Г. Н. ФЕДОРОВА

ДЕРЕККӨРЛАРДЫ ЖОБАЛАУДЫҢ НЕГІЗДЕРІ

«Білім беруді дамытудың федералды институты» федералды мемлекеттік дербес мекемесі «Ақпараттық жүйелер (салалар бойынша)» мамандығы бойынша орта кәсіптік білім беру бағдарламаларын іске асыратын білім беру мекемелері оқу үдерісінде пайдалануға арналған оқу құралы ретінде ұсынылған.

«БДФИ» ФМДМ 2014 жылғы 24 сәуірдегі рецензиясының тіркеу нөмірі 140

2-ші басылым, стереотиптік

ACADEMA

Мәскеу
«Академия» баспа орталығы
2016

ӘОЖ 004.4(075.32)
КБЖ 32.973 -018.2я723
Ф333

Бұл кітап Қазақстан Республикасының Білім және ғылым министрлігі және «Кәсіпқор» холдингі» КЕАҚ арасында жасалған шартқа сәйкес «ТжКБ жүйесі үшін шетел әдебиетін сатып алуды және аударуды ұйымдастыру жөніндегі қызметтер» мемлекеттік тапсырмасын орындау аясында қазақ тіліне аударылды. Аталған кітаптың орыс тіліндегі нұсқасы Ресей Федерациясының білім беру үдерісіне қойылатын талаптардың ескерілуімен жасалды.Қазақстан Республикасының техникалық және кәсіптік білім беру жүйесіндегі білім беру ұйымдарының осы жағдайды ескеруі және оқу үдерісінде мазмұнды бөлімді (технология, материалдар және қажетті ақпарат) қолдануы қажет.Аударманы «Delta Consulting Group» ЖШС жүзеге асырды, заңды мекенжайы: Астана қ., Иманов көш., 19, «Алма-Ата» БО, 809С , телефоны: 8 (7172) 78 79 29, эл. поштасы: info@dcg.kz

Пікір беруші —

Поволжье мемлекеттік телекоммуникациялар және информатика университетінің информатика және есептеу техникасы кафедрасының меңгерушісі, техника ғылымдарының докторы, доцент Н. Ф. *Бахарева*

Федорова Г.Н.

Ф333 Дерекқорларды жобалаудың негіздері: орта кәсіби білім беру мекемелері студенттеріне арналған оқу құралы /Г. Н. Федорова. — 2-ші басылым, стер. — М. : «Академия» баспа орталығы, 2016. — 224 б.

ISBN 978-601-333-334-2 (каз.)
ISBN 978-5-4468-3076-3 (рус.)

Оқу құралы «Ақпараттық жүйелер (салалар бойынша)», ОП.07 «Дерекқорларды жобалаудың негіздері» мамандығы бойынша орта кәсіби білім берудің федералды мемлекеттік білім беру стандартына сәйкес құрылған.

Деректерді ұсыну үлгілеріне сипаттама берілді. Реляциялық үлгісі жан-жақты қарастырылған. Дерекқорларды жобалаудың теориялық негіздері баяндалған. Материалда мысал өте көп келтірілген, бұл оны терең игеруге септігін тигізеді.

Орта кәсіби білім беретін мекемелердің студенттеріне арналған.

ӘОЖ 004.4(075.32)
КБЖ 32.973-018.2я723

ISBN 978-601-333-334-2 (каз.)
ISBN 978-5-4468-3076-3 (рус.)

ҚҰРМЕТТІ ОҚЫРМАН!

Бұл оқу құралы «Ақпараттық жүйелер (салалар бойынша)» мамандығы бойынша оқу-әдістемелік жинақтың бір бөлігі болып табылады.

Оқу құралы ОП.07 «Дерекқорларды жобалаудың негіздері» жалпы кәсіптік пәнді оқуға арналған.

Жаңадан шығып жатқан оқу-әдістемелік жинақтар жалпы білім беретін және жалпы кәсіптік пәндер мен кәсіптік модульдерді оқытуды қамтамасыз ететін дәстүрлі және инновациялық оқу материалдарынан тұрады. Әр жинақта жұмыс берушінің талаптарын ескерумен, жалпы және кәсіби құзыреттіліктерді игеруге қажетті оқулықтар мен оқу құралдары, оқыту мен бақылау құралдары бар.

Оқу басылымдары электронды білім беру ресурстарымен толықтырылады. Электронды ресурстарда интерактивті жаттығулары мен жаттықтырушылары бар теориялық және тәжірибелік модульдер, мультимедиялық объектілер, Ғаламтордағы қосымша материалдар мен ресурстарға сілтемелер бар. Оларға терминдер сөздігі мен оқу үдерісінің негізгі өлшемдері: жұмыс уақыты, бақылау және тәжірибелік тапсырмаларды орындау нәтижесі жазылып отыратын электронды журнал енгізілген. Электронды ресурстар оқу үдерісіне оңай енгізіліп, түрлі оқу бағдарламаларына бейімделе алады.

Алғы сөз

Дерекқорларды құру – жоғары білікті мамандардың қатысуын талап ететін күрделі әрі еңбекті көп қажет ететін жұмыс. Ұсынылып отырған оқу құралының мақсаты – студенттерге дерекқорларды жобалау қағидаларын зерттеуге және олардың тұтастығын қамтамасыз етуге, сонымен бірге дерекқорлармен жұмыс жасаудың қазіргі әдістері мен құралдарымен жұмыс істеуді игеруге көмектесу болып табылады.

Оқу құралы алты тараудан тұрады.

1 тарауда дерекқорлар мен ақпараттың жүйелер теориясының негізгі түсініктеріне анықтама беріледі. Деректерді басқару жүйелерінің өзіне тән белгілері, ДБЖ қызметтері сипатталады. Деректерді өңдеу технологияларының негізгі даму кезеңдері мен оларды одан әрі пайдаланудың перспективалары қарастырылады. Ақпаратты нысандандырылған ұсыну мәселелері талқыланып, деректердің көп деңгейлі үлгісі, деректердің физикалық және қисынды тәуелсіздігі қағидалары енгізіледі. Қазіргі заманғы ДБЖ-ге шолу жасалынады.

2 тарауда деректер үлгісі мен деректер құрылымы түсініктері енгізіледі. Дерекқорлар жүйесінде пайдаланылатын үлгілердің жіктелімі беріледі. Деректерді ұсыну үлгілеріне жалпы сипаттама беріледі. ДБЖ-да пайдалануға болатын деректер үлгілерінің теориялық негіздері сөз болады. Ертеректегі дерекқорларды бақылау жүйелерінде пайдаланылған деректердің теориялық-графтық үлгілері және көпөлшемді мен нысанға бағытталған деректер үлгілерге сипаттама беріледі. Реляциялық үлгіге сипаттама беріліп, оның негізгі құрамдас бөлшектеріне анықтамалар келтіріледі.

3 тарауда кеңінен таралған реляциялық деректер үлгісінің негізгі түсініктері мен құрамдас бөлшектері қарастырылады. Қатынастардың өзіне тән белгілері ашылады. Көптеген мысалдарды пайдаланумен реляциялық алгебра негіздеріне түсініктеме беріледі. Индекстеу механизмдері, кестелерді байланыстыру тақырыптары ашылып, сілтемелік тұтастық түсінігі сипатталады. Реляциялық дерекқорда тұтастықты сақтаудың қағидалары түсіндіріледі. Реляциялық деректер үлгісінің артықшылықтары мен кемшіліктері көрсетіледі.

4 тарауда дерекқорларды жобалау мәселелері қарастырылады. мәндік саланы талдаудан бастап дерекқорларды іс жүзінде іске асыруға дейінгі жобалаудың негізгі кезеңдері жан-жақты сипатталады. Тұжырымдамалық және логикалық үлгілерді құру мысалдары беріледі. Кестелерді бөлшектеп байланыстыру әдісімен дерекқор құрылымын әзірлеу сияқты қалыптандыру үдерісі сипатталынады. Дерекқорларды қалыптандыру қағидаларының негізінде жобалауға мысал келтіріледі. Дерекқорларды жобалаудың автоматтандырылған құралдарын пайдаланудың қажеттілігіне негіздеме беріледі. CASE-технологиялардың негізгі түсініктеріне анықтама беріліп, CASE-технологиялардың түрлі жіктелімдерінің сипатталары сөз болады.

5 тарауда дерекқорлардың түрлі сәулеттерінің артықшылықтары мен кемшіліктері қарастырылып, «клиент-сервис» сәулет үлгісін құру мен пайдаланудың ерекшеліктері сипатталады. Бұл тарауда деректердің тұтастығын қамтамасыз ету мәселесі де қозғалады. Деректерге бір уақытта кірген кезде туындайтын мәселелер мен оларды шешу жолдары баяндалады. Транзакциялар механизмі мен транзакцияларды басқарудың көмегімен деректердің тұтастығын қамтамасыз етудің тәсілдері сөз болады. Транзакцияларды оқшаулау деңгейлеріне анықтама беріледі. Өзгерістерді журналдау рәсімі сипатталады.

6 тарауда SQL құрылымдалған сұратулар тілін пайдаланумен дерекқорларды басқарудың негізгі тәсілдері қарастырылады. Шарттар бойынша іріктеу, деректерді түрлендіру сияқты түрлі қызметтерді орындауға арналған SQL тілі командаларының негізгі санаттарына сипаттама беріледі. Кесте деңгейінде шектеулерді жариялаудан, дерекқорды тұтас күйінде сақтау үшін триггерлерді әзірлеуден бастап транзакцияларды құруға дейін деректердің тұтастығын қамтамасыз ету мәселесіне баса назар аударылады.

Бұл оқу құралында материалдың басым бөлігі қазіргі таңда ДБЖ-ның барлық негізгі жеткізушілері қолдайтын дерекқорлар тілінің жалғыз танылған стандарты – SQL тілінің негіздерін баяндауға арналған. Бұнымен бүгінгі күні дерекқорлар мен қосымшаларды кәсіби әзірлеушілердің SQL тілін жақсы меңгеруге міндетті екендігімен түсіндіруге болады.

Оқу құралының әр тарауына өздік жұмысы кезінде студентке оқылған материалды бекітуге мүмкіндік беретін бақылау сұрақтары енгізілген.

Оқу құралы «Ақпараттық жүйелер» 230401 мамандығының «Дерекқорларды жобалаудың негіздері» пәні бойынша ОКБ (орта кәсіби білім беру) мемлекеттік білім беру стандартына сай дайындалып, автордың тәжірибелік қызметі барысында жинақтаған

материалдарға негізделеді. Аталған мамандық бойынша оқып жатқан студенттер бұл оқу құралын «Дерекқорларды жобалаудың негіздері» пәнін ғана емес, дерекқорлар мен ақпараттық жүйелерді жобалаумен және пайдалануға енгізумен байланысты басқа да пәндерді оқыған кезде пайдалана алады. Ұсынылып отырған материал кәсіпорындарда дерекқорларды әзірлеумен және пайдаланумен айналысатын көптеген мамандарға да пайдасын әкелері анық.

ДЕРЕККӨРЛАР ТЕОРИЯСЫНЫҢ НЕГІЗДЕРІ

1.1. ДЕРЕККӨРЛАР МЕН АҚПАРАТТЫҚ ЖҮЙЕЛЕР. НЕГІЗГІ АНЫҚТАМАЛАР

Қазіргі уақытта ақпарат кез келген қызмет аясының тиімділігін анықтаушы фактор болып отыр. Ақпараттық ағындар көбейіп, деректерді өңдеу жылдамдығына қойылатын талаптар артуда. Көптеген операцияларды енді қолмен ғана орындау жеткіліксіз, олар ақпаратты сақтау мен өңдеу жүйесі үлкен рөл атқаратын озық компьютерлік технологияларды қолдануды талап етеді.

А қ п а р а т — қоршаған ортадағы объектілер, құбылыстар, үдерістер, оқиғалар туралы білімнің белгісіздігін азайтатын деректер. Ақпарат толыққанды, анық, уақтылы, қарама-қайшы келмейтін, дәлме-дәл болуы тиіс. Шешімдерді қабылдау үшін анық ақпаратты дер кезінде беру – ақпараттық жүйелердің негізгі мақсаты.

А қ п а р а т т ы қ ж ү й е — кез келген саладағы міндеттерде ақпараттың жиналуын, сақталуын, өңделуі, іздеуі мен берілуін қамтамасыз ететін техникалық және бағдарламалық құралдардың жиынтығы. Ақпараттық жүйелер ақпараттың ұзақ уақыт сақталуын және сапалы талдауын қамтамасыз етіп, мәселелерді шешуге және жаңа ақпараттық өнімді жасауға көмектеседі.

П ә н с а л а с ы — сол туралы деректер ақпараттық жүйеде сақталынып, пайдаланылып жатқан шынайы өмірдің бөлігі. Пән саласы басқаруды ұйымдастыру және ақыр соңында автоматтандыру үшін зерттелінуі тиіс. Пән саласы сол нысандарды пайдаланатын нысандардың, үдерістердің жиынтығымен, сонымен қатар пән саласына көзқарасы бір көптеген пайдаланушылармен сипатталынады, кез келген ақпараттық жүйені құрмас бұрын пән саласы талданады. Нақты ақпараттық жүйенің пән саласы пайдаланушылардың қызығушылығын туғызатын нақты нысандардың жиынтығы ретінде қарастырылады. Бұл нысандардың

әрқайсысы белгілі бір қасиеттер мен белгілерге ие.

Ақпараттық нысан — шынайы өмірде бар немесе болып жатқан пән саласының мәнін – нысанды, үдерісті, құбылыс немесе оқиғаны суреттеу. Ақпараттық нысан қисынды байланысқан ақпараттың жиынтығы болып табылады, яғни ақпараттық нысандар арасында түрлі байланыстар болуы мүмкін.

Ақпараттық жүйелерде күрделі құрылымды ақпараттың үлкен көлемдері айналып жатады. Жалпылама түрде ақпараттық жүйені келесі блоктардан тұратын схема түрінде көрсетуге болады (1.1-сурет):

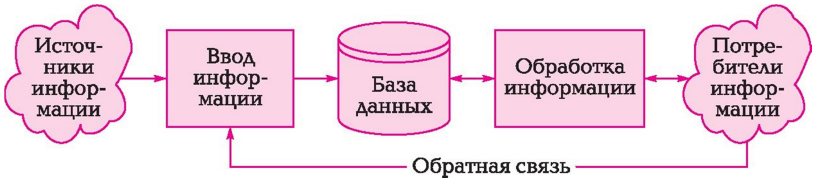
- ақпаратты енгізу (сыртқы орта нысандарының күйі туралыақпаратты жинау);
- дерекқор (деректер қоймасы);
- ақпаратты өңдеу (ақпаратты іздеу, іріктеу, сұрыптау, біріктіру, талдау, шығару);
- кері байланыс (кіру ақпаратын түзету үшін тұтынушы қайта өңдеген ақпаратты беру).

Дерекқор ақпараттық жүйенің негізі, оның орталық тізбегі болып табылады.

Деректер — одан кейін өңдеуге, беруге және сақтауға болатын белгілі бір қалыпқа келтірілген ақпарат (мысалы, ЭЕМ жадындағы немесе ЭЕМ-ге енгізу үшін дайындалған ақпарат).

Дерекқор (Д) — объектілердің жай-күйі мен олардың қандай да бір пән саласындағы қатынастарын көрсететін, өзара байланысқан бірнеше пайдаланушы қолданатын деректердің атаулы жиынтығы.

Деректер құрылымы деп деректердің жекелеген бөлшектері арасындағы байланыстарды көрсететін ережелер мен шектеулердің жиынтығын айтады.



1.1-сурет. Ақпараттық жүйенің жалпы бейнесі

Источники информации – ақпарат көздері
Ввод информации – ақпаратты енгізу
База данных – дерекқор
Обработка информации – ақпаратты өңдеу
Потребители информации – ақпаратты тұтынушылар
Обратная связь – кері байланыс

Деректерді өңдеу — деректер ауқымын түрлендіретін міндеттер жиынтығы. Деректерді өңдеуге ЭЕМ-де деректерді енгізу, қандай да бір өлшемдер бойынша деректерді іріктеу, деректер құрылымын түрлендіру, ЭЕМ-нің сыртқы жадында деректерді ауыстыру, есептеудің немесе басқа түрлендірулердің нәтижесі болып табылатын деректерді шығару кіреді (кесте немесе пайдаланушы үшін қолайлы басқа түрде).

Деректерді өңдеу жүйесі (ДӨЖ) — деректерді басқару міндеттерін атқаратын аппараттық және бағдарламалық құралдардың жинағы.

Деректерді басқару — деректерді өңдеу жүйесінің табысты жұмыс істеуі үшін қажетті деректермен жасалынатын операциялардың барлық түрлері.

Метадеректер — дерекқордың өз құрылымының сипаттамасы. Оларды «деректер туралы деректер» деп те атайды. Бұл дерекқордың барлық нысандары жайлы ақпарат сақталған жүйелік кестелер. Метадеректер деректердің оларды өңдейтін бағдарламалардан тәуелсіздігін қамтамасыз етеді. Егер деректердің сипаттамасы деректермен қоса сақталынып тұрса, деректер құрылымын өңдейтін қосымша бағдарламаларды жазбай-ақ оларға сұрау салып, түрлендіруге болады.

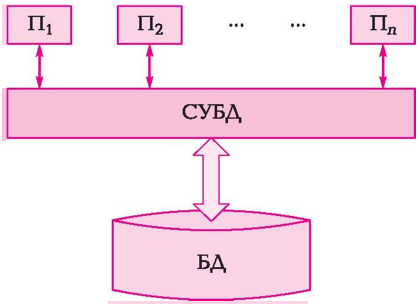
Сақталынып, өңделетін деректер құрылымының сапасы мен сенімділігіне қойылатын талаптар өте жоғары. Дерекқордың құрылымы ішіндегі деректер толық, қарама-қайшылықсыз және тұтас болуы тиіс. Бұған қоса дерекқорлар деректерді дұрыс енгізуге болатындай және анық ақпаратты уақтылы алу мүмкіндігін қамтамасыз ететіндей етіп жобалануы керек. Сондықтан дерекқор құрылымын жобалау және оған кіру әдістерін таңдау аса маңызды міндет болып табылады. Ақпараттық жүйеде деректерді қате немесе бұрыс ұйымдастыру бұл шарттардың орындалмауына әкеліп, жүйенің өзін пайдалануға болмайтындай етеді.

Дерекқорларды басқару жүйесі түсінігі дерекқор түсінігімен тығыз байланысты.

Дерекқорларды басқару жүйесі (ДБЖ) — дерекқорларды құру мен пайдалануды басқаруға арналған тілдік және бағдарламалық құралдардың жиынтығы.

Деректқорларды басқару жүйелері деректерді алып шығуға және өңдеуге, құрылымын өзгертуге және талдауға арналған инструменталдық құралдар болып табылады. ДБЖ деректерге қолайлы, жылдам және, ең маңыздысы, бақыланатын қолжетімділікті ұсынады, деректқорларда ақпаратты енгізу және түрлендіру мүмкіндігін, оның пайдаланушыға берілуін қамтамасыз етеді, деректердің тұтастығын және деректқорды жұмыс күйінде сақталуын қамтамасыз ететін құралдарға ие. Деректқорды басқару жүйесінде әдетте деректерді құрылымдау және басқарудың автоматтандырылған құралдары көп пайдаланушы бар ортада ақпараттың құпиялылығын, қалпына келтіру мен сақталуын қамтамасыз ету құралдарысмен үйлесіп жатады.

Деректқорды басқару жүйелері негізінен кәсіби әзірлеушілерге арналған. Қатардағы пайдаланушы деректқорды басқару мен оның деректерімен жұмысты қосымшалар деп аталатын арнайы қолданбалы бағдарламалардың көмегімен жүзеге асырады. Қосымшалар деректқорларға қол жеткізу құралдарын пайдаланатын бағдарламалау жүйесінің көмегімен ДБЖ ортасында немесе одан тыс жасалынуы мүмкін. Жалпы жағдайда бір деректқормен бірнеше түрлі қосымша жұмыс жасай алады. Мысалы, егер де деректқор қандай да бір кәсіпорынды үлгілесе, онымен жұмыс істеу үшін бірнеше қосымша құрылуы мүмкін: кадрларды қызметке алу қосалқы жүйесіне, қызметкерлердің еңбекақысын есептеу қосалқы жүйесінің жұмысына немесе қоймалық есепке алу қосалқы жүйесіне және т.б. қызмет көрсететін. Бір деректқормен жұмыс істейтін қосымшаларды қарастырған кезде олар параллель және бір-бірінен тәуелсіз жұмыс істей алады деп болжанады. Тап ДБЖ бірнеше қосымшалардың бірыңғай деректқормен жұмысты олардың әрқайсысы түзу орындалып қана қоймай, басқа қосымшалар енгізіп жатқан деректқордағы барлық өзгерістерді ескеретіндей етіп қамтамасыз етуі тиіс (1.2-сурет).



1.2-сур.Қолданбалы бағдарламалар (қосымшалар) мен деректқордың өзара әркеттестігі СУБД - ДБЖ, БД - Деректқор (Д)

Бір пайдаланушылық жүйе – бір уақытта деректқорға бір ғана пайдаланушы қол жеткізе алатын жүйе.

Көп пайдаланушылық жүйе- бір уақытта деректқорға бірнеше пайдаланушы қол жеткізе алатын жүйе.

Көп пайдаланушылық жүйенің басым көпшілігінің негізгі міндеті – әр жеке пайдаланушыға

бір пайдаланушылық жүйемен жұмыс істеп жатқандай мүмкіндік беру. Бір пайдаланушылық пен көп пайдаланушылық жүйелердің айырмашылығы олардың ішкі құрылымында жатыр, түпкілікті пайдаланушыға олар көрінбейді.

Ақпаратты сақтау және қол жеткізуді ұйымдастырудың қазіргі түрі деректер банкі болып табылады. Оның түрлі анықтамалары бар. Олардың ең анығын келесідей келтіруге болады.

Деректер банкі (ДБ) — бұл деректерді бір орталықта жинақтауды және ұжымдық көп мақсатта пайдалануды қамтамасыз етуге арналған арнайы ұйымдастырылған деректер (дерекқорлар), бағдарламалық, тілдік, ұйымдық-әдістемелік құралдардың жүйесі.

Деректер банкі құрамына деректерді автоматтандырылған өңдеуге арналған барлық қосалқы жүйелер кіретін күрделі жүйе болып табылады. Бұл анықтама деректер банкінің негізгі өзіне тән ерекшеліктерін де қамтиды. Біріншіден, дерекқорлар бір пайдаланушының қандай да болсын міндетін шешу үшін емес, көптеген мақсаттарда пайдалану үшін құрылады. Екіншіден, пайдаланушылар үшін деректерді сақтауды ұйымдастырумен, олардың түзетумен және қол жеткізумен байланысты барлық операциялардың орындалуын жеңілдететін арнайы тілдік және бағдарламалық құралдардың болуы деректер банкінің ерекшелігі болып табылады. Сонымен, дерекқор –дерекқорларды тиісті басқару жүйесі бар дерекқордың жиынтығы. Дерекқор – белгілі бір түрде ұйымдастырылған ақпараттың ЭЕМ-гі орталықтандырылған сақтау орны. ДБЖ – дерекқорды құру қызметін, оның жұмыс істеуін, өңдейтін бағдарламаларға қажетті ақпаратты беруді жүзеге асыратын бағдарламалардың арнайы кешені.

Деректер сөздігі (ДС) деректердің құрылымдары, дерекқордағы файлдардың бір-бірімен өзара әрекеттесуі, деректердің түрлері мен оларды ұсыну форматтары, деректерге қол жеткізуді шектеу және т.б. туралы ақпаратты орталықтандырылған сақтауға арналған деректер банкінің қосалқы жүйесі болып табылады. Деректер сөздігі дерекқорды жобалау, пайдалануға енгізу және дамыту кезінде деректер туралы ақпаратты сақтауға және бақылауға мүмкіндік беретін құрал болып табылады.

Кейбір дерекқорларда деректерді орталықтандырылған сақтау емес, компьютер желісінде үлестіру қарастырылады.

Үлестірілген дерекқор (ҮД) — компьютер желісінің түрлі тораптарында орналасқан және түрлі ДБЖ басқарылуы мүмкін бірнеше дерекқордан құралатын бөліктерден тұратын база.

Пайдаланушылар мен қолданбалы бағдарламалар тұрғысынан қарағанда үлестірілген дерекқор қалыпты жергілікті дерекқор болып көрінеді.

Үлестірілген дерекқор жойылған деректерді бірлесіп пайдалану мүмкіндігін, жүйенің сенімділігін, қолжетімділігі мен өнімділігін арттырып, қаражатты үнемдеуге және жүйенің көлемденуін жақсартуға мүмкіндік береді.

Дерекқорлар теориясының негізгі түсініктерінің мән-мағынасы ашылғаннан кейін, оларды әзірлеу мен іске асыру мәселелеріне көшуге болады. Дегенмен де жоғарыда аталған түсіндірмелер мен анықтамаларды толық меңгеру үшін оларды анда-санда қайталау ұсынылады (курстың келесі тарауларын өткен соң). Соңында дерекқорларды пайдаланушылар мен әзірлеушілердің негізгі санаттарын анықтап, дерекқордың қызмет етуіндегі рөлдерін атап кетейік.

Түпкілікті пайдаланушылар — солар үшін дерекқор құрылатын пайдаланушылардың негізгі санаты. Бұл дерекқорды кей уақытта қандай да бір ақпаратты алу үшін пайдаланатын кездейсоқ пайдаланушылар (мысалы, өнімдер немесе қызметтер каталогын қарайтын фирма клиенттері) немесе тұрақты пайдаланушылар (мысалы, лауазымдық міндеттерін атқарған кезде олар үшін арнайы әзірленген бағдарламалармен жұмыс істейтін қызметкерлер). Түпкілікті пайдаланушылардан есептегіш техника және бағдарламалау саласында қандай да бір арнайы білім талап етілмеуі тиіс.

Қосымшаларды әзірлеушілер мен әкімшілер — дерекқорды жобалау, жасау және қайта құру кезінде әрекет ететін пайдаланушылар тобы. Қосымшалардың әкімшілері нақты қосымшаны немесе қызметтік қосалқы жүйеге біріктірілген қосымшалар тобын әзірлеген кезде әзірлеушілердің жұмысын үйлестіреді. Нақты қосымшаларды әзірлеушілер дерекқор ақпаратының нақты қосымша үшін қажетті бөлігімен ғана жұмыс істейді.

Пайдаланушылардың барлық түрлері әрқашан айқындала бермейді. Жеке ДБЖ пайдаланумен ақпараттық жүйелерді әзірлеген кезде деректер банкінің әкімшісі, қосымшалар әкімшісі және әзірлеуші көп жағдайда бір адам болады. Дегенмен де ірі фирма немесе корпорацияда барлық бизнес-үдерістерді немесе олардың басым бөлігін автоматтандыру үшін пайдаланылатын заманауи күрделі корпоративтік дерекқорларды құрған кезде қосымшалар әкімшілерінің топтары және әзірлеушілер бөлімдері болуы мүмкін.

Дерекқор әкімшісінің тобына ең күрделі міндеттер жүктеледі.

Дерекқорлар әкімшілері – дерекқорды әзірлеудің бастапқы кезеңінде оның тиімді ұйымдастырылуына және түпкілікті пайдаланушылардың бір уақытта жұмыс істеуіне жауап беретін пайдаланушылар тобы.

Дамыту және қайта құру кезеңінде пайдаланушылардың бұл тобы дерекқордың ағымдағы пайдаланылуын өзгертпей немесе тоқтатпай оны дұрыс қайта құру мүмкіндігіне жауап береді.

Дерекқор әкімшісі тобының құрамында келесілер болуы тиіс:

- жүйелік талдаушылар;
- деректер құрылымын және деректер банкіне қатысты сыртқы болатын ақпараттық қамсыздандыруды жобалушылар;
- деректерді өңдеуді технологиялық үдерістерін жобалаушылар;
- жүйелік және қолданбалы бағдарламашылар;
- операторлар және техникалық қызмет көрсететін мамандар.

1.2. ДЕРЕКТЕРДІ ӨҢДЕУ ТЕХНОЛОГИЯЛАРЫНЫҢ

Деректерді өңдеу технологияларын дамыту мен жетілдіру есептегіш техниканың эволюциясымен тікелей байланысты. Деректерді басқару жүйелерінің тарихы қырық жылдан астам уақытты қамтиды. Алғашқы дерекқорлар қажетті ақпараттың үлкен көлемдері жинақталған әскери өнеркәсіпте пацда болды. 1968 жылы IBM фирмасының алғашқы өнеркәсіптік ДБЖ пайдалануға енгізілді. Ол кезде дерекқорлар орталық есептегіш машинаның сыртқы жадында сақталынатын. Дерекқормен жұмыс консольді терминалдар арқылы интерактивті тәртіпте жүзеге асырылды, олардың жеке есептегіш қорлары (процессоры, сыртқы жады) болмай, тек орталық ЭЕМ үшін енгізу-шығару құрылғылары қызметін ғана атқарды. Ол кезде шартты болса да, міндеттерді қатар орындау мүмкіндігі қамтамасыз етілген болатын. қорларды үлестіруді басқару қызметн негізінен операциондық жүйе атқарды. Сол кезде деректермен жұмыс жасауға арналған жоғарғы деңгейлі алғашқы тілдер пайда бола бастады, бірақ оларға стандарттар болған жоқ.

Дерекқорлар теориясының одан арғы дамуына үлкен үлес қосқан американдық математик, реляциялық деректер үлгісінің негізін қалаушы - Э.Ф. Кодд болды. Ол үлгі қазіргі күнге дейін пайдаланушылар мен әзірлеушілер арасында үлкен танымалдыққа ие болып отыр.

1970-ші жж. ортасынан бастап дерекқорларды сараптамалық жүйелер мен білім базалары жүйелері саласында пайдалана бастады. Дерекқорларда жасанды ақыл-ой саласында әзірленген білімді ұсыну және нысанға бағдарланған дерекқор үлгісін құру әрекеттері жасалына бастады.

Шынайы өмірді үлгілеу әдістемесінің дамуы әсерінен дерекқорлар технологиясына білімге негізделген жүйелермен байланысты идеяларды көшіру үдерістері дами бастады.

Дербес компьютерлердің пайда болуымен ақпараттық технологиялар дамуының жеке кезеңі басталды. Ол кезде кәсіби, тұрмыстық салада болсын, адамның дербес ақпараттық қажеттіліктерді қанағаттандыру неізгі мақсат болатын. Кәсіби емес пайдаланушыларға арналған бағдарламалық қамсыздандыру құрыла бастады. Бұл дерекқорлармен жұмыс жасаудың тәсілдеріне әсер етпей қоймады. Дербес компьютерлер бағасының қолжетімділігі пайдаланушылар ауқымын кеңейтіп, деректерді дамыған қолайлы өңдеу бағдарламаларына деген сұранысты туындатты, бұл бағдараламалық қамсыздандыруды жеткізушілерді жаңа ДБЖ әзірлеуге, жаңа мүмкіндіктерді ұсынып, интерфейсін жақсартып, жылдам жұмыс етуін ұлғайта отырып, оларды жетілдіруге мәжбүрледі. Дерекқорларды басқарудың түрлі жүйелері ұқсас қызметтерді атқарды, сондықтан түрлі ДБЖ түрлі форматтағы деректерін экспорттау-импорттау әдістері әзірленді. Деректермен әрекет етудің көп деңгейлі тілдерінің стандарттары құрылды. ДБЖ басым бөлігінің дамыған әрі қолайлы пайдаланушылық интерфейсі, бағдарламалаусыз дайын қосымшаларды әзірлейтін құралдар жинағы болды. Құралдар ортасы тез арада бірыңғай кешенге жинақтала алатын экранды пішіндердің шаблондары, есеп берулер, графикалық құрастырушылар түріндегі қосымшалардың дайын бөлшектерінен құралды. Бұл кезеңнің жағымды бір ерекшелігі дербес ДБЖ тарапынан аппараттық қамсыздандыруға қоятын салыстырмалы түрде қарапайым талаптары болды. Оларды толыққанды ДБЖ деуге де келмейді. Бұл қатардыңбұған дейін кеңінен пайдаланылып келе жатқан өкілдері - СУБД dBase, FoxPro, Clipper, Paradox. Бұл кезеңде барлық ДБЖ бір пайдаланушыға арналған дерекормен жұмыс жасауға арналған болатын, не болмаса бірнеше пайдаланушы кезектесіп жұмыс жасаған болатын. Бір пайдаланушылық тәртіп дерекқорға әкімшілік жүргізу қызметінің жойылуына және соған байланысты – дерекқорды қорғайтын құралдардың жоқтығына әкелді.

Дербес және үстелдік дерекқорлар кезеңінің орнына бірігу үдерісі келді – жергілікті желілер пайдалануға енгізіліп, өңделетін ақпараттың көлемі артты. Түрлі орындарда сақталынып, өңделетін бір-бірімен қисынды байланысқан деректерді келісу қажеттілігі туындады. Бұл міндеттерді табысты шешу үстелдік ДБЖ-дың барлық артықшылықтарын сақтап, ақпараттың қатар өңделуін және дерекқордың тұтастығын сақтауға мүмкіндік беретін үлестірілген дерекқорлардың пайда болуына әкеледі. 1990-шы жылдардың

екінші жартысында бөлшектер физикасы, молекулалық биология мен ғылымның басқа салаларынағы зерттеу жобалары аясында құрылған аса үлкен дерекқорлар пайда бола бастады. Ірі мультимедиялық дерекқорлар көптеген электронды кітапханалар құрамында құрыла бастайды.

Деректермен жұмыс істеу технологиялары дамуының қазіргі кезеңі деректерге қол жеткізудің жаңа тәсілдерімен сипатталынады. Енді пайдаланушыға дерекқормен жұмыс жасау үшін мамандандырылған клиенттік бағдарламалық қамсыздандыруды қолданудың еш қажеті жоқ – стандартты ғаламтор-браузері болса жеткілікті. Мұндай әдісті дерекқорларға алыстан қол жеткізу үшін ғана емес, кәсіпорын немесе ұйымның жергілікті желісінде де пайдаланады.

Деректерді өңдеу технологиялары үнемі жетілдіріліп отырады. Қазіргі ДБЖ көбісі түрлі сәулетті компьютерлерде және түрлі операциялық жүйелермен жұмыс жасай алады. Бұл кезде пайдаланушылар үшін ДБЖ түрлі платформаларда басқарып жатқан деректерге қол жеткізуде еш айырмашылық жоқ. Әкімшілік жүргізу мен деректерді қорғау құралдары дамуда.

Дерекқорлар технологияларының дамуы барысында жаңа бағыттар туындауда.

Соңғы уақытта грид (Grid) деп аталатын орта тұжырымдамасының белсенді дамуына байланысты ең жаңа бағыттардың бірі осы грид-технологиялармен байланысты. Бұл үлестірілген есептеулердің түрі – түрлі тораптардың ресурстары (есептегіш, ақпараттық) пайдаланушыларға бірыңғай интерфейс арқылы қолжетімді болатын орталықтандырылмаған компьютерлік желілік орта. Бұл технология үлкен есептегіш ресурстарды талап ететін ғылыми, математикалық міндеттерді шешу үшін, сонымен қатар экономикалық болжам, сейсмоталдау, жаңа дәрі-дәрмектердің қасиеттерін зерттеу сияқты еңбекті көп қажет ететін міндеттерді шешу үшін коммерциялық инфрақұрылымда қолданылады.

Дерекқорлар технологияларындағы қарқынды дамып жатқан жаңа бағыттардың бірі XML стандарттарының Web-кешені деген жаңа технологиялық платформаның құрылуымен байланысты туындады. Бұл бағытқа XML-бағдарланған ДБЖ деген атқа ие болған түптұлғаларды және коммерциялық бағдарламалық өнімдерді зертеу, әзірлеу жатады. Мұнда тәжіибе жүзінде қолданылатын технологиялар қалыптасып, көптгене қосымшалар бар. Дегенмен де жоғары өнімділікті қамтамасыз ететін қызметі жағынан толыққанды XML-бағдарланған жүйелерді құру мәселесі әлі шешімін таппай отыр.

Сымсыз байланыс арналарын пайдалануға мүмкіндік беретін телекоммуникациялар технологияларының дамуы мобильді дерекқорлар жүйелері деп те аталатын мобильді сәулеті бар дерекқорлардың үлестірілген жүйелерін әзірлеуге септігін тигізді. Бұл келешегі зор дерекқорлар технологияларының бірі болып табылатыны сөзсіз.

Қазіргі таңда дерекқорлар ең үлкен сұранысқа ие ақпараттық технологиялардың бірі болып табылады. Кейбір авторлардың пікірінше, дерекқорлардың пайда болуы бағдарламалық қамсыздандыру саласындағы ең үлкен жетістік болды. Дерекқорлар жүйесі көптеген ұйымдардың жұмысын түбегейлі өзгертті, олар қолданылмайтын қызмет саласы қалмады деуге болады.

1.3. ДЕРЕКҚОРЛАРДЫ БАСҚАРУ ЖҮЙЕЛЕРІ. ДБЖ НЕГІЗГІ ҚЫЗМЕТТЕРІ

Қолданбалы бағдарлама тұрғысынан алғанда файл – жазуға болатын және деректерді оқуға болатын сыртқы жадының атаулы облысы. Файлдарды атау ережелері, файлдарда сақтаулы деректерге қол жеткізу тәсілі мен сол деректердің құрылымы файлдарды басқаратын нақты жүйеге және файлдың түріне байланысты. Файлдарды басқару жүйесі сыртқы жадыны үлестіру, файлдардың атын сыртқы жадыдағы тиісті мекенжайларға кіргізу мен деректерге қол жеткізуді қамтамасыз ету қызметін атқарады. Алайда бұл қарапайым болсын ақпараттық жүйелерді құруға жеткіліксіз. Файлдарды басқару жүйелері қисынды келіскен файлдар жинағын қолдау мүмкіндігін, деректермен әрекет ету тілін, түрлі бұзылулардан кейін ақпаратты қалпына келтіруді, бір уақытта бірнеше пайдаланушылардың жұмысын қамтамасыз ете алмайды. Қолданбалы бағдарлама осы қасиеттерге ие деректерді басқарудың қандай да бір жүйесіне сүйенуі тиіс. Ондай жүйе дерекқорларды басқару жүйесі болып табылады.

Деректерді өңдеу жүйелеріне деректердің өздері, дерекқорды басқару жүйесі мен деректермен дерекқорды басқару жүйесі арқылы қатынас жасайтын қолданбалы бағдарламалық қамсыздандыру кіреді.

Дерекқорларды басқару жүйелерінің келесі қызметтерін бөліп көрсетуге болады:

- деректерді сыртқы жадыда басқару;
- жедел жадыда деректерді басқару;

- транзакцияларды басқару;
- журналдау,резервтік көшірме мен қалпына келтіру;
- дерекқор тілдерін қолдау.

Сыртқы жадыда (дискілерде) деректерді басқару. ДБЖпайдаланушыларға келесі мүмкіндіктерді беруі тиіс:

- дерекқорда деректерді сақтау, алу және жаңарту;
- деректерге қолжетімділікті басқару;
- бірнеше пайдаланушылардың қатар жұмыс істейін қамтамасыз ету;
- деректер тұтастығын қолдау.

Дерекқорда деректерді сақтау, алу және жаңарту мүмкіндігі – ДБЖ-нің ең іргелі қызметі. Бұл қызметті іске асыру тәсілі түпкілікті пайдаланушыдан құпия сақталынуы тиіс (ДБЖ файлдық жүйені пайдалана ма, файлдар қалай құрылған және т.б.).

Деректерге қолжетімділікті бақылау — парольмен қорғауды, дерекқор мен оның жекелеген бөлшектеріне қолжетімділік деңгейін сақтауды пайдалана отырып, дерекқорға тек рұқсат етілген қолжетімділікті қамтамасыз ету мүмкіндігі. Әр пайдаланушы дерекқордың оның пайдаланушылық құқықтарына сай қолжетімді болатын деректеріменғана жұмыс жасау мүмкіндігіне ие боулы тиіс. Деректерді басқару жүйесіндегі қауіпсіздік осылайша қамтамасыз етіледі.

Параллельділікті басқарудың мәні ДБЖ-да бір уақытта қол жеткізу кезінде көптеген пайдаланушыларға деректерді дұрыс жаңартуға кепілдік беретін механизмдердің бар болуында. Бірлесіп жұмыс жасау кезіндегі қайшылықтар деректердің қисынды тұтастылығының бұзылуына әкелуі мүмкін, сондықтан жүйе деректерді басқа біреу пайдаланып жатқан кезде пайдаланушыға оларды жаңартуға жол бермейтін шараларды қарастыруы тиіс. Сөз болған жағдайларда аталмыш «оқшаулаулар» пайдаланылады. Оқшаулаудың алуан түрлері бар – оқшауланған жазбалардың санымен ерекшеленетін кестелік, беттік, жолдық және т.б.

Деректер тұтастығын қолдау деректермен олардағы өзгерістер берілгне ережелерге сай болуы үшін бақылау құралдарымен жүзеге асырылады. Дерекқордың тұтастығы – дерекқорда толық, қарама-қайшылықсыз және пәндік саланы дәлме-дәл көрсететін ақпарат барын білдіретін қасиет. Дерекқордың тұтастығын қолдауға дерекқорда қарама-қайшылықтар туындаған жағдайда оның тұтастығын тексеру мен қалпына келтіру кіреді. Дерекқордың тұтастық күйі қорда сақтаулы деректер қанағаттандыруы тиіс шарттар түріндегі тұтастықты шектеу көмегімен сипатталады. Деректері дерекқорда сақталынатын объектілер атрибуттарының мүмкін мәндері ауқымдарын шектеу немесе реляциялық дерекқор

кестелерінде қайталанатын жазбалардың жоқтығы осындай шарттарға мысал бола алады.

Жедел жадыда деректерді басқару. ДБЖ жұмыс жасайтын дерекқордың көлемі айтарлықтай үлкен – әдетте жедел жадының қолжетімді көлемінен әлдеқайда үлкен болады. Деректердің кез келген бөлшегімен қатынас жасаған кезде сыртқы жадымен алмасу жүргізілсе, барлық жүйе сыртқы жады құрылғысының жылдамдығымен жұмыс істейтіні түсінікті. Бұл жылдамдықты өсірудің жалғыз тәсілі жедел жадыда деректерді буферлеу болып табылады. Буферлер сыртқы және жедел жады арасындағы алмасуды жылдамдатуға арналған жедел жадының бөліктері болып табылады. Буферлерде дерекқордан алынған бөліктер уақытша сақталынады, олардан алынған деректерді ДБЖ-мен жұмыс жасаған кезде пайдалану немесе өңделгеннен кейін қорға жазу жоспарланады. Дамыған ДБЖ-да буферлерді алмастырудың өз ережелері бар жедел жады буферлерінің жеке жинағы қолданылады. Болашақта компьютерлердегі жедел жадының көлемі буферлеу жайлы алаңдамауға мүмкіндік беретіндей қомақты болады.

Транзакцияларды басқару. Транзакция — ДБЖ біртұтас дүние ретінде қарастыратын дерекқормен жасалынатын іс-әрекеттердің жиынтығы. Басқаша айтқанда, бұл деректермен барлық операциялар орындалатын, не болмаса олардың ешқайсысы орындалмайтын операциялардың кезектілігі(«барлығы немесе ешқайсысы» қағидасы). Егер де транзакция табысты орындалса, ДБЖ сол транзакциямен жасалған дерекқордағы өзгерістерді сыртқы жадыда көрсетеді. Егер де транзакция аясындағы барлық өзгерістер жойылса, олардың ешбірі дерекқордың жай-күйінде көрсетілмейді. Транзакция түсінігі дерекқордың қисынды тұтастығын сақтап тұру үшін қажет. Банк жүйесінде ақшаны бір шоттан екіншісіне аудару операциясы транзакцияға мысал бола алады. Не барлық әрекеттерді орындау керек (бір клиенттің шотын көбейтіп, екіншісінікін азайту) не болмаса бұл әрекеттің ешқайсысын орындамау керек. Бір шоттағы ақшаның сомасын азайтып, екіншісінде ұлғайтпауға болмайды. Бірінші әрекетті орындағаннан кейін (шоттағы ақшаны азайтқан соң) бір ақау шықты делік. Мысалы, клиент компьютерінің дерекқормен байланысы үзіліп немесе клиенттің компьютерінде операциялық жүйенің қайта жүктелуіне әкелген жүйелік ақау орын алуы мүмкін. Бұл жағдайда дерекқормен не болды? Бірінші клиенттің шотында ақшаны кемсіту командасы орындалды, ал басқа шотта ақшаны көбейтуге деген екінші команда жоқ, бұл дерекқордың қарама-қайшылықты, өзексіз күйіне әкелетін еді. Алдымен бір шоттан ақша шешіліп, оларды басқа шотқа қосады. Бұл әрекеттің біреуі табысты

орындалмаған жағдайда операцияның нәтижесі дұрыс болмай, шоттар арасындағы теңгерім бұзылады. Сондықтан бұл жағдайда жүйе алдыңғы күйге операциялар кезегі орындалғанға дейін оралуы керек.

Бір пайдаланушылық және транзакциялар қатар қосылуы мүмкін көп пайдаланушылық ДБЖ-да транзакцияларды бақылау маңызды.

Бірнеше транзакция қатар орындалған кезде қарама-қайшылықтар туындауы мүмкін, оларды шешу де ДБЖ қызметі болып табылады. Мұндай жағдайлар орын алған кезде әдетте транзакцияны «кері қайтару» болады, яғни бір немесе бірнеше транзакциямен жасалынған өзгертулерді жою. Деректердің тұтастығын қамтамасыз ету мәселелері зерделенген кезде транзакциялар механизмі мен оларды басқарау анағұрлым толық қарастырылатын болады.

Журналдау, резервтік көшірме мен қалпына келтіру. ЭЕМ жұмысы барысында ақаулар (мысалы, электр қуатын өшірудің салдарынан) және машиналық деректер құрылғыларының бұзылуы орын алуы мүмкін. Бұл кезде деректер бұзылып, одан кейінгі жұмыс жасау мүмкін болмай қалады. ДБЖ-ға қойылатын талаптардың бірі деректердің сыртқы жадыда сақталуының сенімділігі, соның ішінде физикалық және қисынды тұтастықты қорғау болып табылады. Сақтаудың сенімділігі дегеніміз ДБЖ кез келген аппараттық немесе бағдарламалық ақаудан кейін (қисынды немесе физикалық бас тартулар) соңғы келісілген дерекқордың күйін қалпына келтіру мүмкіндігіне ие болуы керек.

Физикалық тұтастықты қорғауға өзгерістерді журналдау, резервтік көшіру мен ақаулардан кейін дерекқорды қалпына келтіру кіреді.

Өзгерістерді журналдау қарапайым жағдайда сыртқы жадыға дерекқорда орын алып жатқан барлық өзгерістерді кезекпен жазу болып табылады. Өзгерістің реттік нөмірі, түрі мен уақыты; транзакцияны сәйкестендіргіш; өзгерген нысан (сақталынған файл нөмірі және ондағы деректерді оқшаулағыш нөмірі, оқшаулағыш ішіндегі қатар нөмірі); нысанның алдындағы жай-күйі мен объектінің жаңа күйі сияқты ақпарат жазып алынады. Дерекқордағы өзгерістер журналы – бұл осылайша қалыптасып жатқан ақпарат. Журналда транзакцияның басталу мен аяқталу туралы белгі және бақылау нүктесін қабылдау туралы белгілер болады. Ол ДБЖ пайдаланушылар қол жеткізе алмайтын дерекқордың ерекше бөлігі болып табылады. Журнал ерекше мұқияттылықпен жүргізіледі, оған дерекқордың негізгі бөлігінде болып жатқан өзгерістер туралы жазбалар келіп түседі. Кейде түрлі физикалық дискіде орналасатын

журналдың екі көшірмесі құрылып, толтырылады.

Дерекқорды резервтік көшіру – дерекқорлар бұзылған немесе жойылған жағдайда деректерді түпнұсқалы немесе жаңа орында қалпына келтіруге арналған тасымалдағышта деректердің көшірмесін құру үдерісі.

Дерекқорды қалпына келтіру – қисынды және физикалық ақаулар орын алған жағдайда дерекқорды өзекті күйге келтіретін ДБЖ қызметі. Жоғарыда аталып өткендей,ақау жүйенің немесе есте сақтайтын құрылғының, аппараттық және бағдарламалық қамсыздандырудың істен шығуынан болуы мүмкін. Мұндай жағдайда ДБЖ дерекқорды қалпына келтіру және қарама-қайшылықсыз күйге қайтару механизмін ұсынуы қажет. Егер өзгерістер журналы да, дерекқордың өзі де бұзылған физикалық бас тарту орын алған жағдайда, резервті көшірмені орындау сәтінде ғана қалпына келтіру мүмкін.

Дерекқор тілдерін қолдау. Дерекқорлармен жұмыс жасау үшін жалпы алғанда дерекқорлардың тілдері деп аталатын арнайы тілдер пайдаланылады. Қазіргі ДБЖ-да әдетте дерекқорды құрудан бастап дерекқормен жұмыс жасауға қажетті барлық құралдары бар және дерекқорлармен негізгі пайдаланушылық интерфейсті қамтамасыз ететін бірыңғай біріккен тіл қолданылады. Қазіргі таңда кеңінен таралған реляциялық ДБЖ-дың стандартты тілі – SQL тілі (StructuredQueryLanguage).

Негізгі қызметтерден басқа ДБЖ түрлі қосалқы қызметтердің жинағын да ұсынады. Қосалқы утилиталар әдетте дерекқорға тиімді әкімшілік жүргізуге арналған, оған келесілер кіреді:

- деректерді экспорттау/импорттау;
- дерекқорға мониторинг жүргізу (дерекқорды қызмет ету мен пайдалану сипаттамаларына бақылау жүргізу);
- дерекқордың өнімділі мен пайдалану дәрежесін бағалауға мүмкіндік беретін статистикалық талдау;
- индекстерді қайта құру;
- «қоқысты» (пайдаланылмайтын жазбаларды) жинау мен жойылған жазбаларды сақтағыш құрылғылардан жою үшін жадыны үлестіру, босаған кеңістікті біріктіру және қажет болған жағдайда жадыны қайта үлестіру.

Дерекқорды басқарудың типтік жүйесін ұйымдастыру және оның құрамдас бөлшектерінің құрамы біз қарастырып кеткен қызметтер жинағына сәйкес келеді.

Дерекқорларды басқарудың қазіргі жүйесінде ДБЖ өзегін, дерекқор тілінің процессорын, орындау уақытын сақтау қосалқы жүйесін, сонымен қатар дерекқорға қызмет көрсету бойынша

ларды (сыртқы утилиталар) бөліп көрсетуге болады. Кейбір жүйелерде бұл бөліктер анық, басқаларында – көмескі көрініс табады, дегенмен мұндай бөлуді дерекқорларды басқарудың барлық жүйелерінде жүргізуге болады.

ДБЖ өзегі деректерді сыртқы жадыда басқаруға, жедел жады буферлерін басқаруға, транзакцияларды басқаруға және журналдауға жауапты. Басқаша айтқанда, ДБЖ өзегі – бұл дерекқорды құруға және сақтауға қажетті және жеткілікті бағдарламалық модульдердің жинағы, пайдаланушыларға ақпараттық қызмет көрсету бойынша қалыпты міндеттерді шешетін әмбебап бөлігі.

Сервистік бағдарламалар сипатталынатын пәндік салаға және нақты пайдаланушының қажеттіліктеріне байланысты болатын бірқатар қосымша мүмкіндік пен қызметті ұсынады.

Дерекқор тілінің процесоры дерекқорлар тілінің операторларын машиналық кодтарда ұсынылатын қандай да бір орындалатын бағдарламаға құрастырады.

Дерекқордың жекелеген утилиттеріне әдетте дерекқорды жүктеу және шығару, статистиканы жинақтау, дерекқордың тұтастығын жаһандық тексеру сияқты және т.б. рәсімдерді бөліп шығарады. Утилиттер ДБЖ өзегінің интерфейсін пайдаланумен, кейде тіпті өзектің ішіне енумен бағдарламаланады.

1.4. ДЕРЕКҚОРДЫҢ СӘУЛЕТІ. ФИЗИКАЛЫҚ ЖӘНЕ ҚИСЫНДЫ ТӘУЕЛСІЗДІК

БДЖ көп деңгейлі сәулеті туралы идеялар алғаш рет 1975 жылы жарияланған Американдық ұлттық стандарттар институтының (ANSI) Стандарттау комитетінің есеп беруінде сөз етілген болатын. Онда сыртқы, тұжырымдамалық және физикалық (ішкі) деңгейлерден тұратын ДБЖ сәулетінің жалпыланған үш деңгейлі үлгісі ұсынылды (1.3-сурет). Мұндай сәулетті енгізудің негізгі мақсаты – дерекқордың пайдаланушылық түсінігін оның физикалық түсінігінен бөліп айыру болып табылады.

ДБЖ сәулетінің түрлі деңгейлерінде деректер абстракицясының түрлі деңгейі қолданылады.

ДБЖ сәулеті ең алдымен пайдаланушылық пен жүйелік деңгейлердің бөлінуін қамтамасыз етуі тиіс.

1.3-сурет. ANSI ұсынған дерекқорды басқару жүйесінің үш деңгейлі үлгісі

Сыртқы үлгілердің деңгейі – әр үлгінің өз деректер ұсынысы бар ең жоғарғы деңгей. Пайдаланушылардың жеке топтары осы қосымша аясында қолжетімділігі бар деректермен ғана жұмыс жасайды, яғни әр қосымша соған қажетті деректерді ғана көріп, өңдей алады. Мысалы, егер дерекқор қандай да бір кәсіпорынды үлгілесе, онымен жұмыс жасау үшін бірнеше қосымша құрылуы мүмкін. Кадрларды есепке алу қосалқы жүйесіне қызмет көрсететін қосымша қызметкерлердің жеке деректерін – туған жылын, төлқұжат деректерін, мекенжайын және т.б. пайдаланады. Қоймадағы тауарға есеп жүргізетін қосымшада қызметкерлер тауарды қабылдаушы және жіберуші тұлғалар ретінде қарастырылып, олар туралы жеке дерек қажет болмайды, есесіне тауар мен жеткізіп берушілер туралы ақпарат жазылуы тиіс. Бұл ақпараттың бәрі бір дерекқорда сақталынады. Түрлі қосымшалар түрлі бағдарламалау тілін пайдаланумен құрылуы мүмкін. Қолданбалы бағдарламашылар негізінен жоғары деңгейлі тілдерді, не болмаса ДБЖ арнайы тілдерін қолданады. Бір дерекқормен жұмыс жасайтын қосымшаларды қарастырған кезде, олар қатар және бір-бірінен тәуелсіз жұмыс жасай алады деп болжамданады. Тап ДБЖ бірнеше қосымшаның бірыңғай дерекқормен жұмысын әрқайсысы дұрыс орындалып, дерекқорға басқа қосымшалар енгізіп жатқан өзгерістерді ескеретіндей қамтамасыз етуі тиіс. Пайдалану-

шылар тобы үшін деректерді ұсынуды сипаттау сыртқы схема деп аталады. Дерекқор жүйесінде пайдаланушылардың түрлі топтары немесе міндеттері үшін бірнеше сыртқы схема бір уақытта қолданылуы мүмкін.

Сәулеттің тұжырымдамалық деңгейі негізгісі болып табылады және дерекорды жалпы түрде оның барлық қосымшаларына ұсыну қызметін атқарады. Тұжырымдамалық деңгей үш деңгейлі сәулеттегі аралық деңгей болып табылады және дерекқордағы барлық ақпараттың дерексіз түрде ұсынылуын қамтамасыз етеді.Бұл пәндік саланыңнысандандырылған ақпараттық-қисынды үлгісі, яғни деректерді ұсыну мен сақтау тәсілдеріне тәуелді болмайтын пәндік саланың деректеріне талаптарды толық ұсыну. Бұл деңгейде дерекқорды сипаттау тұжырымдамалық схема деп аталады. Тұжырымдамалық схемаға нысандар мен олардың атрибуттары, нысандар арасындағы байланыстар, деректерге қойылатын шектеулер, деректер туралы мағыналық ақпарат, деректердің қауіпсіздігі мен тұтастығын қамтамасыз ету кіреді. Тұжырымдамалық схема деректердің барлық бөлшектері мен олардың арасындағы қарым-қатынастардың бірыңғай қисынды сипаттамасы, бүкіл дерекқордың қисынды құрылымы болып табылады. Кез келген үлгі сияқты, тұжырымдамалық схема шынайы әлем нысандарының өңдеу жағынан тек қомақты ерекшеліктерін ғана көрсетеді.

Сәулеттің физикалық (ішкі) деңгейі дерекқордың сақтау ортасында ұсынылуын қолдайды. Дерекқордың физикалық деңгейдегі сипаттамасы ішкі схема немесе сақтау схемасы деп аталады.Физикалық деңгей — файлдарда немесе ақпараттың сыртқы тасымалдағыштарында орналасқан беттік құрылымдарда орналасқан деректердің өздері. Бұл дерекқорды физикалық іске асыру. Бұл деңгейде оңтайлы өнімділікке қол жеткізіп, диск кеңістігін үнемді пайдалануды қамтамасыз ету қажет. Физикалық деңгейде деректерді есте сақтау құрылғыларында орналастыру, индекстерді орналастыру, деректерді алу және т.б. мақсатта ДБЖ-нің операциялық жүйенің қолжетімділік әдістерімен өзара әрекеттесуі жүзеге асырылады. Физикалық деңгейде деректер мен индекстерді сақтау үшін диск кеңістігін үлестіру туралы ақпарат, жазбаларды сақтаудың сипаттамасы (деректердің сақталынатын бөлшектерінің нақты өлшемдерін көрсетумен), жазбаларды орналастыру туралы деректер, деректерді қысу мен шифрлеу әдістер туралы деректер сақталынады. Физикалық деңгейді ДБЖ басшылығымен операциялық жүйе бақылайды. Қазіргі таңда физикалық деңгей толықтай ДБЖ-мен қамтамасыз етіледі. Дерекқорды жобалаған кезде басты назар тұжырымдамалық деңгей үлгісін құруға аударылады.

ANSI сәулет үлгісінде ДБЖ-да деректердің деңгейаралық көрінісін қамтамасыз ететін механизмдер бар деп қарастырылады. Бұл механизмдердің қызметтік мүмкіндіктері деректердің дерексізденуін қамтамасыз етіп, деректердің барлық деңгейдегі тәуелсіздігінің дәрежесін айқындайды.

Деңгейаралық көріністерге сай деректердің қисынды және физикалық тәуелсіздігін бөліп көрсетуге болады.

Қисынды тәуелсіздік сыртқы схемалардың тұжырымдамалық схемаға енгізілетін өзгерістерден қорғалғанын білдіреді. Бұдан басқан қисынды тәуелсіздік бір дерекқормен жұмыс істеп жатқан басқа қосымшаларды түзетпей бір қосымшаның өзгеру мүмкіндігін және физикалық деректерге қолжетімділік механизмін қайта құруды білдіреді.

Физикалық тәуелсіздік тұжырымдамалық схеманың сақтау схемасына енгізіліп жатқан өзгерістерден қорғалғанын білдіреді. Физикалық тәуелсіздік те дерекқормен жұмыс жасап жатқан барлық қосымшалармен жұмыс жасауды сақтай отырып, сақтаулы ақпараттың бір тасымалдағыштан басқасына ауыстыру мүмкіндігін білдіреді.

БАҚЫЛАУ СҰРАҚТАРЫ

1. Келесі түсініктерге анықтама беріңіз: ақпарат, ақпараттық жүйе, ақпараттық нысан және ақпараттық технология.
2. Дерекқор деген не және оның ақпараттық жүйедегі орны қандай?
3. Деректер мен метадеректер арасындағы айырмашылық?
4. Дерекқорларды басқару жүйелерінің міндеті қандай?
5. Қолданбалы бағдарламалар дерекқормен қалай әрекеттеседі?
6. Деректер банкінің дерекқордна айырмашылығы қандай?
7. Деректер банкінің құрамына қандай құрамдас бөлшектер кіреді?
8. Деректер сөздігі не үшін пайдаланылады?
9. Дерекқорды пайдаланушылар мен әзірлеушілердің негізгі санаттарын атап шығыңыз. Дерекқордың қызмет етуіндегі олардың атқаратын рөлдері қандай?

10. ДБЖ дамуының әр кезеңінің ерекшеліктерін атап шығыңыз.
11. Дерекқор технологияларының дамуында дербес компьютерлердің пайда болуы қандай рөл атқарды?
12. Деректермен жұмыс жасау технологиялары дамуының қазіргі кезеңі немен сипатталынады?
13. Дерекқор дамуының келешегі бар бағыттары қандай?
14. Дерекқорларды басару жүйелерінің қызметтері қандай?
15. Дерекқорлардағы журналдау не үшін қажет?
16. Транзакция деген не?
17. ДБЖ қандай қосалқы қызметтерді ұсынады?
18. Дерекқордың тұтастығы деген не?
19. ДБЖ-ның үш деңгейлі сәулеті деген не?
20. Сыртқы үлгілер деңгейінің ерекшелігі неде?
21. Тұжырымдамалық деңгейдің ерекшелігі неде?
22. Физикалық деңгейдің ерекшелігі неде?
23. Дерекқордың схемасы деген не?
24. Деректердің қисынды және физикалық тәуелсіздігі нені білдіреді?

ДЕРЕКТЕР ҮЛГІЛЕРІ

2.1. ДЕРЕКТЕР ҮЛГІЛЕРІ ТҮСІНІГІ

Дерекқорлардың классикалық теориясында деректер үлгісі – бұл дерекқорды басқару жүйесіндегі деректерді ұсыну және өңдеудің формалды теориясы.

Ұзақ уақыт бойы «деректер үлгісі» термині ешбір ресми анықтамасыз қолданылып келді. Бұл түсінікке нақты анықтама берген алғашқы мамандардың бірі Э.Кодд болды. Алайда деректер үлгісі түсінігін Кодденгізгенмен, деректер үлгісінің ең таралған түсіндірмесі Кристофер Дейтке тиесілі. Оның анықтамасына сәйкес деректер үлгісіне үш аспект кіреді:

1) құрылым аспектісі: дерекқордағы деректердің түрлерін және қисынды құрылымдарын сипаттау әдістері;

2) манипуляция аспектісі: деректермен жұмыс жасау әдістері;

3) тұтастылық аспектісі: дерекқор тұтастылығын сипаттау және қолдау әдістері.

Құрылым аспектісі дерекқордың қисынды тұрғыда не болып табылатынын анықтайды.

Манипуляция аспектісі дерекқор сұранысын жазу үшін арналған бір немесе бірнеше тілдің ерекшелігіне ие. Бұл тілдер ешқандай нақты пысықталған синтаксисі жоқ, дерексіз болуы мүмкін. Деректер үлгісінің манипуляциялық бөлігінің негізгі міндеті – көркемділік деңгейі осы үлгіге сай келетін ДБЖ іске асырылуында қолданылуы тиіс дерекқордың эталонды тілін қамтамасыз ету болып табылады. Манипуляция аспектісі дерекқор күйлері арасындағы ауысу тәсілдерін айқындайды – деректерді түрлендіру тәсілдері және деректерді дерекқордан шығару тәсілдері

бөлігінде бұл үлгіге сай келетін ДБЖ іске асырылуында міндетті түрде қолданылуы тиіс тұтастылықты шектеудің механизмдері сипатталынады.

Дерекқорлар теориясында деректер үлгісі кез келген дерекқордың өзегі болып табылады. Деректер үлгісі пәнді саланың ақпараттық нысандарын, олардың арасындағы өзара байланысты сипаттап:

- дерекқорды басқарудың қисынды және логикалық аспектілері арасындағы шекараны айқындауға (деректердің тәуелсіздігі);
- түпкілікті пайдаланушылар мен бағдарламашыларға деректер мәнін жалпы түсіну мүмкіндігін және құралын қамтамасыз ету (байланысқа бейімділік);
- жазбалардың үлкен жиынтықтарында (жалпы жағдайда әр типті деректерді) бір типті операцияларды бірыңғай операция ретінде (көптегендерін өңдеу) орындау мүмкіндігінқамтамасыз ететін жоғары деңгейлі тілдік түсініктерді айқындауға мүмкіндік береді.

Деректер үлгісінде барлық нақты ДБЖ мен олар басқаратын дерекқорлар ие болуы тиіс түсініктер мен белгілердің бір динағы сипатталынады. Деректер үлгісінің болуы бір тілді пайдалана отырып, нақты іске асырылған әрекеттерді салыстыруға мүмкіндік береді.

Иерархиялық, желілік, реляциялық деректер үлгісі классикалық болып табылады. Бұған қоса соңғы уақытта көп өлшемді және нысанға бағдарланған үлгілер тәжірибе жүзінде белсенді енгізіле бастады.

Танымал үлгілерді кеңейтетін басқа деректер үлгілеріне негізделген алуан түрлі жүйелер әзірленуде. Олардың қатарында нысандық-реляциялық, дедуктивті-нысанға бағдарланған, семантикалық, тұжырымдамалық және бағдарланған үлгілерді атауға болады. Бұл үлгілердің кейбірі дерекқорларды, білім қорларын және бағдарламалау тілдерін біріктіру қызметін атқарады. Жекелеген ДБЖ бір уақытта бірнеше деректер үлгісін қолдайды.

2.2. ТЕОРИЯЛЫҚ-ГРАФТЫҚ ДЕРЕКТЕР ҮЛГІСІ

2.2.1. Иерархиялық үлгі

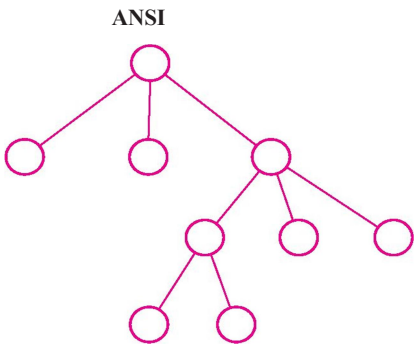
Иерархиялық деректер үлгісі тораптары тігінен «арғы ата-ұрпақ» қатынасымен байланысқан иерархиялық ағаш түрінде дерекқордың пәндік саласының ұсынысын пайдаланады.

Ағаш — циклдары жоқ байланысты бағдарланбаған граф. Ағашпен жұмыс жасаған кезде нақты бір шыңды бөліп көрсетіп, оны ағаштың тамыры ретінде айқындап, оны ерекше қарастырады – бұл шыңға ешбір қабырға кірмейді. Бұл жағдайда ағаш бағдарланғанға айналып, бағдар тамырынан айқындалады.

Түпкілікті шыңдар, яғни ешбір доға шықпайтын шыңдар ағаштың жапырақтары деп аталады. Ағаштың түрлі бұтақтарында түбірінен бастап жапырақтарға дейінгі жолдағы шыңдардың саны түрлі болуы мүмкін. Иерархиялық деректер үлгісінде түбірден бастап жапырақтарға қарай ағаш тәріздес құрылым бағдары пайдаланылады. Дерекқор схемасының графикалық диаграммасы анықтама ағашы деп аталады. Иерархиялық үлгідегі деректер арасындағы байланыстар ұсынысының қарапайым түрі 2.1-суретте көрсетілген.

Иерархиялық құрылым деректер арасындағы тепе-теңсіздікті білдіреді – біреулері басқаларына қатаң бағынады. Шынайы өмірдегі көптеген байланыстар бір нысан аталық болып, онымен көптеген бағыныңқы нысандар байланысты болатын иерархияға сай болады деген негіздеме иерархиялық үлгінің негізі болып табылады. Дерекқордағы навигация бір құрылымда тігінен және көлденең қозғалуды білдіреді. Дерекқорларды басқарудың ең танымал иерархиялық жүйелерінің бірі IBM компаниясының 1968 жылы шыққан InformationManagementSystem (IMS) болды.

«Ағаш» түрі *Сибағдарламалау тілдерінің* «құрылым» және Паскаль тілінің «жазба» түрлеріне ұқсас. Оларда әрқайсысы белгілі бір деңгейде орналасқан түрлердің салымдылығына жол беріледі. «Ағаш» түрі құрамдас болып табылады. Оған өз кезегінде әрқайсысы «ағаш» түрі болып табылатын қосалқы түрлер («қосалқы ағаштар») кіреді. «Ағаш» түрінің әрқайсысы бір «түбірлік» түрден және бағыныңқы түрлердің тәртіптенген жинағынан (мүмкін, бос) тұрады. «Ағаш» түріне енгізілген элементарлы түрдің әрқайсысы «жазбаның» қарапайым немесе құрамдас түрі болып табылады. «Қарапайым» жазба бір, мысалы сандық түрден тұрады, ал құрамдас «жазба» түрлердің жиынтығын, мысалы, тұтас, таңбалар жазбасы және көрсеткіш (сілтеме). Иерархиялық үлгідегі физикалық жазбалар ұзындығы мен құрылымы бойынша ерекшеленеді. Ағаштың әр шыңы пәндік саланың мәніне сәйкес келеді.



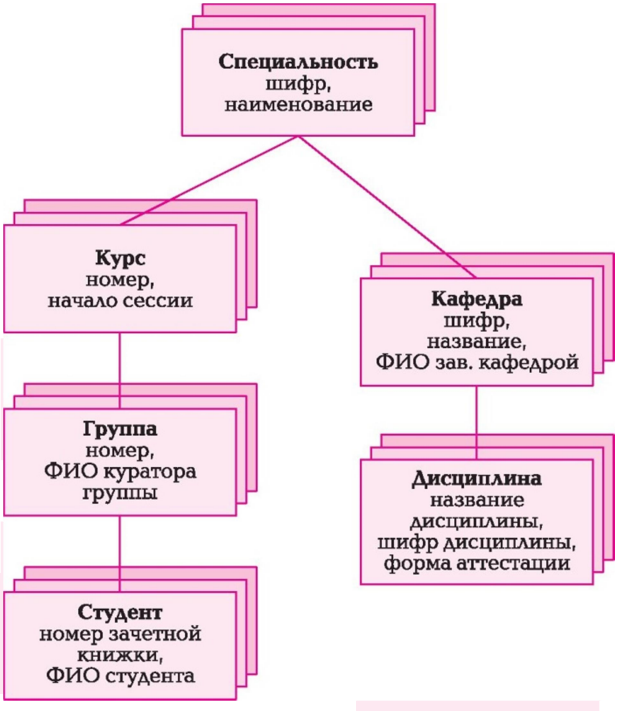
2.1-сурет Иерархиялық үлгідегі байланыстарды ұсыну

Бұл мән атрибуттардың еркін санымен сипатталады.

Бағыныңқы түрлері бар және өзі қосалқы түр болып табылмайтын түр түбірлі болып аталады. Бағыныңқы түр (қосалқы түр) ол үшін арғы ата (ата-ана) болатын түрге қатысты ұрпақ болып табылады. Бір түрден тараған ұрпақ бір-біріне қатысты егіз болады.

Шыңның түрі мәннің түрімен және оның атрибуттарының жинағымен айқындалады. Ағаштың әр шыңында мәндердің даналары – жазбалар сақтаулы. Дерекқордағы тәуелді топтың әр данасына ата-ана жазбалары даналарының бірегей саны – жоғары тұрған деңгейлер шыңдарының әр түрінен бір данадан сәйкес келеді.

Иерархиялық дерекқордың мысалы 2.2-суретте келтірілген.



2.2-сурет Иерархиялық базадағы деректер

Специальность, *шифр, наименование –* ***мамандық,*** *шифр, атауы*

Курс*, номер, начало сессии –* ***курс,*** *нөмірі, сессияның басталуы*

Группа, *номер, ФИО куратора группы –* ***топ,*** *нөмірі, топ тәлімгерінің ТАӘ*

Студент, *номер зачетной книжки, ФИО студента –* ***студент****, есеп кітапшасының нөмірі, студенттің ТАӘ*

Кафедра, *шифр, название, ФИО зав. кафедрой – кафедра, шифр, атауы, кафедра меңгерушісінің ТАӘ*

Дисциплина, *название дисциплины, шифр дисциплины, форма аттестации –****пән****, пәннің аты, пәннің шифрі, аттестация түрі.*

Иерархиялық деректер үлгісінде арнайы навигация тәсілдері де қарастырылған. Ағашпен қозғалу әрқашан түбірлі шыңнан басталады, одан келесі деңгейдің кез келген шыңының нақты данасына ауысуға болады. Бұл шың ағымдағы шың, ал дана - ағымдағы дана (жазба) болады. Бұл жазбадан бұл шыңның басқа жазбасына, ата-ана шыңы жазбасының данасына немесе бағыныңқы шың жазбасының данасына ауысуға болады.

Иерархиялық ДБЖ-да келтірілгеннен бөлек терминология пайдаланылуы мүмкін. Мәселен, IMS жүйесінде «жазба» түсінігіне «сегмент» термині сай келеді, ал «дерекқор жазбасы» ретінде «ағаш» түрінің бір данасына жататын жазбалардың барлығы түсініледі.

ЭЕМ жадында иерархиялық деректерді физикалық орналастыруды ұйымдастыру үшін келесі әдістер тобы пайдаланылуы мүмкін:

- жадының кезекті үлестірілуімен желілік тізіммен ұсыну (мекенжайлық арифметика, сол тізімдік құрылымдар);
- байланысты желілік тізімдермен ұсыну (көрсеткіштер мен анықтамалықтарды пайдаланатын әдістер).

Иерархиялық түрде ұйымдасқан деректерге манипуляция жасаудың негізгі операцияларына келесілер жатады:

- дерекқордың көрсетілген данасын іздеу;
- бір ағаштан екіншісіне көшу;
- ағаштың ішінде бір жазбадан екіншісіне көшу;
- аталған позицияға жаңа жазбаны енгізу;
- ағымдағы жазбаны өшіру және т.б.

«Ағаш» түрінің анықтамасына сәйкес, арғы аталар мен ұрпақтар арасындағы байланыстардың тұтастығын бақылау автоматты түрде сақталынады деген қорытындыға келуге болады. Тұтастықты бақылаудың негізгі ережесі келесідей түсіндіріледі: ұрпақ ата-анасыз өмір сүре алмайды, ал кейбір ата-ананың ұрпағы болмауы да

мүмкін. Түрлі ағаштар жазбаларының арасында байланыстардың тұтастығын сақтау механизмдері жоқ.

Иерархиялық деректер үлгісінің артықшылықтарына ЭЕМ жадын тиімді пайдалану мен деректермен негізгі операцияларды орындау уақытының жақсы көрсеткіштері жатады. Иерархиялық деректер үлгісі иерархиялық жағынан тәртіптенген ақпаратпен жұмыс жасауға қолайлы.

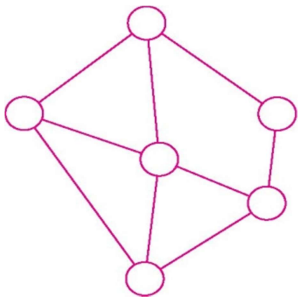
Иерархиялық деректер үлгісінің басты кемшілігі – деректердің қосарлануы. Ол әр мән (атрибут) бір ғана ата-аналық мәнге жатуға болатынымен түсіндіріледі. Мәселен, егер де қызметкердің балалары туралы деректерді сақтау керек болса, ал кәсіпорында әкесі де, анасы да қызмет атқарса, балалар туралы ақпаратты екі мәрте сақтауға тура келеді. Бұл балалар туралы деректерге өзгеріс енгізу кезінде дерекқордың қисынды тұтастығын бұзуы мүмкін. Егер деректер табиғи ағаш тәріздес құрылымға ие болса, иерархиялық деректер үлгісін пайдалану қиындық туғызбайды. Тәжірибе жүзінде болса иерархиялықтан ерекшеленетін деректер құрылымын іске асыру қажет болады. Бұл міндеттерді шешу үшін иерархиялық деректер үлгісіне негізделген нақты ДБЖ еркін ұйымдасқан деректердің ұсынысын жеңілдететін қосымша құралдарды іске қосады.

Иерархиялық үлгінің тағы бір кемшілігі күрделі қисынды байланыстары бар ақпаратты өңдеу үшін тым үлкен болуы мен қатардағы пайдаланушыға түсінудің ауырлығы болып табылады.

2.2.2.

Желілік деректер үлгісінде кез келген нысан бір уақытта басты және бағыныңқы бола алады. Ол басқа нысандармен байланыстардың кез келген санын құруға қатыса алады. Желілік дерекқор жазбалар жинағы мен бұл жазбалар арасындағы байланыстар жинағынан, нақтырақ айтатын болсақ – дерекқор схемасында берілген жазба түрлері жинағынан әр түрден даналардың жинағынан және берілгенбайланыс түрлері жинағынан әртүрден даналар жинағынан тұрады.

Деректерді ұйымдастырудағы желілік тәсіл иерархиялықтың кеңейтілген нұсқасы болып табылады. Иерархиялық құрылымдарда жазба-Ұрпақ бір ғана арғы атаға ие болуы керек; деректердің желілік құрылымында «ұрпақ»-та «арғы ата»-ның кез келген саны болуы мүмкін.



2.3-суретЖелілік үлгідегі байланыстарды ұсыну

Деректердің желілік үлгісі сол арқылы иерархиялық деректер үлгісін жалпылай отырып, деректер бөлшектерінің еркін граф түріндегі алуан түрлі өзара байланысын көрсетуге мүмкіндік береді (2.3-сурет).

Үлгінің негізгі нысандары деректер бөлшегі, деректер агрегаты, жазба, деректер жинағы болып табылады.

Деректер бөлшегі — ДБЖ пайдаланып жатқан пайдаланушыға қолжетімді ең төменгі ақпараттық бірлік. Деректердің әр бөлшегіне оның түрі айқындалуы тиіс (мысалы, таңбалық, сандық, қисынды және т.б.).

Деректер агрегаты — біртұтас дүние ретінде қарастыруға болатын жазба ішіндегі деректер бөлшектерінің атаулы жиынтығы. Агрегат құрамына деректер бөлшектері ғана кіретін қарапайым және деректер бөлшектерімен қоса басқа да агрегаттар кіретін құрамдас болуы мүмкін. Деректер агрегаты ауқым, жазбалар, кешенді сандар және т.б. түріндегі қарапайым немесе иерархиялық құрылыммен ұсынылады. Деректер агрегатының аты бар және жүйеде оны атымен атауға болады. «Вектор» түріндегі агрегат деректер бөлшектерінің желілік жинағына сай келеді. Мысалы, «Мекенжай» агрегаты келесідей ұсынылуы мүмкін: Қала, Көше, Үй, Пәтер. «Қайталанатын топ» түріндегі агрегат деректер векторларының жиынтығына сай келеді. Мысалы, «Еңбекақы» агрегаты 12 рет (ай, еңбекақы сомасы) қайталанатын «қайталанатын топ» түріне сәйкес келеді.

Жазба — деректер бөлшектерінің немесе деректер мен агрегаттар бөлшектерінің шынайы әлем нысандарының белгілі бір сыныбын бейнелейтін атаулы жиынтығы. Жазба – бұл басқа ешбір агрегаттың құрамына кірмейтін агрегат. Ол күрделі иерархиялық құрылымға ие болуы мүмкін, себебі агрегацияның бірнеше мәрте қолдануға жол беріледі. Жазба түрі мен жазба данасын, яғни деректер бөлшектерінің нақты мәндері бар жазбаны бөліп жарады. Бір жазба пәнді саланың бір нысанының – дананың қасиеттерін сипаттайды.

Деректер бөлшектерінің арасында бір немесе бірнеше негізгісі бөліп ажыратылады. Деректердің енгізгі бөлшектерінің мәндері (негізгі атрибуттары) нақты жазба тиесілі болатын нысанды жіктеуге мүмкіндік береді. Бірегей мәнді кілттерді потенциалды деп атайды. Әр кілт деректер агрегаты бола алады. Кілттің бірі бастапқы, қалғандары – қосалқы болып табылады. Бастапқы кілт жазба данасын сәйкестендіреді және оның мәні бір түр жазбалары аясында бірегей болуы тиіс. Кейде «жазба» терминін «топ»

жазбаларының екі немесе бірнеше түрлері арасындағы қатынасты (байланысты) білдіреді. Жинақтың әр түрі үшін жазбаның бір түрі жинақтың иесі болып жарияланып, қалған жазба түрлері жинақ мүшелері болып жарияланады. Әр жинақ данасында иесінің тек бір ғана жазба данасы және иесімен байланысты жинақ мүшелері түріндегі жазбалардың сонша данасы болуы керек. Топтық қатынас үшін де түр мен жананы ажыратады. Мысал ретінде екі жинақты ұйымдастырып, олар арасындағы байланысты анықтайық.

«Сабақ жүргізеді» жинақ даналарының саны оқытушылардың санына, ал «оқиды» жинақ даналарының саны топ санына тең болады.

Желілік дерекқорлардың бірқатар артықшылықтары бар:

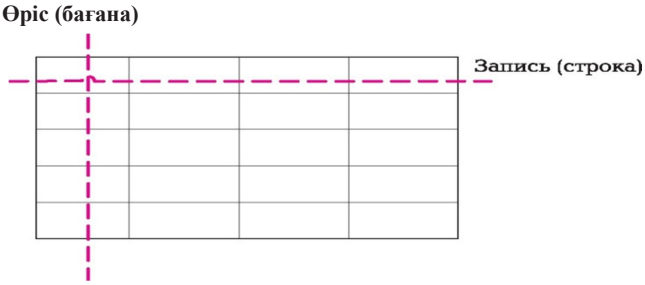
- икемділік. Көптеген арғы ата/ұрпақ қатынастары желілік дерекқорға құрылымы қарапайым иерархиядан күрделі болатын деректерді сақтауға мүмкіндік береді;
- стандарттау;
- жылдам әрекет ету. Аса күрделі болғанына қарамастан, желілік дерекқорлар иерархиялық дерекқордың жылдам әрекет етуімен салыстыруға болатын нәтижелерге жетіп отырды.

Желілік деректер үлгісінің кемшіліктері – оған негізделіп құрылған дерекқор схемасының аса күрделі әрі қатаң болуы, бұл қатардағы пайдаланушыда дерекқорда ақпаратты өңдеуді түсінуде және орындауда қиындық туғызады. Бұған қоса желілік деректер үлгісінде жазбалар арасында еркін байланыстарды орнатуға жол берілетіндіктен, байланыстардың тұтастығына бақылау жасау әлсіз.

2.3. РЕЛЯЦИЯЛЫҚ ҮЛГІ

Реляциялық үлгі теориялық-көптік деректер үлгісіне жатады және қазіргі таңда пайдаланушылар арасында болсын, кәсіби әзірлеушілер арасында болсын ең танымал болып табылады. Реляциялық үлгі дерекқорды анағұрлым икемді етуге деген талпыныстан туындаған. Бұл үлгі деректер байланысын қолдаудың қарапайым әрі тиімді механизмін ұсынды. Дерекқорларды басқарудың реляциялық жүйелері деректерді өңдеудің ең табысты технологияларының бірі болып табылады. Бизнес әлеміндегі деректердің басым бөлігі реляциялық түрде сақтаулы.

Реляциялық үлгі ғана деректердің (ақпараттық нысандардың) кесте түрінде ұсынудың біркелкілігін қамтамасыз етеді. Дерекқордың әр кестесі – жолдар мен бағандардың жиынтығы. Жолдар (жазбалар) нысан данасына, нақты оқиға немесе құбылысқа сай келеді. Бағаналар (өрістер) атрибуттарға, яғни нысан, оқиға



2.4-сурет. Дерекқор кестесінің схемасы

Запись (строка) – жазба (жол)

немесе қандай да бір құбылыстың белгілеріне, сипаттамаларына немесе өлшемдеріне сай келеді (2.4-сурет).

Кестеде бірде бір жол болмауы да, яғни бос болуы да мүмкін. Бір кестенің барлық жолдары бірдей құрылымға ие.

Реляциялық дерекқордағы кестеде кем дегенде бір бағана болуы керек. Әр бағана нақты деректер түріне ие. Дерекқорларда қатардағы бағдарламалау тілдерінде қолданылатынға ұқсас деректер түрлерінің саны пайдаланылады. Кесте өрістерінде тұтас ақпарат болуы тиіс.

Дерекқордың әр кестесінде бастапқы кілт болуы керек.

Бастапқы кілт — нысанның әр данасын немесе жазбаны бір мағынада сәйкестендіретін өріс (өріс жинағы). Дерекқор кестесіндегі бастапқы кілттің мәні бірегей болуы керек, яғни кестеде бастапқы кілттің бірдей мәнімен екі немесе одан көп жазбаға жол берілмейді. Ол барынша жеткілікті болуы керек, яғни жойылуы бірегейлікке ешбір әсерін тигізбейтін өрістері болмауы тиіс.

Бір кестенің көмегімен деректер арасындағы байланыстың қарапайым түрін, атап айтқанда сол туралы ақпарат кестеде сақталынып тұрған бір нысанның (құбылыс, мән, жүйе) әрқайсысына кестенің жолы немесе жазбасы сәйкес келетін көптеген қосалқы нысандарға бөлінуін сипаттау қолайлы. Бұл кезде нысандардың әр өкілі жазба өрістерінің сәйкес мәндерімен сипатталатын бірдей құрылым немесе қасиетке ие болады. Мысалы, кестеде колледж студенттерінің тобы туралы деректер болуы мүмкін, оладың әрқайсысы жайлы келесі сипаттамалар белгілі: тегі, аты, әкесінің аты, жынысы, жасы мен білімі. Пәндік саладағы деректердің анағұрлым күнделі қисынды құрылымдарын бір кесте аясында сипаттау мүмкін болмағандықтан, кестелер байланысын қолданады.

Реляциялық дерекқорларда деректерді сыртқы тасымалдағыштарда физикалық орналастыру қарапайым файлдар арқылы жүзеге асырылады.

Реляциялық деректер үлгісінің артықшылығы қарапайымдылығында, түсініктілігінде және ЭЕМ-де іске асырудың қолайлылығында. Тап осы қарапайымдылығы мен түсініктілігінің арқасында реляциялық үлгі пайдаланушылар арасында кеңінен танымал болды. Бұл түрдегі деректерді өңдеу тиімділігімен туындайтын мәселелердің техникалық жағы да шешімін тапты. Реляциялық үлгінің басты кемшілігі иерархиялық және желілік байланыстарды сипаттаудың қиындығы. Реляциялық ДБЖ-дың соңғы нұсқалары нысанға бағдарланған жүйелердің кей қасиеттеріне ие. Мұндай ДБЖ-ды көп жағдайда нысандық-реляциялық деп те атайды. Ондай жүйенің мысалы ретінде Oracle8.x. өнімдерін алуға болады.

2.4. ПОСТРЕЛЯЦИЯЛЫҚ ДЕРЕКТЕР ҮЛГІСІ

Классикалық реляциялық үлгі кестелер жазбаларының өрістерінде сақтаулы деректердің бөлінбеуін білдіріп, олардың шамадан тыс болуына жол бермейді. Постреляциялық үлгі деректердің бөлінбеуіне шектеулер жоқ кеңейтілген реляциялық үлгі болып табылады. Бұл үлгіде құрамдас (көп мәнді) өрістер болуы мүмкін, яғни өрістің мәні қосалқы мәннен тұруы мүмкін. Көп мәнді өрістер мәнінің жинағы негізгі кестеге енгізілген дербес кесте болып есептелінеді.

Постреляциялық үлгінің артықшылығы – өзара байланысқан реляциялық кестелер жиынтығын бір постреляциялық кестемен ұсынудың мүмкіндігі. Бұл ақпаратты ұсынудың жоғары көрнекілігін және оны өңдеу тиімділігінің артуын қамтамасыз етеді. Постреляциялық үлгінің кемшілігі сақталынатын деректердің тұтастылығы мен қарама-қайшылықсыздығын қамтамасыз ету мәселесін шешудің қиындығы. Бұл мәселе ДБЖ-ға тиісті механизмдерді қосумен шешіледі.

Постреляциялық деректер үлгісін uniVers, Bubbaи Dasdb сияқты дерекқорларды басқару жүйелері қолдайды.

2.5. КӨП ӨЛШЕМДІ ДЕРЕКТЕР ҮЛГІСІ

Ақпараттық жүйелер тұжырымдамаларының дамуында келесі екі бағытты бөліп көрсетуге болады: жедел өңдеу жүйелері мен аналитикалық өңдеу жүйелері (шешім қабылдауды қолдау жүйелері). Реляциялық ДБЖ ақпаратты жедел өңдеудің ақпараттық

жүйесіне арналған және бұл салада ең тиімді болып табылады. Алайда аналитикалық өңдеу жүйелерінде олар икемділігін көрсете алмады. Бұл жерде деректердің көп өлшемді жүйелері анағұрлым тиімді. Көп өлшемді жүйелер талдауды жүргізу мен шешім қабылдау үшін ақпаратты жедел өңдеуге мүмкіндік береді. Реляциялық үлгіге қарағанда деректерді көп өлшемді ұйымдастыруанағұрлым жоғары көрнекілік пен ақпараттылыққа ие.

Көп өлшемді ДБЖ-да деректер көп өлшемді ауқымдар – гиперкубтар түрінде ұйымдасқан. Олар өтінімдердің пайдаланушылардың нақты тобына қажетті деректердің салыстырмалы түрде шағын блоктарына түсуінің есебінен деректерді сұрауына анағұрлым жылдам әрекет етуді қамтамасыз етеді.

Көп өлшемді дерекқорлар технологиясын жаңа технология деп атауға болады. Қорда деректерді ұсынуға деген көп өлшемді тәсіл реляциялық әдіспен қатар пайда болғанына қарамастан, іс жүзінде жұмыс істеп жатқан көп өлшемді ДБЖ бастапқыда аз болатын. 1990-шы жылдардың ортасынан бастап қана оларға көпшілік қызығушылық таныта бастады.

Деректер үлгісінің көп өлшемдігі визуалды ұсынуды емес, деректермен іс-әрекет жасау операцияларында және сипаттау кезінде ақпарат құрылымының қисынды ұйымдасуын білдіреді. Мысалы ретінде 2.5-суретте бір деректердің реляциялық (а) және көп өлшемді (б) ұсынылуы келтірілген.

Көп өлшемді деректер үлгілерінің негізгі түсініктері – өлшеу, өлшем мен кесікті қарастырайық.

Үлгі	Ай	Көлем
Шкаф	Маусым	12
Шкаф	Шілде	24
Шкаф	Тамыз	5
Диван	Маусым	2
Диван	Шілде	18
Үстел	Шілде	19

а

Үлгі	Маусым	Шілде	Тамыз
Шкаф	12	24	5
Диван	2	18	
Үстел		19	

б

2.5-сурет. Деректерді реляциялық және көп өлшемді ұсыну

Өлшеу — гиперкубтың бір қырын құрайтын бір типті деректердің жиынтығы. Басқа сөзбен айтқанда, өлшеулер көп өлшемді координаталар жүйесінің осьтері болып табылады. Ең жиі пайдаланылатын уақыттық өлшеулер – күндер, айлар, тоқсандар мен жылдар. Географиялық өлшеулер ретінде қалалар, аудандар, өңірлер мен елдер жиі қолданылады. Көп өлшемді деректер үлгісінде өлшеулер нақты мәндерді гиперкуб ұяшықтарында сәйкестендіру қызметін атқаратын идекстер рөлін атқарады.

Өлшем (ұяшық, көрсеткіш) — өлшеулердің белгіленген жинағымен айқындалатын өріс. Өрістің түрі көп жағдайда сандық болып айқындалған. Бұл сандық немесе ақшалай көрініс тапқан сатулар көлемі, қоймадағы қалдық, шығындар және т.б. болуы мүмкін.

Үш өлшемді кубтағы өлшем ретінде (2.6-сурет) сатулар сомасы, ал өлшеулер ретінде – уақыт, тауар мен қойма қолданылған. Өлшеулер белгілі бір деңгейлерде топтармен ұсынылған: тауарлар түрі жағынан, ал операциялардың жасалу уақыты туралы деректер – айлар бойынша топтастырылған.

Куб міндетті түрде үш өлшемді болуы тиіс емес. Ол міндетіне қарай екі және көп өлшемді болуы мүмкін.

Алайда кубтың өзі талдау үшін жарамсыз. Үш өлшемді кубтың өзін қызықтырып тұрған өлшемдердің мәндері көрінетіндей етіп компьютер экранында көрсету қиынға соғады. Үштен асатын өлшеуі бар кубтарға не деуге болады? Кубта сақталынып тұрған деректерді визуалдандыру үшін әдетте жолдар мен бағаналардың күрделі иерархиялық атаулары бар әдеттегі екі өлшемді кестелік ұсыныстар қолданылады. Бұл операция кубты «кесу» деп аталады. Талдаушы оны қызықтыратын бағыттар бойынша кубтың өлшеулерін «кеседі». Пайдаланушы шынайы, түсінікті деректер үлгісіне қол жеткізеді. Бұл бизнес үдеріс туралы жинақ немесе толық деректерді алуға және талдау барысында басқа әрекеттерді жүзеге асыруға мүмкіндік береді.

2.

Февраль

Январь

	Склад 1	Склад 2	Склад 3	Склад 4
Мебель	1 000	200	100	200
Аксессуары	500	50	25	50
Ткань	500	50	25	50

Февраль - ақпан мебель - жиһаз ткань - мата склад 1 - 1 қойма

Январь - қаңтар аксессуары - аксессуарлар

	1 қойма	2 қойма	3 қойма	4 қойма
Қаңтар	2 000	300	150	300
Ақпан	3000	1000	200	1000

2.7-сурет. Бір өлшемге арналған кубтың екі өлшемді кесігі

Талдаушы бұл тәсілмен кубтың екі көлемді кесігін алып, сонымен жұмыс істейді. Ағаш кесушілер осылай ағаш кесігіндегі жыл сақиналарын санайды. Сәйкесінше, кесте өлшеулерінің саны бойынша екі өлшеу ғана «кесілмеген» болып қалады.

К е с і к бір немесе бірнеше өлшеуді бекітудің нәтижесінде алынған гиперкубтың қосалқы жиынтығы болып табылады. «Кесіктердің» қалыптасуы пайдаланушы қолданатын мәндерді шектеу үшін орындалады, себебі гиперкубтың барлық мәндері ешқашан бір уақытта пайдаланылмайды. 2.7- суретте «Сан» өлшемі үшін және «Қойма» мен «Ай» екі «кесілмеген» өлшеу үшін кубтың екі өлшемді кесігі суреттелген. 2.8-суретте «Қойма» деген бір ғана «кесілмеген» өлшеу ұсынылған, бірақ бұл жерде «Сан», «Сома» деген екі өлшемнің мәні көрініс табады. Екі өлшеуден артық «кесілмеген» болып қалған жағдайда да куб екі көлемде ұсынылуы мүмкін. Бұл кезде кесік осьтерінде (жолдар мен бағаналарда) «кесілетін» кубтың екі немесе одан көп өлшеулері орналасатын болады (2.9-сурет).

Көп өлшемді деректер үлгісінің кемшілігі – ақпаратты қатардағы жедел өңдеудің қарапайым міндеттері үшін тым ауыр болуы.

Көп өлшемді ДБЖ негізгі артықшылықтары мыналар:

- жүйенің жалпы қарапайымдылығы, бұл көп өлшемді ДБЖ технологияларын қосымшаларға жылдам енгізуге мүмкіндік береді. Көп өлшемді дерекқорға негізделген жүйелер әзірлеу мен әкімшілік жүргізуде арнайы дағдыларды көп қажет етпейді;
- құнының салыстырмалы алғанда төмен болуы, сондай-ақ инвестицияларды тез қайтарып алу;

	1 қойма	2 қойма	3 қойма	4 қойма
Саны	2000	100	50	300
Сомасы	3000	1000	200	1000

2.8-сурет. Бірнеше өлшемге арналған кубтың екі өлшемді кесігі

	Қаңтар				Ақпан			
	1 қойма	2 қойма	3 қойма	4 қойма	1 қойма	2 қойма	3 қойма	4 қойма
Саны	200	100	50	300	200	100	50	300
Сомасы	3000	1000	200	1000	3000	1000	200	1000

2.9-сурет. Бірнеше өлшеулері бар кубтың бір осьтегі екі өлшемді кесігі

- көп өлшемді ДБЖ пайдаланған жағдайда деректерді іздеу және іріктеу реляциялық дерекқорға көп өлшемді тұжырымдамалық көзқарасқа қарағанда айтарлықтай жылдам жүзеге асырылады, себебі көп өлшемді дерекқор сұралып отырған ұяшықтарға оңтайлы қолжетімділікті қамтамасыз етеді;
- көп өлшемді ДБЖ ақпараттық үлгіге алуан түрлі енгізілмелі қызметтерді қосу міндеттерін оңай атқарады, алSQL тілінің объективті түрдегі шектеулері бұл міндеттерді реляциялық ДБЖ негізінде орындауды айтарлықтай күрделі, ал кей кезде орындалмайтын етеді.

Деректерді көп өлшемді құрылымдарда сақтау кезінде бос мәндерді сақтаудың салдарынан «толып кету» мәселесі туындауы мүмкін (2.5-суретті қараңыз). Егер көп өлшемді аумақта өлшеу таңбаларының барлық мүмкін комбинацияларына орын сақтаулы болып, шынайы келгенде тек аз бөлігі ғана толықтырылса (мысалы, бірқатар тауар санаулы өңірде ғана сатылса), кубтың басым бөлігінде орын алынғанымен, ол бос болып тұрады. Қазіргі жүйелер бұл мәселені шеше алады.

Көп өлшемді деректер үлгісін қолдайтын жүйелер Essbase (ArborSoftware), MediaMulti-matrix (Speedware), OracleExpressServer (Oracle), Cache (InterSystems). Кейбір бағдарламалық өнімдер, мысалы, Media/MR (Speed- ware), бір уақытта көп өлшемді және реляциялық дерекқормен жұмыс жасауға мүмкіндік береді. Ішкі деректер үлгісі көп өлшемді үлгі болып табылатын Cache ДБЖ-да деректерге қол жеткізудің үш тәсілі іске асырылған: тікелей (көп өлшемді ауқымдар тораптары деңгейінде), объектілік және реляциялық.

2.6. НЫСАНҒА БАҒДАРЛАНҒАН ҮЛГІ

Дерекқорларды басқарудың нысанға бағдарланған жүйелері – дерекқорлардың мүмкіндіктері мен ерекшеліктерін және бағдарламалаудың нысанға бағдарламаланған тілдерінің мүмкіндіктерін қиыстыруды нәтижесі. Нысанға бағдарланған ДБЖ

нысанға бағдарланған бағдарламалау тілдеріндегіндей нысандармен жұмыс жасауға мүмкіндік береді. Олар бағдарламалау тілдерін кеңейтіп, ұзақ уақытты деректерді, деректермен қатар жұмыс жасауды басқаруды, деректерді қалпына келтіруді, қауымдастырылған сұрауларды және басқа да мүмкіндіктердіенгізеді.

Нысанға бағдарланған тілдердегі бағдарламашылардың компьютердің жедел жадына сыймаған нысандарды сақтау құралдарына деген қажеттілігі нысанға бағдарланған ДБЖ-дың шығуына себепші болды. Сонымен қатар қолданбалы бағдарламаны қайталап іске қосу арасында нысандардың күйін сақтау міндеті маңызды болды. Сондықтан дерекқорды басқарудың нысанға бағдарланған жүйелерінің басым көпшілігі кітапхана түрінде көрініс табады, оның деректерін басқару рәсімдері қолданбалы бағдарламаға енгізіледі.

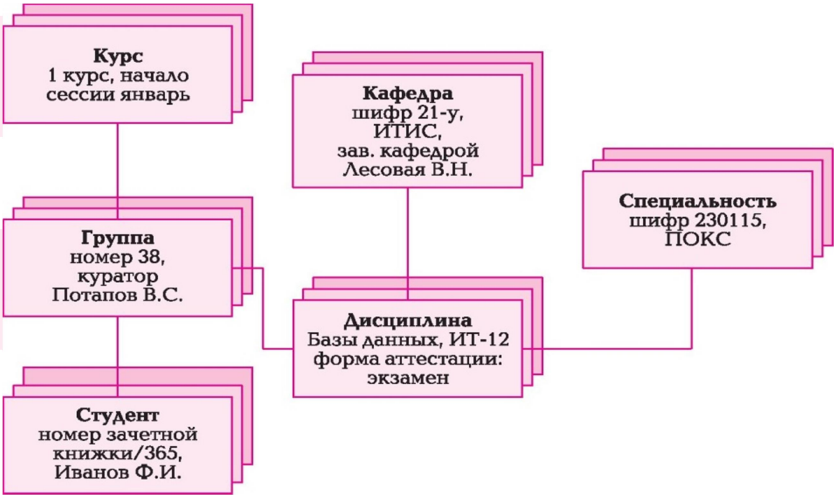
Деректерді ұсыну кезінде нысанға бағдарланған үлгіде қордың жекелеген жазбаларын сәйкестендіру мүмкіндігі бар. Дерекқордың жазбалары мен оларды өңдеу қызметтері арасында бағдарламалаудың нысанға бағдарланған тілдеріндегі құралдарға сай келетін механизмдердің көмегімен өзара байланыс орнатылады.

Нысанға бағдарланған дерекқордың құрылымы тораптары нысандар болып табылатын ағаш түрінде көрсетілген. Нысандар қасиеттері қандай да бір стандартты түрде (мысалы, жолдық,string) немесе пайдаланушымен құрылатын түрмен (class ретінде айқандалады) сипатталады.

Class түріндегі қасиеттің мәні тиісті сыныптың данасы болып табылатын нысанның өзі. Сыныптың әр нысан-данасы қасиет болып анықталған нысанның ұрпағы болып табылады. Сыныптың нысан-данасы өз сыныбына жатады және бір ата-анасы бар. Дерекқордағы туыстық қатынастар нысандардың өзара байланысқан иерархиясын құрайды.

Нысанға бағдарланған дерекқордың қисынды құрылымына мысал 2.10-суретте көрсетілген.

Нысанға бағдарланған дерекқордың қисынды құрылымы иерархиялық дерекқордың құрылымына ұқсас. Олардың басты айырмашылығы деректермен әрекет ету әдістерінде болып отыр. Қарастырылып жатқан дерекқор үлгісінде деректермен әрекет жасау үшін нысанға бағдарланған механизмдермен күшейтілген қисынды операциялар қолданылады.



2.10-сурет Нысанға бағдарланған дерекқордың қисынды құрылымы

***Курс**, 1 курс, начало сессии январь – Курс, 1 курс, сессияның басталуы қаңтар*

***Группа**, номер 38, куратор Потапов В.С. – Топ, нөмір 38, кураторы В.С. Потапов*

***Студент**, номер зачетной книжки/365, Иванов Ф.И. – студент, сынақ кітапшасының нөмірі/365, Ф.И. Иванов*

Нысанға бағдарланған деп аталу үшін ДБЖ келесі сипаттамаларға ие болуы керек:

- күрделі нысандарды қолдау. Жүйеде құрамдас нысандарды құрастырушыны қолданудың есебінен құрамдас нысандарды құру мүмкіндігі қарастырылуы тиіс;
- нысандардың бірегейлігін сақтау. Барлық нысандарда атрибуттарының мәніне тәуелсіз болатын бірегей сәйкестендіргіш болуы керек;
- қапшықтануды қолдау. Бағдарламашылар әдістер интерфейсінің спецификациясына ғана кіру құқығына ие болуының, ал деректер мен әдістерді іске асыру нысандардың ішінде жасырынғанның есебінен дұрыс қапшықтануға қол жеткізіледі;
- түрлер мен сыныптарды қолдау;
- олардың арғы аталарынан түрлер мен сыныптарды мұра етуді қолдау. Қосалқы түр (қосалқы сынып) сәйкесінше оның супер-

түрінен (суперсыныбынан) атрибуттар мен әдістерді мұра етуі керек;

- деректермен әрекет жасау тілі жалпыға ортақ бағдарламалау тілі болуы керек;
- деректер түрлерінің жинағы кеңейтілмелі болуы керек. Пайдаланушы алдын ала айқындалған жүйелік түрлер жинағының негізінде деректердің жаңа түрлерін жасау құралдарына ие болуы керек. Бұған қоса деректердің жүйелік және пайдаланушылық түрлерін пайдалану тәселдерінің арасында ешқандай айырмашылық болмауы керек.

Деректердің нысанға бағдарланған үлгісінің реляциялықпен салыстырғандағы басты артықшылығы - нысандардың күрделі өзара байланыстары туралы ақпаратты көрсетудің мүмкіндігі болып табылады.Деректердің нысанға бағдарланған үлгісі дерекқордың жеке жазбасын сәйкестендіріп, оларды өңдеу қызметтерін айқындай алады. Нысанға бағдарланған үлгінің кемшілігі – түсінуге қиын болуы мен өңдеудің қолайсыздығы.

Нысандық технологиялардың өскелең танымалдылығына деген реляциялық ДБЖ өндірушілердің жауабы әмбебап серверлер деп те аталатын нысандық реляциялық дерекқорларды шығару болды. Нысандық-реляциялық ДБЖ – нысандар, сыныптап мен мұра ету дерекқорлар құрылымында және сұраулар тілінде іске асырылған нысанға бағдарланған тәсілді іске асыратын кейбір технологияларды қолдайтын реляциялық ДБЖ. Кеңінен танымал OracleDatabase, Informix, DB2 нысандық-реляциялық ДБЖ болып табылады.

Нысандық-реляциялық үлгінің негізгі артықшылықтары құрамдас бөлшектерін қайталама және бірлесіп пайдаланудың мүмкіндігі болып табылады.

Нысандық-реляциялық ДБЖ пайдаланатын тәсілдің айқын кемшіліктері – күрделілігі мен онымен байланысты жоғары шығындар. Реляциялық үлгіге сай келетін қарапайымдылық пен анықтық кеңейтудің осындай түрлерін пайдаланған кезде керексіз болып қалады.

БАҚЫЛАУ СҰРАҚТАРЫ

1. Деректер үлгісі деген не?
2. Деректер үлгісі не үшін құрылады?
3. Деректер үлгісіне қандай аспектілер кіреді?
4. Деректерді ұсынудың классикалық және заманауи үлгілірен атап шығыңыз.
5. Иерархиялық деректер үлгісінің артықшылықтары мен кемшіліктерін атап шығыңыз.
6. Иерархиялық дерекқорда деректерді физикалық орналастыру қалай ұйымдастырылады?
7. Желілік деректер үлгісіне сипаттама беріңіз.
8. Реляциялық деректер үлгісіне сипаттама беріңіз.
9. Постреляциялық деректер үлгісінің ерекшелігі неде?.
10. Көп өлшемді деректер үлгісі қайда қолданылады?
11. Көп өлшемді үлгінің артықшылықтары мен кемшіліктерін атаңыз.
12. Көп өлшемді деректер үлгісіне сипаттама беріңіз.
13. Көп өлшемді кестелерге мысал келтіріңіз.
14. Көп өлшемді үлгіде деректермен орындалатын операциялардың мәнін атап, түсіндірінңіз.
15. Нысанған бағдарланған деп атану үшін ДБЖ қандай сипаттамаларға ие болуы тиіс?
16. Деректерді ұсынудың нысанға бағдарланған үлгісінің артықшылықтары мен кемшіліктерін атаңыз.
17. Көп өлшемді ДБЖ-да пайдаланатын түсініктерін мағынасын түсіндіріп беріңіз: өлшеу, ұяшық, кесік.

РЕЛЯЦИЯЛЫҚ ДЕРЕКТЕР ҮЛГІСІ

3.1. РЕЛЯЦИЯЛЫҚ ДЕРЕКТЕР ҮЛГІСІНІҢ ЕРЕКШЕЛІКТЕРІ

3.1.1. Негізгі түсініктері мен компоненттері

Реляциялық деректер үлгісінің теориялық негізі қатынастар теориясы болды. Көптеген қатынас кейбір арнайы операцияларға қатысты жабық, яғни сол операциялармен бірге дерексіз алгебраны құрайды. Қатынастардың бұл маңызды қасиеті реляциялық үлгіде бастапқы алгебрамен байланысты деректермен іс-әрекет жасау тіін әзірлеу үшін пайдаланылды. Американдық математик Э.Ф. Кодд 1970 жылы алғаш рет реляциялық үлгінің негізгі түсініктері мен шектеулерін тұжырымдаған болатын. Кодд ұсыныстарының дерекқор жүйелері үшін тиімді болғаншы соншалық, бұл үлгі үшін есептегіш техниканың теориялық негіздері саласындағы Тьюрингтің беделді сыйлығына ие болды.

Реляциялық дерекқордың негізгі түсініктері мыналар:

- қатынас;
- деректер түрі;
- домен;
- атрибут;
- кортеж;
- бастапқы кілт.

Бұл түсініктердің мәнін колледж студенттері жайлы ақпарат бар «Студенттер» қатынасын мысалға ала отырып көрсетеміз (3.1-сурет).

Қатынас — реляциялық дерекқордың іргелі түсінігі. Сол себепті де үлгі реляциялық деп аталады (лат. *relation*– қатынас, байланыс). Физикалық деңгейде қатынастар ұяшықтарында деректер сақталынатын жолдар мен бағаналар түріндегі екі өлшемді кесте түрінде болады. Әр кестеде бір түрлі нысандар туралы ақпарат болады, ал барлық кестелердің жиынтығы бірыңғай дерекқорды құрайды.

Қатынастың атауы

Атрибуттар

Код	ТАӘ	Топ	Мамандық	Жынысы
1	Иванов Ф.И.	35	Туризм	е
2	Дремина Е.Е.	35	Туризм	ә
3	Латыпова Ж.А.	44	Жарнама	ә
4	Лесовая В.Н.	44	Жарнама	ә
5	Попенко Б. С.	35	Туризм	е
6	Потапов В.С.	35	Туризм	е
7	Федорова Д.С.	35	Туризм	ә
8	Таран О.С.	44	Жарнама	е
9	Дудко О.В.	35	Туризм	ә

«Студенттер» қатынасы*

р Кортеждер

\ Бастапқы кілт

3.1-сурет. Реляциялық тәсілдің негізгі түсініктерінің ара салмағы

Кесте жолы *жазба*, ал кесте бағанасы – *өріс* деп аталады.

Әр өрістің кесте шегінде бірегей аты болуы керек. Кестелерді ұйымдастыру ерекшеліктері дерекқорды құру мен жүргізу үшін пайдаланылатын нақты ДБЖ-ға байланысты болады.

Деректер дерекқорда бірнеше түрлі типтердің бірі түрінде сақталынған жиынтық ақпарат болып табылады. Деректер түрлерінің көмегімен кестенің нақты бағанасындағы деректер үшін негізгі ережелер, соның ішінде олар үшін бөлінетін жадының көлемі де белгіленеді.

Реляциялық үлгідегі «деректер түрі» түсінігі бағдарламалаудағы «деректер түрі» түсінігіне дәлме-дәл келеді. Реляциялық дерекқорда сақталынатын деректердің мәндері типтелген болып табылады, яғни әр сақталынған мәннің түрі белгілі. Әр бағанада өзінің белгілі бір деректер түрі жазылады. Әдетте қазіргі реляциялық дерекқорларда таңбалық, сандық деректерді, мамандандырылған сандық деректерді (мысалы, «ақшалай»), сонымен бірге арнайы деректерді (күні, уақыты , уақыт аралығы) сақтауға болады. Бұған қоса реляциялық жүйелерде пайдаланушыларға өз дерек түрлерін айқындау мүмкіндігі беріледі.

Домен түсінігі бағдарламалаудың кейбір тілдерінде қосалқы түрлерімен аналогтары болғанына қарамастан, дерекқор үшін

анағұрлым тән болып келеді. Кей авторлар домен мен деректер түрі түсініктерін ұқсастырады. Дегенмен кейбір реляциялық ДБЖ-да домен түсінігі мүлдем пайдаланылмайды.

Жалпы алғанда *домен* оның бөлшектері жататын деректердің түрін белгілеу және олардың шектеулерін сипаттау арқылы айқындалады. Әр домен тиісті дерекқордың барлық домендері аттарының арасында бірегей атпен байланысады. Доменді бұл түр мәндерінің ұйғарынды шектеулі қосалқы жиынтығы ретінде айқындауға болады. Мысалы, біздің мысалдағы ТАӘ домені таңбалық жолдардың базалық түрінде айқындалған, алайда оның мәндерінің қатарына орыс әліпбиінің таңбалары ғана кіріп, сандар мен т.б. кіре алмайды. ЖЫНЫС домені де таңбалық болып табылады, бірақ ұзындығы (тек 1 таңба) мен ұйғарынды мәндердің көптігі бойынша («е» және «ә» таңбалары ғана) шектеулері бар.

Домен түсінігінің семантикалық бөлігін де атап кету керек: деректер бір доменге жататын болса ғана, салыстырылатын болып есептелінеді. Мысалы, ТАӘ мен МАМАНДЫҚ домендерін салыстырудың еш қажеті жоқ. Олар таңбалық түрге жататындығына қарамастан, олардың мәндерін салыстыруға келмейді.

А т р и б у т — сол туралы ақпарат дерекқорда сақталынатын пәндік сала нысанының қасиеті (сипаттамасы). Атрибут қандай да бір доменге тиесілі болуы керек атаумен және мәнмен сипатталынады. Нысанның әр данасы әр уақытта атрибуттардың нақты мәндерінің жинағымен сипатталады (3.1-суретін қараңыз).

Қ а т ы н а с с х е м а с ы (қатынас атауы) — «атрибуттың аты, доменнің аты» атаулы жұптардың жиынтығы. Қатынас схемасының дәрежесі – бұл жиынтықтың қуаты, яғни атрибуттардың саны. Қатынастар схемасын кесте бағаналары атауларының жолы ретінде келтіруге болады (3.1-суретті қараңыз).

Қатынастың бұл схемасына сай келетін к о р т е ж – «атрибуттың атауы, мәні» жұптарының жиынтығы, ол қатынас схемасына тиесілі атрибуттың әр атауының бір кіруінен тұрады. Осылайша, кортежді кестенің жолы ретінде көрсетуге болады (3.1-суретін қараңыз). Онда қатынасты бір қатынас схемасына сәйкес келетін кортеждердің жиынтығы ретінде ұсынуға болады.

Кортеждердің еркін жиынтығы қатынастың денесі деп аталады.

Б а с т а п қ ы к і л т (қатынас кілті) — мәндері қатынас кортежін айқындайтын атрибуттардың ең аз жинағы. Әр қатынас үшін оның атрибуттарының толық жинағы бірегей болып табылады. Алайда бастапқы кілтті формальды айқындаған кезде «минималдылық» қажет, яғни бастапқы кілт атрибуттарының жинағына кортежді бір мағыналы айқындау деген негізгі қасиет үшін ешбір әсерсіз алып тастауға болатын атрибуттар кірмеуі керек. Бұны «Студенттер»

қатынасы мысалынан көруге болады (3.1-суретін қараңыз), онда әр кортеж бірегей болғанымен, атрибуттардың барлық жинағынан әр студентті бір мағыналы сәйкестендіретін «Код» дегенін ғана таңдауға болады (әр кортеж немесе жазба).

Кесте үшін кілтті айқындау жазбалардың автоматты сұрыпталуын, жазбалардың негізгі өрістерінде мәндердің қайталануын болдырмауды бақылау мен кестеде іздеу операцияларын орындау жылдамдығын арттыруды білдіреді. Негізгі атрибут белгілі бір мағынаға ие болуы мүмкін. Алайда көп жағдайда негізгі өрістің ешбір мағыналық мәні болмай, кестедегі нысанның сәйкестендіргіші ғана болып табылады. Көбінесе кестелерде бастапқы кілттерді анықтау үшін мәндерді автоматты түрде жинақтауы бар өрістер пайдаланылады («есептегіштер», autoincrementfield). Бұл кезде негізгі өрістің бірегейлігін қолдау бойынша барлық жауапкершілік пайдаланушыдан алынып, дерекқор процессорына жүктеледі. Есептегіш өрістері төрт байтты тұтас сан болып табылады және пайдаланушы кестеге жаңа жазбаны қосқан кезде автоматты түрде бір бірлікке көбейеді.

Сонымен, реляциялық деректер үлгісі негізінде өз бөлшектердің тізімімен және олардың мәндерін атап өтумен белгіленетін қатынас түсінігі жатыр. Қатынас кестемен ұсынылуы мүмкін. Кесте бағаналарының атаулары атрибуттар атауына ие. Олардың аттарының тізімі қатынас схемасының атауына ие. Әр атрибут ол ұсынып отырған деректер түрін айқындайды, ол болса мәндерінің облысымен домен деп аталады. Толықтай кесте қатынас, ал кестенің әр жолы – қатынас кортежі деп аталады. Реляциялық деректер үлгісінің терминологиясы бір қарағанда үйреншікті емес және көбінесе қарапайым кестелердің терминдері пайдаланылады («кесте бағаналары» туралы айтады да, «қатынас атрибуттарын» меңзейді). Біз реляциялық дерекқорларды ұйымдастырудың тәжірибелік мәселелерін және басқару құралдарын қарастыруға көшкен кезде, біз үшін осы үйреншікті терминологияны пайдаланатын боламыз. Бұл терминология коммерциялық реляциялық ДБЖ-дың көбісінде қолданылады.

Қатынас түсінігімен байланысты бірқатар математикалық анықтамаларды енгізейік.

*R,*N-арлық қатынасы немесе *n,*дәрежесінің *Rқатынасы деп* D_1,D_2, ..., D_n(n> 1) жиынтықтарының декарт көбейтіндісінің қосалқы жиынтығын атайды. D_1,D_2, ..., D_n *бастапқы жиынтықтарын үлгіде домен деп атайды.*

Декарт (тура) көбейтіндісі — бөлшектері бастапқы екі жиынтық бөлшектерінің барлық мүмкін тәртіптелген жұптары болып табылатын жиынтық.

$A_1(D_1)$, $A_2(D_2)$, ..., $A_n(D_n)$ — атрибуттардыңатаулары. *Rқатынасының гсхемасы* деп *г* = ={Aj(Dj), A2P2)An(Dn)}атрибуттарын аттарының түпкілікті жиынтығы деп аталады.

D_1x D_2x ... x D_n *декарт көбейтіндісінің Rжиынтығы* D_1, D_2, ..., D_n *түпкілікті жиынтықтарындағы гсхемасы бар қатынас деп аталады.*

Жоғарыда айтылып кеткендей, (d_1, d_2, ..., d_n) қатынасының бөлшектері кортеждер деп аталады. Кортеж (d_1, d_2, ..., d_n)*пқұрамдас бөлшегі бар.*Кортежді белгілеудің қысқартылған түрі де қолданылады d_1, d_2, ..., d_n.*Реляциялық деректер үлгісінде қатынасты айқындау үшін декарт көбейтіндісі түсінігін пайдалану үлгіні конструктивті етеді. Математикалық тұрғыдан бұл үлгінің барлық басқа түсініктері декарт көбейтіндісі негізіндегі қатаң математикалық құрудың шегінде айқындалатынын білдіреді.*

3.1.2.

Бұған дейін келтірілген анықтамалардан келіп шығатын қатынастардың кейбір маңызды қасиеттеріне тоқтала кетейік.

Кортеж-дубликаттардың жоқтығы. Қатынасты көптеген кортеж ретінде анықтаудан келіп шығады. Бұл қасиеттен әр қатынастың бастапқы кілтінің болуы келіп шығады. Қатынастарда кортеждерді олардың құрамдас бөлшектерінің мәні бойынша ғана ажыратуға болады. Бұл да үлгі үшін аса маңызды жайт: ешбір екі кортеж толықтай бір-біріне сай келетін құрамдас бөлшектерге ие бола алмайды. Осылайша, реляциялық үлгіде пәндік сала нысандары туралы деректердің қосарлануына жол жоқ.

Бастапқы кілт түсінігі дерекқорлардың тұтастығы түсінігімен байланысты да аса маңызды болып табылады. Көптеген реляциялық ДБЖ тәжірибе жүзінде іске асырылғанда сұрауларды орындаған кезде нақты орындалмайтын аралық қатынастар үшін кортеждер бірегейлігінің қасиетін бұзуға жол берілетінін атап өту керек. Мұндай қатынастар жиынтық емес, мультижиынтық болып табылады.

Атрибуттарда тәртіптің жоқтығы. Қатынас кортежінде атрибуттың мәніне сілтеме жасау үшін әрқашан атрибуттың аты

пайдаланылады. Қатынастардың атауларын айқындау бойынша көптеген жұптар бар болғандықтан {атрибуттың аты, доменнің аты}, қатынастар атрибуттары бір тәртіпке салынбаған. ДБЖ кортеж атрибуттарының мәндетін қандай тәртіпте сақтау керек екенін өзі шешеді (әдетте әр қатынастың барлық кортеждері үшін бір физикалық тәртіп сақталынады). Бұл қасиет қолданыстағы қатынастар схемаларын жаңа атрибуттарды қосу арқылы ғана емес, қолданылып жатқандарын жою арқылы да түрлендіру операциясын орындауды жеңілдетеді.

Атрибуттар мәндерінің атомдығы. Бұл қасиет доменді деректер простоготипі мәндерінің потенциалды жиынтығы ретінде анықтаудан келіп шығады және атрибут жиынтық бола алмайды дегенді білдіреді. Мысалы, «Иванов И. И., Ленина көшесі, 8» дұрыс емес, себебі көптеген сипаттамалар аталған (ТАӘ, көше, үй).

Кортеждерде тәртіптің жоқтығы. Бұл қасиет қатынас денесін кортеждердің жиынтығы ретінде айқындаудың нәтижесі болывп табылады. Алайда қатынас кортеждерінің жиынтығында тәртіпті сақтауға деген талаптың жоқтығы ДБЖ-ға деректерқорларды сыртқы жадыда сақтау кезінде және дерекқорға сұрау салынған кезде қосымша икемділік береді.

Қатынастар қасиеттерін қолданудың нақты мысалдары келесі тақырыптарда реляциялық дерекқорларды жобалау және қалыптандыру мәселелеріне тоқталған кезде қарастырылатын болады.

3.2. РЕЛЯЦИЯЛЫҚ АЛГЕБРА НЕГІЗДЕРІ

Реляциялық алгебра (Коддың реляциялық алгебрасы) – реляциялық деректер үлгісіндегі байланыстың тұйықталған жүйелік амалы. Реляциялық алгебраның амалын сонымен қатар, реляциялық амал депте атайды. Реляциялық алгебраның негізгі ойы мынада жатыр, егер байланыс көпшілікті болса, онда байланыспен әрекет ету құралы дерекқор үшін айрықша, кейбір арнайы амалдармен толықтырылған, дәстүрлі теоретикалық-көпшілік амалына негізделе алады. Реляциялық алгебра байланыстарға қатысты келесідей

Коддтың реляциялық алгебра амалын екі топқа бөліп қарастыруға болады:

1) базалық теоретикалы-көпшілікті;
2) арнайы реляциялық.

Амалдың бірінші тобына көпшілік теориясының классикалық амалы жатады. Екінші тобына деректерді тартудың шынайы міндеттеріне бағытталған қарапайым теоретикалы-көпшілікті амалының дамуы жатады. Теоретикалық-көпшілік жүйе құрамына келесідей жүйелер кіреді:

- байланысты біріктіру;
- байланыстың қиылысуы;
- байланысты кеміту;
- байланысты тікелей көбейту.

Арнайы реляциялық амалдар келесіні қамтиды:

- байланысты шектеу;
- байланысты проекциялау;
- байланысты бөлу;
- байланысты біріктіру.

Сонымен қатар, алгебра құрамына дерекқор көзінде алгебралық өрнектерді есептеулердің нәтижесін және нәтижелі байланыстың атауын (схемасын) нақты қалыптастыруға мүмкіндік беретін атрибуттарды өзгерту амалын сақтап қалуға көмектесетін иемдену амалы кіреді.

Реляциялық алгебра амалы бір (бір реттік амал) байланысымен немесе (мысалы, проекциялау) немесе екі (екілік амал) байланыспен (мысалы біріктіру) орындалуы мүмкін. Екілік амалды орындау кезінде байланыс амалына қатысушылар құрылымы жағынан сәйкес келуі тиіс.

Кейбір байланыстар ресми түрде атрибуттарының атауы әртүрлі болғаны үшін сәйкес болып келмейді, бірақ атрибуттарды өзгерту амалын қолданғаннан кейін сәйкес болып шығады.

N-дік *f* реляциялық амалды байланысты қайтарушы және *n* байланысы аргумент ретінде болатын қызмет ретінде елестетуге болады: *R = f(Ri, R2Rn).*

Реляциялық алгебра реляциялық амалдың операнты ретінде тұйықталған болып келетіндіктен реляциялық алгебраның басқа өрнектерін алмастыруға болады (түрі жағынан сәйкес келетін):

$$R = f[(fi(Rii, R12, ...), f2(R21, R22, ...), ...].$$

Байланыстың құрылымының сай келуі атрибуттар атауының және сәйкес келетін домендер түрінің үйлесімділігін білдіреді.

Екі үйлесімді бірдей өлшемдегі байланыстың R_j және R_2 *(Rj UNION R2) бірігуі* R байланысы болып табылады, қайталануын есепке алмағандағы алғашқы байланыстардың барлық элементтерін қамтыған, яғни, алынған байланыс операнд байланыстардың жоқ дегенде біреуіне кіретін, барлық кортеждерді есепке алады.

Мысалы, R_j байланысы 35 тобындағы көп студенттер болсын, ал R_2 байланысы – жасөспірімдердің көп студенттері болсын. Сол кезде *R* байланысы 35 тобының студенттерін немесе жасөспірім-студенттерді, не болмаса оныда басқасында есепке алатын көпшілікті білдіреді (3.2 сурет).

Екі үйлескен бірдей өлшемдегі R_j және R_2 (R_j*INTERSECT* R_2) байланыстардың қиылысуы алғашқы екі байланысқада кіретін кортеждерді ала отырып, R байланысын тудырады. Алдыңғы мысалдағы R_j және R_2 байланысы үшін (3.2 сурет) *R* нәтижелі байланысы ер жыныстыы 35 тобының барлық студенттерін көрсететін болады (3.3 сурет).

Бірдей өлшемдегі үйлесімді R_j және R_2 (R_j*MINUS*R_2) байланысын *кемітуі* бірнеше кортеждерден тұратын, R_j жататын, бірақ R_2 байланысына кірмейтін R_j және R_2 (R_j*MINUS*R_2) байланысын *кемітуі*. R_j және R_2 байланыстары үшін байланыстың алдыңғы мысалындағы (3.2 сурет) R_j 35 топтағы әйелдердің

К) қатынасы

ТАӘ	Топ	Жынысы
Иванов Ф.И.	35	е
Кириллова Е.Е.	35	ә
Потапов В.С.	35	е
Дудко О.В.	35	ә

***R*-қатынасы**

ТАӘ	Топ	Жынысы
Иванов Ф.И.	35	е
Потапов В.С.	35	е
Таран О.С.	44	е
Ильин Г.С.	44	е

***R* UNION R_2қатынасы)**

ТАӘ	Топ	Жынысы
Иванов Ф.И.	35	е
Кириллова Е.Е.	35	ә
Потапов В.С.	35	е
Дудко О.В.	35	ә
Таран О.С.	44	е
Ильин Г.С.	44	е

***R*(Jİ, INTERSECT R_2)байланысы**

ТАӘ	Топ	Жынысы
Иванов Ф.И.	35	е
Потапов В.С.	35	е

3.3. суы

(R_t MINUS R_2)байланысы

ТАӘ	Топ	Жынысы
Кириллова Е.Е.	35	ә
Дудко О.В.	35	ә

3.4. сурет. Байланысты кеміту

Атрибуттардың бірдей атауына ие емес, a_1 деңгейіндегі R_1 байланысының және a_2 деңгейіндегі R_2 байланысының(R_j*TIMES* R_2) *тікелей көбеюі* – бұл (a_1 + a_2) деңгейдегі R байланысының атауы R_j және R_2 байланыс атауының ілінісуін көрсетеді, ал дене R_j және R_2 байланыс кортеженің барлық мүмкін бірігулерін ұсынады, кортеждердің алғашқы a_1 элементтері R_1 көпшілігіне жатады, ал соңғы a_2көпшілігі - R_2 көпшілігіне жатады. Қажет болған жағдайда бір немесе бірнеше атрибуттардың бірдей атауларына ие екі байланыстың көбеюін алу үшін өзгерту амалы қолданылады. Байланысты көбейту мысалы 3.5 суретте көрсетілген.

Айта кететін жайт көбейтінді нәтижесін алу амалы тәжірибеде кең таралмаған болып келеді. Біріншіден, оның нәтиже беруші байланысының мықтылығы ауқымды болып келеді, тіпті егерде алғашқылы мықты байланыс болсада, екіншіден амалдың нәтижесі алғашқы байланыстың жалпылама алынғандығына қарағанда ақпаратты емес. Ары қарай көрсетілгендей Коддтың реляциялық алгебра құрамына көбейту амалының қосылуының негізгі мағынасы – оның негізінде байланыстың шыныменде пайдалы амалы анықталады.

Реляциялық алгебраның бірінші арнайы амалы көлденең таңдау, не болмаса іріктеу амалы , не болмаса байланысты шектеу амалы.

Шектеу амалы (таңдау) WHERE екі операндтың болуын талап

***Ri*байланысы**

ТАӘ	Топ	Жыныcы
Иванов Ф.И.	35	е
Таран О.С.	44	е

***R*$_2$байланысы**

Топ	Куратор	Курс
35	Кирсанова Л.Н.	3
44	Никитина Л.П.	4

***R* (*R*$_1$TIMES*R*$_2$)байланысы**

ТАӘ	Топ	Жынысы	1 топ	Куратор	Курс
Иванов Ф.И.	35	е	44	Никитина Л.П.	4
Таран О.С.	44	е	35	Кирсанова Л.Н.	3
Иванов Ф.И.	35	е	35	Кирсанова Л.Н.	3
Таран О.С.	44	е	44	Никитина Л.П.	4

3.5-сурет. Байланысты көбейтінді мысалы

*R*байланысы

ТАӘ	Топ	Жынысы
Иванов Ф.И.	35	е
Кириллова Е.Е.	35	ә
Потапов В.С.	35	е
Дудко О.В.	35	ә
Таран О.С.	44	е
Ильин Г.С.	44	е
Федорова Д.С.	35	ә
Медведева Ж. А.	44	ә
Пушкина А. А.	44	ә

*R*WHERE байланысы Топ= 44

ТАӘ	Топ	Жынысы
Таран О.С.	44	е
Ильин Г.С.	44	е
Медведева Ж.А.	44	ә
Пушкина А. А.	44	ә

3.6-сурет. Байланысты шектеу

f (R WHERE *f*) формуласы бойынша *R* байланысының шектелу (таңдау) амалы R байланысындағы кортеждерден тұратын, логикалық өрнектердің шынайылығын қанағаттандыратын, f формуласымен белгіленген осындай атаумен және денемен жаңа байланыстарды ұсынады. Формуланы басып алу үшін атрибуттардың аты (немесе қатарлар нөмірі), тұрақты шамалар, логикалық амалдар (AND — И, OR — ИЛИ, NOT — НЕ), салыстыру амалдары («=», «^», «>», «>», «<», «<») және жақша қолданылады. 3.6 суретте *R* байланысын шектеу амалының орындалу мысалы көрсетілген – 44 топтағы студенттерді таңдау (WHERE Группа = 44).

А байланысының көпшілік атрибутына деген проекциялау нәтижесі $\{a_t, a_2, ..., a_n\}$ *(PROJECT*А $\{a_j, a_2, ..., a_n\}$) атрибуттардың көптігімен анықталатын $\{a_t, a_2, ..., a_n\}$және a_j: vy a_2: v_2, ..., a_n : v_n>түріндегі кортеждерден тұратын, А байланысында кортеж болатындай атауымен байланыс болып табылады, v_j мағынасы бар a_j атрибуты, v_2 мағынасы бар a_2 атрибуты, v_n мағынасы бар a_n атрибуты. Осылайша проекция амалын орындау кезінде байланыстың «көлденең» таңдауының көшірме кортеждердің өшірілуі пайда болады. Басқаша айтқанда А байланысының проекциясы көптілік бар жердегі атрибуттарда $\{a_j, a_2, ..., a_n\}$ А байланысы тақырыбы атрибуттар толық тізімінің қосалқы жиынтығы болып табылады, қайталанатын кортеждерді есепке алмағанда А байланысындағы кортеждерді ұстанатын,атауымен $\{a_j, a_2, ..., a_n\}$ және денесімен байланысты.

Проекция амалы жазып алулардың келесідей қосымша нұсқаларын көрсетеді:

- атрибуттар тізімінің болмауы барлық атрибуттарды көрсету дегенді білдіреді (тепе-тең проекция амалы);
- бос проекцияның нәтижесі бос жиынтық болып табылады;
- проекция амалы еркін байланысқа қолданыла алады, соның ішінде таңдау нәтижесінеде.

Мысалы, алдыңғы мысалдағы (3.6 сурет) *R* байланысы үшін PROJECT СТУДЕНТТЕР {Топ} проекциясының нәтижесі екі кортеждерден тұратын байланыс болады (3.7 сурет).

Проекция амалы, кейде оны көлденең таңдау амалы депте атайды, ол үлгі объектісінің тек талап етілген сипаттамасын ғана алуға мүмкіндік береді. Мысалы, *R* байланысындағы ТАӘ және Топ атрибутын таңдау. Бәрінен бұрын проекция амалы көлденең таңдау амалында немесе іріктеуде аралық қадам ретінде қолданылады. Сонымен қатар, ол қорытынды кезеңде сауалына өз бетінше жауап алу үшін қолданылады.

Байланыстың *бөлу амалы* жеткілікті түрде қиын және кезеңдік қарастыруды талап етеді.Тіпті екі байланыс берілседа: A және B атауымен бірге {aj, a_2, ..., a_n, b_j, b_2, ..., b_m}және {b_j, b_2, ..., b_m} сай келеді. *bj (i* = 1, 2, ..., m) атрибуттар сол бір ғана иелікте анықталады және бірдей атауға ие, яғни екі байланыс үшін ортақ болып келеді. Атрибуттар жиынтығы ретінде *Щ* деп атасық, құрамдық атрибуты a, жиынтық атрибуты {*bj*} – b құрамдық атрибуты. Сол кезде A және B байланысының бөлінуін (A DIVIDE BY B) *a* атауының және денесінің байланысы деп атайды, оның құрамында барлық {a}кортеждердің жиынтығы бар, {a,b} кортежі болатын, A байланысына жататын {b} кортежі үшін B байланысына жататын. Басқаша айтар болсақ, нәтижесінде A байланысындағы *b* мағынасы B байланысындағы *b –ның* барлық мағынасын иемденеді. Бөлу амалы ыңғайлы болып келеді, егерде жеке атрибуттардың кейбір жиынтығы сипаттамасын салыстыруды талап ететін болса.

Қабайланысы

PROJECT

СТУДЕНТТЕР {Топ}

5

4

3.7-сурет. Байланыс проекциясының нәтижесі

Бөлу амалының орындалу мысалын келтірейік. Дерекқорда R_1{ТАӘ, Пән атауы, Бағалау} байланысы және R_2{Пән атауы} жалғыз байланысы бар (3.8 сурет). Сол кезде бөлу (R_1DIVIDE BY R_2) нәтижесі R_2 байланысының барлық пәнінен баға алған барлық студенттер туралы ақпаратты береді

Байланыс

ТАӘ	Пәннің атауы	Баға
Иванов Ф.И.	Ақпараттық жүйелер	5
Кириллова Е.Е.	Ақпараттық жүйелер	4
Дудко О.В.	Ақпараттық жүйелер	5
Федорова Д.С.	Ақпараттық жүйелер	5
Ильин Г.С.	Ақпараттық жүйелер	4
Федорова Д.С.	Математика	5
Иванов Ф.И.	Математика	3
Кириллова Е.Е.	Математика	5
Федорова Д.С.	Экология	5
Иванов Ф.И.	Экология	4
Дудко О.В.	Экология	4
Федорова Д.С.	Дерекқор	5
Медведева Ж.А.	Дерекқор	4
Иванов Ф.И.	Дерекқор	3

R_2 байланысы

Пәннің атауы
Ақпараттық жүйелер
Экология
Дерекқор

R (i?j DIV1DEBY R_2)байланысы

ТАӘ
Иванов Ф.И.
Федорова Д.

3.8-сурет. Бөлу амалы

R_j және R_2 байланысының *қосу амалы* C_f (R_j, R_2) шарт бойынша *ƒ* формуласымен белгіленген *R* байланысын ұсынады, оны f формуласы бойынша таңдау амалының нәтижесіне R_j және R_2 байланысының декарттық көбейтіндісі арқылы алуға болады. Бұл амал кейбір жағдайлардың немесе формуланың негізінде екі байланысты бірге біріктіру болған уақытта қолдануға арналған. *ƒ* формуласын жазып алу тәртібі таңдау амалы секілді тура сондай болып келеді.

Басқаша айтар болсақ, *a* атрибутындағы R_j байланысын *b* атрибутындағы R_2 байланысымен біріктіру (байланыстарда ортақ аты бар атрибуттар жоқ) (R_JTIMES R_2) WHERE *a Q b* түріндегі амалды орындаудың нәтижесі болып табылады,

Q– бір доменде (негізгі атрибут үшін – бірнеше) анықталынған атрибуттардың логикалық өрнегі. C_f (R_1, R_2) формуласы бар / еркін түрге ие байланыс (ары қарай қарастыратын жиі кездесетн жағдайға қарағанда) Q – байланыс депте аталады. Байланыс амалын мысалдау үшін алдыңғы мысалдарда қолданылған байланыстың атауын және денені кішкене өзгертеміз. Егер дерекқорда R^MO, Топ, Жынысы} және R_2{топ, Куратор, Курс} байланысы бар деп қарастыратын болсақ (3.9 сурет). R_1және R_2 байланысындағы Топ атрибуты бойынша бірігулерін табуымыз керек (оларды ^ деп белгілейміз. Топ және R_2. Топ барлық байланыс үшін сай келеді).

Нәтижесінде R байланысы пайда болады, бірінші және екінші байланыстағы кортеждерді біріктіру арқылы болады және жағдайды қанағаттандырады (^.Группа = R_2.топ), яғни студенттер тізімі және топтардың кураторы.

ТАӘ	Топ	Жынысы
Иванов Ф.И.	35	е
Кириллова Е.Е.	35	ә
Потапов В.С.	35	е
Дудко О.В.	38	ә
Таран О.С.	44	е
Ильин Г.С.	44	е
Федорова Д.С.	35	ә
Медведева Ж.А.	44	ә
Пушкина А. А.	44	ә

Топ	Куратор	Курс
35	Кирсанова Л.Н.	3
44	Никитина Л.П.	4
38	Петрова Е.М.	3
24	Долинская НА.	2

R **байланысы TIMES***R*$_2$**)WHEREД. Топ**= R_2,**Тобы)**

ТАӘ	Топ	Жынысы	Куратор	Курс
Иванов Ф.И.	35	е	Кирсанова Л.Н.	3
Потапов В.С.	35	е	Кирсанова Л.Н.	3
Дудко О.В.	38	ә	Петрова Е.М.	3
Таран О.С.	44	е	Никитина Л.П.	4
Ильин Г.С.	44	е	Никитина Л.П.	4

3.9-сурет. Байланыс амалы

Тек қана 3 курс студенттерін таңдау шартын қоюға болады: (R (R_2TIMESR_2) WHERE (R-^Группа = R_2.rpynna) AND(Курс=3)).

Тәжірибелік көзқарас бойынша біріктірудің жиі кездесетін жағдайындағы маңыздысы эквибіріктіру және табиғи біріктіру.

Эквибіріктіру амалы операндтардың теңдік формуласын беруімен сипатталады. Кейде екі байланыстың эквибіріктіруі атрибуттары екі байланыстада бірдей сәйкес атауларға және домендерге ие қатарлар бойынша орындалады. Мұндай жағдайда ортақ атрибуттар бойынша эквибіріктіру туралы айтылады. Жоғарыда келтірілген мысал бір қатар бойынша эквибіріктірудің жиі кездесетін жағдайын көрсетеді.

Табиғи біріктіру амалы (JOIN амалы)ортақ атрибутқа ие (қарапайым және құрамдық) екі байланысқа қолданылады. Бұл атрибут байланыс кезінде бір ғана атқа ие болады (аттардың жиынтығы) және бір ғана домен негізінде анықталады (домендерде).

Табиғи біріктіру амалының нәтижесі R_jжәне R_2 байланысының эквибіріктіру проекциясын ұсынатын, ортақ атрибут бойынша екі байланыстыңда атрибуттар жиынтығын біріктіретін R байланысы болып табылады.

Енді Коддтың реляциялық алгебра амалы бойынша қысқаша анықтама қалыптастыруға болады:

- бірдей атауға ие екі байланысты біріктіру амалын орындау кезінде (UNION) операндтар байланысының жоқ дегенде біреуінің құрамына кіретін, барлық кортеждерді иемденген байланыс орнатады;
- бірдей атауға ие екі байланысты қиылыстыру амалы (INTERSECT) операнд-байланыстардың екеуінеде кіретін, барлық кортеждерді иемденген байланыс орнатады;
- бірдей атауға ие екі байланыс үшін түрлі болып келетін байланыс (MINUS) бірінші байланысқа да және екінші байланысқада кірмейтін барлық кортеждерді иемденеді;
- атауларының қиылысы бос болып келетін, екі байланыстың көбейтіндісін орындау кезінде (TIMES) бірінші және екінші операндтар кортежін біріктіру арқылы орындалатын кортеждер байланысы орнатылады;
- кейбір жағдайлар бойынша байланыстың шектелу нәтижесі

- екі байланысты кейбір жағдайларға қарай біріктіру кезінде (JOIN) кортеждері бірінші және екінші байланысты біріктіру арқылы орындалатын және сол жағдайды қанағаттандыратын нәтижелі байланыстар пайда болады;
- реляциялық бөлу амалында (DIVIDEBY) екі операнд бар – бинарлық және унарлық байланыс. Нәтижелік байланыстар унарлық кортеждерден тұрады, кортеждердің екінші атрибут мәнісі жиынтығы (бірінші атрибутының белгіленген мәнісінде) екінші операндтың жиынтықты мәнісін енгізе отырып, бірінші атрибутының мәнісін иемденген. Коддтың алгебрасы артық, алгебраның базалық амалы ретінде декарттық туынды амалын енгізу тәжірибесіз оқырмандарды шатастыруы мүмкін. Соның өзінде осы шамалы ескірген және кемшілікті теориясы реляциялық деректер үлгісінің базалық қимыл механизмін реляциялық дерекқорлардың барлық дерлік оқу құралдарында талқылауға кіріседі. Себебі тіл семантикасы SQL (сұраныстың құрылымдық тілі) көбіне дәл осы алгебраға негізделеді, бізге SQL алдын-ала түсініп алғандықтан оқу оңайға түседі.

3.3. ИНДЕКСТЕУ

Деректер индекстеу кестесінде қолданушы қалай енгізсе, сол тәртіпте сақталып қалады. Бұл жазуларды есептеудің физикалық тәртібі. Алайда жиі деректерді басқа, физикалықтан керемет тәртіпте, қандайда бір өріс арқылы іріктей отырып көрсету талап етіледі. Іріктеу ондағы мәннің өсуі немесе кемуі тәртібіндегі белгілі бір өрісте бірыңғайлылық басылымымен жүзеге асырылады. Мысалы, топтардың нөмірімен және/немесе әліпбиімен реттелген студенттер туралы деректі қарау талап етілуі мүмкін (3.10 сурет).

Сонымен қатар, үлкен ауқымдағы жазылған, белгілі бір сынды қанағаттандыратын ақпараттарды табу керек жайттары жиі кездеседі, мысалы студентті туған дылы бойынша табу. Үлкен кестедегі басылымдардан іздеуде қарапайым іріктеп алу ұзақ уақытты талап етуі мүмкін және сол себептен қарқынды болып есептелмейді. Бұны шешудің қарқынды құралы индекстерді қолдану болып табылады.

Индекстерді дерекқордағы арнайы құрылым ретінде елестетуге болады, ол кестедегі белгілі бір өрісте немесе өріс жиынтығында іздеуді және іріктеуді жылдамдатады.

Топ	ТАӘ
35	Иванов Ф.И.
35	Кириллова Е.Е.
35	Потапов В.С.
35	Дудко О.В.
48	Таран О.С.
44	Ильин Г.С.
35	Федорова Д.С.
44	Медведева Ж. А.
44	Пушкина А. А.

а

Топ	ТАӘ
35	Дудко О.В.
35	Иванов Ф.И.
35	Кириллова Е.Е.
35	Потапов В.С.
35	Федорова Д.С.
44	Ильин Г.С.
44	Медведева Ж. А.
44	Пушкина А. А.
48	Таран О.С.

б

3.10-сурет. «Топ» (б) өрісі бойынша кестедегі реттеу басылымы және басылымға ерудің (а) физикалық тәртібі

Индекстер сонымен қатар, деректерді бірегейлікпен қамтамассыз ету үшін қолданылады, яғни алғашқылық немесе бірегейлік кілтті құру үшін. Физикалық индекс жазба мекен-жайын анықтау үшін қолданылады. Индекстер болған кезде, болмаған кезіне қарағанда көптеген жағдайларда деректерді іздеу анағұрлым тезірек орындаады, себебі индекстегі мән реттелген, ал индекстің өзі азғантай. Индекс қолданушымен немесе нақты кесте жүйесімен жасалынады.

Индекстелген кестеде түрлі амалдарды орындау жылдамдығын асырудың басты себебі жұмыстың негізгі бөлігі кестенің өзімен емес, индексті файлдардың өзімен жүргізіледі. Индекстелген кестемен жұмыс жасау барысындағы өнімділікті арттырудың ең бір қарқындысы кесте көлемі бойынша мәнділікке қол жеткізу. Индекс деректер кестесіндегі әрбір жазылымдар үшін кілттік мәнге ие. Кілттік мән кестенің бір немесе бірнеше өрісі арқылы анықталады. Соынмен қатар, индекс кестедегі сай келуші жазуларға қатысты кереметтей сілтемелерге ие, іздеу белгісін қанағаттандыратын жолдарды іздеуге көмектеседі. Индекстерді пайдалану арқылы жұмысты жылдамдату индекстің іздеуге тиімді етілген құрылымын есепке алумен жүзеге асырылады (мысалы, теңдестірілген ағаш).

Индекстерді кітаптағы тақырыптармен салыстыруға болады. Мәтіннің қажетті тұсын іздеген кезде біз қай бөлімде немесе қай параграфта көрсетілгенін іздейміз, кейін мазмұнында көрсетілген кітаптың қажетті бетін ашамыз, сол жерде мәтінді іздестіре бастаймыз. Индекс кесте көрсеткішінің рөлін атқарады, оны қарау кесте жазбаларына мән берумен орын алады. Индекстің көрсетілген мәнімен жүйе деректер массасында қажетті ақпараттың бөлігін табады және қолданушыға көрсетеді. Осылайша, индекс кестедегі деректерді іздестіру үрдісін жылдамдатуға көмектеседі, ал кейде қолданушының сұранысымен алынған деректерді реттеу үрдісін жылдамдатады.

Кестелердегі ақпаратқа физикалық рұқсат беруді ұйымдастыру әдісі көбіне келесі факторларға байланысты:

- индекстік құжат жазылымындағы кілт өрісі құрамының түрі;
- негізгі кесте жазуларындағы қолданылатын сілтемелер (белгілер) түрі;
- қажетті жазылымдарды іздеу әдісі.

Дерекқормен жұмыс істейтін қосымшаларды ойлап табу кезінде қарапайым индекстер көп пайдаланылады. «Студенттер» кестесіндегі қарапайым индекстің мысалы ретінде «Сәйкестендіргіш» (жеке нөмір) жолы қызмет ете алады (3.11 сурет).

Алайда көп жағдайларда деректерді белгілі бір ретпен ұсыну үшін бір-бір өріспен құрылған қарапайым индексті пайдалану жеткілікті, көп жағдайда басты индекссіз жұмыс істеу мүмкін емес

Сәйкестендіргіш	ТАӘ	Топ	Жынысы
101	Иванов Ф.И.	35	е
102	Кириллова Е.Е.	35	ә
103	Потапов В.С.	35	е
104	Дудко О.В.	35	ә
105	Таран О.С.	44	е
106	Ильин Г.С.	44	е
107	Федорова Д.С.	35	ә
108	Медведева Ж. А.	44	ә
109	Пушкина А. А.	44	ә

3.11-сурет. «Сәйкестендіргіш» қатары бойынша қарапайым индекс мысалы

Бөлімше атауы	Тізім нөмірі	ТАӘ	Қызметі	Айлық
001	1	Иванов	Директор	50000
001	2	Потапов	Директор орынбасары	40000
001	3	Таран	Бас инженер	40000
002	1	Ильин	Бас есепші	45 000
002	2	Берковская Ольга Николаевна	Есепші	20000
002	3	Дремина Елена Евгеньевна	Есепші	20000
003	1	Попенко Борис Сергеевич	Бас механик	35000
003	2	Федорова Дарья Сергеевна	Инженер	30000

3.12-сурет. «Бөлімше» және «Тізім нөмірі» қатары бойынша құрамдық индекс мысалы

Құрамдық индекстің жақсы мысалы ретінде «Қызметкерлер» кестесін алуға болады (3.12 сурет). Осындай жағдайда адамның тегі бойынша қарапайым индекс ретінде пайдалануға болмайтындығы түсінікті. Тіпті «Бөлімше нөмірі» және «ТАӘ» өрісіне негізделген құрамдық индексті пайдалану қарқынды болмауы мүмкін, себебі, бұл жағдайда бірдей текті адамдардың болуы мүмкін. Жағдайдан шығудың жолы құрамдық индексті пайдалану болуы мүмкін, мысалы кестенің келесі өрісіне негізделген: «Бөлімше», «Тізбелік нөмір».

Дерекқордағы кестелердің индекстері болмауы да мүмкін. Мұндай жағдайда үлкен кесте үшін нақты бір жазылымды іздеу уақыты мәнді болуы мүмкін және индексті пайдалану қажетті болып табылады. Бір жағынан индекстердің үлкен санын жасауға тым беріліп кеткен дұрыс емес. Индекс санының өсуі қосылу, жаңару, кесте қатарларын өшіру амалдарын баяулатады, себебі осының өзінде индекстің өзін жаңарту керек. Сұраныстың тиімді өнімділігі үшін индекстер әдетте кестенің сұраныста жиі қолданылатын қатарында жасалады. Сонымен қатар индекстер жадтың қосымша көлемін иемденеді, сол себептен индексті жасар алдында сұраныстың өнімділігіндегі жоспарланған ұтыстың индекспен сүйемелденетін компьютер ресурсының қосымша шығынын асып түсетініне көз жеткізген дұрыс.

3.4. КЕСТЕЛЕРДІ БАЙЛАНЫСТЫРУ. СІЛТЕМЕЛІК ТҰТАСТЫЛЫҚ ТҮСІНІГІ

Шынайы әлемнің объектілері арасындағы байланыстар өздерінің бейнелерін дерекқордағы деректер құрылымынан таба алады. Әдетте реляциялық дерекқор өзара байланысқан кесте жиынтығынан тұрады. Екі немесе бірнеше кесте арасындағы байланысты ұйымдастыру *кестені байланыстыру* деп аталады.

Кестелер арасындағы байланыстарды кестені жасау кезінде және қосымшаларды орындау кезінде ДБЖ (дерекқор басқару жүйесі) ұсынылған құралдарды қолдана отырып, орнатуға болады. Екі немесе оданда көп кестені байланыстыруға болады. Бұл тең құқылы байланыс секілді бағыну байланысы болуыда мүмкін, басты кестенің әрбір жазылымы үшін (ата-аналық депте аталатын) бағынышты кестедегі (қыздық) бір немесе бірнеше жазулардың болуы мүмкін. Реляциялық дерекқорда байланысты кестелерден басқа жеке кестелерде болуы мүмкін, ешбір басқа кестемен байланыспаған.

Кестелерді байланыстыру үшін байланыс өрісі қолданылады («сәйкес келуші» өріс). Байланыс өрісі міндетті түрде сәйкестендірілген болуы керек және бір типті болуы тиіс. Бағынышты кестеде басты кестемен байланыс орнату үшін индекс қойылады (сыртқы кілт). Бұл индекстің өріс құрамы басты кестенің индекс өріс құрамымен толықтай немесе жартылай сәйкес келуі керек.

Дерекқор кестесі арасындағы байланыстың үш түрлі түрі бөлінеді:

1) «көпшілікке біреу»;
2) «біреуіне біреу»;
3) «көбісіне - көбі».

«Көпшілікке-біреу» байланысы ата-аналық кесте жазылымының біреуі қыздық жазылымның бірнешеуіне сай келе алғанда көп орынға ие болады. «Көпшілікке-біреу» байланысын кейде «көпшілік-біреуіне» депте атайды. Қай түрін алсақта кестелер арасындағы болмыс өзгеріссіз қалады. «Көпшілікке-біреу» байланысы реляциялық дерекқор үшін ең кең таралғаны болып табылады. Ол деректердің иерархиялық құрылымында үлгілеуге мүмкіндік береді. «Көпшілікке-біреу» байланысына мысал келтіру үшін, екі кестені қолданатын боламыз. Біреуінде оқу орнының топ тізімі жазылады (3.13 сурет), екіншісінде студенттер тізімі жазылады (3.14 сурет).

ID	NAME_GROUP	K_R.UK
1	38-пкс	Федоров С.В.
2	18-пкс	Иванов Ф.И.
3	44-э	Попенко Б.С.
4	36-тг	Филин О.К.

3.13-сурет. «Топтар» кестесі

«Топтар» кестесі келесі құрылымға ие:

ID — жазба сәйкестендіргіші;

NAME_GROUP — топтың атауы, символдық түр алаңы;

K_RUK — сынып жетекшісі, символдық түр алаңы.

«Студенттер» кестесі келесідей алаңды қамтиды:

ID — жазылу идентификаторы, тұтас сандық түрдің кілттік алаңы;

NAME_STUD — студенттің ТАӘ, символдық түр алаңы;

ID_GR ...ық түрдің

ID	NAME_STUD	ID_GRUP
1	Иванов Петр Иванович	1
2	Петров Иван Серегеевич	1
3	Таран Ольга Сергеевна	2
4	Дудко Олеся Владимировна	3
4	Дремина Елена Евгеньевна	1
5	Федорова Дарья Сергеевна	2
6	Лесовая Вера Николаевна	2
7	Семенова Елена Геннадьевна	3
8	Никитина Любовь Петровна	4
9	Курганский Владимир Иванович	4

3.14-сурет. «Студенттер» кестесі

Екі кестенің байланысқан кезінде кестедегі қандайда бір жазбаны таңдауда «Топтар» кестесінде «Студенттер» кестесінде сол топта оқитын студенттердің ғана жазбалары шығады. Мысалы, егер бірінші кестеде көрсеткіш «18-пкс» (ID=2) тобының жазбасында орнатылатын болса, онда студенттер кестесінде тек ID_GROUP=2 үш жазбасы ғана қолжетімді болады (3.15 сурет).

Мысалға, «Топтар» кестесінен жазба өшірілді делік (кездейсоқ немесе әдейі), онда бағынышты «Студенттер» кестесінде сәйкес келетін жазбалар үшін байланыс бұзылатын болады, топтың идентификаторы бойынша бұл студенттердің қай топқа жататындығын анықтау мүмкін емес болады, яғни ақпаратты жоғалту болады, ондайға жол берілмейді. Сол себептен басты кестеден (ата-аналық) жазбаны өшіруге тиым салу керек, егер онда бағынышты кестемен байланысты жазбалар болса, не болмаса бағынышты кестедегі жазбамен бірге басты кестеден сәйкес

ш	NAME_GROUP	K_R.UK
1	**38-пкс**	**Федоров С.В.**
2	**18-пкс**	**Иванов Ф.И.**
3	**44-э**	**Попенко Б.С.**
4	**36-тг**	Филин **О.К.**

ID	NAME_STUD	ID_GRUP
1	**Иванов Петр Иванович**	**1**
2	**Петров Иван Серегеевич**	**1**
3	**Таран Ольга Сергеевна**	**2**
4	**Дудко Олеся Владимировна**	**3**
4	**Дремина Елена Евгеньевна**	**1**
5	**Федорова Дарья Сергеевна**	**2**
6	**Лесовая Вера Николаевна**	**2**
7	**Семенова Елена Геннадьевна**	**3**
8	**Никитина Любовь Петровна**	**4**
9	**Курганский Владимир Иванович**	**4**

3.15-сурет. «Топтар» және «Студенттер кестесіндегі жазбалардың сай келу мысалы

Дәл осылайша бағынышты кестеде қандайда бір деректерді басты кестедегі бір дерекке байланыстырмай тұрып енгізуге болмайды, яғни басты кестеде жоқ мәнді енгізуге немесе байланыс алаңын бос қалдыруға болмайды. Басты кестенің немесе бағынышты кесте байланысының алаңында кілттік алаң мәнісі өзгерген кезде байланыс бұзылатын болады, ол құлдырауға немесе ақпараттың бұрмалануына алып келеді.

«Біреуіне-біреу» байланысы орынға ие болады, егерде ата-аналық кестедегі бір жазбаға қыздық кестедегі бір жазба сай келгенде. Бұл байланыс «көпшілікке-біреу» байланысына қарағанда аз кездеседі. Оны егерде дерекқор кестесі екінші деңгейлі ақпараттан «ұлғайып» кетпесін десе пайдаланады. Мысалы, қандай да бір кестеде қызметкерлер туралы бірнеше жолдары (сипаттамалары) және бірнеше мыңдаған жазбалары бар ақпараттың мұрағаты болады. Бұл мұрағаттан үнемі тек қана ТАӘ, туған жылы және мекен-жайы қолданылады. Әркезде осы деректерді есепке ала отырып, барлық кестені ашып және жадқа деректің үлкен көлемін жүктеуде оперативті жадты қолдану көзқарасы жағынан тиімсіз болушы еді. Сол себептен оны екіге бөлуге болады: біреуінде үнемі сұралатын ақпаратты, ал екіншісінде аз қолданылатын ақпаратты сақтау керек. Сол кезде бұл кестелерді «біреуі-біреуіне» байланысы арқылы байланыстыруға болады. Және ақпараттың кей бөлігін «құпия сақтау» керек кездері болады, сондықтан барлық қолданушыларға қолжетімді болмас үшін құпия деректерді жеке кестеге енгізіп, құпия кілтті қою керек.

«Көбісі-көбісіне» байланысы келесідей жағдайларда қолданылады:

- ата-аналық кестедегі бір жазба қыздықтағы біреуден көп жазбаға сай келеді;
- қыздық кестедегі бір жазба ата-аналықтағы біреуден аса жазбаға сай келеді.

Тәжірибеде «көбісі-көбісіне» байланысы аз қолданылады. Себебі – кестелер арасындағы байланысты ұйымдастыру қиындығы және олардың жазбасы арасындағы өзара әрекеттестіктер. Сонымен қатар, «көбісі-көбісіне» байланысы үшін кестедегі басты және бағынышты түсініктері маңыздылыққа ие емес. Реляциялық дерекқордағы «көбісі-көбісіне» байланысын қосымша кестелерді енгізу көмегі арқылы «көбісі-біреуіне» байланысына алмастыру керек.

Сілтемелік тұтастық реляциялық дерекқорда байланысқан кестелер арасындағы келісімділікті ұсынады. Сілтемелік

Сілтемелік тұтастылықты ұстану үшін, сыртқы кілтпен хабарланған кестедегі кез-келген алаңның ата-аналық кестенің алғашқы кілтінің аумағында мәнді ұстануы керектігі талап етіледі. Сілтемелік тұтастылық қолданушылармен немесе қосымшалармен келісілмеген енгізудің алдын алады. Көптеген реляциялық ДБЖ-да сілтемелік тұтастылықтың түрлі тәртіптері болады, оны екі кесте байланысын жасау кезінде қолдануға болады.

Байланыстағы кестелермен жұмыс келесідей ерекшеліктерге ие:

- байланыс өрісінің өзгеруі (өзгертілуі) кезінде екі кесте арасындағы жазбаның байланысы бұзылуы мүмкін. Сол себептен басты кесте жазбасының байланыс өрісін өзгерту кезінде сәйкесінше барлық бағынышты кестенің байланыс аумағын және байланыс аумағының мағынасын өзгерту қажет (каскадтық өзгеріс);
- басты кестенің жазбасын өшірген кезде бағынышты кестедегі сәйкес келетін жазбалардыда өшіру қажет (каскадтық өшіру);
- бағынышты кестеге жазбаны қосу кезінде олардың байланыс аумағының мәнісі басты кестенің байланыс аумағымен бірдей мәнде орнатылуы керек.

Каскадтық өшіру, каскадтық өзгерту, жаңа мәнді орнату бойынша шектеулер кестелерде оларды жасау кезінде немесе қайта құрылымдау кезінде болуы мүмкін. Бұл шектеулер аумақты және индексті сипаттау және басқа элементтермен бірге кестенің құрылымына кіреді және осы дерекқормен жұмыс жасайтын барлық қосымшалар үшін әрекет етеді. Мұндай шектеулерді бағдарлама көмегіменде жүзеге асыруға болады: кестелер арасындағы байланысты орнату және өшіру, байланыс аумағын бақылау

3.5. РЕЛЯЦИЯЛЫҚ ДЕРЕККОРДАҒЫ ТҰТАСТЫЛЫҚТЫ ҚОЛДАУДЫҢ қағидалары

Деректің тұтастылығы – дерекқор технологиясындағы негізін салушы түсініктердің бірі. Бұл сипаттама дерекқордағы ақпараттың үнемі нақты және толық екендігіне көз жеткізетін құралдың болуынкөрсетеді. Тұтастылық ережесі орнатылуы керек және олар дерекқормен бірге сақталауы керек және жаһандық деңгейде бақыланады. Деректің тұтастылығы деректің қандай жолмен жадқа кіргендігіне қарамастан қамтамассыз етілуі керек (интерактивті тәртіпте импорт құралы арқылы немесе арнайы бағдарлама көмегімен).

Реляциялық үлгіде шынайы әлемнің объектілері өзара байланыстағы қатынастың жиынтығы ретінде сипатталады. Тұтастылықты пәндік аумақтың ақпараттық моделінің шынайы әлемдегі объектісі ретінде және уақыттың кез-келген уақытындағы өзара байланысы ретінде анықтауға болады. Құрылған үлгі үшін маңызды пәндік аумақтағы өзгеріс дерекқорда көрінуі керек. Соның өзінде пәндік аумақ терминіндегі ақпараттық үлгінің бір реттік интерпретациясы сақталып қалуы тиіс.

Тұтастылықтың шектелуі дерекқор жүйесінің бір жағдайдан екінші жағдайға ауысуы кезінде деректердің қайталанбауын қамтамассыз етеді және дерекқорда сақталған деректер арқылы пәндік аумақты сәйкесінше көрсетуге мүмкіндік береді. Тұтастылықты шектеу анық және анық емес болып бөлінеді.

Анық емес шектеулер деректің құрылымымен анықталады. Мысалы, «Студент» секілді жазбалар «Топ» секілді деректер жиынтығының қандайда бір үлгісіндегі міндетті мүшесі болып табылады деген ақиқат болмысы жағынан тұтастылық шектеуіне қызмет етеді, әрбір студент қандай да бір топта міндетті түрде болуы керек дегенді білдіреді.

Анық шектеулер деректерді сипаттау тілі құралы көмегімен дерекқор сызбасында жасалады (DDL, Data Definition Language). Анық шектеулер ретінде көбіне деректердің мәнісіне жататын жағдайлар орын алады. Мысалы, еңбекақы кері болуы мүмкін емес, ал қызметкерді жұмысқа алу күні міндетті түрде оны басқа жұмысқа ауыстыру кезіндегі күнмен сәйкес келеді. Бұл шектеулердің орындалуын ДБЖ өзінің қызметін орындау кезінде қадағалайды.

Тұтастылықтың сонымен қатар, статикалық және динамикалық шектеу түрлері болады. Статикалық шектеу пәндік аумақтың барлық жағдайына тән келеді, ал динамикалық пәндік аумақтың бір күйден екінші күйге ауысу мүмкіндігін қарастырады. Тұтастылықтың статикалық шектеуінің мысалы ретінде куәлік нөмірінің бірегейлігін талап ету немесе алдыңғы күнінен көп бола алмайтын туған күнін шектеу бола алады. Тұтастылықтың динамикалық шектелу мысалы ретінде банктік жүйенің шектелуін алып қарастыруға болады, бұл жерде клиент туралы деректі оның есебі жабылған кезге дейін өшіруге болмайтындығы.

ДБЖ деңгейіндегі деректер тұтастылығын қамтамассыз ету құралына келесілер жатады:

- сілтемелік тұтастылықты қолдау құралы, ол кестелер байланысы туралы ақпаратты жазуды қамтамассыз етеді және сілтемелік тұтастылықтың бұзылуына алып келетін кез-келген жүйені автоматты түрде қиып өтеді.

Кейбір ДБЖ жақсы өңделген алғашқылық кілт бірегейлігі мүмкіндіктерін, жүйені шектеу (болдырмау) және тіпті каскадтық жаңарту және ақпаратты өшіруді жүзеге асыратын ДБЖ процессорына ие. Мұндай жүйелерде аумаққа немесе кестеге бекітілетін нақытылықты тексеру терезелік форма арқылы ақпаратты енгізу уақытында ғана емес, деректерді өзгерткеннен кейін ғана жүргізілетін болады. Бұл қасиетін әрбір аумақ үшін қолдануға болады және жазбаны тұтастай басып алу үшін, ол жеке аумақтардың мәнісін ғана бақылауды емес, сонымен қатар сол жазбаның бірнеше аумағы арасындағы өзара байланысты бақылауға мүмкіндік береді.

Реляциялық деректер үлгісіндегі тұтастылықты қолдау келесідей аспектілерге ие.

Біріншіден, бұл *құрылымдық тұтастылықты қолдау*, реляциялық ДБЖ деректердің тек қана бір реттік «реляциялық байланыс» секілді құрылымымен жұмыс жасауы керек дегенді білдіреді. Сондықтан, «реляциялық байланыс» түсінігі реляциялық дерекқордың классикалық теориясында қойылған барлық шектеулерді қанағаттандыруы керек:

Реляциялық ДБЖ реляциялық байланыс секілді тек ғана деректер құрылымымен жұмыс жасайды. Реляциялық кестеге сәйкес келетін ережелерді ұстанған дұрыс:

- кестеде бірдей кортеждер жоқ;
- қатарлар байланыс атрибуттарына сәйкес келеді;
- алғашқылық кілт үнемі болады;
- әрбір атрибут бірегейлік атқа ие;
- кестедегі қатарлар тәртібі еркін;
- қатарлардың ілесуінің тәртібімен ғана ерекшеленетін екі байланыс ғана бірдей болып есептеледі.

Екінші, *тілдік тұтастылықты қолдау*, реляциялық ДБЖ сипаттау тілдерімен және SQL стандартынан төмен емес деректерді тартумен қамтамассыз етуі керек. Стандартқа сай келмейтін деректерті тартудың басқада төмен деңгейлі құралдары қолжетімді болмауы тиіс. Дәл осы себептен дерекқорда сақталатын ақпаратқа

Үшінден, *сілтемелік тұтастылықты қолдау*, өзара байланыс қатынастарының үлгісі арасындағы берілген қағидаттарының өзара байланысының біреуін қамтамассыз етуді білдіреді:

- бағынышты байланыстағы кортеждер онымен қатынастағы негізгі байланыс кортежін өшіру кезінде жойылады.
- негізгі байланыс кортеждері онымен қатынастағы негізгі байланысты өшірген кезде түрленеді, сондықтан ата-аналық байланыс кілтінің орнына анықталмаған Null мағынасы қойылады.

Сілтемелік тұтастылық қосу немесе өшіру жүйесін орындаудағы деректерді түрлендіру үрдісінде дерекқордың қайшылықсыздық қолдауын қамтамассыз етеді.

Құрылымдық, тілдік және сілтемелік тұтастылық ДБЖ реляциялық құрылым деректері жұмысының тәртібін анықтайды. Тұтастылықтың бұл үш түрін қолдауды талап ету әрбір ДБЖ бұл ережелердің орындалуын қамтамассыз етуі керек дегенді білдіреді, ал жасаушылар реляциялық үлгіні қолдану арқылы және

3.6. РЕЛЯЦИЯЛЫҚ ДЕРЕКТЕР ҮЛГІСІНІҢ ЖЕТІСТІКТЕРІ ЖӘНЕ КЕМШІЛІКТЕРІ

Дерекқорды ұйымдастырудың реляциялық әдісі 1960 жылдардың аяғында Эдгар Коддпен енгізілді және ол бірден танымалдылыққа ие болып, кеңінен таралып кеткен жоқ. Ол уақытта осы аумақ бойынша негізгі теоретикалық нәтижелер ретінде 1970 жылдардың басында қалыптаса бастады және сол кезде реляциялық ДБЖ-ның алғашқы түптұлғалары пайда болды, ұзақ уақыттар бойына мұндай жүйелердің қарқынды дамуына қолжеткізу мүмкін емес болды. Алайда реляциялық әдіс артықшылығы және реляциялық дерекқорды ұйымдастыру және басқару алгоритмдерін және әдістерді дамыту ДБЖ әлемдік нарығында реляциялық жүйенің үстем етуші орынды иемденуіне алып келді. 1980 жылдардың ортасында реляциялық жүйелер жалпы алғашқылық ДБЖ әлемдік нарықтан алып тастады. Және қазіргі таңда реляциялық ДБЖ ең қарқынды болып табылады.

Бұған реляциялық деректер үлгісінің болмыстық жетістіктері әсер етті.

- Кестелік көрініс, оның негізінде кең таралған пәндік аймақтарды үлгілеуге болады. Реляциялық үлгі деректерді сипаттау құралын

Деректердің машиналық көрінісін алу үшін қандайда бір қосымша құрылымдарды енгізудің қажеттілігі жоқ. Сәйкесінше бұл үлгі жоғарғы деңгейдегі деректер тілінің негізін қамтамассыз етеді, ол бір жағынан бағдарламаның максималды тәуелсіздігін қолдайды және екінші жағынан деректердің машиналық көрінісін және ұйымдастыруын қолдайды. Бұл абстракция нақты және ресми анықталуы мүмкін;

- реляциялық деректер үлгісінің басқа жетістігі қарапайым және бір уақытта көпшілік теориясына және математикалық логикаға негізделетін мықты математикалық аппараттың болуы болып табылады және дерекқорды ұйымдастырудағы реляциялық әдістің теоретикалық қорын қамтамассыз етеді. Сонымен қатар, реляциялық әдіс сыртқы жадта дерекқордың нақты бір физикалық ұйымдарын білудің қажетінсіз-ақ деректерді тартудың мүмкіндігін қамтамассыз ете алады.

Жалпымен бірдей жетістіктерімен қатар кемшіліктер қатарыда болады:

- реляциялық жүйеге тән кейбір шектеулер, ол олардың қарапайымдылығының тікелей салдары болып табылады. Әсіресе бұл дәстүрлі емес аумақтағы реляциялық ДБЖ қолдану кезінде сезіледі (мысалы, автоматтандырылған жобалау жүйелерінде), оларда деректердің өте қиын құрылымдары талап етіледі;

- пәндік аймақ семантикасының теңбе-тең бейнесінің болмауы. Бұлда олардың құрылымының қарапайым болуының дәлелі.

- постреляциялық жүйе аумағы бойынша уақытылы зерттеулер басты себеппен дәл осы кемшіліктерді жою үшін арналған.

1. Реляциялық деректер үлгісіндегі негізгі түсініктерге анықтама беріңіз: байланыс, деректер түрі, атрибут, кортеж, алғашқылық кілттердің байланыс сұлбасы.
2. Келесі түсініктерге математикалық анықтама беріңіз: байланыс, деректер түрі, атрибут, кортеж, алғашқылық кілттердің байланыс сұлбасы.
3. Байланыстың негізгі ерекшелігін сипаттаңыз.
4. Реляциялық алгебра дегеніміз не?
5. Реляциялық алгебраның базалық теоретика-жиынтық амалын сипаттаңыз.

6. Арнайы реляциялық амалдарды сипаттаңыз.
7. Унарлы және бинарлы амал дегеніміз не?
8. Байланысты біріктірудің, қиылыстырудың, кемітудің мысалдарын келтіріңіз.
9. Не себепті байланысты көбейту аз қолданылады?
10. Көбейту және біріктіру амалы арасында ортақ не бар?
11. Эквибіріктіру және табиғи біріктіру амалдарының ерекшелігі неде?
12. Дерекқордағы индекс дегеніміз не?
13. Индекстелген кестедегі деректермен жұмыс жасау жылдамдығы ненің арқасында орын алады?
14. Дерекқордағы кестелерді біріктірудің механизмін түсіндіріңіз.
15. «Деректер тұтастылығы» дегеніміз нені білдіреді?
16. Біріктірілген кестедегі жұмыс ерекшелігін сипаттаңыз.
17. «Біреуіне-біреу», «көпшілікке-біреу», «көбісіне көбі» байланыстарының түрін сипаттаңыз.
18. «Біреуіне-біреу» байланысы қандай жағжайда қолданылады?
19. «Көбісіне-көбі» байланысынан не себепті құтылу керек?
20. Реляциялық деректер үлгісінің жетістіктерін және кемшіліктерін атап көрсетіңіз.

4 тарау

ДЕРЕККОРДЫ ЖОБАЛАУ

4.1. ДЕРЕККОРДЫ ЖОБАЛАУ МІНДЕТТЕРІ ЖӘНЕ НЕГІЗГІ КЕЗЕҢДЕРІ

Дерекқорды жобалау үрдісі ұзақ, ол тапсырыс берушінің пәндік аумағы бойынша мамандарының талқылауларын талап ететін болады. Күрделі ұжымдық ақпараттық жүйені өңдеуде дерекқор жобасы барлық жүйе жалпылай құрылатын фундамент болып табылады.

Дерекқорды жобалау деректер үлгісінің өзара байланысқан жиынтығын құрауды көрсетді. Деректер үлгісін құру үрдісі өңдеу үрдісінен және деректермен тарту үрдісінен ажырамас бөлік.

Дерекқорды жобалау екі негізгі фазадан тұрады: логикалық және физикалық үлгілеу. Логикалық үлгі фазасы кезінде жасап шығарушы жасалып жатқан қор үшін талаптарды жиыстырады, пәндік аумақтың сипаттамасын құрастырады және нақты бір ДБЖ тәуелді емес үлгіні ойлап табады. Физикалық үлгі фазасы кезінде жасап шығарушы ДБЖ үшін және қолданушылардың нақты қосымшалары үшін үлгіні ойлап табады.

Ақпараттық жүйе үшін дерекқор жасауда ең маңыздысы ақпараттық жүйе қызметінің жиынтығын орындалуын қамтамассыз ететін деректердің логикалық құрылымын дұрыс өңдеуімен байланысты міндет болып табылады. Нашар ойластырылған дерекқор сәйкесінше қарқынды емес және тіпті қажетсіз болып шығады. Дерекқорды өңдеп шығару – өте қиын міндет. Көбіне оған көп қарама-қайшы талаптар қойылады. Дұрыс логикалық құрылымды жасау деректердің қалыптасуына және өңделуіне әсер ететін барлық факторлардың жиынтықты талдауын қарастырады. Жақсы жобаланған дерекқор:

- дерекқордың құрамына деген қолданушылардың барлық талаптарын қанағаттандыру. Қорды жобалаудан бұрын дерекқордың қызметі туралы қолданушылардың талаптары бойынша зерттеу жүргізу қажет;
- деректердің үйлесушілігіне және тұтастылығына кепілдік береді;
- құрылымын қабылдауға жеңіл, табиғи ақпаратпен қамтамассыз етеді. Қордың сапалық құрылуы қорға деген «көрінбейтін» және қабылдауға жеңіл сұраныстарды жасауға мүмкіндік береді; әлбетте жаңылыс деректерді енгізу қаупі төмендейді және базаның сапалық сүйемелдеуі арта түседі;
- деректер қорының өнімділігіне деген қолданушылардың талаптарын қанағаттандырады. Ақпараттың үлкен көлемі кезінде өнімділікті сақтап қалу мәселесі басты рөлді ойнайды, бірден жобалау кезеңінің барлық кемшіліктерін «түссіздендіре отырып».

Жобаны жасаушы дерекқордың ең тиімді түрін жасап шығару мақсатында осы барлық факторларды есепке алуы керек. Дерекқорды жобалаудың негізгі міндеттерін келесідей түрде қалыптастыруға болады:

- дерекқорда барлық қажетті ақпараттарды сақтауды қамтамассыз ету;
- барлық қажетті сұраныстары бойынша деректерді алуды қамтамассыз ету;
- деректің артықтылығын және көшірілуін қысқарту;
- дерекқордың тұтастылығымен қамтамассыз ету.

Бұл міндеттердің қалай шешілетінін біз әрі қарай талқылайтын боламыз. Бірақ алдымен кейбір терминдардың анықтамасын беріп өту керек.

Пәндік сала – шынайы әлемнің бір бөлшегі, оның деректерін дерекқорда көрсеткіміз келеді. Мысалы, пәндік сала бойынша студенттердің кітапханасын, кәсіпорынның бухгалтериясын, кадр бөлімін, банкті, дүкенді және т.б. алуға болады. Пәндік сала болмыстық ретінде түсінік және деректерді иемденгендей, аз түсінікті немесе мүлдем түсініксіз болады. Егер пәндік сала ретінде қоймадағы тауардың есебін алатын болсақ, онда «жүкқұжат» маңызды болып табылады. Сол уақытта тауарды есепке алу үшін

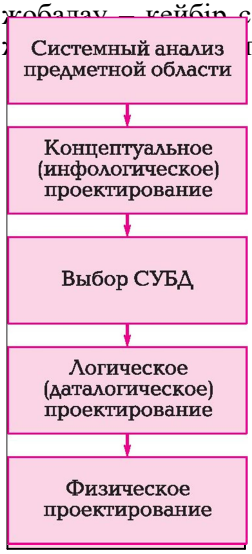
Пәндік саланың үлгісі – бұл пәндік сала туралы қандай да бір құраладар көмегімен көрсетілген, нысандандырылған білімі. Мұндай құралдар сапасы ретінде пәндік саланың мәтіндік сипаттаулары бола алады (қызметтік лауазымдық нұсқаулар, құжат айналымын сипаттау, алғашқылық құжаттар, шығыс есептер мысалы т.б.). Дерекқорды жасау кезіндегі ең ақпаратты және пайдалысы мамандандырылған графикалық жазба шартты белгілері көмегі арқылы орындалған пәндік саланы жазу болып табылады. Пәндік саланың қаншалықты дұрыс үлгіленгендігіне байланысты ақпараттық жүйенің ары қарайғы өңдеулері тікелей байланысты болады.

Дерекқорды жобалау үрдісі пәндік саланың ақпараттық құрылымын биресми ауызша сипаттауынан пәндік саланың объектілерін ресми сипаттауына ауысуы кейбір үлгілер терминінде кезектілікті ұсынады.

Жалпы алғанда жобалаудың келесідей кезекдерін атап көрсетуге болады:

1) пәндік саланың ақпараттық объектілерін ауызша сипаттау және жүйелік таладу;

2) пәндік саланың концептуалды (инфологиялық) үлгісін жобалау – кейбір семантикалық үлгілер терминіндегі объектілерді сипаттау;



3) паталогиялық немесе логикалық жобалау, яғни дерекқорды логикалық деректер үлгісімен қабылданған терминдара сипаттау;

4) физикалық жобалау, яғни қосымшаның қарқынды түрде жұмыс жасауын қамтамассыз ету үшін, дерекқордың сыртқы тасымалдаушыларға қарқынды орнатуларын таңдау.

Осы кезеңдердің әрбірінде сол немесе басқа деректер үлгісі жасалып шығады. Егер біз екінші және үшінші кезеңдер арасында біздің жобамыз қандай стандартты ДБЖ мен жүзеге асырылатыны туралы шешімді қабылдауды есепке алатын болсақ, онда дерекқорды жобалау үрдісін шартты түрде бес сәйкес кезеңдердің орындалуының көрінісі ретінде ала аламыз (4.1 сурет).

4.1 сурет. Дерекқорды жобалау кезеңдері.

Концептуальное (инфологическое) проектирование *- концептуалды (инфологиялық) жобалау;* ***Выбор СУБД*** *– ДБЖ таңдау;* ***Логическое (даталогическое) проектирование*** *– логикалық (даталогикалық) жобалау;* ***Физическое проектирование*** *– физикалық жобалау*

Концептуалдық (инфологиялық) жобалау — пәндік саланың семантикалық үлгісін құру, яғни, абстракцияның ең жоғары деңгейіндегі ақпараттық үлгі. Мұндай үлгі қандайда бір нақты ДБЖ және деректер үлгісіне бейімделусіз-ақ жасалады. «семантикалық үлгі», «концептуалды үлгі» және «пәндік сала үлгісі» терминдері синоним болып табылады. Сонымен қатар, бұл тұжырымда «дерекқор үлгісі» және «пәндік сала үлгісі» сөздері теңқұқылы қолданыла алады (мысалы, «дерекқордың концептуалды үлгісі» және «пәндік саланың концептуалды үлгісі»), мұндай үлгі шынайылықтың бейнесі болғандай, осы шынайылық үшін жобаланатын дерекқор бейнесі.

Дерекқордың концептуалды үлгісінің нақты түрі және мазмұны осы үшін таңдалған ресми аппаратпен анықталады. Әдетте ER-диаграммаға сәйкес келетін графикалық нотациялар қолданылады.

Концептуалды дерекқор үлгісі көбіне келесі негіздерді қамтиды:

- ақпараттық объектілерді және пәндік сала түсініктерін және олардың арасындағы байланысты сипаттау;
- тұтастылық шектеулерін сипаттау, яғни деректердің рұқсат етілген мәнін және олардың арасындағы байланысты талап ету.

Логикалық (даталогиялық) жобалау – нақты деректер үлгісіндегі негізде дерекқор сызбасын жасау, мысалы, деректердің реляциялық үлгісі. Реляциялық деректер үлгісі үшін логикалық үлгі - байланыс сызбасының жиынтығы, әдетте бірінші кілттерді көрсетумен, сонымен қатар сыртқы кілтті ұсынатын байланыстар арасындағы «байланыстар». Концептуалды үлгінің логикалық үлгіге ауысуы ереже бойынша ресми ережеге сай жүзеге асырылады. Бұл кезең белгілі бір деңгейде автоматтандырылады. Логикалық жобалау кезеңінде нақты деректер үлгісінің мамандандырылуы есепке алынады, бірақ ДБЖ арнайлылығы есепке алынбауы мүмкін.

Физикалық жобалау – дерекқорды басқарудың нақты жүйесі үшін дерекқор сызбасын жасау. Нақты бір ДБЖ өзгешілігі дерекқордың атаулық объектілеріне шектеулер қоя алады, қолдауға ие болған деректер түріне шектеу және т.б. қолдана алады. Сонымен қатар нақты бір ДБЖ өзгешілігі физикалық жобалау

Дерекқорды жобалау және өңдеу аумағы бойынша пәндік саланы және декрекқорды үлгілеудің түрлі құралдары қолданылады, тіпті бір нақты жүйе шеңберінде түрлі бағыттағы үлгілердің тұтас жиынтығы қажет. Ары қарай осы кезеңдердің барлығын кеңірек қарастырамыз.

4.2.

Пәндік саланы талдау кез-келген ақпараттық жүйені жасауды негізге алады және оның өңдеуінің бір бөлігі болып табылады. Бұл кезеңде қолданушылардың ақпаратқа деген қажеттіліктері анықталады, өз кезегінде ол болашақ жүйе үшін дерекқордың құрылымын және мазмұнын алдын-ала анықтайды. Пәндік сала – шынайы объектілердің кейбір жиынтығы. Бұл объектілердің әрқайсысы белгілі бір қасиеттерге (атрибуттарға) ие. Пәндік саланың объектілері арасында түрлі мазмұндық мәндегі байланыстар орнауы мүмкін. Ақпараттық жүйе үшін дерекқорды жасай отырып, қолданушы ақпаратты түрлі белгілері бойынша реттеуге тырысады. Бұл қажет болған жағдайда деректердің қажетті жиынтығын шығару үшін қажет – белгілердің қалаулы үйлесуімен таңдауды алу. Мұны жүзеге асыру тек қана құрылым деректері мүмкін болған жағдайда, яғни, деректерді ұсыну әдісі туралы келісімдерге жауап бергенде мүмкін болады.

Пәндік сала үлгісі зерттеліп отырған пәндік сала құрылымын және/немесе қызметін қайталайды және бұл салада дәлме-дәл болуы керек. Пәндік сала үлгісі бойынша ерекше рөлді қалыптастыру сатысындағы болашақ ақпаратты жүйені жасаудың талаптары ойнайды. Пәндік саланың алдын-ала үлгіленуі уақытты және жобалау жұмыстарының өткізілу уақытын қысқартуға мүмкіндік жасайды және ең қарқынды және сапалы жобаныалуға мүмкіндік жасайды. Пәндік саланың үлгілеуін жүргізуінсіз экономикалық шығындарға және келесі қайта жобаланушы жүйелерге үлкен шығын келтіруге алып келетін стратегиялық сұрақтарды шешуде үлкен көлесдегі қате жіберу мүмкіндігі жоғары.

Пәндік саланы зерттеуді өңделіп жатқан жүйе үшін жалпылай алғанда өткізіп тұрған дұрыс, оның бір бөлігі дерекқор болып табылады. Сонықтан деректер үлгісі жүйелердң барлық объектілері, өзара әрекетінің логикасы, жіберілетін ақпараттар ағыны анықталған болса ғана жасалына алады. Дерекқор жұмыс кезінде жүйемен қолданылатын, жіберілетін деректердің сақтаушысы болып табылады.

Дерекқор – бұл жүйенің негізі деп айтуғада болады, ал өз кезегінде оны жасауға өте мұқият қарау керек. Дерекқорды жасау кезінде өте үлкен қателер оның құрылымын жеткіліксіз ойластырғанда және пәндік саланы зерттеуден басталатын жобалаудың кезеңдерін сапасыз орындау себебінен болады.

Жобалау кезеңі кезінде жүйені жасаушылардың арасындағы секілді, пәндік саланың мамандары арасындада, тапсырыс беруші және т.б. арасында өзара түсінушілікке қол жеткізу маңызды, себебі әркімнің жобаға деген өз көзқарасы бар. Осылайша жұмыс объектілердің, жүйенің маңызды қызметінің, ақпараттық ағымының және олардың өзара байланысының жүйесінің кезеңдік бөлінуіне алып келеді.

Қатысушылардың белгілі бір мәселеге қатысты зерттемелері бойынша көзқарастары сәйкес келуі мүмкін, алайда оларды ұсыну формалары түрліше болуы ықтимал, ол бір жобаға қатысты біріккен жұмыстың асқынуына алып келеді. Бұл жағдайдағы маңызды құрал жобалық шешімді ұсынудағы бірдей тілді – үлгілеу тілін қолдану болып табылады. Жобаның барлық қатысушыларына бір-бірін түсінуге мүмкіндік беретін («бір тілде сөйлеу») тағайындау жүйесін, үрдістерді сипаттау тәртібін, объектілерді, құбылыстарды және олардың өзара байланысын анықтаған жөн.

Үлгілеу тілі – бұл графикалық нотация жиынтығы, жобалау үрдісі кезінде үлгілерді сипаттау үшін қолданылады. Нотация үлгілерде қолданылатын графикалық объектілердің жиынтығын ұсынады және үлгілеу тілінің синтаксисі болып табылады. Бір жағынан үлгілеу тілі түсінікті қолданушылармен жобалаушылардың шешімін жасауы керек, екінші жағынан, жобаны жасаушыларға жобалық шешімнің жеткілікті түрде ресми және келісілген құралын ұсыну қажет, тұтастық жүйесін түзетін бағдарламалық жиынтық түріндегі жүзеге асыруларға жататын.

Тәсілдемелердің жиынтығы және бірдей нотацияны қолдану өте маңызды, ол тек қана пәндік саланы үлгілеу кезеңінде ғана емес, сонымен қатар бағдарламалық жүйені жасаудың келесі кезеңдеріндеде маңзыды.

Пәндік саланы зерттеу оған ағылып келуші үрдістердің тікелей бақылауынан, жүйеде таралған құжаттарды зерттеуден, сонымен қатар осы үрдістерге қатысушылардан сұхбат алудан тұрады.

Пәндік саланың үлгісін жасау кезінде кейбір шекаралар

үрдісін қиындатпас үшін босатылады.

Пәндік саланың зерттеуін жүргізудегі нәтижесі жүйелік талаптардың тізімі, мамандандырылуы, ақпараттық ағымдары және олардың сипаттамалары болуы керек. Өте жиі осы үшін пәндік саланы сипаттаудың стандартты әдістері DFD (деректер ағымының диаграммасы),UML (үлгілеудің жүйеленген тілі) қолданылады.

4.3. ТҰЖЫРЫМДАМАЛЫҚ ҮЛГІ

Тұжырымдамалық үлгі — бұл қарастырылып отырған пәндік саланың мәндік құрылымы болып табылатын, көптеген түсініктердің анықтамасы және олардың арасындағы байланыс. Тұжырымдық үлгі болашақ дерекқордың бастапқы түптұлғасы болып табылады, бірақ нақты бір ДБЖ байланыспай-ақ құрылады.

Пәндік саланың тұжырымдық үлгісін өңдеу үрдісі шығармашылық болап табылады және қалыптастыру үрдісіне түсуі қиын. Тұжырымдық үлгіні құрастыру үшін пәндік саланы және және оның семантикасын жақсы білу керек, оның ақпаратының логикалық өзара байланысын түсіну қажет.

Пәндік саланың тұжырымдық үлгісін өңдеу жобаланушы жүйенің қолданушыларын қанағаттандырудың талдауына негізделеді. Пәндік саланы сипаттау үшін құрылымдарымен және сипаттарымен бірге қолданылатын осы түсініктердің түрлері бойынша, жағдайы, сол саланың белгісі және ондағы үрдіс ағынының заңдылығымен жіктелуі өзара байланысқан түсініктер қатарынан тұрады деп айтуға болады.

Пәндік сала үлгісіне келесідей талаптар қойылады:

- нысандандыру, пәндік сала құрылымының бір реттік сипатын қамтамассыз етуші;
- үлгіні бейнелеуге графикалық құралдарды қолдану негізінде тапсырыс берушілерге және жасаушыларға түсініктілігі;
- жүзеге асырушылық, ақпараттық жүйенің пәндік сала үлгісін физикалық жүзеге асыру құралының болуын айтушы;
- үлгілеудің жеңілдігі (үлгіге түрлі себептермен жаңа объектілерді

- пәндік сала үлгісін қарқынды жүзеге асыруды бағалау мүмкіндігін белгілі бір әдістер және ерекше көрсеткіштерімен бірге қамтамассыз ету.

Пәндік сала үлгісінің белсенді болуының басты себебі өңдеріліп жатқан дерекқордағы қызметтік толықтыруымен бекітіледі.

Тұжырымдық үлгі пәндік саланың тек ғана мамандарға оңай «оқытылатын» емес қалыптастырылған сипаттауларға ие болуы керек. Және бұл сипаттаманың мағынасы кең болғаны соншалық, дерекқор жобасын қайта жұмыс істету тереңдігі және нақтылығын бағалауға болатындай болуы керек, әрине бұрындары айтылғандай нақты бір ДБЖ байланысқан болмауы керек. ДБЖ таңдау – бұл жеке міндет, оның оңтайлы шешімі үшін қандай да бір ДБЖ бейімделуінсіз болатын жобаға ие болу қажет.

1970 – 1980 жылдары дерекқорды тұжырымды жобалау әдебиеттерінде «инфологиялық жобалау» және «инфологиялық үлгі» терминдарымен белгіленді. Тұжырымдық үлгіні құру сызбасы

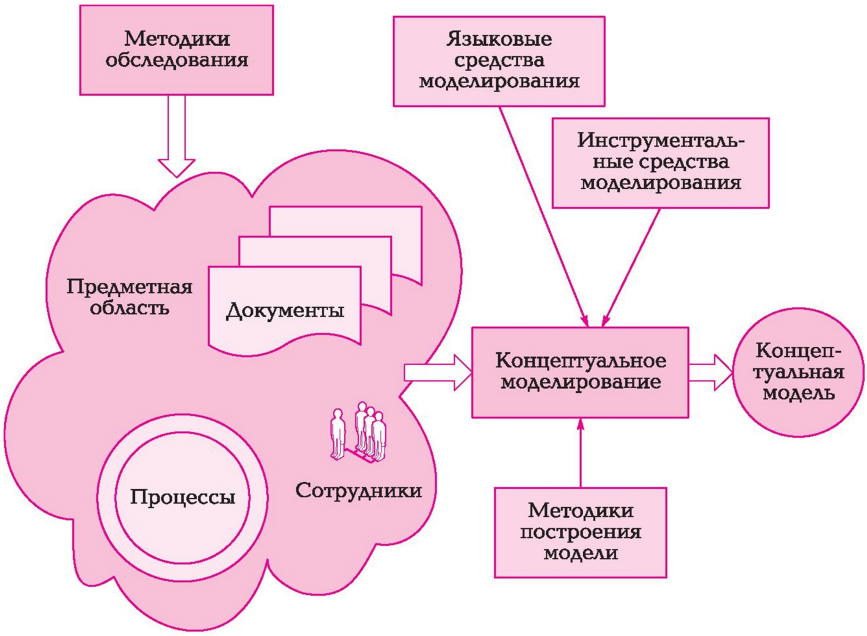


Рис. 4.2. Схема построения концептуальной модели

Методики обследование *– тексеру әдістері;* ***языковые средства моделирования****–үлгілеудің тілдік құралдары;* ***Инструментальные средства моделирования*** *– үлгілеудің инструменталды құралдары;* ***предметная область*** *– пәндік сала;* ***документы*** *– құжаттар;* ***процессы*** *– үдерістер;* ***сотрудники*** *– қызметкерлер;* ***концептуальное моделирование*** *– концептуалды үлгілеу;* ***методики построения модели*** *– үлгіні құру әдістемелері.*

Рис.4.2 Схема построения концептуальной модели – 4.2-сурет. Концептуалды үлгіні құру схемасы

Көбіне дерекқордың тұжырымдық үлгісі келесілерді қамтиды:

- ақпараттық объектілерді сипаттау немесе пәндік сала түсініктерін және олардың арасындағы байланысты сипаттау;
- тұтастылық шектеуін сипаттау, яғни деректердің рұқсат етілген мәнісін және олардың арасындағы байланысты талап ету.

Тұжырымдық жобалау бәрінен бұрын дерекқор үлгісіндегі пәндік саланың семантикасын ұсыну мүмкіндігін байланысты. Дерекқор технологиясы саласындағы мамандарды үнемі семантиканың үлгілердегі болуы мәселесі қызықтыратын. Жетпісінші жылдары семантикалық үлгі деп аталатын деректер ұлгісінің бірнеше түрлері ойластырылды. Барлық үлгілердің өз оң жақтары мен теріс жақтары болды.

1976 жылы Питер Ченмен бірге «мәндік-байланыс» (entity-relationship) деп аталатын үлгі ұсынылды. Ол пәндік саланың негізгі бизнес-тәртіптерін бейнелейтін болмысты және өзара байланысты иемденеді. Қазіргі таңда «мәндік-байланыс» үлгісі үшін жалпыға ортақ ER-үлгісіндегі қысқартылған атауы болды. Ары қарай көптеген авторлар арқылы осы үлгі секілді бірнеше нұсқалар жүзеге асырылды. Бірақ «мәндік-байланыс» үлгілерінің барлық нұсқалары бір ғана ойдан шығады – графикалық суреттеу мәтіндік сипаттауға қарағанда көрнектілеу. Қазіргі таңда ER үлгісі пәндік саланың семантикалық құрылымы үшін жалпы алғанда стандартына айналды – дерекқорды тұжырымдық үлгілеу кезіндегі стандарт.

ER үлгінің семантикалық негізін келесідей пайымдаулар құрайды:

- шынайы әлемнің бір бөлігі, яғни, өзара байланысқан объектілер жиынтығы, олар туралы сипаттаулар дерекқор көзіне жазылуы керек, мәндіктің жиынтығы ретінде де ұсынылуы мүмкін;
- мәндікті мәндік түрлері бойынша жіктеуге болады: мәннің әрбір түрі (кейбір объектілерді көрсететін) белгілі бір анықталған жіктелімкаге қатысты болуы мүмкін – мәнділік түріне, олардың

Бұл жерде тағы да мәндік түсінігінің ақпараттық табиғатын және оның пәндік сала бойынша материалдық немесе қиялдандырылған объектісімен байланысын тағы бір атап кеткен жөн. Пәндік саланың кез-келген объектісі бір бөлігі қолданбалы міндет көзқарасыбойынша маңызды болып бөлінетін құрылғыға ие. Соның өзінде, мысалы, пәндік саланы талдау және жүйелендіру үрдісі кезінде жіктелімдер бөлініп шығады.

Жіктелім – бұл атрибуттарды жинақтау түріндегі сұрастырылатын бірдей құрылғылар жиынтығындағы объектілер жиынтығы. Бұл жіктелімнің барлық объектілері үшін құрылғылардың жиынтығы (тізімі) бірдей болады, алайда бұл сипаттамалардың нақты мағынасы және әсіресе осы мәндердің қиылысуы объектіден объектіге ерекшеленетін болады, соның негізінде бір объектіден екінші объектіге деген бір нұсқаны оңтайландырады.

Пәндік саланы ұсыну деңгейінде, яғни оның тұжырымдық үлгісі, түсінік ретінде қарастырылатын объектісі (адамның санасындағы объект), «мәндік» түсінігі сәйкес келеді; материалдық әлемнің бөлігіндегі объект (және адамның санасынан мәндік түрде тәуелсіз) «мәндік нұсқадағы» түсінік жатады; объектілер жіктелуіне «мәндік түр» жатады.

Болашақта тұжырымдық үлгіде объектілердің жеке нұсқалары қарастырылмайтындықтан, ал жіктелуді біз екі сәйкес деңгейлердің түсінігін бөлмейгендіктен, яғни, «объект» және «мәнділік», «объект құрылымы» және «мәндік құрылым» түсініктерін тепе-теңдігін пайымдаймыз.

ER үлгіні көрімдеу көмегі арқылы стандартты графикалық нотация ретінде «мәндік-байланыс» диаграммасы (ER диаграмма) ұсынылды. ER-диаграмма үш құрылымдық элементтер көмегі арқылы сипатталады: мәнділік, атрибут және байланыс.

Болмыс – бір типті объектілер жіктелуі, олар туралы ақпарат қарастырылып отырған пәндік сала бойынша болмыстық мәнге ие. Әрбір болмыстың жалғыз ғана түрдегі зат есіммен айтылуы керек. Болмыстың атауы үлгі шеңберінде керемет мінсіз болуы тиіс.

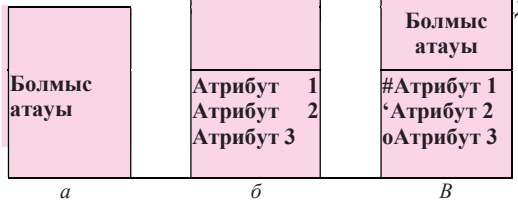
Болмыс бір типті объект ретінде үлгіленетін жіктелу көмегі арқылы «нақты теңдестірілген сала ретінде» анықталады. Осылай әрбір объект секілді құрылымдардың, болмыстың мағына жиынтығы ретінде сипатталады, болмыс атрибуттар жиынтығының жеке болмыстық нұсқасын жеке қарастыруға мүмкіндік беретіндей болып қарастырылуы керек.

Болмыс нұсқасы — сол болмыстың нақты өкілі. Болмыстың әрбір нұсқасы сол болмыстың басқа кез-келген нұсқасының ерекшелігі болуы керек(бұл талап реляциялық кестекортеж-көшірмелеріндегі талаптардан аналогиялық). Болмыс шынайы немесе абстрактілі объектілердің көптеген нұсқасын ұсынады (адамдардың, оқиғаның, саланың және т.б.).

Мысалы, «Қызметкер» нұсқасының болмысы «Иванов И.И. қызметкері» болуы мүмкін. Болмыстың нұсқалары түрліше болуы керек, яғни, осы болмыстың әрбір нұқасы үшін керемет болып есептелетін болмыс кейбір құрылғыларға ие болуы керек. Мысалы, «Қызметкер» болмысының әрбір нұсқасын бірдей деңгейде идентификациялау үшін «Табелдік нөмір» атрибуты енгізіледі, ол өз табиғатының салдарынан кәсіпорын шеңбері бойынша керемет мағынаға ие. Сәйкесінше болмыстың керемет идентификаторы ретінде атрибут, атрибуттар жиынтығы, байланыс жиынтығы немесе байланыстар және атрибуттар жиынтығы болуы мүмкін, болмыстың кез-келген нұсқасын тосы түрдегі басқа болмыстан ерекшелейді.Жоғарыда аталып кеткендей, пәндік саланың объектілі-байланысқан көрінісін сипаттайтын, нотацияның бірнеше жүйесі болады (шартты белгілері, тіледрі).Баркердің нотациясы кең таралған болып есептеледі. Осы нотацияны ұстанатын боламыз. «Мәндік-байланыс» диаграммасын құрастыру үшін реляциялық дерекқор көзінің кейбір теорияларын еске түсіру қажет болады.

Баркер нотациясының диаграммасында болмыс тік бұрыш болып, кейде дөңгеленген бұрыштарымен суреттеледі (4.3, а суреті). Әрбір болмыс бір немесе бірнеше абстарктілерге ие.

Болмыс атрибуты — бұл болмыстың кейбір құрылымы болып табылатын атаулық сипаттамасы. Атрибут атауы жекеше түрдегі зат есімде болады (сипаттаушы баяндауыш болуыда мүмкін). «Қызметкер» болмыстық атрибутының мысалы ретінде алсақ, «Табелдік нөмір», «Тегі», «Аты», «Әкесінің аты», «Қызметі», «Жалақысы» ды. Атрибут



4.3 сурет. Баркердің нотациясындағы болмысты белгілеу:

Атрибут осылайша сипаттаманың немесе құрылымның кейбір түрлерін ұсынады, көптеген шынайы немесе абстрактілі объектілермен туындаған. Құрылым табиғаты болмыстың құрылым байланысының сипаттамасы ретінде (объектісі), түрліше болуы мүмкін. Құрылымның негізгі түрлерін қарастырайық (атрибуттар).

Атрибут көпшілікті немесе жекешеленген болуы мүмкін – яғни, құрылымды белгілейтін атрибут бір уақытта бірнеше мағынаға ие болады немесе сәйкесінше біреуіне ғана, бірақ жалғыз мағына ЖСН.

Атрибут қарапайым (қолданбалы міндет көзқарасы бойынша келешектегі бөлінуіне жатпайтын) немесе құрылымдық – егер оның мағынасы қарапайым құрылымдардың мағынасынан тұратын болса. Мысалы, «Туған жылы» құрылымы қарапайым болады, ал «Мекен-жай» құрылымы – құрамдық, «Қала», «Көше», «Үй» секілді қарапайым құрылымдарды иемденеді. Қажетті атрибут детализация деңгейін қалай анықтауға болады? Мысалы, егер тапсырмада клиенттің мекен-жайын құажттарда көрсету талап етілсе, бірақ толық мекен-жайдың қосымша мекен-жайы, яғни қалалар, үйлер, пәтерлер талап етілмесе, онда біз барлық мекен-жайды қарапайым құрылым ретінде қабылдай аламыз және ол толықтай сақаталады, ал қолданушы дерекқордағы символдардың толық қатары ретінде ғана ала алады. Егерде біздің міндеттерімізде мекен-жайды құраушы бөліктерді талдау болатын болса, мысалы, клиент орналасқан қалалар, онда бізге қаланы деректің жеке элементі ретінде бөліп қарастыруға тура келеді. Тек осы жағдайда қолданушы оған рұқсат ала алады және орындай алады, мысалы, нақты бір қалада ғана тұратын клиент туралы сұранысты іздестіру, мысалы, Санк-Петербург. Алайда егер қолданушыға клиенттің толық мекен-жайы қажет болатын болса, мекен-жай бойынша сәйкес ақпаратты жеке салада сақтаған жөн, ол мысалы, «Қысқартылған мекен-жайы» деп аталуы мүмкін. Мұндай жағдайда дерекқорда әрбір клиент үшін «Қала», «Қысқартылған мекен-жай» сақталуы мүмкін.

Кейбір жағдайларда қорлық және өндірістік құрылымды айыра білген маңызды. Мысалы, «Жеткізуші» оларға жоба бойынша қойылатын «Қойылатын құрылғылардың ортақ саны» деген құрылғының санын есептеумен шығарылатын құрылымға ие бола алады. Егерде болмыстың барлық нұсқасы үшін міндетті болып табылмайтын кейбір құрылғылардың болуы шартты деп аталады. Мысалы, барлық қызметкерлер бірдей «Білім деңгейіне» ие емес.

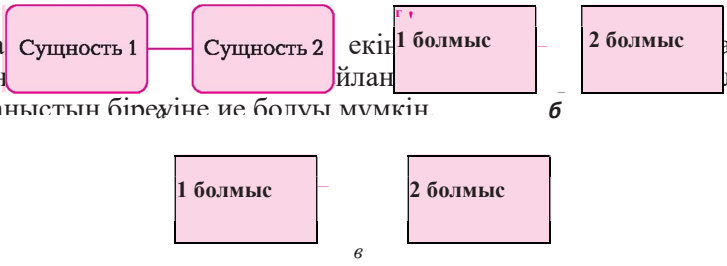
Мысалы, «ЖСН» құрылымы статикалық болып табылса, ал «Мекен-жай» - динамикалық. Құрылым анықталмаған бола алады, егерде ол динамикалық болып табылса, бірақ оның алғашқы мағынасы белгіленбесе.

Атрибут нұсқасы — болмыстың нақты нұсқасының анықталған сипаттамасы, атрибут мағынасы (мысалы, «түс» - бұл атрибут, ал «жасыл» - атрибут нұсқасы).

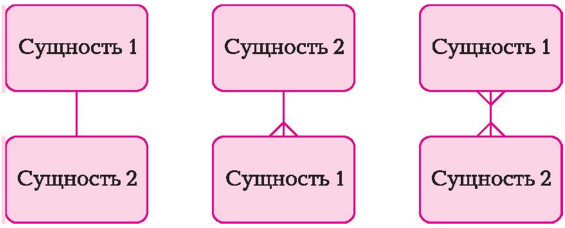
Болмыстың алғашқылық кілті – атрибуттардың артық емес жиынтығы, жиынтықтағы мағынасы болмыстың әрбір нұсқасы үшін керемет болып табылады. Артық еместік кілттің құрамынан кез-келген атрибутты өшіріп тастау оның ерекшелігінің бұзылуына алып келетіндігінде. Алғашқылық кілт болмыстың әрбір нұсқасын ерекше идентификациялау үшін арналған, яғни, объектілерді идентификациялауға болатын, белгілердің жиынтығын ұсынады. Кілттік атрибуттар тізімнің басында белгіленеді және «#» символымен көрестіледі (4.3, в суреті).

Құрылымның бұл пәндік сала дерегінде объектінің бізге қызығушылық танытпау жағдайыда болуы мүмкін. Осыған байланысты ER-үлгілерде ешқандай құрылымы жоқ және тек өзінің идентификаторымен байланысты объектілердің болуы әбден мүмкін.

Атрибуттар кілттік және сипаттаушылық болып бөлінеді. Кілттік атрибуттар – алғашқылық кілт құрамына кіретін атрибуттар. Басқа атрибуттар сипаттаушылық деп аталады. Сипаттаушылық атрибуттар міндетті және міндетті емес болады. Міндетті атрибуттар әрбір болмыс үшін үнемі нақты мағынаға ие, міндетті еместері – анықталмаған болуы ықтимал. Міндетті және міндетті емесе сипаттаушы атрибуттар «*» және «о» символдарымен белгіленеді.



4.4-сурет. Байланыс үлгісі:



а *б* *в*

4.5-сурет. Баркер нотациясындағы байланыстардың түрлі типтерін белгілеу: ***а*** — «біреуі біреуіне»; ***б*** — «біреуі-көпшілікке»; ***в*** — «көпшілікке-көпшілік»

Міндетті емес байланыс – егер бір болмыстың кез-келген нұсқасы басқа болмыстың жоқ дегенде бір нұсқасымен байланыста болса (яғни, біреуден кем емес) (4.4, 4-сурет). Міндетті емес байланыс бір болмыстың нұсқасы басқа болмыстың бір немесе бірнеше нұсқасымен байланысты болуы мүмкін дегенді білдіреді, мүмкін бір нұсқамен байланыста болмауыда мүмкін (4.4, б, суреті). Байланыс түрлі ұштарымен түрлі үлгіде болуы мүмкін (4.4, в суреті).

Әрбір байланыс солдан оңға оқылған секілді, оңнан солғада оқыла алады. Әрбір болмыс үлгінің басқа болмысымен байланысын орната алады. Байланыстың үш түрі айқындалады:

- «біреуі-біреуіне» (1*1);
- «біреуі-көпшілікке» (1*л);
- «көпшілік-көпшілікке» *(n*m)*(4.5-сурет).

«Біреуі-біреуіне» түріндегі байланыс алғашқы болмыстың бір нұсқасы екінші болмыстың бір нұқасымен байланыста дегенді білдіреді. Мұндай байланыс бір болмыстың екіге дұрыс бөлінбеуін куәландырады (алайда кей кездерде байланыстың мұндай түріне тоқталады, егерде деректердің бір бөлігін «құпия сақтау» керек болса). «Біреуі-көпшілікке» түріндегі байланыс бірінші болмыстың әрбір нұсқасы екінші болмыстың бірнеше нұсқасымен байланыста дегенді білдіреді. «Көпшілік-көпшілікке» түріндегі байланыс бірінші болмыстың әрбір нұсқасы екінші болмыстың бірнеше нұсқасымен байланыста дегенді білдіреді және керісінше. Байланыстың мұндай түрі үлгіні өңдеудегі ерте кезеңдерінде рұқсат етілетін байланыстың уақытша түрі болып табылады. Болашақта мұндай байланысты екі «біреуі-біреуіне» секілді байланыспен аралық болмысты көрсету арқылы алмастыру қажет.

4.6-суретБаркер нотациясындағы қауымдық болмысты белгілеу

Сущность - болмыс

Тәуелді болмыс жүйенің басқа болмысына тәуелді деректерді ұсынады, сол себептен ол үнемі басқа болмыстармен байланыста болуы керек.

Қауымдастықты болмыс екі немесе оданда көп болмыс қарым-қатынасымен қауымдасатын деректерді ұсынады. Әдетте болмыстың бұл түрі «көпшілікке-көпшілік» байланысына рұқсат беру үшін үлгіде пайда болады (4.6-сурет).

Егер болмыс нұсқасы толығымен өзінің кілттік атрибуттарымен толықтай иденфикацияланатын болса, онда болмыстың толық идентификациясы туралы айтылып жатыр деген сөз. Тіпті болмаған жағдайда, болмыс идентификациясы болмыспен байланысқан атрибуттарды қолданумен жүзеге асырылады, ол байланыс жолында бөлшек ретінде қалыптасады (4.7-сурет).

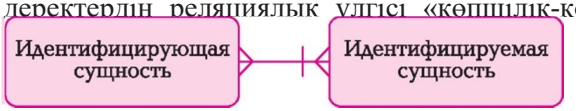
Тұтастылықтың кейбір шектеулері ER-диаграммасында белгіленеді, басқалары табиғи тілде сипатталады.

Мысал ретінде дерекқордың тұрақты клиенттерге тауарды жеткізу және сату үрдісін сипаттайтын кішігірім түрін қарастырамыз. Дерекқор «Көтерме сауда қоймасы» автоматтандырылған ақпараттық жүйесімен жобаланады.

Пәндік сала талдауынан шыға отырып, «Тауар», «Жеткізуші» және «Сатып алушы» болмыстарын көрсетуге болады.

Бірден «Сатып алушылар көп тауарды иемдене алады» және «Тауарлар көптеген сатып алушылармен иемделінеді» деген екі болмыс арасында бірден байланыс пайда болады. Олардың арасындағы қатынас «көпшілік-көпшілікке» түріне жатады (4.8-сурет).

Алайда деректердің реляциялық үлгісі «көпшілік-көпшілікке»



4.7 сурет. Баркер нотациясындағы идентификацияны басқа болмыс арқылы белгілеу

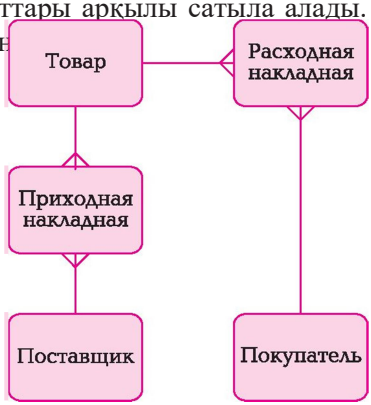
Идентифицирующая сущность - сәйкестендіруші болмыс

Идентифицируемая сущность - сәйкестенуші болмыс

4.8-сурет. ER-диаграмманың бірінші нұсқасы

Объектілер арасындағы байланысты орнатайық. Бір сатып алушы тауарды бір рет ғана алмауы мүмкін, сол себептен «Сатып алушы» және «Шығын жүкқұжаты» объектілері арасында «көпшілікке-біреу» байланысы орнайды. Тауардың әрбір атауы «Тауар» және «Шығын жүкқұжаты» объектілері арасындағы нәтижеден «біреуі-көпшілікке» байланысы орнайтын келісімдерде бірнеше рет қатыса алады. «Тауар» және «Кіріс жүкқұжаты» болмысының байланысы аналогиялық түрде орнайды.

Болмыстың атрибуттарын талдап өтсек. Әрбір жеткізуші және сатып алушы заңды тұлға болып табылады және атауларға, мекенжай, банктік реквизиттерге ие болады. Әрбір тауар өлшем бірлігімен сипатталатын атауларға, бағаларға ие. Әрбір жүкқұжаты ерекше нөмірге, шығару күніне, саны және бағасы бар тауарлар тізімімен, сонымен қатар жүкқұжат бойынша ортақ суммаға ие. Сатып алушылар шығын жүкқұжатын ала отырып, тауарларды сатып алады, оған саны туралы және алынған тауар бағасы туралы деректер енгізіледі. Әрбір сатып алушы бірнеше жүкқұжатын ала алады. Әрбір жүкқұжат бір сатып алушыға есептеледі. Әрбір жүкқұжат жоқ дегенде бір тауарға ие болуы керек («бос» жүкқұжаттың болуы мүмкін емес). Әрбір тауар өз кезегінде бірнеше сатып алушылармен бірнеше жүкқұжаттары арқылы сатыла алады. Талқылаудың аналогиялық байланысын болмысы арасындағы байланысты анықтау үшін алуға болады. Сатып алушы бір уақытта жеткізушіде бола алады, сол себептен бұл екі объект «Контрагент» деген бір болмысқа бірігеді.



4.9-сурет. ER-диаграмманың аралық нұсқасы

Дерекқорды жасаудағы маңызды міндет атрибуттарды анықтау, ол болмыстың әрбір нұсқасын анықтайды, яғни алғашқылық кілттің көрінісі. «Тауар» кестесі үшін тауардың атауы бірінші кілт қызметі бола алмайды, себебі түрлі типтегі тауарлар бірдей атауға ие болуы мүмкін.

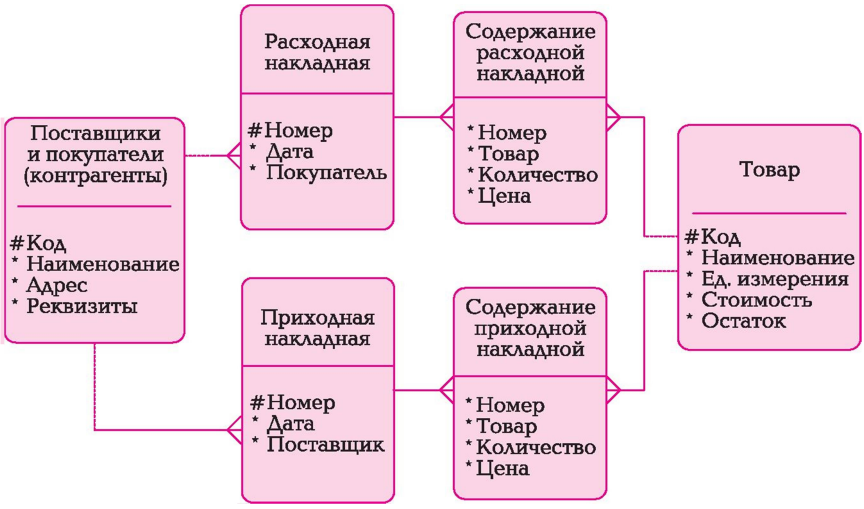
Товар *– тауар* ***Расходная накладная*** *– шығыс жөнелтпе құжаты*
Приходная накладная *– кіріс жөнелтпе құжаты*
Поставщик *– жеткізіп беруші* ***Покупатель*** *– сатып алушы*

Сол себептен «Код» алғашқылық кілтін енгіземіз. «Код» бұл мысалы, тауардың артикулын түсінуге болады. Дәл осылайша «Контрагент» кестесіндегі алғашқы кілт ретінде қызмет ете алмайды. «Кодтың» алғашқылық кілтін жеткізушілер және сатып алушылар үшін енгізелік, әрбір контаргентті анықтайтын, ол арқылы куәлік нөмірін, салық төлеушінің жеке сәйкестендіру нөмірін немесе кез-келген басқа атрибутты түсінуге болады. «Шығын жүккұжаты» және «Кіріс жүккұжаты» кестесі үшін алғашқылық кілт «Нөмір» аймағы болып есептеледі, себебі нөмір бірден әрбір жүкқұжатты анықтайды. Алғашқылық кілт ретінде бір ғана аумақты таңдамай, кейбір аумақтар жиынтығын таңдауға болушы еді, бірақ үлгілеу үрдісінің иллюстрациясы үшін қарапайым алғашқылық кілттермен шектелейік.

Енді мұның барлығын диаграммаға енгізуге болады. Осылайша, анықтаулардан кейін диаграмма келесідей түрде болады (4.10 сурет).

Бұл диаграмма өңделіп жатқан жүйенің деректер ағымының диаграммасында, барлық шығарылған деректерді алу мүмкіндігі көзқарасында тексерілуі тиіс.

«Объект» және «құрылым» түсініктері қатысты болып



4.10-сурет. ER-диаграмманың соңғы нұсқасы

Поставщики и покупатели (контрагенты) *–жеткізушілер мен сатып алушылар (контрагенттер);* ***Наименование, адрес, реквизиты*** *– атауы, мекенжайы, деректемелері;* ***Номер, дата, покупатель*** *– нөмір, күні, сатып алушы;* ***Номер, дата, поставщнк*** *– нөмір, күні, жеткізуші;* ***содержание расходной (приходной) накладной*** *– шығыс (кіріс) жөнелтпе құжатының мазмұны;* ***Номер, товар, количество, цена*** *– нөмір, тауар, саны, бағасы;* ***Товар, код, наименование, ед. измерения, стоимость, остаток*** *– Тауар, код, атауы, өлшем бірлігі, құны, қалдығы*

Баспа туралы басқа ешқандай ақпарат сақталмайды; бұл белгі бойынша ешқандай арнайы өңдеулер жүргізілмейді. Мұндай жағдайда «баспаның» жеке объектісін бөлудің қажеті жоқ, керісінше оны сай келуші объектінің құрылымы ретінде санауға болады – кітаптар. Егерде пәндік салада баспа туралы қосымша ақпарат көрсетілетін болса, мысалы, олардың мекен-жайы, телефон және т.б., онда «баспаны» жеке объект ретінде қарастырған жөн.

Жалпы жағдайда жеке объекті ретінде келесідей ұсыныстарды ерекшелей отырып ER-үлгісін көрсетуге болады.

ER-үлгісіндегі жекелік объекті ретінде қандайда бір олардың құрылымын тиянақтау және бір байланыстан көп жерде қатысатын болмысты белгілеген жөн.

Күмән туындаған кезде жеке объектіні жасау туралы шешім қабылдаған жөн, себебі болашақта ол үлгілердің аз істерін талап ететін болады. Сандық сипаттардың қандай да бір объектінің құрылымы болатынын ескерген жөн, және кей кезде – жекеленген объектілер. Мысалы, туған жылы жекеленген объект ретінде қарастырыла алмайды.

ER-үлгіде осы пәндік салада айтылып жатқанның барлығы болуы керек (кіріс құжаттарында, шығыс құжаттарында және т.б.). Толық ER-үлгіні құрғаннан кейін сақталып отырған көрсеткіштің құрамын анықтау керек. ER-үлгіден логикалық үлгіге ауысу тек сақталушы көрсеткіштер үшін ғана өндірілуі керек.

ER-диаграмманы құрудың тағы бір мысалын қарастырайық. «»

Автоматтандырылған ақпараттық жүйедегі деректер үлгісін құру керек деп ойлайық. Оны құрастырудың мақсаты абитуренттерді қабылдауды ұйымдастырушылар болсын делік. Оны жасаудың мақсаты абитурентті қабылдауды ұйымдастыру, өту емтиханын өткізудің ақпаратын дайындау және қабылдау нәтижесі бойынша қажетті есепті жүргізу. Келіп түсуші деректер жүйеде абитуренттердің өтінішімен, білімі туралы құжаттармен, анықтамамен және т.б. келіп түседі.

Автоматтандырылған ақпараттық жүйедегі өндірілетін деректер үлгісінде «Абитурент» болмысы болуы керек.

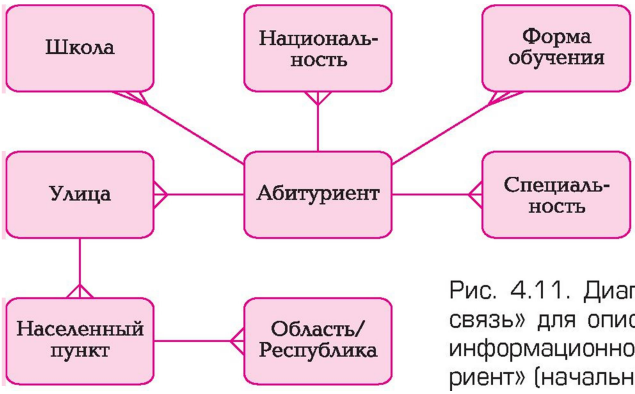
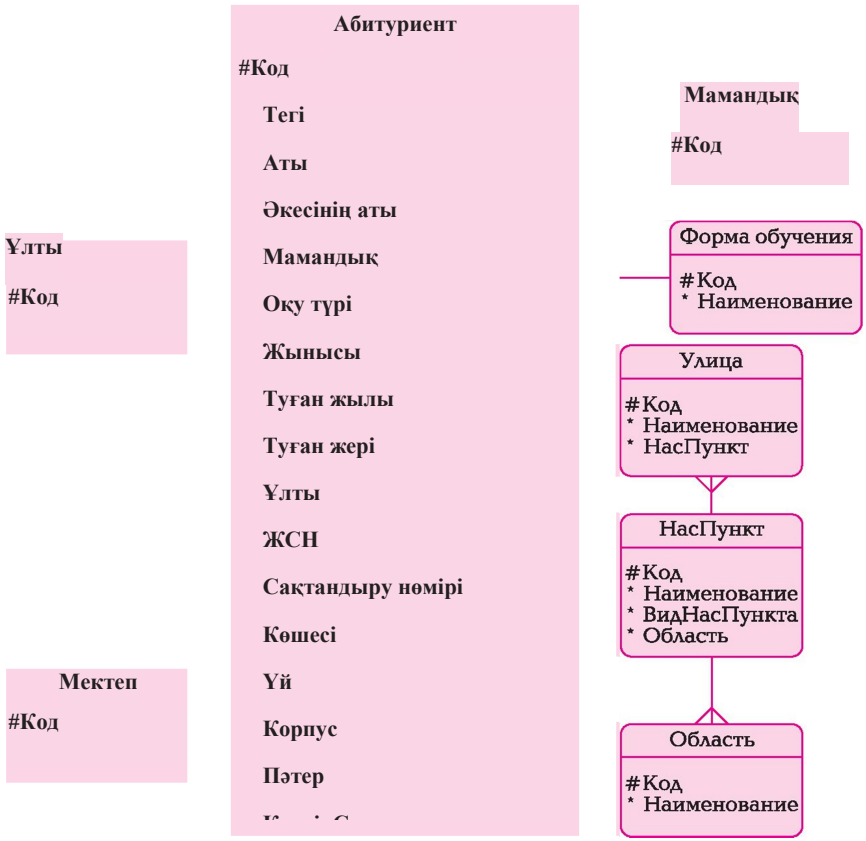


Рис. 4.11. Диаграмма «сущность-связь» для описания базы данных информационной системы «Абитуриент» (начальный вариант)



4.12 сурет. «Абитурент» дерекқорындағы ER-диаграмманың соңғы нұсқасы

4.11-сурет. «Абитуриент» ақпараттық жүйенің дерекқорын сипаттау үшін «мән-байланыс» диаграммасы (бастапқы нұсқасы).

Школа– мектеп*Абитурент* - абитурент

Область/Республика – облыс/республика *Форма обучения* – оқу түрі

Улица– көше*Национальный парк* – ұлттық аумағы

Национальность – ұлты

Специальность - мамандық

Оның ішінде «Мектеп», «Үлгі», «Оқу бөлімінің түрі», «Мамандық», «Көше», «Қала», «Аудан» атрибуттары жеке болмыстардың атауы болып табылады. Сондықтан «Тұрғын аумағы» атрибутының мағынасы толықтай «Көше» болмысының мағынасымен анықталады, ал «Облыс» атрибутының мағынасы «Тұрғын аумағы» болмысының мағынасымен анықталады.

Аталған болмыстардың арасындағы негізгі байланыстар 4.11-суретінде көрсетілген.

Келесі қадамда әрбір болмыстың атрибуттарын анықтаймыз және олардың түрін толықтырамыз (болмыстың қосымша идентификациясы басқа болмыс үшін қолданылатын атрибуттар көрсетілмейді, себебі олар сәйкес болмыста сипатталады). Енді мұның барлығын диаграммаға енгізуге болады (4.12 сурет).

Пәндік саланы суреттеуде ақпаратты нақты беруге тырысу қажет, себебі бұл болашақта дерекқор құрылымын жобалауда нақтыланған шешім беруге мүмкіндік жасай алады. Сонымен, егер «Мекен-жай», «ТАӘ» құрамдық сипаттама болатын болса, онда мұны мүмкіндігіне ER-үлгіде көрсету керек.

Тұжырымдық үлгіні құрастыру «қолмен» орындалғандай, жобалаудың автоматтандырылған құралын қолдану арқылы орындала алады – CASE құрал.

Дерекқордың логикалық құрылымын жобалау кезінде нақты бір ДБЖ ұстанатын деректер үлгісіндегі кіріс тұжырымдық үлгісінің түрленуі және пәндік салада көрсетілген алынған логикалық үлгінің белсенділігін анықтау жүзеге асырылады.

Кез-келген пәндік сала үшін логикалық үлгі бейнелеріндегі жобалық шешімдердің көптеген нұсқалары болады. Логикалық үлгіні жасау кезеңі логикалық жобалау деп аталады. Жобалаудың әдістемесі ең жақын сійкес келетін жобалық шешімді таңдауын қамтамассыз ету керек.

Тұжырымдық (инфологиялық) үлгіден логикалық (даталогиялық) үлгіге ауысу кезінде тұжырымдық үлгінің дерекқорды жобалау үшін қажетті, пәндік сала туралы барлық

Тұжырымдық үлгінің логикалық үлгіге түрленуі ресми ережелер бойынша жүзеге асырылады.

Дерекқордың логикалық үлгісі сол нақты ДБЖ-да рұқсат берілген ақпараттық бірлік терминдарында біз дерекқорды жобалайтын ортада құрылады. ДБЖ тіліндегі дерекқордың логикалық құрылымын сиапттау деректер сызбасы деп аталады.

Дерекқордың логикалық құрылымын жобалау барлық ақпараттық бірліктерді анықтау және олардың арасындағы байланысты анықтау, олардың атауын беру дегенді білдіреді; егер ақпараттық бірлік үшін түрлі типтерді қолдану мүмкін болса, онда олардың түрін анықтаған жөн. Сонымен қатар кейбір сапалық сипаттамаларды берген жөн, мысалы өрістің ұзындығын.

Деректермен басқарудың кез-келген жүйесі оған рұқсат етілген деректердің логикалық бірлігімен жүйеленеді. Сонымен қатар көптеген ДБЖ дерекқор құрылымына сандық және басқа да шектеулер қояды. Сол себептен логикалық үлгіні жасауға кіріспес бұрын ДБЖ ерекшеліктерін, мүмкіндіктерін және оны пайдаланудың мақсатты ұтымдылығын мұқият зерттеп шыққан жөн және де жобалау автоматының бар құралдарын талдауды зерттеу керек.

Алайда логикалық жобалау дерекқордың логикалық құрылымының жобасы болып табылады, оған нақты ДБЖ мен ұсынылатын деректерді физикалық ұйымдастыру мүмкіндіктері әсер етеді. Деректердің физикалық ерекшеліктерін білу логикалық құрылымды жобалаудағы пайдалысы болып табылады.

Дерекқорда нақты бір пәндік сала сипатталады. Сол себептен дерекқорды жобалау үрдісі пәндік саладағы алдын-ала жіктелуін қарастырады, объектілер туралы жүйеленген ақпаратты және олардың арасындағы байланысты қарастырады. Жобалық шешімдерге талап етілетін деректерді өңдеу ерекшеліктері әсер етеді. Сол себептен сәйкес келуші ақпарат белгілі бір бейнеде дерекқорды жобалаудың алғашқы кезеңдерінде талдануы және ұсынылуы керек.

Тұжырымдық үлгіде ақпараттық жүйеде таралатын барлық ақпараттар көрсетілуі керек. Бірақ бұл осының барлығы дерекқорда және болмыстың барлығы сақталу керек деген сөз емес, тіркелген тұжырымдық үлгілер нақты түрде логикалық үлгімен сипатталуы керек. Логикалық үлгіні құрмастан бұрын дерекқорда қандай

Сонымен қатар пәндік салада болатын байланыстың барлық түрлері де нақты бір логикаық үлгіде бейнелене алмайды. Осылайша көптеген ДБЖ-лар элементтер арасындағы «көпшілікке-көпшілік» байланысын ұстанбайды. Бұл жағдайда логикалық үлгіге қосымша, осы байланысты бейнелейтін, көмектесуші элемент қосылады (осылайша «көпшілік-көпшілікке» байланысы осы қайта енгізілген элементпен және шығыс элементімен шартты түрде «көпшілікке-біреу» екі байланысына бөлінеді). Бұған мысал көрсетіліп кеткен.

«Мәндік-байланыс» диаграммасының дерекқордың реляциялық сызбасына түрленудің қадамдық үрдісін сипаттайық.

1. Әрбір болмыс кестеге айналады. Болмыстың атауы кесте атауына айналады. Болмыс түрінің нұсқасы ретінде сәйкес келетін кестенің қатарлары сай келеді.

2. Әрбір атрибут сондай атаумен кесте қатарына айналады; деректерді ұсынудың нақты форматы таңдалынады. Міндетті емес атрибуттарға сәйкес келетін қатарлар анықталмаған мағынаға ие бола алады; міндетті атрибуттарға сәйкес келетін қатарлар – бола алмайды.

3. Болмыстың керемет жиынтығындағы идентификаторлар кестенің алғашқылық кестесіне айналады. Алғашқылық кілт үшін бірнеше мүмкін кереет идентификаторлар болатын болса, онда ең сай келетіні таңдалады. Егер керемет идентификатор құрамына байланыстар кіретін болса, алғашқылық кілт қатарының санына байланыстың екінші ұшында болатын, болмыстың керемет идентификаторының көшірмесі енгізіледі. Осы қатарларды атау үшін байланыс ұштарының аттары және/немесе жұптық болмыстың атаулары қолданылады.

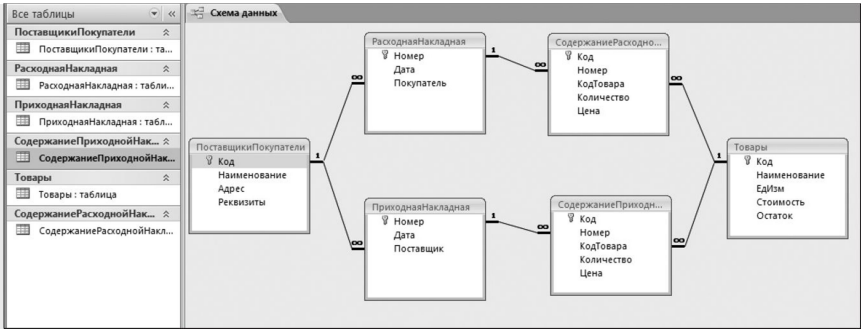
4. «Біреуіне-біреу» байланыстары сыртқы кілт. Міндетті емес байланыстар анықталмаған анықтамалардың болуына жол беретін, сыртқы кілт қатарына сәйкес келеді; міндетті байланыстар – анықталмаған мағыналарға жол бермейтін қатарға сәйкес келеді. Егер А және В екі болмысының арасында «біреуі-біреуіне» байланысы болатын болса, онда сәйкес сыртқы кілт жобаны жасаушының ниетімен А кестесінде көрсетілгендей, В кестесіндеде жарияланады.

5. А және В болмысы түрлерінің арасындағы «көпшілікке-көпшілік» байланысын қолдау үшін қосымша С кестесі екі міндетті қатарымен бірге жасалады, олардың біреуі А болмыс нұқасының керемет идентификаторға ие, ал екіншісі – В болмыс нұсқасындағы

6. Индекстер алғашқылық кілті (керемет индекс) үшін, сыртқы кілті үшін және көбіне сұраныстарды негіздеу ұсынылатын атрибуттармен жасалады. Жоғарыда аталып кеткендей, индекстерді анықтау сұрақтары және басқа көмектесуші деректер құрылымылогикалық жобалауға емес, физикалық кезеңге жатады. Әрине, тәжірибе жүзінде бұл кезеңдер көбіне уақытқа жүктеледі. Айта кететіні SQL-бейімделген ДБЖ индексінде барлық мүмкін және сыртқы кілттер үшін, тәртіпке сай автоматты жүйесі жасалады.

Тұжырымдық үлгісі жоғарыда көрсетіліп кеткен (4.13-сурет) «Көтерме сауда қоймасының» дерекқор сызбасындағы мысалын қарастырамыз. Дерекқор MicrosoftOfficeAccess 2007 ДБЖ жасалған. Логикалық жобалау сатысындағы өңделген байланыс кестелерге түрлендірілген, кілттік атрибуттар берілген және керемет индекстер, атрибуттар үшін деректер түрі анықталған, байланыстар жасалған және т.б.

Логикалық үлгіні сақтау ортасына байланыстыру үшін физикалық деңгейдегі деректер үлгісі пайдаланылады, қысқаша айтсақ, физикалық үлгі деп аталады. Бұл үлгі сақтау ортасындағы деректерді физикалық ұйымдастыру әдістерін анықтайды. Физикалық деңгей үлгісі ДБЖ ұсынылған мүмкіндіктерді есепке алумен құрылады. Дерекқордың физикалық құрылымын сипаттау сақтау сызбасы деп аталады. Дерекқорды жобалаудың сәйкес кезеңі физикалық жобалау деп аталады. ДБЖ деректерді физикалық ұйымдастыруда түрлі мүмкіндіктерге ие. Осыған байланысты нақты жүйелер физикалық жобалаудың күрделілігімен және еңбекке қабілеттілігімен, сонымен қатар орындалатын қадамдар



4.13-сурет. MicrosoftOfficeAccessДБЖ үлгісіндегі логикалық үлгі мысалы (деректер сызбасы)

Деректердің физикалық үлгісі нақты ДБЖ құралдары арқылы деректерді сипаттайды. Физикалық жобалау кезеңінде нақты бір деректер үлгісінің және нақты арнайы ДБЖ мамандандырылу есепке алынады. Логикалық деректер үлгісінің қалыптасу сатысында өңделген байланыстар кестеге ауысады, атрибуттар кестенің қатарына айналады, кілттік атрибуттар үшін керемет индекстер қолданылады, қолданылатын ДБЖ қабылданған домендер деректер түріне айналады. Логикалық деректер үлгісінде бар шектеулер ДБЖ-ның түрлі құралдары арқылы жүзеге асырылады, мысалы, индекс, тұтастылықты шектеу, триггерлер, сақталатын үрдістер арқылы. Логикалық үлгілеу деңгейінде қабылданған шешімдер кейбір шектеулерді анықтайды, олар арқылы физикалық деректер үлгісін дамытуға болады. Осы шектеулер негізінде сонымен қатар, түрлі шешімдерді қабылдауға болады. Мысалы, логикалық деректер үлгісіндегі байланыстар кестеге түрленуі керек, бірақ әрбір кесте үшін деректер қатысу жылдамдығын өсіретін қосымша түрлі индекстерді хабарлауға болады. Бұл жерде көбісі нақты бір ДБЖ-ға байланысты.

Дерекқордың физикалық жобалануы сыртқы тасымалдаушыларға дерекқорды қарқынды орнатуды таңдауын ұсынады, қосымшаның ең қарқынды жұмысы үшін.

Физикалық үлгілеу деңгейінде жүйені қызметке қосу үшін қажетті, шығарушы ресурстарды бағалау, түрін таңдау және ІМК (ішкі моделдеу кезеңдері), амалдық жүйенің түрлері және нұсқалары өндіріледі. Таңдау келесідей қолданушыларға байланысты:

- дерекқордағы шамамен алғандағы көлемі;
- деректер көлемінің өсу динамикасы;
- деректерге сұраныс жасау сипаттамасы (жеке жазбаларды шығару және жаңарту, топтар жазбаларын өңдеу, жеке байланыстарды өңдеу немесе байланыстарды біріктіру);
- сұраныстар түрі бойынша деректерге қарқынды сұраныс жіберу;
- сұраныс түріне қарай жүйелердің уақытында болуын талап ету.

ДБЖ-ны және құралдық бағдарламалық құрылымдарды таңдау дерекқор жобасын жасаудағы ең бір маңызды тұстары, себебі ол қағидалы түрде дерекқорды жобалау үрдісіне және ақпараттық жүйені жүзеге асыруға әсер етеді. Теоретикалық түрде осы таңдауды жүзеге асыру барасында ондаған факторларға назарға алған жөн. Бірақ тәжірибе жүзінде жасаушылар тек жеке ойына

CD

4.1 кесте

Деңгей	Негізгі міндеттері	Негізгі қадамдары	Базалық түсініктері
Тұжырымдық	Деректер талаптарын жинау, талдау және өзгерту	Пәндік саланы зерттеу, оның ақпараттық құрылымын зерттеу. Пәндік саланың барлық фрагменттерін анықтау, олардың әрқайсысы қолданушылық көріністермен, ақпараттық объектілермен және олардың арасындағы байланыспен, ақпараттық объектідегі үрдістермен сипатталады. Бұл кезеңді аяқтағаннан кейін дерекқор құрылымындағы нұсқалық тұжырымдық үлгіні иемденеміз – Е 11-диаграммасын	Сущности. Атрибуты. Связи
Логикалық	Деректер құрылымындағы талаптарды қайта өңдеу	ER-диаграмма кестелер жиынтығында өңделеді, олардың нормалануы жүргізіледі. Шығарда дерекқор құралымының бейімделген ДБЖ және қолданбалы бағдарлама арнайлылылығын аламыз. Бұл кезеңде түрлі ДБЖ қолданылатын дерекқорды жиі үлгілейді және үлгілердің салыстырмалы талдауын жасайды	Кестелер. Жазбалар. Деректер элементі. Жазбалар арасындағы байланыс
Физикалық	Деректерді сақтау, деректерге қол жеткізу ерекшеліктерін анықтау және т.б.	Логикалық үлгінің таңдалған бағдарламаға бейімделуі. Индекстерді таңдау және құрастыру. Протоколдау құралдарын ұйымдастыру және басқалары	Деректерді топтау. Индекстер. Қол жеткізу әдістері

- ДБЖ өнімділігінің сипаты;
- ақпараттық жүйенің ары қарай дамуы үшін қызметтік мүмкіндіктер артықшылығы;
- ДБЖ жарықтығының деректер әкімшілігі қызметкері үшін құрал деңгейі;
- эксплуатация кезіндегі ДБЖ ыңғайлығы және сенімділігі;
- ДБЖ құны және қосымша бағдарламалық қамтамассыз ету құны.

Айтылғандардан қорытынды шығара отырып, негізгі кезеңдердің және дерекқорды жобалау міндеттерінің үрдісін кесте ретінде қарастыруға болады (4.1 кесте). Кезеңдердің әрбірі деректерді ұсыну деңгейімен анықталған жұмыстың белгілі бір кезектілігімен сипатталады.

Жобалау базаны құру кезіндегі маңызы сатының бірі болып табылады, себебі дәл осы кезеңде қарқынды ақпараттық жүйені жасау үрдісіне әсер етуші маңызды стратегиялық шешімдер қабылданады.

4.5. ҚАЛЫПТАНДЫРУ ҚАҒИДАСЫ НЕГІЗІНДЕГІ ДЕРЕКҚОРЛАРДЫ ЖОБАЛАУ

Қалыптандыру теориясымен реляциялық дерекқорды жобалаудың классикалық технологиясы байланысты.

Қалыптандыру деректерді деректер таралуын ликвидациялауға арналған және басқада қарама-қайшылықтардың кестені түрге келтіру мақсатымен, деректерді қарама-қайшылықсыз және нақты өзгертуге арналған қайтадан ұйымдастыру үрдісін көрсетеді. Басқаша айтар болсақ, қалыптандыру дерекқор құрылымын түрге келтіру үшін арналған, азғантай логикалық тарылмды қамтамассыз етеді және жұмыс өнімділігін арттыру немесе кеміту немесе дерекқордың физикалық көлемін арттыру немесе кеміту мақсаты емес. Қалыптандыру мақсаты дерекқордың оптималды құрылымын алуға алып келеді.

Қалыптандыру үрдісінің ортақ бекітулері төменде келтіріледі:

- таралымның кейбір түрлерін алып тастау;
- жаңартудың кейбір аномалиясын орнату;
- дерекқор жобасын жасап шығару, шынайы әлемнің жеткілікті түрде «сапалы» болыпттабылатын, интуициялық түсінікті және келесі кеңею кезінде жақсы негіз бола алады.

- тұтастылықтың қажетті шектеулерін қолдану қажеттілік үрдісін қысқарту.

Қалыптандыру әдісі деректердің реляциялық үлгісі теориясына негізделген. Негізгі нүктесі пәндік саланы бір немесе бірнеше байланыс түрінде ұсыну болып табылады және жобалаудың әрбір қадамында «жақсартушы» құрылымына ие кестелердің кейбір жиынтықтары өндіріледі. Жобалау үрдісі байланыс сызбасының қалыптандыру үрдісін көрсетеді. Басқаша айтар болсақ, алдыңғы нормалық түрде болатын, келесі қалыптандырылған түрдің талаптарын қанағаттандыратын, байланыстардың шығарылымы жүзеге асырылады.

Реляциялық дерекқор теориясында қалыптандырылған түрлердің келесідей кезектілігі көрсетіледі:

- бірінші қалыптандырылған түрі (1 қт, 1 Normal Form, 1NF);
- екінші қалыптандырылған түрі (2 қт, 2 NF);
- үшінші қалыптандырылған түрі (3 қт, 3 NF);
- Бойс Коддтың қалыптандырылған түрі (BCNF);
- төртінші қалыптандырылған түрі (4 тқ, 4 NF);
- бесінші қалыптандырылған түрі немесе байланыстыру-жобаның қалыптандырылған түрі (5 қт, 5 NF, или PJ/NF).

Әрбір қалыптандырылған түрге шектеулердің кейбір анықталған жиынтығы сәйкес келеді. Қатынас ұалыптандырылған түрде болады, егер оған тән шектеулер жиынтығын қанағаттандыратын болса. Әдетте тәжірибеде ары қарай қарастырылатын алғашқы үш қалыптандырылған түрі ғана қолданылады.

Дерекқордың құрылымын жасау кезінде деректер таралуымен және аномалиясымен байланысты мәселелер шығуы әбден мүмкін. Деректердің таралуы деген дерекқорда бар деректерді көшіріп алу. Осылайша қарапайым (таралмаған) көшірулерді және көшірме таралуын ерекшелейді. Деректердің таралуы дерекқор кестесінің кейбір жазбаларында бір ақпараттың қайталануы байқалады.

Таралмайтын көшірілім табиғи және рұқсат етілген. Мысалы, түрлі топтардағы студенттер тізімі (4.14 сурет). Кестедегі топтар нөмірін қайталау көшіру болып есептелмейді.

Студенттік билет нөмірі	ТАӘ	Топ
101	Иванов Ф.И.	35
102	Кириллова Е.Е.	35
103	Потапов В.С.	35
104	Дудко О.В.	35
105	Таран О.С.	44
106	Ильин Г.С.	44
107	Федорова Д.С.	35
108	Медведева Ж.А.	44
109	Пушкина А. А.	44

4.14-сурет. Таралмайтын көшірілім мысалы

Топ нөмірін өзгерткен кезде (мысалы, келесі курсқа өту кезінде) оны осы топта оқитын барлық студенттер үшін өзгерту керек. Егер қандайда бір студент үшін бұл істелінбесе, деректердің сәйкес келмеуі басталады, жаңартудың аномалиясы деп аталады.

Енгізу аномалиясы кестеге жаңа жазбаларды енгізген кезде пайда болады және кестенің кейбір аумақтары үшін шектеулер қойылған орынға ие болады. Аумақ үшін болмайтын мағыналар енгізілуі мүмкін. Мысалы, мағына сұралған диапазонға кірмейді немесе аумақ маынасына қойылмаған, ол міндетті қалыпта толтырылуы керек (бос болуы мүмкін емес). Бұл енгізу аномалиясы деп аталады.

Енді деректердің таралымдық көшіруі бйынша мысал келтірейік. Студенттер тізімін және топтарды сынып жетекшінің қосымша ТАӘ көрсетумен толықтырамыз (4.15-сурет).

Бұл жағдайда деректердің көшірілуін «Сынып жетекші» қатарынан бақылаймыз. Егер кейбір жазбалардағы сынып жетекшілердің тегін «өшіретін» болсақ, біз ақпаратты мүлдем жоғалтпаймыз, себебі бұл ақпараттарды топ нөміріне сәйкес келетін басқа жазбалардан таба аламыз (4.16-сурет).

Кестені осылайша құру кезінде мәселелер қалады. Студенттің сынып жетекшісінің ТАӘ топ нөмірі бойынша басқа кесте қатарынан тауып алуға болады. Сынып жетекші көрсетілген, студенттер деректері туралы қатарды өшірген кезде осы топ

Студенттік билет нөмірі	ТАӘ	Топ	Сынып жетекші
101	Иванов Ф.И.	35	Гришин С.А.
102	Кириллова Е.Е.	35	Гришин С.А.
103	Потапов В.С.	35	Гришин С.А.
104	Дудко О.В.	35	Гришин С.А.
105	Таран О.С.	44	Коровина Л. А.
106	Ильин Г.С.	44	Коровина Л. А.
107	Федорова Д.С.	35	Гришин С.А.
108	Медведева Ж. А.	44	Коровина Л. А.
109	Пушкина А. А.	44	Коровина Л. А.

4.15-сурет. Деректердің таралымдық көшірме мысалы

Бұл мәселелерден мысалы, кестелерді екі жаңа кестеге декомпозициялау жолы арқылы құтылуға болады (4.17-сурет).

Кестенің декомпозиция (бөліну) — тұтастылық деректерін қолдау үшін кестелердің бірнеше кестеге бөліну үрдісі, яғни деректер таралымын жою және аномаляи. Бір-бірімен топ нөмірі бойынша байланыскан кестелер

Студенттік билет нөмірі	ТАӘ	Топ	Сынып жетекші
101	Иванов Ф.И.	35	Гришин С.А.
102	Кириллова Е.Е.	35	
103	Потапов В.С.	35	
104	Дудко О.В.	35	
105	Таран О.С.	44	Коровина Л. А.
106	Ильин Г.С.	44	
107	Федорова ДС.	35	
108	Медведева Ж. А.	44	
109	Пушкина А. А.	44	

4.16-сурет. Таралу көшірмесі бойынша кестедегі деректерді өшіру мысалы

Студенттік билет нөмірі	ТАӘ	Топ
101	Иванов Ф.И.	35
102	Кириллова Е.Е.	35
103	Потапов В.С.	35
104	Дудко О.В.	35
105	Таран О.С.	44
106	Ильин Г.С.	44
107	Федорова Д. С.	35
108	Медведева Ж.А.	44
109	Пушкина А. А.	44

Топ	Сынып жетекші
35	Гришин С.А.
44	Коровина Л. А.

4.17-сурет. Кестені бөлу нәтижесі

Студенттің сынып жетекшісі туралы ақпаратты алу үшін бірінші кестеден оның тобының нөмірін есептейді, одан кейін топ нөмірі бойынша екінші кестеден сынып жетекшінің тегін анықтайды.

Қарастырылып отырған кестенің бөлінуі байланысты қалыптандырудың мысалы болып табылады. Осының өзінде деректер таралуы азайды, бірақ бір уақытта деректерге рұқсат алау уақыты көбейді, енді алдыңғы жағдайдағыдай бір кестемен емес, екі кестемен жұмыс жасау қажет.

Қалыптандыру түрінің әдісін пайдалана отырып дерекқорды жобалау үрдісі кезеңдік болып табылады және байланыстың белгілі бір тәртібі негізінде бірінші қалыптандырылған түрден басқа жоғары деңгейлі қалыптандыру түріне ауысуын білдіреді. Әрбір кезеңдік қалыптандырылған түр қызметтік тәуелділіктің белгілі бір түрін шектейді, дерекқор байланысы бойынша амалдарды орындау кезінде сәйкес аномалияны орнатады және алдыңғы қалыптандыру түрлерін сақтап қалады.

Жобалау барлық объектілерді анықтаумен басталады, олар туралы ақпарат осы объектілердің атрибуттарын анықтау бойынша дерекқорда сақталуы керек. Барлық объектілердің атрибуттары бір енгізу кестесіне сай келеді. Бұл кесте бірінші қалыптандырылған түрдегі талаптарына сай келеді. Кейін бұл кесте екі немесе бірнеше кестеге декомпозициялайд, олар өз кезегінде басқа кестелерге

Осылайша қалыптандыру түріне жауап беретін өзара байланысқан кестелер жиынтығы құрылады. Тәжірибеде әдетте алғашқы үш қалыптандырылған түрлері қолданылады.

Мысал үшін студенттер туралы ақпаратты сақтайтын дерекқорды өңдейміз: ТАӘ, туған жылы, тобы, сынып жетекшісі, шифр және мамандық атауы. Келтірілген кесте өте қарапайым құрылымда, тәжірибеде кесте бұданда көп деректерге ие болушы еді. Бірақ қалыптандыру үрдісін жүзеге асыруда осы ақпараттар жеткілікті. Аумақтың түрі және өлшемі бұл жерде маңызды рөлді атқармайды, сол себептен олардың атауымен ғана шектелеміз. Жасалған кестені бір кестелік дерекқор ретінде қарастыратын боламыз.

Кесте бірінші қалыптандырылған түрге сай болуы үшін келесі шарттар орындалуы керек: кесте аумақтары бөлінбейтін ақпаратты сақтауы керек.

Бірінші қалыптандырылған түрге енгізілген кестені алайық. Сонда «Студенттер» кестесінде 6 қатар болады:

- ТАӘ;
- туған жылы;
- шифр;
- мамандық;
- топ;
- сынып жетекші.

Бірінші қалыптандырылған түрге сәйкес келмеудің мысалы 4.18-суреттегі кесте бола алады, ол жерде топ нөмірі және сынып жетекшінің тегі бір аумақта сақталады.

Екінші қалыптандырылған түрге келесі талаптар қойылады:

- кесте бірінші қалыптандырылған түр талаптарын қанағаттандыру керек
- кез-келген кілттік емес аумақ кілттік аумақтармен иденфикацияланған болуы керек.

Басқаша айтар болсақ, кесте екінші қалыптандырылған түрде болады егерде, бұл кесте бірінші қалыптандырылған кестеге сай келсе және әрбір кілттік емес атрибуттар толығымен біріншілік кілттерге тәуелді болса. Кілттік емес деп байланыстың кез-келген алғашқылық кілті құрамына кірмейтін атрибуттарды айтамыз.

ТАӘ	Туған жылы	Мамандық	Шифр	Топ, сынып жетекші
Иванов Ф.И.	1998	Ақпараттық жүйелер	230401	35и, Попенко Б.С.
Кириллова Е.Е.	1998	Ақпараттық жүйелер	230401	35и, Попенко Б.С.
Потапов В.С.	1998	Ақпараттық жүйелер	230401	35и, Попенко Б.С.
Дудко О.В.	1997	Ақпараттық жүйелер	230401	35и, Попенко Б.С.
Таран О.С.	1998	Металлды қысыммен өңдеу	150412	48о, Демина Е.Е.
Ильин Г.С.	1998	Компьютерлік желілер	230111	44к, Павлова Н.И.
Федорова Д.С.	1998	Ақпараттық жүйелер	230401	35и, Попенко Б.С.
Медведева Ж.А.	1997	Компьютерлік желілер	230111	44к, Павлова Н.И.
Пушкина А. А.	1998	Компьютерлік желілер	230111	44к, Павлова Н.И.

4.18-сурет. Бірінші қалыптандырылған түрге сәйкес келмейтін кесте

ТАӘ	Туған жылы	Мамандық	Шифр	Топ	Сынып жетекші
Иванов Ф.И.	1998	Ақпараттық жүйелер	230401	35и	Попенко Б.С.
Кириллова Е.Е.	1998	Ақпараттық жүйелер	230401	35и	Попенко Б.С.
Потапов В.С.	1998	Ақпараттық жүйелер	230401	35и	Попенко Б.С.
Дудко О.В.	1997	Ақпараттық жүйелер	230401	35и	Попенко Б.С.
Таран О.С.	1998	Металлды қысыммен өңдеу	150412	48о	Демина Е.Е.
Ильин Г.С.	1998	Компьютерлік желілер	230111	44к	Павлова Н.И.
Федорова Д.С.	1998	Ақпараттық жүйелер	230401	35и	Попенко Б.С.
Медведева Ж А.	1997	Компьютерлік желілер	230111	44к	Павлова Н.И.
Пушкина А. А.	1998	Компьютерлік желілер	230111	44к	Павлова Н.И.

4.19-сурет. 1 қт сәйкес келетін кесте

жазбалардың кереметтігін қамтамассыз ету үшін кестеге кілттік аумақты енгіземіз – студенттік билет нөмірі. Осылайша кілттің мағынасы кестедегі әрбір жазбаны идентификациялайтын болады (4.20 сурет).

Бұл кестенің жазбалары маңызды көшірілген деректердің таралымына ие, сынып жетекші барлық топ үшін көрсетілетіндіктен, ал мамандық атауы – мамандықтың әрбір шифры үшін көрсетілетіндіктен. Көшірмеден тек ғана кестені бөлу арқылы ғана құтылуға болады. Кестелер 3 қт сай келеді

Үшінші қалыптандырылған түрдің талаптары келесідей:

- кесте екінші қалыптандырылған түрдің талаптарын қанағаттандыруы керек;
- кілттік аумақ бір-біріне қатысты емес.

4.20 кестеде 3 қт сәйкес келмейді себебі «Сынып жетекші» аумағының мәнісі топ нөміріне байланысты (егер студент басқа топқа ауысатын болса, топ нөмірінің өзгеруіне сынып жетекшіні де ауыстыру болады). Мамандық атауы шифрге байланысты және керісінше. Кестені үшінші қалыптандырылған түрге алып келу үшін

Нөмір	ТАӘ	Туған жылы	Мамандық	Шифр	Топ	Сынып жетекші
101	Иванов Ф.И.	1998	Ақпараттық жүйелер	230401	35и	Попенко Б. С.
102	Кириллова Е.Е.	1998	Ақпараттық жүйелер	230401	35и	Попенко Б. С.
103	Потапов В.С.	1998	Ақпараттық жүйелер	230401	35и	Попенко Б.С.
104	Дудко О.В.	1997	Ақпараттық жүйелер	230401	35и	Попенко Б.С.
105	Таран О.С.	1998	Металлды қысыммен өңдеу	150412	48о	Демина Е.Е.
106	Ильин Г.С.	1998	Компьютерлік	230111	44к	Павлова Н.И.
107	Федорова Д.С.	1998	Ақпараттық жүйелер	230401	35и	Попенко Б.С.
108	Медведева Ж. А.	1997	Компьютерлік желілер	230111	44к	Павлова Н.И.
109	Пушкина А. А.	1998	Компьютерлік желілер	230111	44к	Павлова Н.И.

4.20-сурет. 2 тқ сәйкес келетін форма

4.21-сурет. 3 тқ келтірілгеннен кейінгі дерекқор құрылымы

Специальность - мамандық

Код-код

Шифр специальности - мамандық шифрі

Наименование специальности - мамандық атауы

ФИО- ТАӘ

Үшінші қалыптандырылған түрге ауысқаннан кейін дерекқор 4.21-суретте көрсетілген құрылымға ие болады.

Нөмір	ТАӘ	Туған жылы	Мамандық коды	Топ коды
101	Иванов Ф.И.	1998	10	13
102	Кириллова Е.Е.	1998	10	13
103	Потапов В.С.	1998	10	13
104	Дудко О.В.	1997	10	13
105	Таран О.С.	1998	30	15
106	Ильин Г.С.	1998	20	14
107	Федорова Д.С.	1998	10	13
108	Медведева Ж. А.	1997	20	14
109	Пушкина А. А.	1998	20	14

Мамандық коды	Мамандық	Шифр
10	Ақпараттық жүйелер	230401
30	Металдарды қысыммен өңдеу	150412
20	Компьютер желілері	230111

Топ коды	Топ	Сынып жетекші
13	35и	Попенко Б.С.
15	48о	Демина Е.Е.
14	44к	Павлова Н.И.

4.22-сурет 3 тқ кесте

«Студенттер» кестесінде мамандық және топ туралы толық ақпараттың орнына мамандық коды және топтың коды сақталатын болады (4.22-сурет).

Қалыптандырылған түрдің талаптарына еру үнемі міндетті емес. Кестенің өсу санынан кейін дерекқор құрылымы қиындатылады және деректерді өңдеу уақыты көбейеді. Болған жағдайлар қатарында құрылымды жеңілдету үшін деректерді жартылай көшіруді ұйымдастыруға болады, бірақ олардың тұтастылығын бұзуға жол бермей және қарама-қайшылық болмауын қадағалай отырып.

Қалыптандыру ойы дерекқорды жобалау үшін маңызды болғандығына қарамастан, олар осыдан кейін әмбебап немесе дерекқор жобасының сапасын көтерудің құралы бола алады. Бұл түрлі бейнедегі қалыптандырылған қателердің көп болуымен және дерекқордағы құрылымның жетіспушілігіне байланысты. Алайда осыған қарамастан қалыптандыру реляциялық теорияның және тәжірибенің құнды жетістігі, себебі дерекқор жобасының сапасын қатты ғылыми негізделген санатын және осы сапаны жүзеге асырудағы ресми әдістерін береді. х

Қалыптандыру – бұл жай «жақсы мән» және кез-келген жиынтықты маман өзі «табиғи жолмен» тәуелділік теориясын қолданбай-ақ толықтай қалыптандырылған дерекқорды жобалай алады деген пікірлер бар. Алайда қалыптандыру болмысы жағынан дұрыс мән қағидаларын өзіне біріктіреді, ол арқылы

4.6. ДЕРЕКТЕР ҚОРЛАРЫН ЖОБАЛАУДЫҢ АВТОМАТТАНДЫРЫЛҒАН ҚҰРАЛДАРЫ

4.6.1. Негізгі түсініктер

Дерекқорлары, басқа ақпараттық жүйелер сияқты, жүйенің жобалануынан бастап, жобалау алдындағы тексерулерден бастап, жүйені жобалау, пайдалану және одан кейін - жүйені жаңғыртуды қоса алғанда, өмірлік циклінің түрлі сатыларынан өтеді.

Ірі жобаларды құру жобалауды автоматтандыру құралдарын пайдаланусыз іс жүзінде мүмкін емес.Ақпараттық жүйелердіжобалау мен құруды автоматтандыру үшін 1970-1980 жж. құрылымдық әдіснама кеңінен қолданылған: сызбалар мендиаграммалар арқылы ақпараттық жүйелер үлгілерін сипаттаудың графикалық құралдары пайдаланған. Ақпараттықжүйелерді қолмен жасағанда, мұндай графикалық үлгілерді әзірлеу және пайдалану өте көп еңбекті қажет етеді. Бұл жағдайлар CASE-құралдар деп аталатын және ақпараттық жүйелерді құру әрі қолдау үшін CASE-технологиясын жүзеге асыратын бағдарламалық-технологиялық құралдардың пайда болу себептерінің бірі болып табылады. Заманауи CASE-құралдарының қатарында құрылымдық әдіснамадан басқа жобалаудың нысандық-бағытталған әдіснама қолданылады.

CASE (Computer Aided Software Engineering) термині сөзбе-сөз компьютер арқылы бағдарламалық жасақтаманы әзірлеу болып аударылады.Қазіргі уақытта бұл термін ақпараттық жүйелердің әзірленуін автоматтандыруды білдіретін анағұрлым кең мағынаға ие болды.

Заманауи CASE-құралдары ақпараттық жүйелерді қарапайым талдау құралдарынан және құжаттаудан толық көлемді автоматтандыру құралдарына дейін жобалау үшін көптеген технологияларды қолдайтын кең ауқымды қамтиды. Оларды пайдалану жұмысты жеделдетуге және орындау сапасын арттыруға ғана ықпал етпейді, сондай-ақ CASE-құралдарын жобалаушылар тобының ұжымдық еңбекгін ұйымдастыру үшін құралдарды ұсынады, ақпараттық жүйелерді құру және/немесе сүйемелдеу үрдістерін қолдайтын, мәселен, талаптарды талдау және тұжырымдау, деректер және қосымшалар қорларын жобалау, кодты тудыру, тестілеу, сапаны қамтамасыз ету, конфигурация мен жобаны басқару сияқты бағдарламалық құралдар болып табылады.

CASE-жүйені белгілі бір қызметтік мақсаты бар және бірыңғай бағдарламалық өнім аясында орындалған CASE-құралдарының жинағы ретінде айқындауға болады.

CASE-технология күрделі жүйелерді талдау, жобалау, әзірлеу және сүйемелдеу әдіснамаларының жиынтығын құрайды және оған автоматтандырудың өзара байланысқан құралдар кешенімен қолдау көрсетіледі.CASE-технология – бұл бағдарламалық пен ақпараттық жасақтаманы жобалау және әзірлеу үрдісін автоматтандыра отырып, қағаз бен қарындашты компьютермен алмастырушы жүйелік талдаушыларға, әзірлеушілер мен бағдарламашыларға арналған құрал. Құрылымдық талдаудың әдіснамасын қолдану кезінде шектеулер қатары (түсіну күрделілігі, көп еңбекті қажет етуі және қолдану құны, жобалық ерекшеліктерге өзгерістерді енгізудің

Жобалауды автоматтандыру құралдары пайдаланылатын тіл құралдарының және тұжырымдамалық үлгіні деректер қоры үлгілеріне түрлендіру алгоритмдерінің жиынтығымен ерекшеленеді. Бұл өз кезегінде олардың ортасында үлгіні құру әдістемесіне әсер етеді.

Заманауи CASE-құралдарының көпшілігі ER-үлгісінің формализміндегі деректерді сипаттау үшін аспаптық құралдарды қамтиды.ER-үлгісі дерекқорларды немесе оның жеке сатыларын әзірлеудің толық циклін қолдайтын коммерциялық CASE-өнімдерінің маңызды санының негізі болып табылады.Сонымен бірге олардың көбі әзірленетін жүйе пән саласының тұжырымдамалық жобалау кезеңін ғана қолдап қана қоймайды, сонымен қатар, құралдарымен салынған үлгіге негізделген таңдалған ДБЖ арналған логикалық деректер қорын жобалау сатысын, мысалы, белгілі бір серверге немесе нақты ДБЖ арналған деректер қорларының сызбасын жүзеге асыруға мүмкіндік береді. Пән саласын үлгілеу бұл жағдайда компоненттердің салыстырмалы түрде аз санын, және ең маңыздысы – осындай диаграммаларды құру технологиясын қамтушы графикалық диаграммаларды пайдалануға негізделеді.

CASE-технология мен CASE-құралдарының пайда болуына дейін жобалау әдіснамасы саласында зерттеулер жүргізілген болатын. Бағдарламалау жоғары деңгейлі тілдерді әзірлеу және енгізу, құрылымдық және үлгілік бағдарламалау әдістері, жобалау тілдері және оларды қолдау құралдары, жүйелік талаптар мен ерекшеліктер сипаттамалары туралы ресми және бейресми тілдермен және т.б. жүйелі тәсілдің ерекшеліктеріне ие болды. Бұдан басқа, CASE-технологиясының пайда болуына төменде аталған факторлар да ықпал еткен:

- үлгілік және құрылымдық бағдарламалау саласындағы талдаушылар мен бағдарламашыларды даярлау;
- тиімді графикалық құралдарды пайдалануға және жобалаудың көптеген кезеңдерін автоматтандыруға мүмкіндік беретін компьютерлік өнімділікті кеңінен енгізу және үнемі арттыру;
- жобалаудың бірыңғай үрдісіне жеке орындаушылардың күш-жігерін біріктіруге мүмкіндік беретін желілік технологияны енгізу. Осы мақсатта жоба туралы қажетті ақпаратты қамтитын ажыратылатын дерекқор пайдаланылды.

Деректер қорларын жобалауда аспаптық құралдарды пайдалану ақпараттық жүйелердің өмірлік циклінің әр түрлі сатыларына әсер етеді. Белгілі бір дәрежеде зерттеу үрдісін алдын ала анықтайды

Шетелдік бағдарламалық барлық маңызды жобалар CASE-құралдары пайдалану арқылы жүзеге асырылады, ал таратылған пакеттердің жалпы саны 500 атаудан асады.CASE-жүйелер мен құралдардың негізгі мақсаты бағдарламалық жасақтаманың жобалануын кодтаудан және кейінгі әзірлеу кезеңдерінен (тестілеу, құжаттау және т.б.) бөлу, сондай-ақ бағдарламалық жүйелерді құрудың барлық үрдісін автоматтандыру немесе инжиниринг (ағылшын тілінен *engineering* - әзірлеу) болып табылады.

Бағдарламалардың әзірленуі жүйенің белгілі бір алдын ала нұсқасынан басталады.Осындай нұсқа ретінде арнайы осы үшін әзірленген прототип немесе ескірген және жаңа талаптарға сәйкес келмейтін жүйе ұсынылуы мүмкін. Соңғы жағдайда кейін пайдалану мақсатында бағдарламалық жүйе туралы білімді қалпына келтіру үшін қайталанған әзірлемені – реинжинирингті қолданады.

Қайталанған әзірлеме бағдарламалық кодтарын қарау арқылы бағдарламалық жүйенің бастапқы үлгісін құруға әкеледі. Қолда үлгі бола тұра, оны жетілдіруге, ал одан кейін әзірлеуге қайта көшуге болады. Осылайша жобалау кезінде жиі жасайды. Мұндай түрдің едәуір танымал қағидаттарының бірі «кері жобалау» - Round Trip Engineering (RTE) қағидаты болып табылады.

Қазіргі уақытта графикалық тілдерді пәндік аймақты көрсетуге (тұжырымдамалық үлгіні құруға) және содан кейін таңдалған мақсатты ДБЖ ортасында автоматты түрде тұжырымдамалық үлгіден деректер үлгісіне ауысуды жүзеге асыруға мүмкіндік беретін жүйелер едәуір таратылымға ие болды. Осы түрдегі CASE-жүйелерді пайдалану тек ақпараттық жүйелердің жобалаушыларын ғана емес, сонымен қатар жүйеге тапсырыс берушілерді де біріктіреді, сондықтан жүйенің тұжырымдамалық модельдеу сатысында пайдаланылатын жеке механизмдер, атап айтқанда, шартты белгілер, барлық әзірлеушілер тарапынан сауатты қабылдануы тиіс.

Тікелей жобалауды (forward-engineering) – құрылған ER-үлгі негізінде таңдалған мақсатты ДБЖ арналған деректер қорының құрылымын алу және кері жобалауды (reverse-engineering — реверс-инжиниринг) – ER-үлгісін әрекеттегі деректер қорының негізінде құру үрдісін ажыратады.CASE-құралдары әдетте осы екі үрдісті де қолдайды. Графикалық үлгілеу тілдерін қолдану арқылы құрылған әр түрлі сызбалар нобайдың, жобалау және бағдарламалау тілінің әр түрлі режимдерінде пайдаланыла алады. Бұл деректерді үлгілеу жүйесіне де қатысты нәрсе. Сонымен қатар CASE-құралдар әдетте

низмін таңдауға ықпал етеді.

Әдетте, CASE-құралдарына бағдарламалық жасақтаманың өмірлік циклі үрдістерінің белгілі бір жиынтығын автоматтандыратын кез келген бағдарлама құралын жатқызады. Төменде CASE-құралдары ие болуы тиіс негізгі тән ерекшеліктер беріледі.

1. Ақпараттық жүйелерді сипаттау және құжаттандыру үшін, әзірлеушімен ыңғайлы интерфейс қамтамасыз етуші және оның шығармашылық мүмкіндіктерін дамытушы қуатты графикалық құралдар. CASE-технологиялар тапсырыс берушілерді қоса алғанда, жобаның барлық қатысушыларын қарапайым және анық құрылыммен көзге түсерлік компоненттерді алуға мүмкіндік беруші бірыңғай қатаң, көрнекі және интуитивтік тұрғыдан түсінікті графикалық тілмен қамтамасыз етеді. Бұл жағдайда бағдарламалар тапсырыс берушіге әзірлеу үрдісіне қатысуға, ал әзірлеушілерге – пәндік сала сарапшыларымен тілдесуге, жүйе талдаушыларының, жобалаушылар мен бағдарламашылардың басшылық алдында жобаны қорғауды жеңілдете, сонымен қатар жүйені сүйемелдеу және оған өзгерістерді енгізу жеңілдігін қамтамсыз ете отырып, әрекетті бөлуге мүмкіндік беретін (бірнеше беттік сипаттамалармен салыстырғанда оңай қолданылатын) қарапайым сызбалармен ұсынылады.

2. Жобалық метадеректердің арнайы ұйымдастырылған қоймасын (репозиторияны) қолдану. CASE-технологиясының негізі жоба туралы қол жетімділік құқықтарына сәйкес әзірлеушілер арасында бөлінуі мүмкін барлық ақпаратты сақтауға арналған жоба дерекқорын пайдалану болып табылады. Репозиторийдің мазмұны тек әр түрлі ақпараттық нысандарды ғана емес, сондай-ақ олардың компоненттері арасындағы қатынастарды, сонымен қатар осы компоненттерді пайдалану немесе өңдеу ережелерін қамтиды. Репозиторий құрылымдық диаграммаларды, экрандар мен мәзірлердің анықтамаларын, есептердің жобаларын, деректер сипаттамаларын, өңдеу логикасын, деректер үлгілерін, бастапқы кодтарды, деректер элементтерін және т.б. сақтауы мүмкін.

3. Ақпараттық жүйелерді әзірлеу үрдісінің басқарылуын қамтамасыз ететін CASE-құралдарының жекелеген компоненттерін кіріктіру. Репозиторийдің негізінде CASE-құралдарын кіріктіру және әзірлеушілер арасында жүйелік ақпаратты бөлу жүзеге асырылады. Сонымен бірге репозиторийдің мүмкіндіктерін интеграцияның бірнеше деңгейі қамтамасыз етеді: барлық құралдар бойынша ортақ пайдаланушылық интерфейс, құралдар арасында

4. Ұжымдық әзірлеме мен жобаны басқаруды қолдау. CASE-технологиясы желіде жұмыс істеу мүмкіндігін, жобаның кез-келген фрагменттерін даму және/немесе модификациялау үшін экспорттау мен импорттауды, сондай-ақ жоспарлауды, бақылауды, басшылық етумен өзара әрекеттесуді, яғни жобаларды әзірлеу және сүйемелдеу үрдісінде қажетті қызметтерді қамтамасыз ете отырып жоба бойынша топтық жұмысты қолдайды. Бұл қызметтер сонымен қатар репозиторий негізінде де жүзеге асырылады. Атап айтқанда, репозиторий арқылы қауіпсіздікті бақылау (қолжетімділікті шектеу және артықшылықтар), нұсқалар мен өзгерістерді бақылау және т.б. жүзеге асырылуы мүмкін.

5. Макет жасау (прототиптеу) - болашақ жүйенің макеттерін (прототиптерін) тез құру мүмкіндігі, бұл тапсырыс берушінің әзірлеудің бастапқы кезеңінде өз талаптарына сәйкестігін бағалауға мүмкіндік береді.

Барлық жобалық құжаттама репозиторий қорында автоматты түрде түрленеді.CASE-технологиясының шексіз артықшылығы құжаттама әрдайым ағымдағы істер жағдайын көрсетуінде жатыр, себебі жобаның кез-келген өзгерісі автоматты түрде репозиторийде көрсетіледі.CASE-технологиясы жобаны автоматты түрде жобаның әзірлеудің бастапқы кезеңдеріндегі толықтығы мен дәйектілігін тексеру мен бақылауды қамтамасыз етеді, бұл тұтастай әзірлеудің табысына әсер етеді. Барлық CASE-жүйелерде дерекқорды әзірлеу үрдісін құжаттау үшін дамыған құралдар әзірленді, автоматты есеп генераторлары дерекқор нысандарының толық сипаттамасымен және олардың өзара байланыстарымен графикалық түрде және дайын стандартты баспа есептері түрінде дерекқор жобасының ағымдағы күйі туралы есеп дайындауға мүмкіндік береді, бұл жобаның орындалуын айтарлықтай жеңілдетеді. ER-үлгісі үшін бірыңғай жалпы қабылданған белгілер жүйесі жоқ және әр түрлі CASE-жүйелер әртүрлі графикалық белгілерді қолданады, бірақ бірін түсінгенде, басқа белгілерді де түсіну оңай.

4.6.2. CASE-технологиялар жіктелімі

Барлық заманауи CASE-құралдарды негізінен түрлері және санаттар бойынша жіктеуге болады.Түрі бойынша жіктелу CASE-құралдарының функционалдық бағдарын көрсетеді. Санаттар бойынша жіктелу орындайтын қызметтеріне қарай бірігу дәрежесін анықтайды.

CASE-құралдарын пайдалану құрылатын жобалардың сапасын, ал іс жүзінде ірі корпоративтік жүйелерді құру кезінде сөзсіз жақсарта алады.

Алайда, CASE-құралдарын пайдалану жобалаушыны тек жалпы сипатты ғана емес, сонымен қатар логикалық жобалаудың егжей-тегжейін түсінуден босатпайды.

CASE-құралдарды жіктеу кезінде келесі белгілерді қолданады:

- өмір циклінің кезеңдеріне бағдарлау;
- қызметтік толықтығы;
- пайдаланылған үлгі түрі;
- ДБЖ-дан тәуелсіздік дәрежесі;
- қолайлы платформалар.

Ең жиі қолданылатын белгілер бойынша CASE-құралдарының жіктелуін қарастырайық.

CASE-құралдарының келесі негізгі түрлері өмірлік циклдың кезеңдеріне бағдарлаумен ерекшеленеді:

- пәндік сала үлгілерін құрастыруға және талдауға арналған талдау құралдары;
- жоба сипаттамаларын жасауды қамтамасыз етуші талдау және жобалау құралдары;
- негізгі ДБЖ үшін дерекқорларының деректерін үлгілеуді және сызбаларды әзірлеуді қамтамасыз етуші дерекқорларын жобалау құралдары;
- қосымшаларды әзірлеу құралдары, мысалы.

CASE-жүйесін және құралдарды қызметтік толықтығы бойынша шартты түрде келесі түрлерге бөлуге болады:

- өмірлік циклдің бір немесе бірнеше сатыларында нақты міндеттерді шешуге арналған жүйелер;
- ақпараттық жүйелердің бүкіл өмірлік циклын қолдайтын және жалпы репозиториймен байланысты интеграцияланған жүйелер.

CASE-жүйелерін қолданылатын үлгілер түріне қарай шартты түрде үш негізгі түрге бөлуге болады:

- құрылымдық. Тарихи жолмен бірінші құрылымдық CASE-жүйелері құрылымдық және модульдік бағдарламалау, құрылымдық талдау және синтез әдістеріне негізделген;
- нысанды-бағытталған. 1990 жж. басынан бастап нысанды-бағытталған әдістер мен CASE-жүйелер кеңінен қолданылады. Олар әзірлеу мерзімін қысқартуға, сондай-ақ ақпараттық жүйелер қызметінің сенімділігі мен тиімділігін арттыруға мүмкіндік береді;
- біріктірілген. Біріктірілген инструменталдық құралдар бір

1) тәуелсіз жүйелер – нақты ДБЖ құрамына кірмейтін дербес жүйелер түрінде жеткізіледі;

2) ДБЖ-гекіріктірілген - әдетте ДБЖ құрамына жататын дерекқор форматын қолдайды (дерекқорлардың басқа да форматтарын қолдау мүмкін).

Бүгінгі таңда ресейлік бағдарламалық жасақтама нарығында көптеген дамыған CASE-құралдары бар. Сонымен қатар, нарықта үнемі жүйенің ішкі тұтынушылары үшін жаңа, сондай-ақ жаңа нұсқалар мен жүйелерді үлгілеу пайда болады. Әрі қарай оларға қысқаша шолу жасалады.

4.6.3. CASE-жүйелеріне шолу

AllFusionERwinDataModeler.Осы CASE-құрал дерекқорларын жобалау және құжаттау үшін арналған. Ол дерекқорларын, қоймалар мен дәректер көрме-сөрелерін құруға, құжаттауға және сүйемелдеуге мүмкіндік береді.Деректер үлгілері деректер құрылымын көзбен шолуға көмектеседі, бұл деректерді күрделілігі, дерекқор технологиялары мен орналастыру ортасы ретінде кәсіпорын әрекеттерінің осындай аспектілерін ұйымдастыру, меңгеру және басқару үшін тиімді үрдісті қамтамасыз етеді.

AllFusion ERwin Data Modeler дерекқорды, дерекқор әкімшілерін, жүйе талдаушыларын, дерекқорды жобалаушыларды, әзірлеушілерді, жоба жетекшілерін әзірлейтін және пайдаланатын барлық компанияларға арналған.AllFusion ERwin Data Modeler корпоративтік өзгерістер үрдісінде, сондай-ақ жылдам өзгеретін технология жағдайларында деректерді басқаруға мүмкіндік береді.

AllFusion ERwin Data Modeler (ERwin) көмегімен күрделі деректер құрылымдарын көрнекі бейнелеуге болады. Қолдануға ыңғайлы графикалық орта дерекқорларын әзірлеуді жеңілдетеді және жоғары сапалы және жоғары өндірісті транзакциялық деректер қорын және деректер қоймаларын құру мерзімін азайта отырып, еңбекті көп қажет ететін көптеген міндеттерді автоматтандырады. Аталған шешім деректер қорларының әкімшілері мен әзірлеушілерінің өзара бірлескен жұмысын қатмамасыз ете отырып, ұйымның коммуникациясын жақсартады, сондай-ақ түсіну және қызмет көрсету үшін ыңғайлы форматта деректердің кешенді активтерінің көрнекі түсінікті арттырады.

Sybase компанисының Power Designer. Power Designer құрамына келесі модульдер жатады: Process Analyst, Data Analyst, Application Modeler.

Деректер ағындарымен және деректер қоймаларымен байланысты деректер элементтерін (атауларын, түрлерін, форматтарын) бейнелеуге мүмкіндік бар. Бұл элементтер жобалаудың келесі кезеңіне беріледі, бірақ деректер қоймалары мәндерге автоматты түрде түрленуі мүмкін.

Data Analyst– «мән-байланыс» үлгісін құруға, оның негізінде реляциялық құрылымды автоматты түрде генерациялауға арналған құрал. «Мән-байланыс» үлгісіне арналған бастапқы деректер Process Analyst модулінде құрылған DFD-үлгілерінен алынуы мүмкін. ER-диаграммаларда тек қана бинарлы байланыстырға рұқсат беріледі, байланыстарда атрибуттардың тапсырмасына қолдау көрсетілмейді. SQL тілінің диалекттері шамамен 30 реляциялық ДБЖ үшін қолдау көрсетеді, осымен бірге кестелер, түсініктер, индекстер, триггерлер және т.б. кіріктірілуі мүмкін. Нәтижесінде дерекқордың жобаланған сызбасын әзірлеуші SQL-сценарий (CREATE командаларының бірізділігі) құрылады. ДБЖ-мен ODBC интерфейсі арқылы байланыс орнату мүмкіндігі де бар. Басқа мүмкіндіктер: үлгінің дұрыстығын автоматты түрде тексеру, дерекқор өлшемінің есебі, реинжиниринг (әрекеттегі дерекқорларына арналған үлгілі диаграммаларды құру) және т.б.

Application Modeler- Data Analyst құрылған реляциялық үлгілердің негізінде деректерді өңдеудің программалардың прототиптерін автоматты түрде кіріктіруге арналған құрал. Visual Basic, Delphi, сондай-ақ PowerBuilder, Uniface, Progress сияқты «клиент-сервер» архитектурасында әзірлеу жүйелері және т.б. жүйелерге арналған код алынуы мүмкін. Кодты кіріктіру шаблондардың негізінде жүзеге асырылады, сәйкесінше кіріктіру әрекетін сәйкесінше шаблонды өзгерту есебінен басқаруға болады.

Құрылған үлгілерді сақтау қызметі бұғатталған Power Designer-тің танысу үлгісін ресейлік Sybase веб-серверінен алуға болады.

Design/IDEF пакеті (Meta Software Corp.). Бұл графикалық орта кең ауқымды пайдаланудың күрделі жүйелерін жобалауға және модельдеуге арналған.

Ол жүйелік функцияларды (IDEF0/SADT) сипаттау және модельдеу әдіснамасымен, жүйеде құрылымдар мен деректер ағындарын (IDEF1, IDEF1X, ER) және жүйе тәртібін (IDEF/CPN) қолдайды. Design/IDEF пакеті өндірісті автоматтандырумен және компьютерлеумен, басқару мен бақылаумен, телекоммуникациялар мен әуеғарышпен байланысты күрделі жүйелер жобаларын және

Design/IDEF пакетінің негізгі мүмкіндіктерін анағұрлым жете қарастырайық.

1. Графиканы ұсыну: Design/IDEF стандартты және пайдаланушы объектілерді құруды, объектілерді біріктіру және басқаруды, графикалық объектілердің және мәтіннің атрибуттарын таңдауды қамтитын жылдам және жоғары сапалы графика.Қосымша Design/IDEF деректерді редакциялау және модельдеу үшін қажетті мүмкіндіктер жүзеге асырылған: «резеңке» түріндегі байланыстырушы сызықтарын құру, бағыттау және тегістеу доғалары және т.б.

2. Үлгінің қайшылықсыздығын қамтамасыз ету: Design/IDEF әзірлеушіге IDEF үлгісі дәл, дәйекті және оны жасаудың бүкіл циклі бойынша қайшылықсыз болатынына сенімділік беретін кіріктірілген мүмкіндіктері бар. Мысалы, қызметтік блокқа немесе үлгінің бір бөлігіндегі доғаға тиесілі мәтінді түрлендірген кезде, мәтін үлгінің барлық бетінде динамикалық түрде реттелетін болады.

3. Деректер сөздігін қолдау: Design/IDEF IDEF-үлгісінде ақпаратты сақтауға және деректер қызметі және ағындары туралы есептер құруға мүмкіндік беруші кіріктірілген Деректер сөздігіне ие.Сөздік объектілер туралы бастапқы ақпаратты анықтауға мүмкіндік береді және деректер файлдарының тұтастығын сақтаудың, қалпына келтіру мен сүйемелдеудің түрлі қызметтерінің жинағын ұсынады. Сөздік мүмкіндіктері үлкен икемділікпен ерекшеленеді және пайдаланушыға әрбір нысан үшін өлшемдердің шексіз санын енгізуге мүмкіндік береді. Лазерлі принтерде жоғары сапалы басып шығарумен бірге бұл әзірлеушіге аса жоғары талаптарға жауап беретін жобалық құжаттаманы жасауға мүмкіндік береді.

4. Есептерді әзірлеу: Design/IDEF үлгілерді қолдау және талдау үшін есептердің бес түрін қолдану мүмкіндігін ұсынады; модельдің толықтығын бақылау туралы есеп; қызметтік есеп; дуалдар туралы есеп; сілтемелер туралы есеп; IDEF-есеп. Барлық есептер компьютер экранында көрсетілуі мүмкін, редакцияланады және мәтіндік редактор арқылы басып шығарылады. Design/IDEF диаграммалар мен Сөздік туралы ақпаратты қамтитын мәтіндік файлды жасау үшін деректерді талдайды және таңдайды.Есептердегі ақпарат басқа бағдарламаларда, мысалы, электрондық кестелерде, жұмыс үстеліндегі баспа жүйелерінде және мәтіндік редакторларда пайдалану үшін экспортталуы мүмкін.

5. Ұжымдық жұмысты ұйымдастыру: Design/IDEF бір уақытта үлкен және күрделі IDEF-үлгісін құрушы әзірлеушілердің көп санды топтың жұмысын қолдайды.

Шағын үлгілер бір үлкен үлгіге оңай айналады.

Деректерді модельдеу (IDEF1, IDEF1X және ER - әдіснамалары) Design/IDEF сондай-ақ жүйедегі нақты деректерді және олардың арасындағы байланысты білдіретін ақпараттық үлгілер жасау мүмкіндігін береді.IDEF-үлгілерінде қамтылған ақпарат кез-келген дерекқорға экспортталады, ал үлгілердің өздері Design/CPN - динамикалық модельдеу және күрделі жүйелерді талдау пакетіне экспортталуы мүмкін.

Бағдарламалық жасақтама әзірлеу үшін CASE-пакеті ретінде Design/IDEF бағдарламалық өнімді әзірлеудің бірінші кезеңін қолдайды:

- жобаның талаптары мен мақсаттарын тұжырымдау - жобаланатын жүйенің не істейтінін анықтау;
- ерекшеліктерді әзірлеу - талаптардың формалды сипаттамасы;
- жоба құру - кіші жүйелерді анықтау және олардың өзара әрекеттестігі;
- жобаны құжаттау – жобаның дерекқорын құру, жобаның құрамдас бөліктерін мәтіндік сипаттау;
- жобаны талдау - жобаны толықтығы мен қайшылықсыздығына тексеру.

Design/IDEF пакетінің жұмыс нәтижесі екі бөліктен тұратын бағдарлама жүйесінің жобасы болып табылады:

1) IDEF0-диаграммалары бар иерархиялық байланысты беттерден тұратын және жүйенің барлық үлгілерін (қарапайым қызметтерге дейін) сипаттайтын жүйенің қызметтік құрылымының жобасы, жүйелер, олардың өзара байланысы, кіріс және шығыс өлшемдері;

2) жүйенің жобалық ақпараттық құрылымы - деректердің барлық құрылымдары мен қатынастарын сипаттайтын дерекқордың логикалық үлгісі. Екі жоба да жобалық дерекқорымен және құжаттамамен сүйемелденеді, толықтығына және қайшылықсыздығына тексеріледі.

Design/IDEF түрлі операциялық ортада жұмыс істейді: MS-Windows, Macintosh немесе Unix X Window System үлгілерді IBM PC құруға және бір операциялық ортадан басқасына диаграммаларды көшіруге болады.

Oracle компаниясының Designer/2000. Oracle компаниясының өнімі деректерді өңдеу қосымшаларын жасаудың барлық кезеңдерін едәуір толық қолдайды. Алайда басқа құралдарға қарағанда ол іс

толығымен Oracle Developer/2000 әзірлеу құралымен бірге оны қолдану кезінде ғана жүзеге асырылады.

Oracle Designer/2000 құрамына келесі үлгілер енгізілді:

1) пәндік саланың талдау аспаптары:

- *ProcessModeler*- ұйымның іскерлік белсенділігін талдау құралы. Ұйым құрылымының үлгісін құруға және осы модельге түрлі бөлімшелерде орындалатын қызметтер мен қызметтер арасындағы ақпараттық ағындарды қолдануға мүмкіндік береді;
- *Dataflow Diagrammer*- бұл құралда DFD-диаграммалары негізінде Process Modeler-да сипатталған қызметтер егжей-тегжейлі беріледі;
- *Entity Relationships Diagrammer*- деректерді модельдеу құралы (Dataflow Diagrammer-да анықталған қызметтермен басқарылатын «мән-қатынас» диаграммалары. Баркер шартты белгілері қолданылады);
- *Matrix Diagrammer*– қызметтер және деректер арасындағы байланысты зерттеуге арналған құрал;

2) Құрылымдардың генераторлары:

- *DatabaseWizard*- ER диаграммаларынан реляциялық құрылымдарды қалыптастырады;
- *ApplicationWizard*- қызметтердің иерархиясына негізделген түпкілікті деректерді өңдеу қосымшаларының бағдарламалық модульдерінің иерархиясын қалыптастырады;

Қосымшаны жобалау құралдары:

- *Data Diagrammer*– Баркердің шартты белгілері негізінде деректердің реляциялық құрылымдарды жете өңдеуге арналған құрал;
- *Module Structure Diagrammer*- дайын қосымшаның бағдарламалық модульдерінің құрылымын басқару құралы. Мұнда модуль түрлері (мәзірлер, экран формасы, есеп) және олардың шақыру иерархиясы анықталады;
- *Module Data Diagrammer*– пайдаланатын деректерге негізделген бағдарламалық модульдің экран интерфейсін жобалау құралы. Жасалынған модульдің бағдарламаланусыз сыртқы түрін және тәртібін өте икемді басқаруға мүмкіндік береді;

3) деректер генераторлары:

- *ServerGenerator*- реляциялық үлгілер негізінде дерекқорды жасайды;

4) Module Data Diagrammer орнатылған үлгілерге негізделген код генераторлары Visual Basic, C, Java, сонымен қатар Oracle

- *Preferences Navigator* – бағдарлама модульдерін құру үшін артықшылықты басқару құралы.Жобаға тұтастай көптеген опцияларды (мысалы, экран интерфейсінің элементтерінің сыртқы түрі) және әр модульді бөлек орнатуға мүмкіндік береді.

Oracle Designer/2000-да басқа құралдардың бір қатары бар: SQL-сұраныстарын интерактивті түрде орындау, жобаларды басқару құралдары және тағы басқалар.

БАҚЫЛАУ СҰРАҚТАРЫ

1. Дерекқорын жобалаудың негізгі кезеңдерін атаңыз.
2. Пәндік саласының үлгісі не үшін құрылады?
3. Тұжырымдамалық үлгі деп нені атайды?
4. Пәндік сала үлгісіне қандай талаптар қойылады?
5. Тұжырымдамалық жобалау кезеңінде қандай негізгі ұғымдар пайдаланылады?
6. Тұжырымдамалық жобалау кезеңінде қандай тапсырмалар шешіледі?
7. Дерекқордың тұжырымдамалық үлгісі неден құралады?
8. Тұжырымдамалық жобалаудың қадамдарын атаңыз.
9. ER-үлгінің семантикалық негізін не құрайды?
10. Мән және мәннің данасы ретінде не аталады?
11. Мән атрибуты және атрибут данасы деп не аталады?
12. Мәндер арасындағы байланыс деп нені атайды?
13. Қандай мән ассоциациялық мән болып табылады?
14. Атрибутты детализациялаудың қажетті дәрежесін қалай анықтайды?
15. Байланыстың модальділігі нені білдіреді және ол ER-диаграммасында қалай белгіленеді?
16. Мәнді толықтай сәйкестендіру нені білдіреді?
17. Деректердің логикалық үлгісі деп нені атайды?
18. Логикалық жобалау кезеңінде қандай базалық түсініктер пайдаланылады?
19. Логикалық жобалау кезеңінде қандай тапсырмалар шешіледі?
20. Логикалық жобалау қадамдарын атап шығыңыз.
21. ДҚ қызмет ету үшін қажетті есептеу ресурстарына қойылған талаптарды бағалау қандай кезеңде жүргізіледі?
22. «Мән-байланыс» диаграмасының дерекқорының реляциялық сызбасына түрленуінің типтік қадамдық рәсімін сипаттаңыз.
23. Физикалық деңгейдің деректер үлгісі не үшін қолданылады?
24. Физикалық жобалау кезеңінде қандай негізгі түсініктер пайдаланылады?

25. Физикалық жобалаудың қадамдарын атап шығыңыз.
26. Физикалық жобалау кезеңінде қандай міндеттер шешіледі?
27. Қалыпқа келтіру мақсатын атаңыз.
28. Ақпараттың шамадан тыс қайталануы несімен қауіпті?
29. Қалыпты формалардың негізгі қасиеттерін атаңыз.
30. Қандай кесте шектеулері 1 қф, 2 қф және 3 қф-қа тиесілі?
31. Қалыпты формаларған сәйкес келетін және сәйкес келмейтін кестелердің мысалдарын келтіріңіз.
32. CASE-құралдарына және CASE-технологияға анықтама беріңіз.
33. Болашағы бар CASE-жүйесінің талаптарын белгілеңіз.
34. CASE-құралдарын жіктеу белгілерін атап шығыңыз.

5 тарау

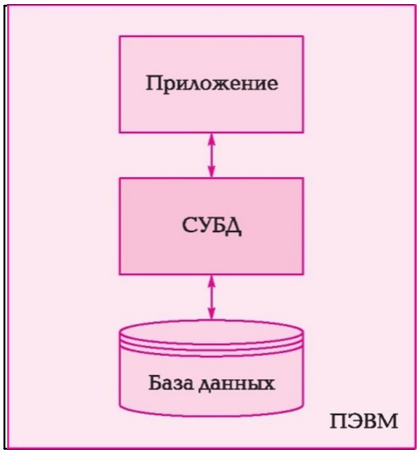
ДЕРЕККОР ТҰТАСТЫҒЫН ҚАМТАМАСЫЗ ЕТУ

5.1. ДЕРЕККОР АРХИТЕКТУРАСЫ

5.1.1. Автономиялық архитектура

Дерекқордың автономиялық архитектурасын пайдаланған кезде ДБЖ және қолданбалы бағдарламасы (қосымшасы) бір компьютерде орналасады (5.1-сурет).Ұйымдастырудың мұндай тәсілі желінің қолдауын қажет етпейді және барлық дербес жұмысқа түседі. Жұмыс келесі түрде құрылған:

- файлдар жиынтығы түріндегі дерекқор компьютердің қатты дискісінде орналасқан;
- сол компьютерде ДБЖ және ДҚ жұмыс істеу үшін қосымшалар орнатылды;
- пайдаланушы бағдарламаны іске қосады.
- қосымшамен ұсынылған пайдаланушы интерфейсін пайдаланып, ақпаратты іріктеу/жаңарту үшін дерекқорға жүгінуді бастайды;



5.1-сурет. Автономиялық архитектура

Приложение-қосымша

СУБД-ДБЖ

База данных-дерекқор

- дерекқорға барлық жүгінулердерекқор физикалық құрылымы туралы барлық ақпаратты жинайтын ДБЖ арқылы өтеді;
- ДБЖ пайдаланушының өтініштері орындалуын қамтамасыз ету (деректер бойынша қажетті операцияларды жүргізу)үшін деректерге жүгінуді бастайды;
- ДБЖ нәтижесі қосымшаға қайтарады;
- пайдаланушы интерфейсін қолдана отырып, қосымша өтініштерді орындау нәтижесін көрсетеді.

Осылайша, осы үлгіде жұмыстың бір пайдаланушылық режим жүзеге асырылады.

5.1.2. «Файл-сервер» архитектурасы

«Файл-сервер» архитектурасында дерекқорлары серверде сақталады, клиент файлдық командаларымен серверге жүгінеді, ал барлық ақпараттық ресурстарды басқару механизмі клиент компьютерінде орналасады (5.2-сурет).

Файл-серверлік дерекқорлар желі арқылы көптеген клиенттерге қолжетімді болуы мүмкін. Дерекқордың өзі бір ғана данада желілік «файл-серверде» сақталады. Жұмыс уақытында әр клиент үшін ол басқаратын деректердің жергілікті көшірмесі жасалады.Бұл жағдайда бірнеше қолданушылардың бірдей ақпаратқа бір уақытта қол жеткізілуімен байланысты өзекті мәселелер пайда болады.Бұл өзекті мәселелерді дерекқор қосымшаларын әзірлеушілер шешеді (әр кезде деректерге жүгінген сайын қолжетімділігі тексеріледі).

«Файл-сервер» архитектурасының елеулі кемшіліктері бар:

5.2-сурет. «Файл-сервер» архитектурасы

База данных - дерекқор

Клиентское предложение 1 - 1 клиенттік ұсыныс

Сұрау тек бір жазбаға қатысты болса да, дерекқордағы барлық жазбалар жаңартылады. Егер дерекқорда көптеген жазбалар бар болса, тіпті клиенттер саны аз болса да, желі мұқият жүктеледі, бұл сұрауды орындау жылдамдығына айтарлықтай әсер етеді. Артық ақпараттың үлкен көлемді желідегі айналымның нәтижесінде желідегі жүктеме күрт артады, оның өнімділігі азаяды және тұтастай алғанда ақпараттық жүйенің жұмысы төмендейді. Елеулі желілік трафик әсіресе төменгі жылдамдықты байланыс арналары арқылы «файл-серверде» дерекқорларына қашықтан қолжетімділікті ұйымдастыру кезінде әсер етеді;

- жұмысты осылайша ұйымдастыру кезінде деректердің тұтастығы туралы қамқорлық бағдарламалық клиенттерге жүктеледі. Жергілікті ДБЖ файл-серверлік жүйелер құрылатын дәстүрлі құралдардың бірі болып табылады. Мұндай жүйелер, әдетте, деректердің тұтастығын қамтамасыз ету талаптарына жауап бермейді, атап айтқанда, олар транзакцияларды (құжаттармен аяқталған операцияларды) өңдеуді қолдамайды. Сондықтан оларды қолдану кезінде деректердің тұтастығын қамтамасыз ету міндеті клиенттік қосымшаларға жүктеледі, бұл өз алдына олардың күрделенуіне алып келеді. Егер олар мұқият ойластырылмаса, қателерді барлық пайдаланушыларға әсер етуі мүмкін дерекқорға оңай енгізуге болады;
- бір пайдаланушыға дерекқорға енгізілген өзгертулер басқа пайдаланушыларға көрінбейді. Бір пайдаланушы белгілі бір жазбаны өңдесе, ол басқа клиенттер үшін бұғатталады. Жазбаларды бұғаттаумен байланысты жеке пайдаланушылардың жұмысын синхрондау қажеттілігі пайда болады;
- дерекқорын басқару әр түрлі компьютерден жүзеге асырылады, сондықтан елеулі деңгейде қолжетімділікті бақылауды, құпиялылықты сақтауды ұйымдастыру қиындатылады, бұл сонымен қатар дерекқорлардың тұтастығын қолдауды күрделендіреді.

Аталған кемшіліктерге қарамастан файлдық-серверлік архитектура қарапайымдылығымен және қолжетімділігімен назар аудартады. Сондықтан, файлдық-серверлік ақпараттық жүйелер әлі де кішігірім жұмыс топтары үшін қызығушылық тудырады және, сонымен бірге, кәсіпорын ауқымында жиі ақпараттық жүйелер ретінде пайдаланылады.

Жиі файлдық-серверлік ДБЖ-ны үстелді деп атайды. Үстелді ДБЖ салыстырмалы шағын міндеттер үшін (өңделетін деректердің шағын көлемі, пайдаланушылардың аз саны) қолданылады. Осыны

салыстырмалы түрдегі жеңілдетілген архитектураға ие, атап айтқанда, «файл-сервер» режимінде қызмет атқарады, ДБЖ мүмкін қызметтерінің барлығын қолдамайды (мысалы, транзакциялар журналы жүргізілмейді, жаңылудан кейін автоматты түрде дерекқорларды қалпына келтіру мүмкіндігі жоқ және т.б.).Дегенмен, мұндай жүйелерде қолданудың айтарлықтай кең ауқымы бар. Ең алдымен бұл мемлекеттік (муниципалдық) мекемелер, білім беру саласы, қызмет көрсету саласы, шағын және орта бизнес.Туындайтын міндеттердің ерекшелігі деректер көлемі тым үлкен еместігіне, жаңартулардың жиілігі өте жоғары еместігіне, ұйым, әдетте, бір шағын ғимаратта орналасқандығына, пайдаланушылардың саны бір адамнан 10 адамға дейін өзгеретіндігіне байланысты.Осындай жағдайларда ақпараттық жүйелерді басқаруға арналған үстелдік ДБЖ-ны пайдалану толығымен негізделген және олар сәтті қолданылуда.

Әр түрлі фирмалар тарапынан әзірленген dBase деп аталатын үйлесімді бағдарламалық жүйелер алғашқы ДБЖ-ның бірі болды.Мұндай кең таралған жүйенің алғашқысы болып dBase III-PLUS жүйесі (Achton-Tate фирмасы) танылды.Жүйені кеңінен таратуға үлес қосқан, бағдарламалаудың дамыған тілі, жаппай пайдаланушыға қолжетімді, ыңғайлы интерфейс.Сонымен бірге, жүйенің түсіндіру режимінде жұмыс істеуі орындалу сатысында төмен өнімділікті тудырды. Бұл dBase III-PLUS жүйесіне жақын жаңа жүйе-компиляторлардың пайда болуына әкеп соқты: *Clipper*(Nantucket Inc. фирмасы), FoxPro (Fox Software фирмасы), FoxBase+ (Fox Software фирмасы), *Visual FoxPro* (Microsoft фирмасы).Бір уақытта PARADOX ДБЖ айтарлықтай кеңінен пайдаланылды (Borland International фирмасы).

Соңғы жылдары Майкрософт корпорациясының Microsoft Office жиынтығы нұсқаларының тұтас қатарына кіретін Microsoft Access дерекқорды басқару жүйесі өте кең таралған (Microsoft фирмасы).

Ірі ұйымдар үшін жағдай түбегейлі өзгереді.

Мұнда жоғарыда сипатталған себептерге байланысты файлдық-серверлік технологияларды қолдану жол берілмейді. Сондықтан едәуір қымбат серверлік ДБЖ пайдаланылады.

5.1.3. «Клиент-сервер» екі деңгейлі архитектурасы

Клиент-сервердің архитектурасының ерекшелігі - сұраудың құрылымдық сұрау (Structured Query Language, SQL) тілінде қабылданатын және іздеу, сұрыптау мен ақпараттауды орындайтын арнайы дерекқор серверлерінің болуы.

Жалпы алғанда, клиент пен сервер әр түрлі компьютерлерде жұмыс істейді деп есептеледі.

Сервер - деректерді анықтау, жазу, оқу, деректерді жою, сыртқы (тұжырымдамалық) және ішкі деңгейлер сызбаларын қолдау, сұратуды орындау және диспетчерлеу, деректерді қорғау сияқты ДБЖ қызметтерін орындайтын бағдарлама.

Клиент - ДББЖ жеткізушісі немесе пайдаланушысы, сыртқы немесе ДБЖ-ға қатысты «кіріктірілген» бағдарлама. Клиент бағдарламасы сыртқы интерфейс арқылы ДБЖ компоненттеріне деректермен операциялар жасау үшін пайдаланылатын қосымша ретінде ұйымдастырылады.

Сұрауды клиенттік және серверлік бөлікке орындау үрдісін бөлу келесі әрекеттерді жасауға мүмкіндік береді:

- әртүрлі қолданбалы (клиенттік) бағдарламалары бір уақытта жалпы дерекқорды пайдаланады;
- ақпаратты қорғау, деректердің тұтастығын қамтамасыз ету, ресурстарды бірлесіп пайдалануды басқару сияқты басқару қызметтерін орталықтандыру;
- сұрау салудың параллельді өңделуін қамтамасыз ету;
- клиенттік компьютерлердің ресурстарын босату;
- мамандандырылған компьютерлер - дерекқор серверлері арқылы деректерді басқарудың тиімділігін арттыру.

«Клиент-сервер» платформасындағы дерекқор көптеген пайдаланушыларға арналған жүйелерде қолданылады.Дерекқордың контекстінде клиент пайдаланушы интерфейсін және қосымша логикасын басқарады, дерекқор қосымшалары іске қосылған жұмыс станциясы ретінде әрекет етеді. Клиент бағдарламасы пайдаланушыдан сұрауды қабылдайды, синтаксисті тексереді және SQL тілінде немесе қосымшаның логикасына сәйкес келетін басқа дерекқор тілінде дерекқорға сұрау жасайды.Содан кейін ол хабарламаны серверге жібереді, жауаптың түсуін күтеді және қабылданған деректерді пайдаланушыға көрсету үшін пішімдейді.Сұраулармен операцияларға бағытталған қуатты сервер қуаттайтын дерекқорға сұрауды қабылдайды және өңдейді, содан кейін алынған нәтижелерді клиентке кері жібереді. Мұндай өңдеу

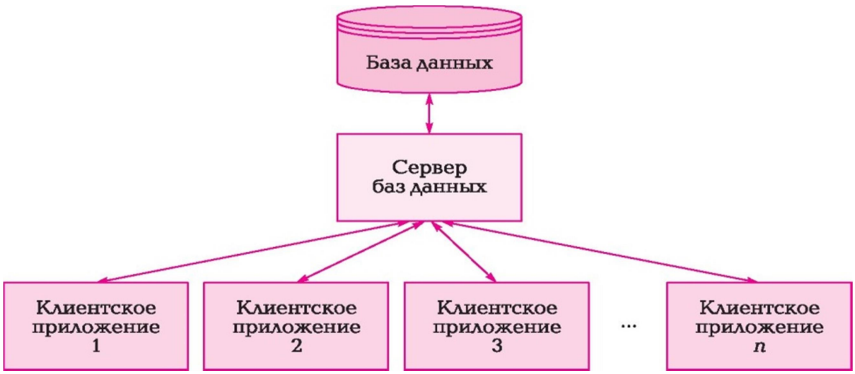
бұдан басқа параллелділік пен қалпына келтіру арқылы басқару қолдау табады.

«Клиент-сервер» архитектурасының дерекқорында қосымша өзекті мәселе туады - қосымшаны сервердің мүмкіндіктерін барынша кеңейтіп, желі арқылы мөлшері аз ақпаратты ғана жібере отырып, желіге анағұрлым аз салмақ түсіретіндей жобалау.

Сипатталған архитектура екі деңгейлі болып табылады және толық клиент деп аталады (5.3-сурет).

Қазіргі уақытта «клиент-сервер» архитектурасы жұмыс топтары мен корпоративтік деңгейдегі ақпараттық жүйелерге қосымшаларды ұйымдастыру әдісі ретінде танымал болды және кеңінен қолдана бастады.Мұндай ұйым дерекқоры серверінің мүмкіндіктерін пайдалану арқылы қосымшалардың өнімділігін жақсартады, желідегі жүктемені азайтады және деректер тұтастығын бақылауды қамтамасыз етеді.Ақпарат қауіпсіздігін арттыру барлық клиенттердің сұрау салуларын серверде орналасқан бірыңғай бағдарламамен өңдейтініне байланысты.Сервер дерекқордың барлық пайдаланушылары үшін ортақ ережелерді орнатады, клиенттің деректерге қол жеткізу режимдерін басқарады, атап айтқанда, әртүрлі пайдаланушылардың бір уақытта бір жазбаны өзгертуге тыйым салады.

Сондай-ақ, клиенттік қосымшалардың күрделілігі дерекқордың бақылануы мен оған қол жеткізуді шектеу туралы кодтың болмауына байланысты азаяды.



5.3-сурет.«Клиент-сервер» екі деңгейлі архитектурасы

База данных-дерекқор, сервер баз данных-дерекқор сервері, клиентское предложение 1 - 1 клиенттік ұсыныс

ДБЖ де желі серверінде орналасады. ДҚ жұмыс істеу үшін клиенттік қосымша орналасқан клиенттік компьютерлерден құрылған жергілікті желі қолданылады.Әрбір клиенттік компьютерде пайдаланушылар қосымшаны іске қосу мүмкіндігіне ие. Қосымша ұсынатын пайдаланушылық интерфейсті қолдана отырып, ол ақпаратты іріктеуге/жаңартуға серверде орналасқан ДБЖ жүгінуге бастамашылық жасайды. Қарым-қатынас үшін SQL сұрау салудың арнайы тілі қолданылады, яғни желі бойынша клиенттен серверге сұрау салудың мәтіні ғана жолданады.

ДБЖ өз ішінде сервердегі ДҚ физикалық құрылымы туралы барлық мәліметтерді инкапсулирлейді және сервердегі деректерге жүгінуге бастамашылық етеді, нәтижесінде барлық деректерді өңдеу серверде орындалады және сұрау салудың нәтижесі тек клиенттік компьютерге көшіріледі. Осылайша, ДБЖ нәтижені қосымшаға қайтарады.

Пайдаланушылық интерфейсті қолдана отырып, қосымша сұрау салуды орындау нәтижесін көрсетеді.

Айтылғанды есепке алатын болсақ, сервер мен клиент арасындағы қызметтердің шектелуін келесі түрде сипаттауға болады: *қосыма-клиент қызметтері:*

- серверге сұрау салуды жолдау;
- серверден алынған сұрау салулардың нәтижелерін түсіндіру;
- нәтижелерді пайдаланушыға қандай да бір түрде ұсыну (пайдаланушы интерфейсі);

серверлік бөліктің қызметтері:

- қосымшалар-клиенттерден сұрау салуларды қабылдау;
- сұрау салуларды түсіндіру;
- ДҚ-ға сұрау салуларды оңтайландыру және орындау;
- қосымшаға-клиентке нәтижелерді жолдау;
- қауіпсіздік жүйесін қамтамасыз ету және қолжетімділікті шектеу;

- ДҚ тұтастығын басқару;
- жұмыстың көп пайдаланушылық режимнің тұрақтылығын жүзеге асыру.

«Клиент-сервер» архитектурасында «өнеркәсіптік» деп аталатын ДБЖ жұмыс істейді.Өнеркәсіптік деп аталатын себебі – осы сыныптың ДБЖ-сы орта және ірі кәсіпорын, ұйым, банк

Олар дерекқорлардың физикалық сипаттамаларын басқарады, ДҚ түрлі компоненттерінің оңтайландырылуын, бапталуын және қайта айқындалуын жүзеге асырады, жаңа ДҚ құрайды, әрекеттегілерін өзгертеді және т.б., сонымен қатар түрлі пайдаланушыларға артықшылықтарды (нақты ДҚ-ға, SQL-серверге белгілі бір деңгейіне қолжетімділік рұқсатын) береді.

«Клиент-сервер» архитектурасының негізгі қасиеттерін қысқаша «файл-сервер» архитектурасымен салыстыру бойынша келесі түрде келтіруге болады:

- әрекеттегі дерекқорларға анағұрлым кең қолжетімділік қамтамасыз етіледі;
- жүйенің жалпы өнімділігі артады. Клиенттер мен сервер түрлі компьютерлерде орналасқандықтан, олардың процессорлары қосымшаларды қатарлас орындауға қабілетті. Дерекқормен ғана жұмыс орындалатын жағдайда сервермен компьютердің өнімділігін баптау жеңілдетіледі;
- аппараттық жасақтаманың құны төмендейді.Үлкен сақтау құрылғысы бар айтарлықтай қуатты компьютерге дерекқорды сақтау және басқару сервер үшін ғана қажет, қуатты жұмыс станцияларына қажеттілік жоқ;
- желінің жүктелуі азаяды. Қосымшалар клиенттік компьютерлердегі операциялардың бір бөлігін орындайды және желі бойынша жіберілген деректердің көлемін едәуір азайтуға мүмкіндік беретін желі арқылы тек қана дерекқорға сұрау салады;
- деректердің қайшылықсыздық деңгейі артады. Сервер деректердің тұтастығын тексеруді өз бетінше басқара алады, өйткені тек сонда ғана барлық шектеулер анықталады және тексеріледі. Осымен бірге әрбір қосымша өз тексерісін орындауға мәжбүр болмайды.

Кемшіліктер қатарына аппараттық және бағдарламалық жасақтаманың жоғары қаржылық шығындарын, сондай-ақ әртүрлі жерлерде орналасқан барлық клиенттік компьютерлерде клиенттік қосымшаларды уақтылы жаңарту қиындықтарын жатқызуға болады.

Дегенмен, «клиент-сервер» архитектурасы іс жүзінде өз-өзін жақсы жағынан көрсете алды, қазіргі кезде осы архитектураға сәйкес құрылған ДҚ көптеген саны бар және қызмет атқарады.

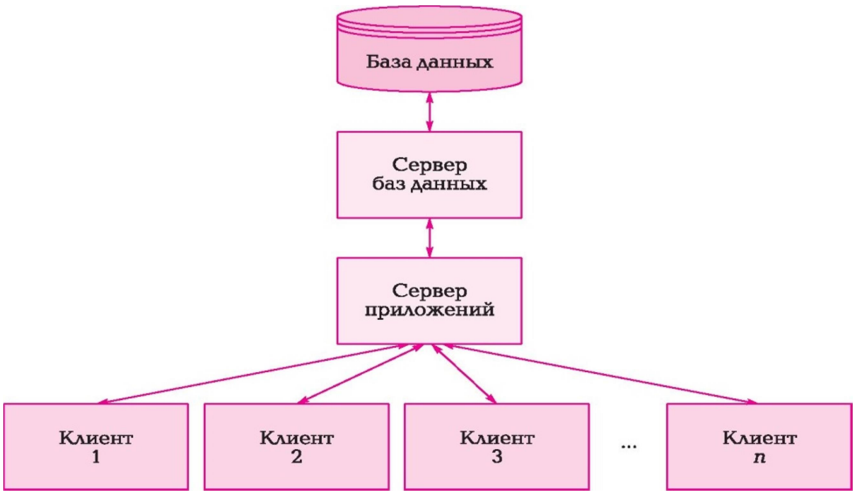
«Клиент-сервер» архитектурасының екі деңгейлі сызбалары көптеген пайдаланушылардың және логиканы шатастыратын

5.1.4. «Клиент-сервер» үш деңгейлі архитектурасы

Екі деңгейлі «клиент-сервер» архитектурасын одан әрі кеңейту «қалың» («зияткерлік») клиенттің функционалдық бөлігін екі бөлікке бөлуін жобалайды (5.4-сурет).Үш деңгейлі «клиент-сервер» архитектурасында жұмыс бекетінде «жұқа» («зияткерлік емес») клиент пайдаланушы интерфейсі тарапынан ғана басқарылады, ал деректерді өңдеудің орташа деңгейі барлық қалған қосымша логикасын басқарады. Үшінші деңгей - дерекқор сервері.Бұл үш деңгейлі архитектура кейбір орталарда - мысалы, Internet және Intranet желілері үшін анағұрлым қолайлы болып табылады, мұнда клиент түрінде қарапайым web-браузер әрекет ете алады.

Төменгі деңгейде пайдаланушылардың компьютерлерінде қызметтерді орындауға және бағдарламаны орташа деңгейге шақыру үшін бағдарламалық интерфейсті қамтамасыз етуші логиканы қарауға арналған клиенттік қосымшалар орналасқан.

Дерекқормен деректерді өңдеу логикасы операцияларды орындаушы, қолданбалы логика орындалатын қосымшалар сервері орташа деңгейде орналасқан, яғни бұл деңгей пайдаланушылар мен бөлінген дерекқорлары арасындағы деректердің алмасуын



5.4-сурет. «Клиент-сервер» үш деңгейлі архитектурасы

База данных-дерекқор, сервер баз данных-дерекқор сервері;

сервер приложений - қосымшалар сервері:

Қосымшалар сервері барлық клиенттерге қолжетімді желі торабында орналасады.

Үшінші жоғары деңгейде қосымшалар серверінен ақпаратты қабылдаушы дерекқордың қашықтықтағы арнайы сервер орналасқан. Дерекқорлар сервері деректер мен файлдық операцияларды өңдеу қызметін көрсетуге арналған.

«Клиент-сервер» үшбуынды ДБЖ жұмысын қысқаша келесі түрде сипаттауға болады:

- файлдар жинағы түріндегі дерекқоры арнайы бөлінген компьютердің (желі серверінің) қатты дискісінде орналасқан;
- сондай-ақ ДБЖ желі серверінде орналасады;
- арнайы бөлінген қосымшалар серверінде бағдарламалық жасақтама (бизнес-логика) орналасады;
- көптеген клиенттік компьютерлердің әрқайсысында «жіңішке» клиент – пайдаланушы интерфейсін жүзеге асырушы клиенттік қосымша орнатылған. Қосымша тарапынан ұсынылатын пайдаланушы интерфейсін қолдана отырып, ол қосымшалар серверінде орналасқан бағдарламалық жасақтамаға жүгінуге бастамашылық етеді.
- қосымшалар сервері пайдаланушы талаптарын талдайды және ДҚ-ға сұрау салуларды құрастырады.
- қарым-қатынас үшін SQL сұрау салудың арнайы тілі қолданылады, яғни желі бойынша қосымшалар серверінен ДҚ серверіне сұрау салудың тек мәтіні ғана жолданады;
- ДБЖ серверде орналасқан ДҚ физикалық құрылымы туралы барлық мәліметтерді өз ішінде қапшықтайды;
- ДБЖ қосымшалар серверіне сұрау салуды орындау нәтижесі көшірілетін серверде орналасқан деректерге жүгінуге бастамашылық етеді;
- қосымшалар сервері клиенттік қосымшаға (пайдаланушыға) нәтижені қайтарады;
- пайдаланушы интерфейсті қолдана отырып қосымша сұрау салуларды орындау нәтижесін көрсетеді.

Үш деңгейлі архитектураның артықшылықтары:

- қосымшалар серверіне көшірілген операциялардың бір бөлігін орындаудан дерекқорлар серверін жеңілдету;
- артық кодтан босату есебінен клиенттік қосымшалар санын азайту;
- барлық клиенттердің бірыңғай әрекеті;
- клиенттер баптауын жеңілдету – қосымшалар серверінің жалпы

кемшіліктерін жояды. Ол желіге түскен жүкті одан да көп теңдестіруге ықпал етеді. «Клиент-сервер» жүйелерінің артуымен үш деңгейлі ұйымға қажеттілік анағұрлым айқын болып келеді.

5.2. ДЕРЕККҚОР НЫСАНДАРЫ

Дерекқор нысандары оның құрылымы мен деректері туралы барлық ақпаратты қамтиды.Дерекқор нысандары сондай-ақ метадеректер деп те беріледі. Дерекқор кестелер, домендер, сақталатын рәсімдер, триггерлер және т.б. сияқты түрлі нысандардан құрылады. Төменде серверлік дерекқорлары нысандарының түсініктемелері беріледі. Осы нысандардың кейбіреулері кеңінен қарастырылады.

Кесте (table) – кез келген реляциялық дерекқордың негізгі нысаны. Кестелерде дерекқордың барлық деректері және метадеректері сақталады.Кесте - жолдардың ерікті санын қамтитын жазық екі өлшемді құрылым.Жиі жолды жазба (record) деп атайды. Кестеде бірде-бір жол болмауы, яғни бос болуы мүмкін.Бір кестенің барлық жолдары бірдей құрылымға ие. Олар бағандардан (өрістерден) тұрады. Реляциялық дерекқордағы кестеде кемінде бір баған болуы мүмкін.Әрбір бағанда нақты деректер түрі бар. Дерекқорында әдеттегі бағдарламалау тілдерінде қолданылатын деректерге ұқсас деректердің бірнеше түрлері пайдаланылады.

Домен (domain) - бағанның кейбір сипаттамаларын сипаттайтын дерекқор нысаны. Жасалатын кестенің бағандарын сипаттағанда немесе дерекқордағы кесте бағандарының сипаттамаларын өзгерткен кезде доменге сілтеме жасауға болады. Бұл жағдайда доменнің барлық сипаттамалары бағанға көшіріледі.

Индекс (index) - кестеден деректерді іріктеу нәтижелерін реттеуді тездету және/немесе деректерді іріктеу нәтижелерін жылдамдатуға арналған дерекқор нысаны.Әрбір индекс бір нақты кесте үшін жасалады.Индекс - реттелген жолдардың жиынтығы, олардың әрқайсысы индексті құрайтын өрістердің мәнін және осы өрістердің тиісті мәндерін қамтитын кесте жолына көрсеткішті қамтиды. Индекстерді белсендіруге, қайта белсендіруге, олардың сипаттамаларын жақсартуға арналған құралдар әрекет етеді. Бастапқы, бірегей және сыртқы кілттер үшін жүйе индекстерді автоматты түрде жасайды.

Генератор (generator) - жасанды бастапқы кілттің немесе кейде бірегей кілттің мәнін қалыптастыру үшін әдетте пайдаланылатын

Сақталатын рәсім (stored procedure) - SQL тілінің рәсімдік кеңейтілімінде жазылған бағдарлама (сақталған рәсімдер мен триггерлер тілі деп те аталады) және дерекқордағы деректермен әртүрлі әрекеттерді орындауға мүмкіндік беретін дерекқорының метадеректер аймағында сақталады. Сақталған рәсімдерге осы дерекқордың, пайдаланушы (клиент) бағдарламаларының сақталған рәсімдері арқылы жүгінуге болады. Сақталған рәсім сервер жағында орындалады, бұл көптеген жағдайларда желі трафигін күрт төмендетіп, пәндік сала міндеттерін шешу жылдамдығын едәуір арттырады.

Триггер (trigger), сақталатын рәсім сияқты, метадеректер аймағында сақталған және серверде орындалатын SQL тілінің рәсімдік кеңейтілуіне жазылған бағдарлама болып табылады. Алайда, триггерге тікелей қол жеткізу мүмкін емес. Кестелердегі деректердің өзгеруіне байланысты оқиғалар фазаларының бірі немесе дерекқорға қосылумен байланыстыратын және транзакциялармен жұмыс істейтін оқиғалар автоматты түрде шақырылады. Кесте оқиғалары – деректерді қосу, кесте жолын өзгерту, жолды жою. Әрекетті орындауға «дейін» (before) және әрекетті орындағаннан «кейін» (after) фазалар болып табылады.

Пайдаланушылық ерекшеліктер (exception) - дерекқормен бағадарламалар жұмыс істеу үрдісінде нақты жағдай туындаған кезде пайдаланушыға хабарларды жасауға және беруге мүмкіндік беретін реляциялық дерекқордың нысаны. Бұл дерекқордағы кез келген бұзушылықтардан немесе дерекқорда деректерді өңдеудің кез келген ерекше жағдайынан пайда болатын қате жағдай болуы мүмкін. Бұл ерекшеліктер тек сақталатын рәсімдерде және триггерлерде қолданыла алады.

Дерекқордың оқиғалары (event) – нақты дерекқормен жұмыс істеуші барлық клиенттік қосымшаларға бағытталған кейбір хабарламаларды сақталатын рәсімдерден және триггерлерден жолдау мүмкіндігі бар. Бұл құрал көптеген жағдайларда деректерді көрсетуді үндестіруге немесе клиенттік қосымшалардың өзара әрекеттесуі үшін күрделі әрекеттерді орындауға мүмкіндік береді, олар бір уақытта дерекқормен бірге жұмыс істейді.

Көрініс (view; шолу, қарау) – нақты жиі едәуір күрделі критерийлердің негізінде дерекқорының бір немесе одан да көп кестесінен деректерді іріктеу нәтижесі. Көрініс негізі – кез кел күрделіліктің SELECT операторы. Көрініс – дерекқорында іс жүзінде сақталмайтын виртуалды кесте деп айтуға болады.Көрініс –

Бұдан басқа, көріністер пайдаланушыдан қарауына арналмаған кестелердің кейбір мәндерін жасыруға мүмкіндік береді.

Пайдаланушы белгілеген қызметте (User Defined Functions, UDF) - кез келген бағдарламалау тілінде жазылған және дерекқордан тыс сақталған, бірақ осы дерекқорда сипатталған қызметтер.SQL тілінің мүмкіндіктерін және тиісті бағдарламалау тілдерін кеңейту үшін қолдануға болады. Дерекқордың деректерімен (немесе кіріс параметрлері ретінде берілген деректермен) айтарлықтай күрделі әрекеттерді жиі сипаттауға ықпал етеді.

5.3. 5.3.1. Дерекқордың тұтастығы және транзакциялар механизмі

ДБЖ-ны серверлік дерекқорлармен жұмыс істеу үшін қолдану ДҚ тұтастығы сияқты өзекті мәселенің ерекше маңыздығын тудырды.

ДҚ тұтастығы ретінде дерекқорда бар ақпараттың оның ішкі логикасына, құрылымына және барлық анық көрсетілген ережелеріне, яғни дұрыс ДҚ мазмұнының құрылымы мен қайшылықсыздығын түсінеміз.Дерекқордың ықтимал күйіне белгілі бір шектеу қойған әрбір ереже тұтастығын шектеу деп аталады.Тұтастықты бұзу, мысалы, бағдарламалық қателер немесе техникалық ақаулар арқылы туындауы мүмкін, себебі бұл жағдайда жүйе қалыпты өңдеуді немесе дұрыс деректерді бере алмайды.

Тұтастықтың екі аспектісін - жеке объектілер мен рәсімдер және тұтастай алғанда дерекқор деңгейінде қарастырайық.

Тұтастықтың бірінші аспектісі деректер құрылымы және ДБЖ тілдік құралдарының жеке операторлары деңгейінде қамтамасыз етіледі.Мұндай тұтастық бұзылған кезде тиісті оператор (мысалы, сандық өріске жолдарды енгізу әрекеті) қабылданбайды.Кейбір тұтастық шектеулерін нақты құрылымдарда көрсету қажет емес, себебі олар деректер құрылымдарына енгізілген. Алайда дерекқор жиі бір ғана емес, бірнеше рәсімдердің міндетті түрде орындалуын талап ететін тұтастық шектеулеріне ие болуы мүмкін. Дерекқор шектеулері жағдайында тұтастықты қамтамасыз етуге белгілі бір жеке рәсімдер емес, транзакциялар механизмі әрекет етеді.

Дерекқордағы деректермен және метадеректермен жүргізілетін кез келген әрекеттер белгілі бір транзакция контекстінде орындалады.

Транзакция – деректерге әсер ету тұрғысынан бөлуге келмейтін деректермен айла-шарғы жасау рәсімдерінің бірізділігі.Пайдаланушы үшін транзакция «барлығы немесе ештеңе» негізінде жүзеге асырылады.Немесе транзакция толығымен орындалады және дерекқорды бір толық (қайшылықсыздық) күйден басқа тұтас күйге ауыстырады, немесе қандай да бір себептермен транзакция әрекеттерінің біреуі, немесе жүйелік ақаулық орын алса, дерекқор транзакция басталғанға дейін болған бастапқы жағдайға, яғни транзакция кері қайтарылады.

Транзакцияның «тіршілігі» үрдісінде осы транзакцияның басқаруымен немесе, транзакция контекстінде орындалатын дерекқордағы деректермен жасалатын әрекеттер оны растағанға дейін басқа қоса жүретін үрдістерге көрінбейді. Транзакция контекстінде орындалған әрекеттер расталған жағдайда өзгерістер басқа қоса жүретін үрдістерге де көрінеді.

Транзакция түсінігі ДҚ тұтастығы сияқты түсінікпен тікелей байланыста болады. Транзакцияға жиі өзара байланысты өзгерістерді әр түрлі кестелерге енгізу бойынша әрекеттер жасалған кезде бірнеше кесте рәсімдері біріктіріледі. Бір кестеден екіншісіне жазбаларды көшіру жүзеге асырылып жатыр деп ұйғарайық. Жазба бірінші кестеден алдымен жойылып, кейін екінші кестеге енгізілсе, ақау туындаған кезде, мысалы, компьютерді электр қуатымен қамтудың үзілуіне байланысты, жазба жойылған, бірақ екінші кестеге түспеген жағдай орын алуы мүмкін. Жазба алдымен екінші кестеге енгізілген, кейін бірінші кестеден жойылған жағдайда ақау кезінде жазба екі кестеде де болу жағдайы орын алуы мүмкін. Екі жағдайда да ДҚ тұтастығы мен қайшылықсыздығының бұзылуы орын алады.

Мұндай жағдайдың алдын алу үшін бір кестеден жазбаны жою және екінші кестеге енгізу рәсімдері бір транзакцияға біріктіріледі. Бұл транзакцияны орындау кез келген нәтиже кезінде ДҚ тұтастығының бұзылмауына кепілдік береді.

Белгіленген бақылау нүктелері құрылған және транзакцияны қайтаруды транзакцияның басына емес, белгіленген бақылау нүктесіне орындау мүмкін болған жағдайда кіріктірілген

Транзакциялар ұзақ және қысқа болуы мүмкін. Дерекқормен жұмыс істеген кезде клиенттік бағдарламалар ұзақ және қысқа транзакцияны қолдануы мүмкін. Ұзақ транзакция айтарлықтай көп уақыт аралығында белсенді болып табылады. Оның контекстінде дерекқормен көптеген рәсімдер орындалуы мүмкін. Қысқа транзакцияның белсенділігі басталған сәттен бастап, оны растау немесе кері қайтару сәтіне дейін секундтық үлесті құрайды. Әдетте, мұндай транзакция үрдісінде дерекқорда өзгерістердің айтарлықтай шектеулі саны орындалады.

Ұзақ транзакциялар, әдетте, дерекқор деректерін оқу кезінде, қысқа транзакциялар - дерекқор кестелеріне өзгерістер енгізген кезде қолданылады.

Дерекқоры біреу көп пайдаланушылық жүйелерде бір уақытта бірнеше пайдаланушы немесе қолданбалы бағдарламалар жұмыс жасауы мүмкін. ДБЖ міндеті – пайдаланушылардың оқшауланuын қамтамасыз ету, яғни әрбір пайдаланушы тек өзі ғана дерекқормен жұмыс істеп жатқандығы туралы сенімді және нанымды елес жасауы.

5.3.2. Деректерге параллельді қолжетімділіктің өзекті мәселелері

ДҚ-мен бір уақытта бірнеше пайдаланушы жұмыс істесе, онда ДБЖ жеке транзакцияларды дұрыс орындап қана қоймай, ақаулардан кейін ДҚ келісілген жай-күйін қалпы келтіруге, сондай-ақ бірдей деректермен барлық пайдаланушылардың дұрыс қоса жүретін жұмысын қамтамасыз етуі тиіс.Теориялық тұрғыдан, әр қолданушы және әрбір транзакция оқшаулау сипатына ие болуы керек. Және заманауи ДБЖ құралдары осы жолмен пайдаланушыларды бір-бірінен оқшаулауға мүмкіндік береді. Алайда, транзакцияларды параллельді өңдеу кезінде өзекті мәселелер туындайды, оларды едәуір егжей-тегжейлі қарастыратын боламыз. Серверде бір дерекқормен бір уақытта бірнеше клиент жұмыс істеп жатқан кездегі классикалық өзекті мәселелер ретінде төменде аталған мәселелерді атауға болады:

- бастапқы мәніне негізделе отырып бірдей жолды бірнеше пайдаланушы өзгерткен кезде соңғы өзгеріске байланысты өзекті мәселе туындайды; онда кейбір деректер жоғалып кетеді, себебі әрбір кейінгі транзакция алдыңғы жасаған өзгерістерді қайта жазады. Бұл жағдайдан шығу дәйекті түрде өзгерістерді енгізуде жатыр;
- логикалық дұрыс күйге түсер алдында деректердің көп өзгеруін

деректерді өзгерту кезінде басқа пайдаланушы оны оқыса, ол логикалық тұрғыдан бұрыс ақпаратты алуы мүмкін. Мұндай өзекті мәселелерді жою үшін деректерді оқу әрекетін барлық өзгерістер аяқталғаннан кейін орындау қажет;

- қайталанбаған оқылым мәселесі транзакция бойынша бірдей мәліметтерді бірнеше оқу нәтижесі болып табылады. Бірінші транзакцияны орындау барысында басқасы деректерге өзгерістер енгізуі мүмкін, сондықтан оны қайта оқығанда, бірінші транзакция тұтастықтың немесе логикалық сәйкессіздіктің жоғалуына әкелетін деректердің басқа жиынтығын алады;
- фантомдарды оқу мәселесі бір транзакция кестеден деректерді таңдағаннан кейін және екіншісі біріншісі аяқталғанша жолдарды енгізгеннен немесе жойғаннан кейін пайда болады. Кестеден таңдалған мәндер дұрыс болмайды.

Мұндай өзекті мәселелерге тап болмау үшін төменде аталған ережелерге сәйкес болуы тиіс параллельді транзакцияларды келісімді орындау рәсімін құрастырған жөн:

1) транзакцияны орындау барысында пайдаланушы тек қана келісілген деректерді көреді. Пайдаланушы келісілмеген аралық деректерді көрмеуі тиіс;

2) ДҚ-да екі транзакция қоса орындалса, ДБЖ транзакциялардың орындалу нәтижелері 1 транзакциямен, содан кейін 2 транзакция, немесе керісінше, алдымен 2 транзакция, содан кейін 1 транзакциямен бірдей болады деп мәлімделген транзакцияларды дербес орындау қағидатына кепілдік береді.

5.3.3. Транзакцияларды оқшаулау деңгейлері

Транзакцияларды дұрыс сақтау дерекқорлардың тұтастығын қамтамасыз етудің негізі болып табылады, сонымен қатар көп қолданушы жүйелердегі пайдаланушыларды оқшаулау үшін базисті құрайды.Дерекқордың тұтастығын сақтау үшін міндетті талаптарды сақтай отырып, транзакцияларды оқшаулаудың бірнеше деңгейлері болуы мүмкін.

Транзакцияларды оқшаулау деңгейі – транзакцияға келісілмеген деректер енгізілген кезде деңгейді, яғни бір транзакцияның басқадан оқшаулану деңгейін айқындайтын мән. Оқшаулаудың едәуір жоғары деңгейі деректердің дәлділігін арттырады, бірақ осымен бірге қоса орындалатын транзакциялар саны төмендеуі

екінші жағынан, оқшауланудың неғұрлым төмен деңгейі көптеген қоса жүргізілетін транзакциялар жасауға мүмкіндік береді, бірақ деректердің дәлділігін азайтады.

Дерекқордың тұтастығын сақтаудың міндетті талабын орындау кезінде транзакцияларды оқшаулаудың келесі деңгейлері орын алуы мүмкін:

1. Оқшаулаудың ең жоғары деңгейі транзакцияларды сериалау хаттамасына сәйкес келеді. SERIALIZABLE (реттелушілік) деп аталатын бұл деңгейі транзакциялардың толық оқшауланууын және қоса жүретін транзакцияларды толық дұрыс өңдеуді қамтамасыз етеді.

2. Келесі оқшаулау деңгейі REPEATABLE READ расталған оқылым деңгейі деп аталады. Бұл деңгейде транзакция басқа транзакциялардың аралық немесе соңғы нәтижелеріне қол жеткізе алмайды, сондықтан жоғалған өзгерістер, аралық немесе келісілмеген деректер сияқты мұндай өзекті мәселелердің орын алуы мүмкін емес. Алайда транзакцияны орындау кезінде басқа транзакция арқылы дерекқорына енгізілген жолды көре аласыз. Сондықтан бір транзакция барысында орындалған бірдей сұрау салу әр түрлі нәтиже беруі мүмкін, яғни елес-жолдарға қатысты мәселе қалады. Алайда мұндай мәселе сыни жағдай болса, оны алгоритмдік түрде, өңдеу алгоритмін өзгерте отырып, бір транзакцияда сұрау салудың қайталама орындалуына жол бермей орындаған жөн.

3. Оқшаулаудың басқа деңгейі аяқталған оқылыммен байланысты және READ COMMITED (тіркелген деректер оқылымы) деп аталады. Бұл оқшаулау деңгейінде транзакция басқа транзакциялардың аралық нәтижелеріне қол жеткізе алмайды, сондықтан жоғалған жаңартулар мен аралық деректердің өзекті мәселелері туындауы мүмкін емес.Алайда, басқа транзакцияларды орындау кезінде алынған соңғы деректер біздің транзакцияға қол жетімді болуы мүмкін.Бұл жағдайда, «жағымсыз» оқылым (яғни COMMIT командасымен дерекқорда сақталмаған деректерді бір пайдаланушының оқуы) жоқ. Дегенмен, бір транзакцияның жұмыс үрдісінде басқасы табысты аяқталуы және жасалған өзгерістер тіркелуі мүмкін. Нәтижесінде бірінші транзакция деректердің басқа жиынтығымен жұмыс істейді. Бұл қайталанбайтын оқылымның өзекті мәселесі. Осы оқшаулау деңгейінде транзакция басқа транзакциямен жаңартылған жолды жаңарта алмайды. Мұндай жаңартуды орындауға талпыныс жасалған кезде транзакция жоғалған жаңартуға қатысты өзекті мәселенің туындауына жол бермеу үшін автоматты түрде жойылады.

4. Соңында, оқшауланудың ең төменгі деңгейі расталмаған немесе «жағымсыз» оқылым деңгейі деп аталады. Ол READUNCOMMITED болып белгіленеді. Оқшауланудың осы деңгейінде ағымдағы транзакция аралық және келісілмеген деректерді көре алады және оған елес-жолдар да қолжетімді. Бірнеше транзакция бір уақытта бірдей жолды өзгертуге тырысқан жағдайда соңғы нұсқада жол соңғы табысты орындалған транзакциямен айқындалған мәнге ие болады. Алайда оқшаулаудың осы деңгейінің өзінде ДБЖ жоғалған жаңарулардың алдын алады.

Транзакциялардың оқшауланууын қамтамасыз ету үшін ДБЖ-да өзара орындалуын реттеудің белгілі бір әдістері қолданылуы тиіс.

Жүйеде бір уақытта транзакциялардың белгілі бір саны орындалсын. Әртүрлі транзакциялар рәсімдері кезектесетін немесе қоса орындалатын транзакциялар жиынтығын орындау тәсілі осы транзакциялардың белгілі бір бірізді орындалу нәтижесіне транзакцияларды өзара орындау нәтижесі баламалы болса, сериалы деп аталады.

Транзакцияларды сериалау – оларды белгілі бір сериалы жоспар бойынша орындау механизмі. Мұндай механизмді қамтамасыз ету транзакцияларды басқаруға жауапты ДБЖ компонентінің негізгі қызметі болып табылады. Транзакциялардың сериалануы сақталатын жүйе пайдаланушылардың шынайы оқшауланууын қамтамасыз етеді.

5.3.4. Журналға өзгерістерді енгізу және деректерді қалпына келтіру

Журналға өзгерістерді енгізу - логикалық немесе физикалық сәтсіздік жағдайында бұрынғы келісілген күйге дерекқорды қалпына келтіру үшін қажетті ақпаратты сақтайтын ДБЖ қызметтерінің бірі. Өзгерістерді журналға енгізу, яғни барлық дерекқорды түрлендіру туралы ақпаратты сыртқы жадында сақтау, транзакцияларды басқарумен тығыз байланысты. Ең қарапайым жағдайда өзгерістерді журналған енгізу дерекқорда жасалған барлық өзгерістерді сыртқы жадыға дәйекті түрде жазуды білдіреді.Өзгерістерді енгізу журналы дерекқордағы барлық деректерді өзгерту рәсімдерінің, атап айтқанда, түрлендірілген нысанның ескі және жаңа мәнін, объектіні түрлендіретін транзакцияның жүйелік нөмірін қамтиды. Осылайша қалыптасатын

Жүйе бақылау нүктелерін кезеңді түрде орнатады. Бұл үрдісті орындау барысында барлық тіркелмеген деректер сыртқы жадқа ауыстырылады және бақылау нүктесі журналға жазылады.Осыдан кейін бақылау нүктесінің алдында жазылған журналдың мазмұны жойылуы мүмкін.

Өзгерістерді енгізу журналы сыртқы жадқа тікелей жазылмай, жедел жадыда жинақталуы мүмкін.Егер транзакция расталса, ДБЖ журналдың қалған бөлігі сыртқы жадқа жазылғанша күтеді.Осылайша, растау белгісінен кейін енгізілген барлық деректер дискі кэшінен барлық өзгертілген блоктардың қайта жазылуын күтпестен сыртқы жадқа аударылатынына кепілдік беріледі.

Логикалық бас тарту жағдайында немесе бір транзакцияны кері қайтару белгісі кезінде журнал кері бағытта сканерден өтеді, жойылатын транзакцияның барлық жазбалары транзакцияның басталуы туралы белгісіне дейін журналдан алынады. Алынған ақпаратқа сәйкес транзакция әрекеттерін жоятын әрекеттер орындалады, ал журналға өтеуші жазбалар жазылады.

Физикалық бас тарту жағдайында, журналдың өзі, дерекқор зақымданбаса, өткізу (rollforward) үрдісі орындалады. Журнал алдыңғы бақылау нүктесінен бастап тура бағытта сканерленеді. Барлық жазбалар журналдан журналдың соңына дейін алынады.Журналдан алынатын ақпарат сыртқы деректер жадының блоктарына енгізіледі, онда өзгеру санының белгісі журналда жазылғаннан аз. Егер өткізу үрдісі барысында ақаулық қайтадан іске қосылса, журналды сканерлеу қайтадан басталады, бірақ іс жүзінде қалпына келтіру тоқтаған жерінен жалғасады.

Дерекқорды қалпына келтіру қажеттілігі жағдайларын қарастырайық.

- транзакцияны дербес кері қайтару транзакцияның өзімен (ROLLBACK пәрмені бойынша) немесе транзакция жұмысында кез-келген қате орын алған жағдайда ДБЖ жүйесімен (мысалы, нөлге бөлу) транзакцияның кері қайтарылуы басталуы мүмкін;
- жүйенің «жұмсақ» ақауы (бағдарламалық жасақтаманың апаттық жағдайда істен шығуы) жүйе жедел жадының жоғалуымен сипатталады.Осымен бірге ауқау кезінде барлық орындалатын транзакциялар зақымдалады, дерекқордың барлық буферлерінің мазмұны жоғалады. Дискіде сақталатын деректер зақымданбаған

- жүйелердің «қатты» ақауы (аппаратураның аппатық жағдайда істен шығуы) жадтың сыртқы сақтау құралдарының зақымдануымен сипатталады. «Қатты ақау», мысалы, дискілік сақтау құралдарының сынуы немесе электр қуатының аппаттық ажыратылуы нәтижесінде орын алуы мүмкін.

Барлық үш жағдайда транзакциялар журналы тарапынан қамтамасыз етілетін деректер артықтығы қалпына келудің негізі болып табылады.

5.4. ДЕРЕККҚОРЛАРДАҒЫ АҚПАРАТТЫ ҚОРҒАУ

Ақпараттық қауіпсіздік көбінесе XXI ғасырдың негізгі ақпараттық өзекті мәселелердің арасында көрсетіледі. Шынында да, ақпаратты ұрлау, оны қасақана бұрмалау және жою мәселелері жиі зардап шеккен тарап үшін фирманың ойсырауына және банкротқа ұшырауына, тіпті адам өліміне әкелетін қайғылы салдарға әкеліп соғады. Ақпараттық қауіпсіздікті бұзудан болатын қиындықтардың тізімі үлкен – жүздеген миллион доллар шығынға әкелген мыңдаған коммерциялық компьютерлік қылмыстар, құпия ақпаратты ұрлаумен байланысты моральдық шығындар. Бұрын революцияның немесе төңкерістің табысты жүзеге асырылуы үшін пошта мен телеграфты басып алу маңызды болатын, ал енді компьютерлік телекоммуникациялар жүйелерін әлсірету қажет.

Ақпараттық жүйенің қауіпсіздігі - ақпараттың құпиялылығы мен тұтастығын қамтамасыз ету, яғни оны ашуға, өзгертуге немесе жоюға рұқсат етілмеген қол жетімділікке қатысты ақпаратты қорғау сияқты жүйенің қабілетін қамтушы қасиет.

Дерекқорлардағы ақпаратты қорғауға қатысты өзекті мәселені есептеу жүйесі мен жалпы жүйенің қорғауға байланысты мәселемен бірге қарастырған жөн. Бұған қоса, бағдарламалар мен деректерді қорғау әдістері мен құралдары тең дәрежеде ДҚ-дағы бағдарламалар мен деректерге жатады. Есептеу жүйесінің әр түрлі компоненттерімен ұсынылатын қорғау және мүмкіндіктер жүйесін құру қағидаларын (операциялық жүйені, қызмет көрсету бағдарламаларын, ДБЖ, мамандандырылған қорғаныс пакеттері мен жеке құрылғыларды) білу ақпараттық жүйенің осалдығын бағалауға және құпия ақпаратты қорғауды сауатты ұйымдастыруға ықпал етеді.

Ақпараттық жүйелерге және, сәйкесінше, дерекқорларға қатысты барлық қауіптерді үш топқа біріктіруге болады:

1) ашылу қаупі - ол туралы білмеуі керек тұлғаға ақпараттың белгілі болуы;
2) тұтастық қаупі - есептеу жүйесінде сақталған немесе бір жүйеден екіншісіне берілетін деректерді қасақана рұқсатсыз өзгерту (түрлендіру немесе жою);
3) қызметтен бас тарту қаупі – есептеу жүйесінің белгілі бір ресурсына қолжетімділікті бұғаттау қаупі.

Ықтимал қауіп-қатерлерге байланысты ақпаратты қорғаудың үш негізгі міндетін атап өтуге болады: ұрлықтан, жоғалудан, ақаулар мен істен шығудан қорғау.

Ұрлаудан ақпаратты қорғау ақпарат құралдарын және ақпарат сақтау құралдарын физикалық ұрлаудың алдын алуға, ақпараттың рұқсатсыз шығарылуына (көшіру, тыңшылық, ұстап қалу және т.б.) және бағдарламалар мен ақпаратты заңсыз таратуға жол бермейді.

Ақпаратты жоғалтудан қорғау ақпараттың тұтастығы мен дұрыстығын сақтауды, яғни ақпараттың физикалық, логикалық және семантикалық тұтастығын қамтамасыз етуді білдіреді.

Жүйедегі ақпарат пайдаланушылар жүйесіне, бағдарламаларға рұқсатсыз (оның ішінде компьютерлік вирустардың) кіру жағдайында, пайдаланушылар мен бағдарламаларының, қызмет көрсетуші қызметкерлер құрамының қате іс-әрекеттері, сондай-ақ желідегі ақаулар мен істен шығу кезінде жоғалуы мүмкін.

Жүйенің аппараттық-бағдарламалық жасақтамасының *ақаулар мен істен шығудан қорғау* жүйенің қалыпты қызмет ету үшін қажетті жағдайлардың бірі болып табылады. Егер есептеу жүйесі сенімсіз болса, ондағы ақпарат жиі зақымданады және кейде жоғалады.Жүйедегі ақаулықтардан және істен шығудан жақсы қорғауды қамтамасыз ететін негізгі жүктеме жүйелік аппараттық-бағдарламалық компоненттерге жатады: процессор, негізгі жады, сыртқы сақтау құрылғылары, кіріс-шығыс құрылғылары және басқа құрылғылар, сондай-ақ операциялық жүйе бағдарламалары.Жүйе құралдары айтарлықтай сенімді болмаған жағдайда ақаулардан қорғауды қолданбалы бағдарламаларда қарастырған жөн.

Ақпараттық жасақтаманың сенімділігі оған жүктелген қызметтердің нақты және уақтылы орындалу қабілетін білдіреді. Ақпараттық жасақтама сенімділігінің деңгейі әзірлеме үрдісін

Ақпараттық қауіпсіздікті қамтамасыз ету құралдарын іске асырылу тәсіліне байланысты әдістердің келесі сыныптарына бөлуге болады:

- ұйымдастырушылық әдістер ұтымды конфигурацияны, жүйені ұйымдастыру мен басқаруды қамтиды. Ең алдымен, ол желілік ақпараттық, операциялық жүйелерге, желілік әкімшінің өкілеттіктеріне, желісіне қол жеткізу тәртібін және жұмыс істеуін анықтайтын міндетті нұсқаулардың жиынтығына қолданылады;
- желілік басқаруды жүзеге асыру, ақпараттық ресурстардың қауіпсіздігін бақылау және аудит жүргізу, пайдаланушыларды тіркеу, келіп түскен ақпаратты іріктеу және вирусқа қарсы өңдеу электронды журналдарды жүргізу технологияларын қамтитын технологиялық әдістер;
- рұқсатсыз кіруден жүйенің физикалық қорғанысын жүзеге асырушы аппараттық әдістер, жүйенің перифериялық терминалдары мен пайдаланушыларды сәйкестендірудің аппараттық қызметтері, желілік компоненттерді қосу режимдері және т.б.;
- бағдарламалық әдістер – бұл ақпаратты қорғаудың едәуір кең таралған әдістері (мысалы, пайдаланушыларды сәйкестендіру, құпия сөзді қорғау және өкілеттіктерді тексеру бағдарламалары, брандмауэрлер, криптохаттамалар және т.б.).Бағдарлама компонентін пайдаланбай, әдістердің алғашқы үш тобын қоса алғанда, бірде-біреуі іс жүзінде мүмкін емес (таза түрде ұйымдастырушылық, технологиялық және аппараттық қорғау әдістері, әдетте, жүзеге асырыла алмайды - барлығы бағдарламалық компонентті қамтиды). Сонымен қатар, ақпараттың қорғалуын қамтамасыз ету үшін бағдарламалық желілік көптеген шешімдерді жүзеге асыру құны шығын бойынша аппараттық, технологиялық және, ең бастысы, ұйымдастырушылық шешімдердің құнынан айтарлықтай асып кететінін (әрине, егер сіз «қарақшылық» бағдарламаларды емес, лицензияланған бағдарламаларды пайдалансаңыз) ескерген жөн.

Сондай-ақ қуаттау тізбектері, компьютер немесе монитордың электрмагниттік сәулелену арналары бойынша ақпаратты жайылып кетуінен (үй-жайларды экрандау, шулы сәулелену генераторларын пайдалану, сәулеленуі анағұрлым аз мониторлар мен компьютердің толымдаушы бөліктерін арнайы іріктеу қолданылады), компьютердің толымдаушы бөліктеріне тікелей орнатылатын электронды «қоңыздардан» қорғау құралдары белсенді дамып келеді.

түрлі аспектілері мен қолданбалы сұрақтарына арналған мыңдаған басылымдар жарық көрді, халықаралық және мемлекеттік деңгейлерде ақпараттың қауіпсіздігін қамтамасыз ету жөніндегі көптеген заңдар қабылданған.

БАҚЫЛАУ СҰРАҚТАРЫ

1. Транзакция дегеніміз не?
2. Бірнеше транзакцияның өзара бірлескен жұмысының өзекті мәселелерін атап шығыңыз.
3. Дерекқордың тұтастығы деп нені атаймыз ?
4. «Жағымсыз» оқылымның өзекті мәселесі неде жатыр?
5. Қоса жүргізілетін транзакциялардың келісілген орындалуы рәсімі қандай ережелерге сәйкес келуі қажет?
6. Транзакцияны оқшаулау деңгейі деп нені атаймыз?
7. Транзакцияларды оқшаулаудың әрбір деңгейіне сипаттама беріңіз.
8. Транзакциялар журналына қандай ақпарат жазылады?
9. Транзакцияларды орындау қажеттілігін көрсетуші мысалды келтіріңіз.
10. Транзакциялар механизмін түсіндіріп беріңіз.
11. Логикалық істен шығу жағдайында не болады?
12. Физикалық істен шығу жағдайында не болады?
13. Транзакцияларды қолданудың мақсаты қандай?
14. Дерекқорды қалпына келтіру қандай жағдайда талап етіледі?
15. Ақпаратты қорғаудың үш негізгі міндетін атап беріңіз.
16. Жүзеге асыру тәсіліне байланысты ақпараттық қауіпсіздікті қамтамасыз ету әдістерінің қандай сыныптары бар?

6 тарау

SQL НЕГІЗДЕРІ

6.1. SQL ТІЛІНЕ КІРІСПЕ

1970 жж. басында IBM компаниясының зертханаларының бірінде дерекқорларды басқарудың эксперименталдық реляциялық жүйесі әзірленген, одан кейін оған арналып SQL арнайы тілі (Structured Query Language — құрылымдық сұрау салу тілі) құрылған болатын. SQL тіліне арналған стандарт 1986 жылы Американдық ұлттық стандарттар институты тарапынан шығарылған (ANSI), ал 1987 жылы Халықаралық стандарттар ұйымы (ISO) оны халықаралық стандарт ретінде қабылдаған.

Қазіргі күні SQL тілі дерекқорлармен жұмыс істеудегі алғашқы және әзірге жалғыз стандартты тіл болып табылады. SQL тілі едәуір әр түрлі есептеу платформаларға арналып әзірленген көптеген ДБЖ қолдау көрсетеді. ДБЖ-ның айтарлықтай барлық ірі әзірлеушілері қазіргі уақытта SQL тілін пайдалануға бағытталған өнімдерін шығарады.

SQL негізінен реляциялық дерекқорларда сақталған деректерді сипаттауға, өзгертуге және шығаруға арналған ақпараттық-логикалық тіл болып табылады. SQL - бейрәсімдік тіл, сонымен қатар, оны бағдарламалау тілі деп атауға болмайды.SQL тілі логикалық өзара байланысты кестелер ретінде ұсынылған деректермен орындалатын операцияларға бағдарланған, бұл өз алдына сөйлемдердің шағын ғана жиынтығынан құралған жинақы тілді құруға мүмкіндік берді. Деректерді өңдеудің рәсіміне емес, өңдеудің түпкілікті нәтижесіне бағдарлау SQL тілі құрылымының маңызды ерекшелігі болып табылады.SQL тілінің өзі деректердің, индекстердің қай жерде орналасқанын және тіпті нәтижелерді алу үшін операциялардың қандай ең тиімді тізбектерін пайдалану қажет екенін анықтайды, сондықтан бұл мәселелерді дерекқорға сұрау

Бұдан басқа, SQL тілі деректерге сұрау салуларды орындау үшін де, сондай-ақ қолданбалы бағдарламаларды құру үшін қолданыла алады.

Қызметтік мүмкіндіктерін кеңейту мақсатында қабылданған стандарттарды сақтаушы көптеген әзірлеушілер SQL стандартты тіліне түрлі кеңейтімдерді қосады.

Сәйкесінше өндірушінің SQL бағдарламалық өнімі SQL тілінің жүзеге асырылуы деп аталады.

Тілдің барлық нақты жүзеге асырылу әрекеттері бір-бірінен біршама өзгешеленеді. Пайдаланушылардың тасымалдау және жұмыс істеу ыңғайлылығы тұрғысынан оларды жүзеге асыру ANSI заманауи стандарттарына сәйкес келуіне кепілдік беру өндірушілердің мүддесіндегі жағдайда. Дегенмен, SQL әрбір іске асыру белгілі бір дерекқор серверінің талаптарына жауап беретін жетілдірулерді қамтиды.SQL тілінің бұл жетілдірулері немесе кеңейтімдері стандартты пакетке толықтырулар болып табылатын және осы іске асыруда қол жетімді болатын қосымша пәрмендер мен опцияларды білдіреді.

SQL тілі тек деректерді айқындау және айла-шарғы жасауға арналған командаларды қамтиды және есептеулердің барысын басқару үшін қандай да бір командалардан тұрмайды. Мұндай міндеттер бағдарламалау тілдері арқылы немесе интерактивті түрде пайдаланушылардың іс-әрекеттері арқылы шешілуі тиіс. SQL тілі екі тәсіл арқылы қолданыла алады:

- пайдаланушы тарапынан жеке SQL-операторларды енгізуді қамтитын интерактивті жұмыс;
- рәсімдік тілдерге құрылған бағдарламаларға SQL-операторларын енгізу.

SQL тілі салыстырмалы түрде меңгеру жағынан қарапайым. Бұл бейрәсімдік тіл болғандықтан, мұнда ақпаратты қалай алуға болатынды емес, қандай ақпарат алынуы тиіс екенін көрсету қажет. Басқа сөзбен айтқанда, SQL деректерге қолжеткізу әдістерін көрсетуді талап етпейді. SQL тілін көптеген кәсіби мамандар, соның ішінде дерекқор әкімшілері, қолданбалы бағдарламашылар және бағдарламалау дағдылары жоқ басқа пайдаланушылар пайдалана алады.

Орындалатын іс-әрекеттер түріне байланысты келесі рәсімдер ерекшеленеді:

SQL тілі командаларының негізгі санаттары әртүрлі қызметтерді, соның ішінде дерекқор нысандарын құру және айла-шарғы жасауды, кестелерге деректерді бастапқы жүктеуді, қолданыстағы ақпаратты жаңарту және жоюды, дерекқорға сұраныстарды толтыруды, оған қол жеткізуді бақылау және оны жалпы басқаруды орындауға арналған.

SQL тілі командаларының негізгі санаттары:

- DDL — деректерді анықтау тілі;
- DML— деректермен айла-шарғы жасау тілі;
- DQL — сұрау салу тілі;
- DCL — деректерді басқару тілі;
- деректерді басқару командалары;
- транзакцияларды басқару командалары.

DDL (DataDefinitionLanguage) деректерді анықтау тілі дерекқор нысандарының құрылымын құруға және өзгертуге, мысалы, кестелерді құруға және жоюға мүмкіндік береді. DDL тілінің негізгі командалары:

CREATE TABLE — кесте құру;

ALTER TABLE — кестені өзгерту;

DROP TABLE — кестені жою;

CREATE INDEX —индекс құру;

ALTER INDEX —ндексті өзгерту;

DROP INDEX —индексті жою.

DML (Data Manipulation Language) деректермен айла-шарғы жасау тілі төменде аталған үш негізгі команда арқылы реляциялық дерекқор нысандарының ішіндегі ақпаратпен айла-шарғы жасау үшін қолданылады:

INSERT — жазбаларды қою;

UPDATE — жазбаларды жаңарту;

DELETE — жазбаларды жою.

Кез-келген деректер үлгісі кейбір ДҚ-ны іске асыруда орындалуы мүмкін әрекеттер жиынтығын бір күйден екіншісіне түрлендіру үшін анықтайды.

Іріктеу шарты - бұл деректер элементінің логикалық позициясы, оның мәні және деректер арасындағы қатынастар пайдаланылуы мүмкін деректерді таңдаудың белгілі бір өлшемі.

DCL (Data Control Language) деректерді басқару тілі дерекқордағы ақпаратқа қолжетімділікті басқаруға мүмкіндік беретін деректерді басқару пәрмендерін қамтиды.Әдетте, олар деректерге қол жеткізуге байланысты объектілерді жасау үшін пайдаланылады, сонымен қатар пайдаланушылар арасында артықшылықтарды бөлуді басқару үшін қызмет етеді. Деректерді басқару пәрмендері келесідей:

GRANT — қол жету құқығын орнату;

REVOKE — қол жету құқығын жою.

Бұл пәрмендердің синтаксисі ДБЖ-ға байланысты. Қол жеткізуді басқару үрдісін жеңілдету үшін көптеген ДБЖ пайдаланушыларды топтарға біріктіру немесе рөлдерді белгілеу мүмкіндігін ұсынады.

Роль - пайдаланушыға және/немесе басқа рөлдерге берілген артықшылықтар жиынтығы. Бұл тәсіл нақты пайдаланушыға өзіне жүктелген тапсырмаларға сәйкес құқықтардың жиынтығы бар белгілі бір пайдаланушылар тобына белгілі бір рөл немесе өзектілігін беруге мүмкіндік береді.Деректерді басқару пәрмендерінің көмегімен пайдаланушы дерекқормен әрекеттердің орындалуын бақылайды, дерекқор рәсімдерін, жүйенің жұмысын талдайды және т.с.с. Деректерді басқару мен дерекқорды басқару бірдей емес екенін атап кеткен жөн. Дерекқорды басқару дерекқорды жалпы басқару болып табылады және барлық деңгейдегі командаларды пайдалануды білдіреді.

Транзакцияларды басқару пәрмендері келесі пәрмендерді қамтиды:

COMMIT — транзакцияны растау;

ROLLBACK — транзакцияны кері қайтару;

SAVEPOINT — үзіліс нүктесін орнату (толық емес қайтару);

SET TRANSACTION — транзакцияны бастау.

SQL тілі көптеген ДБЖ негізі болып табылады, себебі ол физикалық құрылымды қалыптастыру және деректерді дискіге

SQL операторы сақталған сөздерден, сондай-ақ пайдаланушы анықтаған сөзден тұрады.Сақталған сөздер SQL тілінің үнемі бөлігі болып табылады және тіркелген мәнге ие.Пайдаланушы анықтаған сөздерді пайдаланушының өзі (синтаксистік ережелерге сәйкес) белгілейді және әртүрлі дерекқор нысандарының сәйкестендіргіштерін немесе аттарын көрсетеді. Операторлардағы сөздер де белгіленген синтаксистік ережелерге сәйкес орналастырылады.

SQL тіл сәйкестендіргіштері дерекқордағы нысандарды белгілеуге арналған және кестелердің, көріністердің, бағандардың және басқа дерекқор нысандарының атаулары болып табылады.SQL стандарты әдепкі қалпы бойынша қолданылатын таңбалар жиынын анықтайды. Ол латын әліпбиінің кіші және бас әріптерін (A-Z, a-z), сандарды (0-9) және сызу белгісін (_) қамтиды. Сәйкестендіргіш пішіміне келесі шектеулер қолданылады:

- сәйкестендіргіш ұзындығы 128 таңба болуы мүмкін;
- сәйкестендіргіш әріптен басталуы тиіс;
- сәйкестендіргіш бос орындарды қамти алмайды.

Тілдің көптеген компоненттері регистрді ескермейді.

SQL тілінің сипаттамасы берілген тіл, метатіл деп аталады.Синтаксистік анықтамалар әдетте Бэкус-Наур формулалары (БНФ) деп аталатын арнайы металингвистикалық таңбалар арқылы беріледі. Сақталған сөздерді жазу үшін бас әріптер қолданылады. Кіші әріптер пайдаланушы тарапынан айқындалатын сөздерді жазу үшін пайдаланылады. БНФ шартты белгілерінде қолданылатын таңбалар және олардың белгіленуі 6.1-кестеде

6.1 кесте

Таңба	Белгіленуі
:: =	Анықтау бойынша тең
I	Берілген бірнеше мәндерден біреуін таңдап алу қажеттілігі
<...>	Метатіл арқылы сипатталған тіл құрылымы
	Тізімнен белгілі бір құрылымды міндетті түрде таңдау
[...]	Тізімнен белгілі бір құрылымды міндетті емес таңдау
[,n]	Нөлден бастап бірнеше мәртеге дейін құрылымның қайталанудың міндетті емес мүмкіндігі

Бұрын біз деректерді ДҚ-да әртүрлі түрдегі сақталған жиынтық ақпарат ретінде анықтадық. Деректер түрлерін қолдана отырып, кестенің нақты бағанындағы деректерге, оның ішінде бөлінген жады көлеміне қатысты негізгі ережелер белгіленеді.

SQL тілі стандарт бойынша анықталған алты скалярлық дерек түрін қамтиды (6.2-кесте).

Таңбалар деректері ДБЖ жасаушыларымен анықталған таңбалар жинағына кіретін таңбалар тізбегінен тұрады. Таңбалар жиынтығы SQL тілінің түрлі диалектілеріне тән болғандықтан, таңба түрінің деректер мәндеріне енгізілетін таңбалардың тізімі де нақты іске асыруға байланысты болады.

Таңбалар деректер түрімен бағанды анықтаған кезде, «ұзындық» өлшемі берілген бағанда орналастырылуы мүмкін таңбалардың жоғары санын көрсету үшін пайдаланылады. Таңба жолы тіркелген (CHAR) немесе айнымалы (VARCHAR) ұзындығы ретінде анықталуы мүмкін.Егер жол бекітілген ұзындық мәндерімен анықталса, онда оған аздаған таңбаларды енгізген кезде, мән көрсетілген ұзындыққа дейін оң жақтан қосылатын бос орындармен толықтырылады.Егер жол айнымалы мәндер мәнімен анықталса, дерекқорда азырақ таңбаларды енгізу кезінде енгізілген таңбалар ғана сақталады, бұл белгілі бір сыртқы жадтың үнемделуіне қол жеткізуге мүмкіндік береді.

Бит жолдарын анықтау үшін бит деректер түрі, яғни әрқайсысы 0 немесе 1 мәніне ие екілік сандардың (биттердің) кезектілігі пайдаланылады.

6.2 кесте

Деректер түрі	Хабарландыру
Таңбалық	CHAR|VARCHAR
Биттік	BIT|BIT VARYING
Нақты сандар	NUMERIC|DECIMAL|INTEGER|SMALLINT
Дөңгелектенген сандар	FLOAT|REAL|DOUBLE PRECISION
Күні/уақыты	DATE|TIME|TIMESTAMP
Аралық	INTERVAL

Нақты сандық түр деректері бөлшек бөліктің дәлдігі мен ұзындығымен анықталады. Дәлдік ондық нүктенің өзін ескермей, бүтін және бөлшек бөлігінің ұзындығын қамтитын санның маңызды ондық сандарының жалпы санын көрсетеді. Көлемі бөлшек санның ондық таңбалы санын көрсетеді.

NUMERIC және DECIMAL түрлері сандарды ондық форматта сақтауға арналған. Әдепкі қалпы бойынша бөлшек бөлік ұзындығы нөлге тең, ал әдепкі қалпы бойынша қабылданатын дәлдік іске қосуға байланысты болады. INTEGER (INT) түрі үлкен оң немесе теріс бүтін сандарды сақтауға арналған. SMALLINT түрі шағын оң немесе теріс бүтін сандарды сақтау үшін арналған; бұл жағдайда сыртқы жадты тұтыну айтарлықтай азаяды.

Дөңгелектелген сандар түрі компьютерде нақты көрсетіле алмайтын деректерді, атап айтқанда, нақты сандарды сипаттау үшін қолданылады.Дөңгелектелген сандар немесе өзгермелі нүктелер сандары ғылыми белгілерде ұсынылады, онда сан онның белгілі бір дәрежесіне көбейтілген (рет) мантисса арқылы жазылады.

«Күні/уақыты» деректер түрі белгілі бір белгіленген дәлдікке сәйкес уақыт кездерін анықтау үшін қолданылады. SQL стандарды келесі форматты қолдайды.DATE деректер түрі YEAR (жыл), MONTH (ай) және DAY (күн) өрістерін қамтитын күнтізбелік күндерді сақтау үшін пайдаланылады.TIME деректер түрі HOUR (сағат), MINUTE (минут) және SECOND (секунд) өрістерін қамтитын уақыт белгілерін сақтауға арналған.TIMESTAMP деректер түрі күн мен уақытты бірге сақтау үшін пайдаланылады.

INTERVAL түріндегі деректер уақыт кезеңдерін көрсету үшін

6.2. КЕСТЕЛЕРМЕН ЖҰМЫС. ТҰТАСТЫҚТЫ ШЕКТЕУ

6.2.1. Домендермен жұмыс

Домен – кесте бағандары сипаттамаларын сипаттауға мүмкіндік беруші реляциялық дерекқорлар нысаны. Бағандарды сипаттағанда кез келген кестені жасағанда немесе өзгерткенде, барлық сипаттамаларын кестедегі бағанға көшіру үшін қолданыстағы доменге сілтеме жасауға болады. Кейде дерекқорлардың бірнеше кестесінде сипаттамалары бірдей бағандар кездеседі.

Бұл жағдайда осындай бағанның түрін және доменді пайдаланып, іс-әрекетін алдын ала сипаттауға, одан кейін бірдей бағандардың әрқайсысын домен атымен салыстыруға болады. Доменде сипатталған жеке сипаттамалар бағанға көшіру кезінде өзгертілуі және толықтырылуы мүмкін. Домендерді сонымен қатар сақталатын рәсімдер мен триггерлерде жергілікті айнымалылар мен өлшемдерді сипаттау кезінде пайдалануға болады.

Деректер түрі доменнің негізгі сипаттамасы болып табылады.Деректер түрлері домендерді, кестелер бағандарының сипаттамаларын жасау немесе өзгерту, сақталатын рәсімдер мен триггерлердегі ішкі айнымалы мәндерді сипаттау кезінде анықталады.Деректер түрлерін сандық, жолдық топтарға (санаттарға), күн мен уақытқа, сондай-ақ, топтағы BLOB деректердің бірегей екілік түріне біріктіруге болады.

Домен келесі форматтағы CREATEDOMAINоператоры тарапынан айқындалады:

CREATE DOMAIN <домен атауы>[AS] <деректер түрі>

[DEFAULT <әдепкі қалпындағы мән>]

[NOT NULL] [CHECK (<шектеу шарты>)]

[COLLATE <сұрыптау тәртібі>]

DEFAULT ұсынымы әдепкі қалпы бойынша кесте жазбасын жасау кезінде доменмен байланыстырылған бағанға енгізілген өрнекті анықтайды.Бұл мән пайдаланушы кез келген жолмен өзгерпегенінше жазбаның тиісті бағанында қалады. Әдепкі қалыптағы мәндер литеральді мән (сандық, жолдық немесе күн) ретінде көрсетілуі мүмкін.

CHECK ұсынымы доменмен байланысты әрбір бағанның мәніне қойылатын талаптарды анықтайды. CHECK ұсынымында жүктелген шектеулерді қанағаттандырмайтын мәндер бағанға тіркеле алмайды.Доменмен байланыстырылған өрістердің мәндері үшін жүктелген шектеудің форматы:

<доменді шектеу шарты>::= {

VALUE<оператор><мән>

|VALUE ⌈NOT⌉ BETWEEN <1 мән>AND <2 мән>|VALUE ⌈NOT⌉ LIKE

<домен шектеулері>

<домен шектеулері>OR <домен шектеулері><домен шектеулері>AND <домен шектеулері>

}

мұнда

<оператор> = { = | < | > | <= | >= | !< | !> | <> | != };

VALUE <оператор><мән> - доменмәні<оператор>өлшеміменанықталғанқатынастарда<мән>өлшеміндеорналасады.Басқашаайтқанда, VALUE доменменбайланыстыбағанғатағайындалуымүмкінбарлықдұрысмәндердібілдіреді;

BETWEEN <1 мән>AND <2 мән> - домен мәні, олардың ішінде <1 мән> және <2 мән> арасында орналасуы тиіс;

LIKE<1 мән>[ESCAPE <2 мән>] - кез келген ұзындықтың кез келген мәнін көрсететін «%» белгісімен ұқсастық үлгісін, ал «_» таңбасы кез келген жалғыз таңбаны анықтайды.LIKE операторында қалыпты таңбалар ретінде «%» және «_» таңбалары көрсетілген жағдайда ESCAPE пайдаланылады.Бұл жағдайда <2 мән> белгілі бір таңбасы таңдалынады, содан кейін қызметтік нышандар ерекше мәртебесін жоғалтады және әдеттегі өлшемдер ретінде іздеу жолына енгізіледі. Мысалы, CHECK (VALUE LIKE «%!%»ESCAPE «!»)- «!» таңбалардың кез келген саны «%» аяқталады;

IN (<1 мән> [, <2 мән>...] – домен мәні тізімде көрсетілген өлшемдердің бірімен сәйкес келуі тиіс;

IS [NOT] NULL – бағандар міндетті түрде бос орыннан өзгеше белгілі бір мәндерді қамтуы тиіс;

CONTAINING <мән> - домен мәні міндетті түрде қай жерде екені маңызды емес, бірақ <мән> өлшемінің енгізілуін қамтуы тиіс;

STARTING [WITH] <мән> - домен мәні мән өлшемінен басталуы тиіс.

Доменнің мәні сәйкес келетін шарттар жиынтығы берілуі мүмкін. Бұл жағдайда жеке шарттар AND немесе OR операторы

Домен түсінігіне өзгеріс енгізу үшін келесі оператор қолданылады:

ALTER DOMAIN <атауы>

{ [S E T DEFAULT {литерал |NULL | U S E R }]

| [DROP DEFAULT]

| [ADD [CONSTRAINT] CHECK (<домен шектеулері>)]

| [DROP CONSTRAINT]}

Оператор бұған дейін CREATE DOMAIN операторы тарапынан белгіленген домен өлшемдерін өзгертуге мүмкіндік береді.Дегенмен, егер бағандар орнатылған деректер түріне сәйкес келмейтін мәндер немесе олар бос болса, деректер түрін және NOT NULL анықтамасын өзгертуге болмайды. Жасалған барлық өзгерістер осы доменді пайдалану арқылы анықталған барлық бағандар үшін ескеріледі. Бұл мәлімдемеде:

[SET DEFAULT] — CREATE DOMAIN операторында жасалатындай әдепкі мәндерді орнатады;

[DROP DEFAULT] — әдепкі қалпы бойынша тағайындалған ағымдағы әдепкі мәндерді жою;

[ADD [CONSTRAINT] CHECK (<домен шектеулері>)] — доменмен байланыстырылған баған мәндеріне сәйкес келуі тиіс шарттар қосу. Бұл жағдайда CREATE DOMAIN операторының CHECK ұсынысы үшін жоғарыда қарастырылған шарттарды

6.2.2. Кестелерді басқару

Жалпы дерекқор құрылымын жасағаннан кейін, дерекқор жобасының құрамына жататын қатынастарды көрсететін кестелерді жасауды бастауға болады. Естеріңізге сала кетейік, кесте – реляциялық дерекқорда ақпаратты сақтау үшін негізгі нысан болып табылады.Ол деректерді қамтитын жолдар мен бағандардан тұрады, дерекқордағы физикалық кеңістікті алады және тұрақты немесе уақытша болуы мүмкін. Реляциялық дерекқордағы баған деп аталатын өріс белгілі бір деректер түрінің тағайындалған кестесінің бөлігі болып табылады. Әр дерекқор кестесінде кемінде бір баған болуы керек. Деректер жолы – бұл дерекқор кестесіндегі жазба. Ол бір кесте жазбасынан деректерді қамтушы өрістерді қамтиды.

Дерекқор кестелерін жасамас бұрын, кестенің барлық бағандары мен әр бағанның сипаттамаларын, индекстерді, басқа кестелерге қатысты тұтастығы шектеулерін анықтау керек.Алдын ала домендер кестелерде пайдаланылған жағдайда анықталуы керек. Құрылатын кестенің қосылатын дерекқоры ашық болуы, яғни онымен белсенді байланыс орнатылуы керек.

ДҚ кестесін құру CREATE TABLE операторы тарапынан жүзеге асырылады. Кесте жасау операторының негізгі синтаксисі келесі түрге ие:

CREATE TABLE <кесте атауы>

({<бағанды анықтау>|<кестені шектеуді анықтау>}

[,...,{<бағанды анықтау>|<кестені шектеуді анықтау>}]}

Кестенің атауын үтірмен көрсетіп болғаннан кейін, кестенің жеке элементтерін анықтайтын жақшалардағы барлық ұсыныстар - бағандар немесе тұтастық шектеулері аталуы тиіс:

<кесте атауы> — құрылатын кесте сәйкестендіргіші;

<бағанды анықтау> — кестедегі жеке бағанның атауын, деректер түрін және өлшемдерін көрсету. Баған атаулары сәйкестендіргіштерге арналған ережелерге сәйкес келуі және кесте шегінде бірегей болуы керек;

<кестені шектеуді анықтау> — кесте деңгейінде белгілі бір тұтастық шектеулерін орнату.

<Бағанды анықтау> ұсынысын пайдалану арқылы баған сипаттары көрсетіледі:

<бағанды анықтау> ::=

<баған атауы><деректер түрі>

[<баған шектеуі>][,...,<баған шектеуі>]

Өлшемдердің мақсатын және пайдалануын қарастырайық.

<баған атауы> — кесте бағанының атауын белгілеуші сәйкестендіргіш;

<деректер түрі> — баған деректерінің түрі;

{[DEFAULT <мән>]

|[NULL|NOT NULL]

|[PRIMARY KEY|UNIQUE]

|[FOREIGN KEY REFERENCES <басты кестенің атауы>

[(<баған атауы>[,..,n])]

[ON DELETE {NO ACTION|CASCADE|SET DEFAULT|SET NULL}] [ON

Өлшемдер мәндерін қарастырайық:

CONSTRAINT — баған мәніне шектеу атауы (<сілтемелік тұтастық атауы>) көрсетілетін міндетті емес негізгі сөз.Аталар дерекқор шегінде бірегей болуы керек. Сілтеме тұтастығының атауы міндетті болып табылмайды.Ол тұтастықты бұзу туралы жүйелік хабарламаларда орналасады және кестелер құрылымын өзгерту кезінде пайдаланылуы мүмкін. Егер бұл атау болмаса, жүйе атауы орнатылады. Атаусыз тұтастықты жою үшін оның жүйелік атауын пайдалануға тура келеді;

DEFAULT — бағана үшін әдепкі мәнді анықтайды. Бұл мән баған үшін ешқандай мән көрсетілмесе, жолды кірістіру кезінде пайдаланылады;

NULL|NOT NULL —NULL мәндер бағанында сақтауға рұқсат беретін немесе қабылдамайтын негізгі сөздер. NULL кілт сөзі бұл бағанда NULL мәндері болуы мүмкін екенін көрсету үшін пайдаланылады.NULL мәні бос орыннан немесе нөлден ерекшеленеді - деректердің қолжетімсіз, қабылданбаған немесе жарамсыз екендігін көрсету қажет болғанда қолданылады.Егер NOT NULL кілт сөзі көрсетілсе, осы бағандағы NULL мәнін қою әрекеті қабылданбайды. Егер NULL өлшемі көрсетілсе, бағанға NULL мәндерін енгізуге рұқсат етіледі. Әдепкі қалпы бойынша SQL стандарты NULL кілт сөзін талап етеді;

PRIMARY KEY — бастапқы кіл сөзінің анықтамасы.Егер бағанда бастапқы кілт орнатылған болса, онда бағанңа PRIMARY KEY төлсипаты тағайындалуы мүмкін.Егер бастапқы кілт бір бағанды қамтыса, біліктеме баған анықтамасына орналастырылады.Бастапқы кілт құрамына бірнеше баған енгізілуі қажет болса, біліктеме барлық бағандарды айқындағаннан кейін қойылады. Кез келген жағдайда, бастапқы кілт құрылатын жолдар

UNIQUE — бағанда екі бірдей мән болмайтынын білдіретін төлсипат. Бірегей кілт бағана бастапқы кілт құрамына жатпаған кезде, алайда оның мәні әрқашан бірегей болуы тиіс жағдайда бағана (бағаналар) бойынша құрылады. Бұл төлсипатпен жарияланған баған, бастапқы кілт сияқты, бас және еншілес кестесінің арасындағы сілтеме тұтастығын қамтамасыз ету үшін пайдаланылуы мүмкін. Бас кестемен біріктіру үшін еншілес кестеде сыртқы кілт құрылады. Кестеде UNIQUE тұтастықтың бірнеше шектеулері құрылуы мүмкін;

FOREIGN KEY — тәуелді кестеде сілтемелік тұтастықты қамтамасыз ету үшін сыртқы кілт құрылады.

Сыртқы кілтті анықтау форматы:

```
FOREIGN KEY (<тәуелді кесте бағандарының тізімі>) REFERENCES
<басты кестенің атауы>

[<басты кесте бағандарының тізімі>]

[ON DELETE {NO ACTION|CASCADE|SET DEFAULT|SET NULL}]
[ON UPDATE {NO ACTION|CASCADE|SET DEFAULT|SET
NULL}]
```

Бағыныста кестенің бағандарының тізімі сыртқы кілтіне енгізілген өрістерді қамтиды. Негізгі кестедегі бағандар тізімі кестелерді байланыстыру кілті болып табылатын өрістерді қамтиды (басты кестесімен байланыс бастапқы кілтпен орындалса, тізім жойылмауы мүмкін).

ON DELETE, ON UPDATE міндетті емес өлшемдері басты кестенің бастапқы кілті жойылған және тиісінше өзгертілген кезде сервердің не істеу керектігін көрсетеді:

NO ACTION — егер басты кестеде бағынышты жазбалар болса, бас кестесінің тиісті жазбаларын жоюға немесе өзгертуге тыйым салынады;

CASCADE — басты кестедегі жазбаларды жойған кезде еншілес кестедегі барлық бағынышты жазбалардың жойылуы жүзеге асырылады. Бас кестедегі жазбаларды өзгерткен кезде, еншілес кестенің барлық бағынатын жазбаларындағы кілт өрісінің мәндері өзгереді;

SET NULL — бос NULL мәні еншілес кестесінің бағынатын жазбаларының негізгі өрісіне енгізіледі.

Кестенің бағандарына салынған шектеулер жалпы форматы төменде көрсетілген CHECK ұсынысы (<баған шарты>) арқылы анықталады:

<баған шарты> : : = {

VALUE<оператор><мән>

|VALUE [NOT] BETWEEN <1 мән>AND <2 мән>|VALUE [NOT] L I K E <мән>[ESCAPE <мән>]

|VALUE [NOT] IN (<1 мән>[, <2 мән>...]

Бұл жағдайда бағаның мәніне қатысты шектеулер доменді анықтау кезіндегідей бірдей форматта сипатталады:

<оператор> = { = | < | > | <= | >= | !< | !> | <> | != };

VALUE <оператор><мән> — домен мәні <оператор> параметрімен анықталған қатынастарда <мән> параметрімен анықталады;

BETWEEN <1 мән>AND <2 мән> — домен мәні <1 мән> және <2 мән> арасында орналасуы тиіс;

L I K E <1 мән>[ESCAPE <2 мән>] - кез келген ұзындықтың кез келген мәнін көрсететін «%» белгісімен ұқсастық үлгісін, ал «_» таңбасы кез келген жалғыз таңбаны анықтайды.LIKE операторында қалыпты таңбалар ретінде «%» және «_» таңбалары көрсетілген жағдайда ESCAPE пайдаланылады.Бұл жағдайда <2 мән> белгілі бір таңбасы таңдалынады, содан кейін қызметтік нышандар ерекше мәртебесін жоғалтады және әдеттегі өлшемдер ретінде іздеу жолына енгізіледі. Мысалы, CHECK (VALUE L I K E «%!%»ESCAPE «!») - «!» таңбалардың кез келген саны «%» аяқталады;

IN (<1 мән> [, <2 мән>...] – домен мәні тізімде көрсетілген өлшемдердің бірімен сәйкес келуі тиіс;

IS [NOT] NULL – бағандар міндетті түрде бос орыннан өзгеше белгілі бір мәндерді қамтуы тиіс;

CONTAINING <мән> - домен мәні міндетті түрде қай жерде

STARTING [WITH] <мән> - домен мәні мән өлшемінен басталуы тиіс.

Доменнің мәні сәйкес келетін шарттар жиынтығы берілуі мүмкін.Бұл жағдайда жеке шарттар AND немесе OR операторы арқылы қосылады.

Кестелерді жасағаннан кейін, дерекқордағы бар кестелерді сипаттауды (бағандардың сипаттамалары, олардың тәртібі, әртүрлі кілттердің болуы немесе CHECK шектеулері) өзгертуге болады.

Жоғарыда айтылғанды мысалдармен дәлелдеп көрейік.

Нөмір және Атауы жолдарымен БӨЛІМДЕР кестесін белгілеп алайық:

```
CREATE TABLE БӨЛІМДЕР
(НөмірINTEGER NOT NULL,

АтауыV A R C H A R ( 2 0 ) )
```

Нөмір, Атауы және Нөмір жолы бойынша бастапқы кілтпен берілген жолдар орналасқан БӨЛІМДЕР КЕСТЕСІН айқындайық:

```
CREATE TABLE БӨЛІМДЕР

(НөмірINTEGER NOT NULL PRIMARY K E Y ,

АтауыV A R C H A R ( 2 0 ) )
```

Сол кесте, бірақ екі өрісте тұрған бастапқы кілтпен (кесте деңгейіндегі шектеулерді анықтау) беріледі:

```
CREATE TABLE БӨЛІМДЕР
(НөмірINTEGER NOT NULL,

АтауыVARCHAR(20) NOT NULL,

PRIMARY KEY (Нөмір, Атауы)
```

Енді домен кестесіндегі қолдану мысалын келтірейік. Артық немесе 100 тең шектеуімен бүтін түрге ие Домен_Нөмір доменін құрамыз:

```
CREATE DOMAIN Домен_НөмірAS INTEGER CHECK
(VALUE>=100);
```

ҚЫЗМЕТКЕРЛЕР кестесінің түсініктемесінде Нөмір жолы Домен_Нөмір доменімен ұқсастырылады:

```
CREATE TABLE ТАУАРЛАР (ТауарVARCHAR (20)
NOT NULL,

БағаINTEGER NOT NULL,

PRIMARY KEY (Тауар))
```

САТЫЛЫМДАР тәуелді кестесі ТАУАРЛАР кестесімен сілтемелік тұтастықты қамтамасыз ету үшін Тауар сыртқы жолы және Нөмір жолы бойынша бастапқы кілтке ие. Басты кестеде байланыс жолы көрсетілмегендіктен, байланыс үшін ТАУАРЛАР кестесінің бастапқы кілті қолданылады:

```
CREATE TABLE САТЫЛЫМДАР (НөмірINTEGER
NOT NULL PRIMARY KEY,

КүніDATE,

ТауарVARCHAR (20) NOT NULL,

FOREIGN KEY (Тауар) REFERENCES ТАУАРЛАР)
```

Басты және тәуелді кестелердің жалпы жолдарының анықтамалары дәл сәйкес келуі тиіс. Егер рәміздердің сұрыптау тәртібінде айырмашылықтар бар болса, байланыс бағандары іс жүзінде бірдей болмайды, бұл сілтеме тұтастығының бұзылуына әкеледі.

ТАУАРЛАР және САТЫЛЫМДАР кестелері арасындағы байланысты құрудың тағы бір мысалы:

Бағынышты кестеде бастапқы кілт құрамындағы мәнді өзгерту және басты кестедегі жазбаны жою кезіндегі сервер іс-әрекеттері көрсетіледі:

```
CREATE TABLE САТЫЛЫМДАР (НөмірINTEGER
NOT NULL PRIMARY KEY,

КүніDATE,

ТауарVARCHAR (20) NOT NULL,

CONSTRAINT FK1 FOREIGN KEY (Тауар) REFERENCES ТАУАРЛАРON
UPDATE NO ACTION ON DELETE NO ACTION)
```

Қолданыстағы кестелердің құрылымын өзгерту үшін төменде жеңілдетілген синтаксисі көрсетілген ALTER TABLE операторы қолданылады:

ALTER TABLE <кесте атауы>

{ [A D D <баған атауы><деректер түрі> [

NULL|NOT NULL]]

| [D R O P [COLUMN] <баған атауы>] }

Бір операторда кестедегі өзгерістердің ерікті санын орындауға болады. Бар кестені өзгертуге арналған түрлі операциялар бір-бірінен үтірлермен бөлінеді. ALTER TABLE операторы келесі әрекеттерді орындауға мүмкіндік береді:

- жаңа бағанның анықтамасын қосу;
- кестеден бағанды жою;
- кестенің немесе жеке баған тұтастығының төлсипатын жою;
- тұтастықтың жаңа төлсипаттарды қосу.

Баған сипаттамаларын өзгерту, сондай-ақ бағанды жою сәтсіз аяқталуы мүмкін, егер:

- баған PRIMARY KEY немесе UNIQUE төлсипаттарын иеленеді, бірақ бағандағы ескі мәндер бірегей деректердің талаптарын бұзады;
- жойылған баған бастапқы немесе сыртқы кілтке бір бөлігі ретінде енгізілген, бұл кестелеер арасындағы сілтемелік тұтастықтың бұзылуына әкеп соқты;
- бағанға кесте деңгейіндегі тұтастық шектеулері қоса жазылған;
- баған қараулардағы, триггерлердегі, есептелген бағандардағы өрнектердегі басқа дерекқор компоненттерінде пайдаланылады.

Осылайша, егер қажет болса, баған төлсипаттарын өзгерту немесе бағанды жою жағдайында алдымен кесте мен дерекқордың салдарын өзгерту немесе жою сияқты өзгерістерді неге әкелуі мүмкін екенін мұқият талдау қажет. Жаңа бағанды қосу:

ALTER TABLE <кесте атауы>ADD <бағанды белгілеу>

Тұтастықтың жаңа шектеуін қосу:

ALTER TABLE <кесте атауы>ADD [CONSTRAINT <шектеу атауы^ <тұтастықты белгілеу>

Бағанды жою:

Тұтастық шектеуін жою:

```
ALTER TABLE <кесте атауы>DROP <шектеу атауы>Кестені жою
```

оператор арқылы жүзеге асырылады:

```
DROP TABLE <кесте атауы>
```

Мысалы, кестеге бағанды қосу келесі команда арқылы жүзеге асыруға болады:

```
ALTER TABLE KLIENT ADD ADRES VARCHAR(50)
```

Тұтастық (бастапқы кілт) шектеуін кестеге қосуды келесі команда арқылы орындауға болады:

```
ALTER TABLE KLIENT

ADD CONSTRAINT PK_ KLIENT

PRIMARY KEY (ID)
```

Кестені жою мысалы:

```
DROP TABLE KLIENT
```

Байланысты кестелерде (кезінді кезде жойылмаған) кестелердің

6.3. ДЕРЕКТЕРДІ ІРІКТЕУ. SELECT ОПЕРАТОРЫ

SELECT операторы DML деректеріне айла-шарғы жасау тілінің бөлігіне жатады.Бұл ең күрделі және қуатты SQL операторларының бірі.Бұл оператор өте күрделі және дамыған құрылымға ие. Дерекқорлармен қандай да бір түрде байланысқан кез келген маманға белгілі болуы керек.

Бұдан әрі мысал ретінде пайдаланылатын дерекқор үлгісі 6.1-суретте берілген.

Тауарлар кестесі Тауар өрісіне арналған Сатылымдар кестесімен «бір-көп» қатынаспен байланысты. Сол сияқты, Клиенттер кестесі Клиент өрісі үшін Сатылымдар кестесімен «бір-көп» қатынаспен байланысты. Сұрау салуларды орындау нәтижесін анағұрлым көрнекі көрсету үшін мысал ретінде толтырылған кестелерді (6.2

Өріс	Өріс түрі	Сипаттама
Тауар	Жол	Тауар атауы
ӨБ	Жол	Өлшем бірлігі
Баға	Сан	Баға

а

Өріс	Өріс түрі	Сипаттама
Клиент	Жол	Клиент
ЖСН	Жол	Клиенттің ЖСН-ы
Қала	Жол	Қала
Телефон	Жол	Телефон

б

Өріс	Өріс түрі	Сипаттама
Нөмір	Сан	Құжат нөмірі
Күні	Күні	Құжат күні
Тауар	Жол	Тауар атауы
Сан	Сан	Тауар саны
Клиент	Жол	Клиент

в

6.1-сурет. Мысалдар үшін қолданылатын кестелер құрылымы: ***а*** — ТАУАРЛАР кестесі; ***б*** —КЛИЕНТТЕР кестесі; ***в*** — САТЫЛЫМДАР кестесі

SELECT (ағылш. таңдау) — берілген шартты қанағаттандыратын дерекқордан деректер жиынын (таңдауды) қайтаратын SQL тілінің операторы.Бұл деректерді бір немесе бірнеше дерекқор кестелерінен таңдауға және алынған нәтижелерді қажетті пішінге түрлендіруге мүмкіндік береді. Бұл оператор реляциялық алгебра операторларына баламалы әрекеттерді орындай алады. Оның көмегімен түрлі кестелерден деректерді таңдаудың күрделі және ауқымды шарттарын жүзеге асыруға болады. Егер таңдау бірнеше кестеден орындалса, онда біріктіру әрекеті туралы айтады.SELECT сұрауын жасаған кезде пайдаланушы деректердің қажетті жиынтығын (бағандар жиыны, жазбаларды таңдау критерийлері, топтау мәндері, жазбалардың шығысын ресімдеу және т.б.) сипаттайды.

	Халва	кг	45
	Қант	кг	60
	Бидай ұны	кг	50
	"Мишка"кәмпиттері	кг	150
	"Коровка"кәмпиттері	кг	90
	"Кроха"мармелады	кг	85
	Шұжықтар	кг	260
	"Рощинские"қысқа шұжығы	кг	300
	"Штурвал" майбалығы	дана	56

а

Клиент	ИНН	Қала	Телефон I
"Горизонт"ЖАҚО	025500123152	Мәскеу	4890605
Привалов И.И. ЖК	025501025666	Санкт-Петербург	2560245
"Түймедақ" ЖАҚ	025501244555	Мәскеу	3658815
"Перевал" ЖАҚ	145889898622	Тверь	221588
Иванов Ф.И. ЖК	025558000554	Мәскеу	1155488
Таран О.С.	360224005454	Мәскеу	1215648
Федорова Д.С.	500255510055	Санкт-Петербург	4449702
Лесовая В.Н.	212585832187	Мәскеу	3021402
БМЖТ	555878970025	Санкт-Петербург	4353822
Дремина Е.Е.	025578721058	Мәскеу	6582209
Газимова К.К. ЖК	025587978766	Мәскеу	6521588

б

1 Нөмір	Клиент	Тауар	СаныКүні	
4§1	Федорова Д.С.	"Коровка" кәмпиттері	50	02.04.2013
156	Газимова К.К. ЖК	"Мишка" кәмпиттері	30	02.04.2013
162	Федорова Д.С.	Халва	10	15.02.2013
200	Лесовая В.Н.	"Кроха" мармелады	20	07.02.2013
25	Таран О.С.	Халва	5	04.01.2013
85	Лесовая В.Н.	Бидай ұны	50	02.04.2013
112	"Түймедақ" ЖАҚ	Халва	40	15.02.2013
254	Лесовая В.Н.	Қант	50	01.02.2013
140	Газимова К.К. ЖК	"Коровка" кәмпиттері	15	09.01.2013
144	Таран О.С.	Шұжықтар	30	01.04.2013

В

6.2-сурет.Кестелердің бастапқы жағдайы:

а — ТАУАРЛАР кестесі; ***б*** — КЛИЕНТТЕР кестесі; ***в*** — САТЫЛЫМДАР кестесі

Біріншіден барлық жазбалар кестеден шығарылады, содан кейін жинақтың әрбір жазбасы үшін көрсетілген критерийге сәйкестігі тексеріледі. Егер бірнеше кестеден біріктіру жүзеге асырылса, алдымен кестелердің көбейтіндісі құрылады, содан кейін қажетті жазбалар алынған жиындардан алынады. Тұтастай алғанда, SELECT операторы келесі синтаксиске ие:

```
SELECT [ALL|DISTINCT ]{*|[баған_атауы [AS жаңа_ атау]]} [,..n]

FROM кесте_атауы [[AS] лақап ат] [,...n]

[WHERE <жағдайлар>]
```

[HAVING <топтарды таңдау қағидаттары>]

[ORDER BY баған_атауы [,...n]]

SELECT операторы сұраудың нәтижесіне қосылатын өрістерді (бағандарды) анықтайды. Тізімде олар үтірлермен бөлінеді және сұраныстың нәтижесі ретінде ұсынылатын тәртіпте беріледі. Егер бос орындарды немесе бөлгіштерді қамтушы өріс атауы қолданса, оны төртбұрышты жақшаға алған жөн. Барлық өрістерді * белгісі арқылы таңдауға және өріс атауының орнына бірнеше атаулардан тұрған өрнекті қолдануға болады. Егер бірнеше кесте өңделсе, онда әр түрлі кестелерде атауы бір жолдар болған кезінде өрістер тізімінде өрістің толық ерекшелігі, яғни Кесте атауы, Өріс атауы қолданылады.

SELECT операторы элементтерінің өңделуі келесі ретте орындалады:

FROM — деректер іріктелетін дерекқор кестелерінің тізімі;

WHERE — берілген шарттарға сәйкес нысан жолдарын іріктеу орындалады;

GROUP BY — көрсетілген бағанда мәні бір жолдар топтары құрылады;

HAVING — көрсетілген шартқа сәйкес нысан жолдарының топтары іріктеледі;

SELECT — шығу деректерінде қандай бағандар болуы тиіс екендігі белгіленеді;

ORDER BY — операторлардың орындау нәтижелерін бірізділікке келтіру айқындалады.

SELECT операторындағы сөйлемдер мен сөз орамдарының тәртібін өзгертуге болмайды. SELECT және FROM екі сөйлем ғана міндетті болып табылады, қалғандары түсірілуі мүмкін. SELECT — жабық рәсім: кестеге сұраныстың нәтижесі басқа кестені құрайды. Төмендегі мысалдарда көрсетілгендей, осы операторды жазудың көптеген нұсқалары бар.

Мысалы, Клиенттер кестесінің барлық бағандары мен жазбаларынан құралған деректер жиынтығын төменде көрсетілген оператор арқылы құруға болады:

Клиент	ИНН	Город	Телефон
"Горизонт" ЖАҚО	0255001231521	Мәскеу	4890605
Привалов И.И. ЖК	0255010256	Санкт-	2560245
"Түймедақ" ЖАҚ	0255012445	Мәскеу	3658815
"Перевал" ЖАҚ	1458898986	Тверь	221588
Иванов Ф.И. ЖК	0255580005	Мәскеу	1155488
Таран О.С.	3602240054	Мәскеу	1215648
Федорова Д.С.	5002555100	Санкт-	4449702
Лесовая В.Н.	2125858321	Мәскеу	3021402
БМЖТ	5558789700	Санкт-	4353822
Дремина Е.Е.	0255787210	Мәскеу	6582209
Газимова К.К. ЖК	0255879787	Мәскеу	6521588

6.3-сурет. КЛИЕНТТЕР кестесінің барлық жазбалары мен бағандарын іріктеу **ИИН - ЖСН; Город-Қала**

Сұрауды орындау нәтижесі қайталанатын мәндерді қамтуы мүмкін, себебі SELECT операторы деректерді алуды орындау кезінде қайталанатын мәндерді есептен шығармайды. DISTINCT предикаты таңдалған өрістерде қайталанатын жазбалары бар деректер блоктарын тастау талап етілген жағдайларда пайдаланылуы керек.

SELECT нұсауындағы өрістердің әрқайсысы үшін мәндер бірегей болуы керек, себебі олардың құрамындағы жазба шығыс жиынтығына енгізіле алуы қажет. DISTINCT пайдаланудағы шектеудің себебі - оны қолдану сұраныстардың орындалуын күрт төмендетуі мүмкін.

SELECT DISTINCT КЛИЕНТТЕР. Қала FROM КЛИЕНТТЕР

Нәтижесінде қалалар тізімін аламыз — клиенттердің орналасу жері (6.4-сурет).

WHERE операторы қажетті ғана жазбаларды деректер жиынтығына енгізу үшін қолданылады. Бұл жағдайда SELECT операторы келесі форматқа ие:

SELECT {* | <1 мән> [, <2 мән> . ..]}

FROM <1 кесте> [, < 2 кесте> ...]

WHERE <жағдай>

Город
Москва
Санкт-Петербург
Тверь

6.4-сурет. Клиенттер қалаларының тізімі

Басты WHERE сөзінен кейін іздеу шарттарының тізімі беріледі.SELECT операторымен қайтарылатын деректер жиынтығына WHERE-ден кейін көрсетілген іздеу шарттарына сәйкес келетін

Клиент	ИНН	Город	Телефон
ИП Привалов И.И.	025501025666	Санкт-Петербург	2560245
Федорова Д.С.	500255510055	Санкт-Петербург	4449702
БМСТ	555878970025	Санкт-Петербург	4353822

6.5-сурет. Санкт-Петербургтен клиенттерді таңдау

Іздеу шарттарының (немесе предикаттардың) негізгі түрлері:

- *салыстыру:*бір өрнек есептеудің нәтижелерін басқа есептеу нәтижелерімен немесе әртүрлі бағандардың мәнімен салыстыру;
- *диапазон:* баған мәнінің немесе өрнекті бағалау нәтижесінің белгілі бір мәндер ауқымына түсуі тексеріледі;
- *жиынтыққа тиесілігі:* берілген мәндер жиынтығына баған мәнінің немесе өрнекті есептеу нәтижесінің сәйкес келуі тексеріледі;
- *шаблонға сәйкестігі:* берілген шаблонға белгілі бір өріс мәнінің жауап беруі тексеріледі;
- *NULL мәні:* берілген бағанның NULL (белгісіз мәнді) анықтағышын қамтуы тексеріледі.

Мысалы, КЛИЕНТТЕР кестесінен Санкт-Петербург клиенттерінің тізімін сұрау салу арқылы алуға болады (6.5-сурет):

```
SELECT*

FROMКЛИЕНТТЕР

WHEREКЛИЕНТТЕР.Қала = «Санкт-Петербург»
```

SQL тілінде келесі салыстыру операторларын қолдануға болады:

= тең< кем> артық<= кем немесе тең>= артық немесе тең<> тең емес

Мысалы, сатылған тауар саны 10 артық тауарларды сатудың барлық операцияларды көрсету:

```
SELECT *

FROM
САТЫЛЫМДАРWHERE
Саны>10
```

- мән солдан оңға қарай есептеледі;
- алдымен жақшалардағы шағын мәндер есептеледі;
- NOTоператорлары AND және OR операторларын орындауға дейін орындалады;
- AND операторлары OR операторларын орындауға дейін орындалады.

Кез келген ықтимал белгісіздікті жою үшін жақшаларды қолдану ұсынылады. Іздеудің айтарлықтай күрделі шарттарын қолдану мысалдарын келтірейік. Мынадай сұрау нәтижесі бойынша бағасы 50-ден артық немесе 100-ден кем немесе тең тауарлардың тізімі көрсетіледі.

```
SELECT Тауар, БағаFROM ТАУАРЛАР

WHERE Баға>=50 AND Баға<=100
```

Осы сұрауды BETWEEN операторы арқылы орындауға болады. BETWEEN кілт сөзі ең аз және ең жоғары мәндермен анықталған белгілі бір аралығындағы мәнді іздеу үшін пайдаланылады.Бұл жағдайда көрсетілген мәндер іздеу шартына қосылады:

```
SELECT Тауар, БағаFROM ТАУАРЛАР

WHERE БағаBETWEEN 50 And 100
```

Алынған үлгі бұрынғы мысалдағы үлгімен ұқсас болады(6.6-сурет).

NOT BETWEEN терістеуін қолданған кезде көрсетілген ауқым шегінен тыс мәндер алынады.

IN операторы белгілі бір мәнді көрсетілген мәндер тізімімен салыстыру үшін пайдаланылады және өрнекті бағалау нәтижесі берілген тізімдегі мәндердің біріне сәйкес келетінін тексереді.IN операторын пайдалану арқылы сол нәтижені OR операторын пайдалану жағдайында алуға болады, алайда IN операторы

Тауар	Баға
▶ Қант	60
Бидай ұны	50
"Коровка" кәмпиттері	90
"Кроха" мармелады	85
"Штурвал" майбалығы	56

6.6-сурет. Сұрау салуда BETWEEN қолдану нәтижесі

Мысалы, Мәскеуде немесе Санкт-Петербургте өмір сүрмейтін клиенттердің тізімін көрсету үшін сұранысты орындау қажет:

```
SELECT Клиент, ҚалаFROM КЛИЕНТТЕР

WHERE ҚалаNOT IN ("Мәскеу","Санкт-Петербург")
```

LIKE операторы арқылы белгі-орын алмастырушыларды қолдануға рұқсат берілген шаблонмен мәнді салыстыруды орындауға болады:

- «%» таңбасы — таңбалардың кез-келген саны осы таңба үшін ауыстырылуы мүмкін;
- «_» таңбасы жолдың бір таңбасын ауыстырады.

Мысалы, келесі сұрау салуды орындау нәтижесінде телефон нөмірлерінің екінші саны 4 болатын клиенттердің тізімі шығады:

```
SELECT *

FROM КЛИЕНТТЕР

WHERE КЛИЕНТТЕР.Телефон LIKE '_4%'
```

Нәтижесінде 4449602 телефонымен клиент Федорова Д.С. табылатын болады.

Тегінде немесе атауында «ва» буыны кездесетін клиенттерді таңдау үшін «%» екі белгісін қолданумен сұрау салуды құрау қажет, нәтижесінде келесі іріктеу алынады (6.7-сурет):

```
SELECT *

FROM КЛИЕНТТЕР

WHERE КЛИЕНТТЕР.Клиент LIKE '%ва%',
```

Клиент	ЖСН	Қала	Телефон
Привалов И.И. ЖК		025501025666 Санкт-Петербург	2560245
"Перевал" ЖАҚ		145889898622 Тверь	221588
Иванов Ф.И. ЖК		025558000554 Мәскеу	1155488
Федорова Д.С.		500255510055 Санкт-Петербург	4449702
Лесовая В.Н.		212585832187 Мәскеу	3021402
Газимова К.К. ЖК		025587978766 Мәскеу	6521588

6.7-сурет. Шаблон бойынша клиенттерді іріктеу

IS NOT NULL сөз тіркесі белгілі бір өрісте мәннің болуын тексеру үшін қолданылады.

Мысалы, телефоны көрсетілмеген клиенттерді табу қажет (Телефон өрісі ешқандай мәнді қамтымайды):

```
SELECT Клиент

FROM КЛИЕНТТЕР

WHERE Телефон IS NULL
```

Телефоны бар клиенттерді іріктеу (Телефон өрісі белгілі бір мәнді қамтиды):

```
SELECT КлиентFROM КЛИЕНТТЕР

WHERE ТелефонIS NOT NULL
```

Тұтастай алғанда, SQL сұрау деректерінің нәтиже жинағындағы жолдар ешқандай түрде реттелмеген. Дегенмен, оларды белгілі бір тәртіпте сұрыптауға болады. Бұл үшін ORDER BY негізгі сөзі шығару кестесінің деректерін көрсетілген ретпен сұрыптайтын SELECT операторына орналастырылады. Сұрыптауды бірнеше өрістерде орындауға болады, бұл жағдайда олар үтірлермен бөлінген ORDER BY негізгі сөзінен кейін тізімделеді.Сұрыптау әдісі (өсуі немесе кемуі бойынша) ORDER BY өлшемінде көрсетілген негізгі сөз бойынша сұрыптау орындалатын өрістің атауымен анықталады. Әдепкі қалпы бойынша өсу тәртібінде сұрыптау орындалады. Ол ASC негізгі сөзімен айқын түрде беріледі. Сұрыптауды кері тәртіпте орындау үшін DESC негізгі сөзін ол орындалатын өріс атауынан кейін көрсету керек.ORDER BY тармағы таңдаулы жазбаларды кез келген бағанның немесе бағандардың тіркесімі мәндерінің өсу немесе кему тәртібімен, осы бағандардың нәтижелер жиынында немесе жоқ екеніне қарамастан ұйымдастыруға мүмкіндік береді. ORDER BY сөйлемі әрдайым SELECT операторындағы соңғы элемент болуы керек. Мысалы, тауарлар тізімін әліпбилік ретпен тізімдеу керек (6.8-сурет):

```
SELECT *

FROM ТАУАРЛАР

ORDER BY ТАУАРЛАР.Тауар
```

П	"Коровка" кәмпиттері	кг	90
	"Мишка" кәмпиттері	кг	150
	"Кроха" мармелады	кг	85
	Бидай ұны	кг	50
	"Рощинские" шұжықтары	кг	300
	Қант	кг	60
	Шұжықтар	кг	260
	Халва	кг	45
	"Штурвал" майбалығы	дана	56

6.8-сурет. ТАУАРЛАР кестесін іріктеу нәтижесі

```
SELECT*

FROMСАТЫЛЫМДАР

ORDER BY САТЫЛЫМДАР.Тауар, САТЫЛЫМДАР.Күні
```

Келесі сұрау салуда іріктеу алдымен сатып алушының атауы бойынша, одан кейін тауар атауы бойынша және, соңында, күні бойынша жүзеге асырылады:

```
SELECT Клиент, Тауар, Күні, Саны

FROM Сатылымдар

ORDER BY Клиент, Тауар, Күні
```

Біріктіру — екі немесе одан да көп кестенің біреуіне біріктіру үрдісі. Ақпаратты бірнеше кестелерден немесе сұраулардан деректердің бірыңғай логикалық жиынтығы түрінде біріктіру мүмкіндігі төменде талқыланатын SQL мүмкіндіктерінің кең ауқымын тудырады. Бұрын WHERE операторын сұрау нәтижесінің шектеулерін енгізу құралы ретінде анықтадық. Бірақ шектеу шарты ретінде кестелер арасындағы нақты бағандар арасында байланыс орнатылса, бұл екі кестенің қосылуы болады.

Нөмір	Клиент	"Коровка" кәмпиттері	СаныКүні	
45	Федорова Д.С.	"Коровка" кәмпиттері	50	02.04.2013
156	Газимова К.К. ЖК	"Мишка" кәмпиттері	30	02.04.2013
200	Лесовая В.Н.	"Кроха" мармелады	20	07.02.2013
85	Лесовая В.Н.	Бидай ұны	50	02.04.2013
254	Лесовая В.Н.	Қант	50	01.02.2013
144	Таран О.С.	Шұжықтар	30	01.04.2013
25	Таран О.С.	Халва	5	04.01.2013
162	Федорова Д.С.	Халва	10	15.02.2013
112	"Түймедақ" ЖАҚ	Халва	40	15.02.2013

6.9-сурет. Екі өріс бойынша іріктеу нәтижесі

Екі кестеден алынған деректердің блоктары көрсетілген өрістерде сәйкес мәндер табылған кезде біріктіріледі.

САТЫЛЫМДАР кестесінен тауарлар сату үшін барлық жазбаларды және ТАУАРЛАР кестесінен оның бағасын көрсету үшін әрбір өнім үшін келесі сұрауды орындауға болады:

```
SELECT САТЫЛЫМДАР.*, ТАУАРЛАР.БАҒА

FROM САТЫЛЫМДАР, ТАУАРЛАР

WHERE САТЫЛЫМДАР.ТАУАР = ТАУАРЛАР.ТАУАР
```

Осы операторды орындаған кезде САТЫЛЫМДАР кестесіндегі әрбір жазба үшін ТАУАРЛАР кестесіндегі жазба ізделінеді, оның Тауар өрісіндегі мәні САТЫЛЫМДАР кестесінің ағымдық жазбадағы Тауар өрісіндегі мәнімен бірдей келеді (6.10-сурет).

Осымен бірге кестелерді іздеуді қандай тәртіпте тізімдеу керек, яғни қандай кестелердің сол жағында және қайсысының оң жағында көрсетілетіні маңызды емес. Біріктірілетін кестелердің бұл тәсілі *ішкі байланыс* деп аталады. Кестелердің ішкі байланысының тағы бір мысалын келтірейік. Өнімді тұтыну туралы барлық жазбаларды САТЫЛЫМДАР кестесінен таңдап, әрбір клиент үшін өз Қаласын КЛИЕНТТЕР кестесінен көрсетейік:

```
SELECT САТЫЛЫМДАР.*, КЛИЕНТТЕР.ҚалаFROM
САТЫЛЫМДАР, КЛИЕНТТЕР

WHERE КЛИЕНТТЕР.Клиент = САТЫЛЫМДАР.Клиент
```

Мысалдан көріп отырғандай, алынған деректер жиынтығы КЛИЕНТТЕР кестесіндегі жазбаларды қамтымайды, олар үшін

Номер	Клиент	Тауар	СаныКүні		Баға
140	Газимова К.К. ЖК	"Коровка" кәмпиттері	15	09.01.2013	90
45	Федорова Д.С.	"Коровка" кәмпиттері	50	02.04.2013	90
156	Газимова К.К. ЖК	"Мишка" кәмпиттері	30	02.04.2013	150
200	Лесовая В.Н.	"Кроха" мармелады	20	07.02.2013	85
85	Лесовая В.Н.	Бидай ұны	50	02.04.2013	50
254	Лесовая В.Н.	Қант	50	01.02.2013	60
144	Таран О.С.	Шұжықтар	30	01.04.2013	260
25	Таран О.С.	Халва	5	04.01.2013	45
162	Федорова Д.С.	Халва	10	15.02.2013	45
112	"Түймедақ" ЖАҚ	Халва	40	15.02.2013	45

6.10-сурет. Кестелерді ішкі біріктіру нәтижесі

[[Нөмірі [клиент		Тауар	Саны	Күні	Қала
140	Газимова К.К. ЖК	"Коровка" кәмпиттері	15	109.01.2013	Мәскеу
156	Газимова К.К. ЖК	"Мишка" кәмпиттері	30	02.04.2013	Мәскеу
254	Лесовая В.Н.	Қант	50	01.02.2013	Мәскеу
200	Лесовая В.Н.	"Кроха" мармелады	20	07.02.2013	Мәскеу
85	Лесовая В.Н.	Бидай ұны	50	02.04.2013	Мәскеу
112	"Түймедақ" ЖАҚ	Халва	40	15.02.2013	Мәскеу
25	Таран О.С.	Халва1	5	04.01.2013	Мәскеу
144	Таран О.С.	Шұжықтар	30	01.04.2013	Мәскеу
162	Федорова Д.С.	Халва	10	15.02.2013	Санкт-
45	Федорова Д.С.	"Коровка" кәмпиттері	50	02.04.2013	Санкт-

6.11-сурет. САТЫЛЫМДАРжәне КЛИЕНТТЕР кестелерін біріктеру

Алынған деректер жиынтығынан WHERE сөйлеміндегі іздеу шартына сәйкес келмейтін барлық жазбалар жойылады.

Кестелерді SELECT сөзінен кейін және бағанның атауынан бұрын WHERE-ден кейін іздеу күйінде қайтарылған бағандар тізіміндегі кестелерді байланыстырудың жоғарыдағы мысалдарында кестенің атауы нүкте бойынша жазылады, мысалы:

```
WHERE КЛИЕНТТЕР.Клиент = САТЫЛЫМДАР.Клиент
```

Баған атауынан бұрын әр түрлі кестелерде бірдей бағандар болған жағдайда кестенің атауын көрсетуіңіз керек (бұл мысалдағы сияқты). Бағандарды анықтау үшін кесте атауын пайдалану ыңғайсыз, себебі бұл оны қиын және күрделі етеді. Әрбір кестеге қысқа атау - лақап атты тағайындау әлдеқайда жақсы. Лақап аттар нақты кесте атауынан FROM негізгі сөзінен кейін бастапқы кестелер тізіміндегі бос орын арқылы бөлінеді. Мысалы, сұрау салу

```
SELECT САТЫЛЫМДАР.*,
КЛИЕНТТЕР.ҚалаFROM
САТЫЛЫМДАР, КЛИЕНТТЕР

WHERE КЛИЕНТТЕР.Клиент = САТЫЛЫМДАР.Клиент
```

лақап аттар кестесін енгізгеннен кейін айтарлықтай жинақы көрінеді:

```
SELECT С.*, К.ҚалаFROM
САТЫЛЫМДАР С,
КЛИЕНТТЕР К WHERE
К.Клиент = С.Клиент
```

Алынған деректер жиынының есептелген бағандарының мәндерін есептеу үшін арифметикалық өрнектер қолданылады. Бұл жағдайда SELECT операторынан кейін есептелетін баған атауының орнына баған тізімінде өрнек көрсетіледі. Мысалы, төмендегі сұраныс Сатылымдар кестесінен тауарларды сатуға қатысты барлық

Номер	Клиент	Товар	Количество	Дата	MULTIPLY
140	Газимова К.К. ЖК	"Коровка" кәмпиттері	15	09.01.2013	1350
45	Федорова Д.С.	"Коровка" кәмпиттері	50	02.04.2013	4500
156	Газимова К.К. ЖК	"Мишка" кәмпиттері	30	02.04.2013	4500
200	Лесовая В.Н.	"Кроха" мармелады	20	07.02.2013	1700
85	Лесовая В.Н.	Бидай ұны	50	02.04.2013	2500
254	Лесовая В.Н.	Қант	50	01.02.2013	3000
144	Таран О.С.	Шұжықтар	30	01.04.2013	7800
162	Федорова Д.С.	Халва	10	15.02.2013	450
112	"Түймедақ" ЖАҚ	Халва	40	15.02.2013	1800

6.12-сурет. Есептелетін өрісті құру

```
SELECT С.*, Т.Баға, С.Саны * Т.БағаFROM САТЫЛЫМДАР С,
ТАУАРЛАР Т WHERE С.Тауар = Т.Тауар
```

Есептелетін өрісті құру үшін қосу, алу, көбейту және бөлудің арифметикалық амалдары, сондай-ақ SQL тілінің бекітілген қызметтері қолданылады. Кестедегі кез келген бағанның атын көрсетуге, бірақ тек FROM тармағында тізімделген кестенің немесе сұраудың баған атауын пайдалануа болады. Күрделі өрнектерді салу кезінде жақшалар қолданылады. Алынған кестенің баған атауларын айқын көрсетуге болады, олар үшін AS сөз тіркесі қолданылады. Келесі сұраныс сатылатын жылы мен айы белгіленген өнімдер тізімін көрсетеді (6.13-сурет):

```
SELECT Т.Тауар, Year(С.Күні) AS Жылы, Month(С.Күні) AS Ай

FROM ТАУАРЛАР Т, САТЫЛЫМДАР С WHERE Т.Тауар=С.Тауар
```

Сұрау салуда күнінен жылы мен айын белгілеу үшін `Year` және `Month` кіріктірілген қызметтер пайдаланылған.

1 Тауар	Жыл	Ай
► "Коровка" кәмпиттері	2 013	1
"Коровка" кәмпиттері	2 013	4
"Мишка" кәмпиттері	2 013	4
"Кроха" мармелады	2 013	2
Бидай ұны	2 013	4
Қант	2 013	2
Шұжықтар	2 013	4
Халва	2 013	2
Халва	2 013	2

6.13-сурет. Сұрау салулардағы кіріктірілген қызметтерді пайдалану

Сұрау салуларды қалыптастыру кезінде агрегаттық (қорытынды) қызметтерді қолдануға болады.

Агрегаттық (қорытынды) қызметтер деректер жиынтығының барлық жазбалары бойынша операциялардың жалпы мәндерін есептеуге арналған. Агрегаттық қызметтерге келесі қызметтер жатады:

COUNT (<мән>)SQL-сұрау салуының шығу жиынтығындағы жазбалар санын айқындайды (тіркелуші деректер жиынтығының барлы жазбаларына мәндерді енгізу саны;

SUM (<мән>) өрнек мәндерін жинақтайды;

AVG (<мән>) жазбалар сұрау салуымен таңдалған белгілі бір өрісте сақталатын көптеген өрнектің орташа мәнін есептеуге мүмкіндік береді. Ол арифметикалық орташа мән, яғни олардың санына бөлінген мәндер сомасы болып табылады;

MAX (<мән>) ең жоғарғы мәнді айқындайды;

MIN (<мән>) ең төменгі мәнді айқындайды.

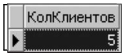
Барлық осы қызметтер кестенің бір бағанында (бірнеше кесте бағандарында) немесе арифметикалық өрнектегі мәндермен жұмыс істейді және бір мәнді қайтарады.Кез келген қызметтердің нәтижелерін есептеу кезінде, барлық бос мәндер алдымен жойылады, содан кейін қажетті операция бағанның қалған нақты мәндеріне ғана қолданылады. COUNT(*)нұсқасы - COUNT қызметін пайдаланудың арнайы жағдайы, оның мақсаты бос, қайталанатын немесе кез келген басқа мәндерден тұратындығына қарамастан, алынған кестенің барлық жолдарын санау болып табылады. Келесі сұрау салуды орындау нәтижесінде САТЫЛЫМДАР кестесінде жолдар саны есептеледі:

```
SELECT Count(*) AS Сатылымдар Саны
FROM САТЫЛЫМДАР
```

Бірдей жазбалар тобынан біреуін ғана есепке алу қажет болса, онда өрнек алдында жақшаға DISTINCT негізгі сөзін енгізеді:

```
COUNT(DISTINCT Клиент)
```

DISTINCT-тің MIN және MAX қызметтері үшін мәні жоқ, алайда оны пайдалану SUM және AVG қызметтерін орындау нәтижелеріне ықпал етуі мүмкін, сондықтан ол әрбір нақты жағдайда болуы тиіс



6.14-сурет. Жазбалар санын

6.15-сурет. Есептелетін өріс мәндері

```
SELECT COUNT(DISTINCT САТЫЛЫМДАР.КЛИЕНТ) AS
Клиенттер саны FROM САТЫЛЫМДАР
```

Басқа мысалда бір күн ішінде сатылған тауарлардың жалпы құны есептеледі (6.15-сурет):

```
SELECT SUM(С.Саны*Т.Баға) AS Жалпы құныFROM
САТЫЛЫМДАР С,ТАУАРЛАР Т
WHERE (С.Тауар^.Тауар) AND (С.Күні='02.04.2013')
```

COUNT, MIN, MAX қызметтері сандық және сандық емес өрістерге де қолданылады. SUM және AVG қызметтері сандық өрістер жағдайында ғана пайдаланылуы мүмкін.

Кейде сұраулар бойынша аралық қорытындыларды жасау, яғни барлық деректер жиынтығы үшін емес, белгілі бір топтар үшін қызметті орындау нәтижесін алу (мысалы, әрбір клиент, әрбір тауар үшін белгілі бір күнге жиынтықты есептеу үшін) керек. Бұл мақсатта SELECT операторында GROUP BY сөйлемі қолданылады. GROUP BY қатысатын сұрауды топтау сұрауы деп атайды, себебі деректер SELECT әрекеті нәтижесінде топтастырылған. Әрбір жеке топ үшін бір жиынтық жолы жасалады.SELECT операторынан кейін топтағы мәндерді өзгертпейтін сол бағандар тізімге енгізіледі, яғни олар топтауға қатысады.Мысалы, САТЫЛЫМДАР кестесіндегі тауарларға арналған сұранысты топтастыру арқылы SELECT сөйлеміндегі күнді көрсетуге болмайды, себебі әр өнімге сатудың әртүрлі күндері сәйкес болуы мүмкін және сәйкесінше мұндай сұрау дұрыс болмайды:

```
SELECT С.Тауар, С.Күні, SUM(С.Саны)
FROM САТЫЛЫМДАР С GROUP BY С.Тауар
```

Сатылған тауардың жалпы санын алу үшін сұрау салуды қолданған жөн (6.16-сурет):

Тауар	(SUM
"Коровка" кәмпиттері	65
"Мишка" кәмпиттері	30
"Кроха" мармелады	20
Бидай ұны	50
Қант	50
Шұжықтар	30
Халва	55

6.16-сурет. Топтастырылатын сұрау салу нәтижесі

Ал әрбір күн үшін сатылатын тауарлардың санын алу үшін алдыңғы сұрауды келесі жолмен түзету қажет:

```
SELECT С.Тауар, С.Күні, SUM(С.Саны)

FROM САТЫЛЫМДАР С

GROUP BY С.Тауар, С.Күні
```

Басқаша айтқанда, SELECT сөйлемі тізімінде тізілген барлық өріс атаулары, жиынтық қызметінде баған атауы пайдаланылған жағдайды қоспағанда, GROUP BY сөйлемінде болуы керек. Сонымен қатар GROUP BY сөйлемінде SELECT сөйлемі тізімінде жоқ баған атаулары болуы мүмкін.

Әр клиенттің сатып алу санын есептеуді сипаттайтын сұраулардағы топтауды қолданудың тағы бір мысалы (6.17-сурет):

```
SELECT Клиент, COUNT(*)

FROM САТЫЛЫМДАРGROUP BY Клиент
```

Егер GROUP BYбірге WHEREсөйлемі қолданылса, ол бірінші өңделеді, іздеу шартын қанағаттандырушы ғана жолдар топтастыруға жатады.

Клиент	COUNT
Газимова К.К. ЖК	2
Лесовая В.Н.	3
"Түймедақ" ЖАҚ	1
Таран О.С.	2
Федорова Д.С.	2

6.17-сурет.Әрбір клиент үшін сатылым (мәміле деректерінің) санын есептеу

Тауар	(SUM
"Коровка" кәмпиттері	5850
"Мишка" кәмпиттері	4500
"Кроха" мармелады	1700
Бидай ұны	2500
Қант	3000
Шұжықтар	7800
Халва	2475

6.18-сурет. Шарты белгіленген топтастыру нәтижесі

Күні	COUNT
04.01.2013	1
09.01.2013	1
01.02.2013	1
07.02.2013	1
15.02.2013	2
01.04.2013	1
02.04.2013	3

6.19-сурет. Әрбір күнге сатып алушылардың санын есептеу

```
SELECT C.Тауар,

SUM(C.Саны*Т.Баға)

FROM САТЫЛЫМДАР C,
ТАУАРЛАР Т
WHERE T.Тауар = C.Тауар
GROUP BY C.Тауар
```

Әрбір күнге сатып алушылардың санын анықтау үшін келесі сұрау салуды орындау қажет (6.19-сурет):

```
SELECT Күні, COUNT(DISTINCT Клиент)
FROM САТЫЛЫМДАР
GROUP BY Күні
```

Егер нәтиже жиынынан алынған жолдардың нәтижелеріне шектеулер қою керек болса, онда HAVING сөйлемі GROUP BY сөйлемінен кейін пайдаланылады.Бұл шығу жиынтығын «сүзу» үшін қосымша мүмкіндік. Қорытынды қызметтерді SELECT сөйлемі тізімінде және HAVING сөйлемінде ғана қолдануға болатындығын ескеру керек, ал WHERE сөйлемінде мұндай қызметтерді көрсету мүмкін емес. Мысалы, келесі сұраныс 3000 рубльден артық сатылған тауарлардың тізімін көрсетуге мүмкіндік береді (6.20-сурет):

```
SELECT C.Тауар, SUM(C.Саны*Т.Баға)

FROM САТЫЛЫМДАР C, ТАУАРЛАР Т WHERE 1.Тауар = C.Тауар

SELECT Күні,COUNT(*)

FROM САТЫЛЫМДАР

WHERE Саны>=10
GROUP BY Күні HAVING
```

	Тауар	SUM
1	"Коровка" кәмпиттері	5850
	"Мишка" кәмпиттері	4500
	Шұжықтар	7800

6.20-сурет. 3 000 руб. артық сомаға сатылған тауар тізімі

Дата	COUNT
15.02.2013	2
02.04.2013	3

6.21-сурет. Сатылым күні және сатылған тауар саны

Кейде салыстыру әрекеттерін пайдаланған кезде сұрауларда сұрау нәтижесін салыстыруға қажетті мән алдын-ала анықталмайды және бір емес, бірнеше мәндерді білдіреді немесе басқа SELECT операторын орындау арқылы алынады. Мұндай жағдайларда ішкі сұраулар (кіріктірілген сұраулар) қолданылады.

Ішкі сұрау (кіріктірілген сұрау) - бұл уақытша кестені жасау құралы, оның мазмұны сыртқы операторомен шығарылады және өңделеді. Ішкі сұраудың мәтіні жақшаға алынуы керек. Ішкі сұраныстағы SELECT операторы келесі пішінге ие:

```
SELECT . ..

FROM . ..

WHERE <салыстырылатын мән><оператор>(SELECT . ..)
```

Ішкі сұраудың екі түрі бар:

- скалярлық ішкі сұрау жалғы мәнді қайтарады. Негізінде жалғыз мәнді көрсету талап етілген жердің бәрінде қолданыла алады;
- кестелік ішкі сұрау көптеген мәндерді, бірден артық жолдарда орналасқан кестенің бір немесе бірнеше бағанының мәндерін қайтарады. Ол кестенің болуына рұқсат етілген барлық жерде орналасуы мүмкін.

Ішкі сұраулар келесідей болады:

- корреляцияланбаған - жоғары деңгейлі сұрауға сілтемелерді қамтымайды, жоғарғы деңгейлі сұрау үшін бір рет есептеледі;
- корреляцияланған – негізгі сұрау салудағы жолдар мәніне тәуелді шарттарды қамтиды; жоғары деңгейлі сұрау салудың әрбір жолы үшін есептеледі.

Кіріктірілген сұрау негізгі немесе сыртқы сұрау сияқты дәл сондай синтаксиске ие, сонықтан кіріктірілген сұрау да ішкі сұрау салуды қамтуы мүмкін.Бір SELECT операторы басқа SELECT операторына енетін сияқты. Сыртқы SELECT операторы барлық рәсімнің соңғы нәтижесінің мазмұнын анықтау үшін кші операторлардың орындау нәтижесін қолданады. Ішкі сұраулар SELECT сыртқы операторлардың WHERE және HAVING сөйлемдеріне салыстыру (=, <, >, <=, >=, <>) операторынан кейін тікелей орналастырылуы мүмкін. Бұдан басқа, SELECT ішкі операторы INSERT, UPDATE және DELETE операторларында (жазбаларды қою, жаңарту және жою) қолданылуы мүмкін.

Ішкі сұрауларға келесі ережелер мен шектеулер қолданылады:

- ORDER BY қолдануға болмайды, бірақ бұл сөз тіркесі сыртқы сұрауда болуы мүмкін;
- SELECT сөйлеміндегі тізім EXISTS негізгі сөзі ішкі сұрауда болған кездегі жағдайларды қоспағанда, олардан тұратын жеке бағандар немесе өрнектердің атауларынан тұрады;
- әдепкі қалпы бойынша ішкі сұраудағы баған атаулары FROM сөйлемінде аты көрсетілген кестеге жатады. Дегенмен, сыртқы сұраудың FROM сөз тіркесінде көрсетілген кестенің бағандарына сілтеме жасауға рұқсат етіледі.
  Бұл үшін көрсетілген баған атаулары (яғни, кесте) немесе олардың лақап аты қолданылады;

- ішкі сұрау салыстыру операциясына қатысатын екі операнданың бірі болса, онда сұрау осы операцияның оң жағында көрсетілуі керек.
  Келесі сұрау тауардың ең көп саны сатылған күнін қайтарады:

```
SELECT Саны, Күні

FROM САТЫЛЫМДАР

WHERE Саны = (SELECT MAX(Саны)

FROM САТЫЛЫМДАР)
```

Әлбетте, кіріктірілген SELECT операторы тізімді емес, бір мәнді қайтаруы керек. Кіріктірілген ішкі сұраныс кезінде тауардың ең жоғарғы саны анықталады. Сыртқы шағын сұрау салуда – тауардың саны ең жоғарғы санына тең күн айқындалады. Ұсынысты тікелей қолдануға болмайтынын айта кету қажет,

```
WHERE Саны = Max(Саны),
```

себебі WHERE ұсынымында жалпылаушы қызметтерді қолдануға тыйым салынған.

```
SELECT С.Саны, С.Күні, К.Клиент, К.ҚалаFROM
САТЫЛЫМДАР С, КЛИЕНТТЕР К
```

	СаныКүні	Клиент	Қала
▶	50101.02.2013	Лесовая В.Н.	Мәскеу
	50 02.04.2013	Лесовая В.Н.	Мәскеу
	50 02.04.2013	Федорова Д.С.	Санкт-

6.22-сурет.Кіріктірілген сұрау салудың мысалы

6.23-сурет. Есептелуші бағандағы сұрау салуды кіріктіруді қолдану мысалы

Күні	Саны	Ауытқу
02.04.2013	50	20
02.04.2013	50	20
15.02.2013	40	10
01.02.2013	50	20

```
WHERE(С.Клиент = К.Клиент)

AND
```

Алдыңғы мысалмен салыстырғанда, сұрауға САТЫЛЫМДАР мен КЛИЕНТТЕР кестелерінің ішкі қосылымы енгізілген.

Ішкі сұраныстың тағы бір мысалы: тауардың саны жалпы орташа мәннен асып кету және ауытқу мөлшерін анықтайтын сатып алу күндерін анықтау керек (6.23-сурет):

```
SELECT Күні, Саны,

Саны-^ELECT AVG(Саны) FROM САТЫЛЫМДАР)

AS АуытқуFROM САТЫЛЫМДАР WHERE Саны>
```

Жоғарыда келтірілген мысалда барлық сатылым фактілері бойынша тауарлар санының орташа мәні болып табылатын нәтижесі сыртқы SELECT операторында орташа мөлшерден ауытқуды есептеу үшін және күн туралы ақпаратты таңдау үшін пайдаланылады.

Жиі WHERE немесе HAVING ұсыныстарымен салыстырылатын мән бір мәнді емес, бірнеше мәндер болып табылады - кіріктірілген ішкі сұранымдар аралық уақытша кестені құрайды. Ішкі сұрауға қолданылатын рәсімдер өз кезегінде көбіне пайдаланылатын рәсімдерге негізделген, атап айтқанда:

```
[NOT] IN;

{ALL | SOME | ANY};

[NOT] EXISTS;
```

IN операторы белгілі бір мәнді мәндер тізіміне салыстырып, салыстырылған тізімде мән бар ма немесе салыстырылатын мән тексерілген тізімнің элементі емес екенін тексеру үшін пайдаланылады.

```
SELECT C.*

FROM САТЫЛЫМДАРCWHERE С.КлиентIN

(SELECTC1.Клиент FROMCATЫЛЫМДАРС1
WHEREC1.Саны =

(SELECT^^^.Саны) FROMCATЫЛЫМДАРС2))
```

Бұл сұрау салуды орындау логикасын түсіндірейік. Алдымен Саны бағанында ең жоғарғы мәні анықталады:

```
SELECT ^^^.Саны) FROM САТЫЛЫМДАР С2
```

Әрі қарай келесі сұрауда осы сатылымды жүзеге асырған клиенттер анықталады:

```
SELECTC1.Клиент FROMCATЫЛЫМДАРС1
WHEREC1.Саны =

(SELECT^^^.Саны) FROMCATЫЛЫМДАРС2)
```

Одан кейін негізгі сұрау INоператоры арқылы САТЫЛЫМДАР кестесінен табылған клиенттермен барлық жазбаларды таңдайды, яғниСАТЫЛЫМДАРкестесінен кіріктірілген сұраулардан алынған көп санына Клиент өрісінің мәні жататын жазбалар ғана деректердің нәтижелі жиынтығына енеді (6.24-сурет).

NOT IN арқылы кіріктірілген сұраулардан алынған салыстырмалы мәні көп санына жатпайтын жолдар таңдауын алуға болады.

ANY және ALL негізгі сөздері сандардың бір бағанын қайтарушы ішкі сұрауларды пайдалана алады.

Ішкі сұрау арқылы қайтарылатын мәндер мен салыстырмалы мәннің қатынасы ALL және SOME (ANY) операторларымен

Нөмір	Клиент	Тауар	Саны	Күні
▶ 451	Федорова Д.С.	"Коровка" кәмпиттері	50	02.04.2013
162	Федорова Д.С.	Халва	10	15.02.2013
200	Лесовая В.Н.	"Кроха" мармелады	20	07.02.2013
85	Лесовая В.Н.	Бидай ұны	50	02.04.2013
254	Лесовая В.Н.	Қант	50	01.02.2013

6.24-сурет. Тауардың ең жоғарғы санын сатып алған клиенттердің сатып алған заттарының тізімі

Нөмір	Клиент	Тауар	Саны	Күні
▶	45 Федорова Д.С.	["Коровка"кәмпиттері	50	02.04.2013
	85 Лесовая В.Н.	Бидай ұны	50	02.04.2013
	254 Лесовая В.Н.	Қант	50	01.02.2013

6.25-сурет. ALL негізгі сөзін қолдану мысалы

SOME (немесе ANY) — салыстырмалы мән ішкі сұрау арқылы қайтарылған кем дегенде бір мәнмен қажетті қатынаста болған жағдайда, яғни ішкі сұраудың нәтижелік бағандағы аз дегенде бір мән үшін орындалған кезде іздеу шарты шынайы болады.

Ішкі сұрауды орындау нәтижесінде бос мән алынса, онда ALL негізгі сөзі үшін салыстыру шарты орындалған, ал ANY негізгі сөзі үшін орындалмаған деп есептеледі. SOME негізгі сөзі ANY сөзінің синонимі болып табылады.

Жоғарыда аталғанды мысалдармен дәлелдеп көрейік. Сатылатын тауар бірлігі орташа мәннен артық тауарларды сатудың барлық деректерін айқындайық (6.25-сурет):

```
SELECT * FROM САТЫЛЫМДАР C1 WHERE
C1.Саны>ALL (SELECT AVG(C2.Саны)

FROMСАТЫЛЫМДАРC2 GROUPBYC2.Клиент)
```

HAVING ұсынысын және ALL негізгі сөзін қолданудың тағы бір мысалы. Тауарлардың айтарлықтай санын сатып алған клиент туралы мәліметтерді көрсету қажет болса:

```
SELECT К.*

FROM КЛИЕНТТЕР К WHERE К.Клиент =

(SELECT С.Клиент FROM САТЫЛЫМДАР С GROUP
```

	212585832187 Мәскеу	3021402

6.26-сурет. HAVING ұсынысы мен ALL негізгі сөзін қолданудың нәтижесі

Нөмір	Клиент	Тауар	Саны	Күні
► 45	Федорова Д.С.	"Коровка" кәмпиттері	50	02.04.2013
156	Газимова К.К. ЖК	"Мишка" кәмпиттері	30	02.04.2013
200	Лесовая В.Н.	"Кроха" мармелады	20	07.02.2013
85	Лесовая В.Н.	Бидай ұны	50	02.04.2013
112	"Түймедақ" ЖАҚ	Халва	40	15.02.2013
254	Лесовая В.Н.	Қант	50	01.02.2013
144	Таран О.С.	Шұжықтар	30	01.04.2013

6.27-сурет. SOME негізгі сөзін қолдану мысалы

Сатылатын тауар бірлігінің саны кем дегенде бір тауарды сатудың орташа мәнінен артық болатын тауарларды сатудың барлық деректерін атап өтейік (6.27-сурет):

```
SELECT * FROM САТЫЛЫМДАР C1 WHERE
C1.Саны>SOME (SELECT AVG(C2.Саны)

FROMСАТЫЛЫМДАРC2 GROUPBYC2.Клиент)
```

EXISTS және NOTEXISTS негізгі сөздері тек қана ішкі сұраулармен бірге қолдануға арналған. Оларды өңдеу нәтижесі TRUE немесе FALSE логикалық мәнді құрайды. Ішкі сұрау арқылы қайтарылатын нәтижелі кестеде кем дегенде бір жол орналасқан жағдайда ғана EXISTS негізгі сөзі үшін нәтиже TRUE тең.Алынған сұраудың нәтижелі кестесі бос болса, EXISTS әрекетін өңдеу нәтижесі FALSE мәні болады. NOT EXISTS кілт сөзі EXISTS кілт сөзіне қарсы өңдеу ережелерін қолданады. EXISTS және NOT EXISTS кілт сөздері нәтижесінде алынған ішкі сұрау кестесінде жолдардың болуы ғана тексерілгендіктен, бұл кестеде бағандардың ерікті саны болуы мүмкін. EXISTS кілт сөзін пайдаланып, кемінде бір рет тауарды сатып алған барлық клиенттер тізімін құруға болады (6.28-сурет):

```
SELECT *

FROM КЛИЕНТТЕР KWHERE EXISTS
```

Клиент	ЖСН	Қала	Телефон
"Ромашка"ЖАҚ	025501244555	Мәскеу	3658815
Таран О.С.	360224005454	Мәскеу	1215648
Федорова Д.С.	500255510055	Санкт-Петербург	4449702
Лесовая В.Н.	212585832187	Мәскеу	3021402
Газимова К.К. ЖК	025587978766	Мәскеу	6521588

6.28-сурет. EXISTS кілт сөзін қолдану нәтижесі

Клиент	ЖСН	Қала	Телефон
"Горизонт" ЖАҚ	025500123152	Мәскеу	4890605
Привалов И.И. ЖК	025501025666	Санкт-Петербург	2560245
"Перевал" ЖАҚ	145889898622	Тверь	221588
Иванов Ф.И. ЖК	025558000554	Мәскеу	1155488
БМЖТ	555878970025	Санкт-Петербург	4353822
Дремина Е.Е.	025578721058	Мәскеу	6582209

6.29-сурет. NOTEXISTS қолдану нәтижесі

```
(SELECTC.САТЫЛЫМДАР

FROMСАТЫЛЫМДАРС

WHEREК.Клиент = С.Клиент)
```

Сатып алуды жасамаған клиенттер тізімін алу үшін, NOT кілт сөзін сол сұрауға енгізу қажет (6.29-сурет):

```
SELECT *

FROM КЛИЕНТТЕРКWHERE NOT EXISTS

(SELECT С.САТЫЛЫМДАР

FROM САТЫЛЫМДАРС

WHEREК.Клиент = С.Клиент)
```

Егер іздеу шартында тек қана бір мәнді қайтаратын кестеден тек жазбаларды таңдау қажет болса, SINGULAR ұсынысы көрсетіледі. Алдыңғы сұрауды сәл өзгертіп, мүлдем басқа нәтиже алуға болады (6.30-сурет):

```
SELECT *

FROM КЛИЕНТТЕРКWHERE SINGULAR

(SELECT С.САТЫЛЫМДАРFROM САТЫЛЫМДАРСWHERE
К.Клиент = С.Клиент)
```

Сұраулыорындаунәтижесітекбірғанатауартүрінсатыпалғанбарлы

ООО "Ромашка"	025501244555	Мәскеу

6.30-сурет. SINGULAR кілт сөзін қолдану нәтижесі

Бұл жағдайда бірінші және екінші кестелердің қатарындағы декарттық көбейтінді құрастырылған және бірлестіктің жағдайын қанағаттандыратын жазбалар деректер жинағынан таңдалады.

Қосылған кестелерінің тағы бір түрі бар - сыртқы байланыс. Бұл келесі ерекшелікке сәйкес FROM ұсынысында анықталады:

```
SELECT . ..

FROM <1 кесте><бірігу түрі>JOIN <2 кесте>

ON <бірігу жағдайы>
```

Сыртқы байланыс ішкі байланыстан ондағы кестелердің біреуі жетекші кесте болуымен ерекшеленеді. Деректердің нәтижелі жиынтығына басқа кестенің бос көптеген жазбаларымен біріге алатын байланыстың жетекші кесте жазбалары енгізіледі.

Сол жақтағы сыртқы байланыс(LEFT) — бірінші кесте жетекші болып табылатын қосылым, FROM ұсынысында қосылым түрінің сол жағында орналасады.

Тиісінше, екінші кестенің жалпы бағандарында бірдей мәндері жоқ жетекші (бірінші) кестенің жолдары деректердің нәтиже жиынына қосылады.

Оң жақтағы сыртқы байланыс(RIGHT) — FROM ұсынысында қосылым түрінен оң жақта орналасқан екінші кесте жетекші болып табылатын байланыс. Нәтижелі қатынаста оң жақтағы кестенің барлық жолдары орналасады.

Толық сыртқы байланыс(FULL) — барлық кестелер жетекші кестелер болып табылатын қосылым.

Деректер жиынтығы екі кестенің барлық жазбаларын қамтиды.Қосылымның жағдайын қанағаттандыратын бірінші кестенің жазбасы үшін екінші кестенің жазбалары бар болса, екі кестенің осындай жазбалары тіркесімінің барлық комбинациясы алынған деректер жиынына қосылады. Әйтпесе, бос жазбаға қосылған бірінші кестенің жазбасы деректердің нәтиже жинағына қосылады. Екінші жағынан, қосылым жағдайын қанағаттандыратын

Нөмір	Клиент	Тауар	СаныКүні Клиент!			ИНН	Қала	Телефон
<null		<nj>	<null>	<null>	ООО	025500123152	Мәскеу	489060
<	<null>	<null>	<null>	<null>	Привалов И.И.	0255010256	Санкт-	256024
^					ЖК "Ромашка"	66	Петербург	5
^	ООО	Халва	40	15.02.201	ЖАҚ "Перевал"			
	"Ромашка"			null><null	ЖАҚ Иванов	0255012445	Мәскеу	365881
	<null>	<null>	<null>	>	Ф.И. ЖК Таран	55		5
				04.01.201	О.С.		Тверь	
	<null>	<null>	<null>	3		1458898986		221588
					Таран О.С.			
200	Лесовая В.Н,	"Кроха"	20	07,02,201	Лесовая В.Н.	2125858321	Мәскеу	302140
85	Лесовая В.Н,	Бидай ұны	50	02,04,201	Лесовая В.Н.	2125858321	Мәскеу	302140
254	Лесовая В.Н.	Қант	50	01,02,201	Лесовая В.Н.	2125858321	Мәскеу	302140
<null	<null>	<null>	<null>	<null>	БМЖТ	5558789700	Санкт-	435382
<null	<null>	<null>	<null >	<null>	Дремина Е,Е,	0255787210	Мәскеу	658220
156	Газимова К.К.	"Мишка"	30	02.04.201	Газимова К.К.	0255879787	Мәскеу	652158
140	Газимова К.К.	"Коровка"	15	09.01.201	Газимова К.К.	0255879787	Мәскеу	652158

6.31-сурет. КЛИЕНТТЕР мен САТЫЛЫМДАР кестелерінің сыртқы сол жақтағы байланыс нәтижесі мынадай нәтижелі деректер жинағының құрылуына алып келеді (6.31-сурет): **ИИН - ЖСН**

```
SELECT САТЫЛЫМДАР.*, КЛИЕНТТЕР.*
FROM КЛИЕНТТЕР LEFT JOIN САТЫЛЫМДАР ON
САТЫЛЫМДАР.Клиент = КЛИЕНТТЕР.Клиент
```

Нәтижеден көріп отырғандай, КЛИЕНТТЕР кестесінің кейбір жолдарында қосылым күйін қанағаттандыратын САТЫЛЫМДАР кестесінде жұпталған жазбалар жоқ. Сондықтан осы КЛИЕНТТЕР кесте деректері бос жазбалармен бірге көрсетіледі.

Сол сияқты, оң жақтағы сыртқы бірігу де жүзеге асырылады:

```
SELECT САТЫЛЫМДАР.*, КЛИЕНТТЕР.*

FROM КЛИЕНТТЕР RIGHT JOIN САТЫЛЫМДАР ON
САТЫЛЫМДАР.Клиент = КЛИЕНТТЕР.Клиент,
```

және толық сыртқы бірігу де жүзеге асырылады:

```
SELECT САТЫЛЫМДАР.*, КЛИЕНТТЕР.*
FROM КЛИЕНТТЕР FULL JOIN САТЫЛЫМДАР ON
САТЫЛЫМДАР.Клиент = КЛИЕНТТЕР.Клиент
```

Кейде SELECT операторларынан кейін қайтарылған деректердің екі немесе одан да көп жиынтығын біріктіру қажет.Мұндай біріктіру UNION операторы арқылы орындалады. Біріктірілген деректер жиынтығы бірдей құрылымға, яғни бірдей өріс құрамына (түрі, өлшемі) ие болуы тиіс.Егер алынған деректер жинақтарында бірдей жазбалар болса, біріктірілген жиында бір жолға жазылады.

```
SELECT САТЫЛЫМДАР.*, КЛИЕНТТЕР.*

FROM КЛИЕНТТЕР FULL JOIN САТЫЛЫМДАР
ON САТЫЛЫМДАР.Клиент =
КЛИЕНТТЕР.Клиент.
```

SELECT операторын пайдалана отырып, әртүрлі кестелердегі

6.4. ДЕРЕКТЕРДІ ӨЗГЕРТУ. INSERT,UPDATE,DELETE ОПЕРАТОРЛАРЫ

Дерекқорды пайдаланушы деректерімен толтыру және бұрыннан бар деректерді өзгерту және жою үшін, DML шағын бөлігінің SQL операторларын пайдалануға болады.Операторлар дерекқордың деректерімен не істеу керек екендігін мұны қалай істеу керектігін нақты көрсетпей сұрайды. INSERT операторы жаңа жолдарды кестелерге немесе дерекқор көріністеріне қосу үшін пайдаланылады. Дерекқор кестелеріндегі деректерді өзгерту үшін UPDATE операторы пайдаланылады. Кестелердің жолдарын жою үшін DELETE операторы қолданылады.

Деректерді өзгерту бойынша барлық әрекеттер белгілі бір транзакция контекстінде (басқаруында) орындалады.Бұл SET TRANSACTION операторы арқылы қажетті сипаттамалармен немесе деректермен және метадеректер дерекқорымен кез келген операцияларды орындау кезінде жүйе автоматты түрде бастамаған әдепкі транзакциямен алдын ала басталған транзакция болуы мүмкін.

SQL тілі жазбалар тобында операцияларды орындауға бағытталған, бірақ кейбір жағдайларда операция жеке жазбада орындалуы мүмкін.Сондықтан, жалпы жағдайда жазбаларды қосу, өзгерту және жою туралы операторлар жазбалардың топтарына сәйкес операциялар жасайды.

Тұрақты кестеге немесе көріністің негізінде жатқан кестеге жаңа жолдарды қосу INSERT операторымен жасалады. Оның синтаксисі келесі түрде беріледі:

```
INSERT INTO <нысан>

[(1 баған[, 2 баған . ..])]

{VALUES (<мән>[,<2 мән>...]) |
```

Бағандардың тізімі алынып тасталуы мүмкін. Бұл жағдайда нысанның барлық бағандары және олар осы нысанда анықталған тәртіпте көрсетіледі.

Мәндердің тізімдерін екі бағанға екі жолмен қоюға болады: VALUES сөзінен кейін мәндерді нақты көрсетуге және SELECT операторын пайдаланып мәндер тізімін жасауға болады.

Мағлұматтар тізімінің анық көрсеткіші бір жазбаны қосу үшін қолданылады және ол келесі түрде беріледі:

```
INSERT INTO <нысан>

[(1 баған[, 2 баған . ..])]

VALUES (<мән>[,<2 мән>...])
```

Мәндер бағандарға екі және басқалардағы тәртіпте тағайындалады: бірінші бағанға бірінші мән, екінші баған екінші мән және т.с.с. Мысалы, келесі оператормен ТАУАРЛАР кестесіне жаңа жазбаны қосуға болады:

```
INSERT INTO
ТАУАРЛАР (Тауар,
ӨБ, Баға)

VALUES («Қант», «кг.», 100)
```

ТАУАРЛАР кестесінің бағандары толығымен және ТАУАРЛАР кестесін жасаған кезде тізілген тәртіпте келтірілгендіктен, оператор жеңілдетілуі мүмкін:

```
INSERT INTO ТАУАРЛАР

VALUES («Қант», «кг.», 100)
```

INSERT операторының екінші формасы ретінде келесі форматты атауға болады:

```
INSERT INTO <нысан>

[(1 баған[, 2 баған . . .])]

<SELECT операторы>
```

Бұл жағдайда, бағандарға тағайындалған мәндер SELECT

CREATE TABLE САТЫЛЫМДАР (НөмірINTEGER
NOT NULL,

КлиентVARCHAR(20) NOT NULL COLLATE PXW_CYRL,

ТауарVARCHAR(20) NOT NULL COLLATE PXW_CYRL,
СаныINTEGER NOT NULL,

КүніDATE NOT NULL,

PRIMARY KEY (Нөмір)

Осы кестеден деректерді күнде белгілі бір аумақ бойынша қашықтықтағы дерекқорға, мысалы САТЫЛЫМДАР_ГЛ, енгізу қажет деп болжам жасайық. Онда САТЫЛЫМДАР кестесінен САТЫЛЫМДАР_ГЛ кестесіне жазбаларды күнде енгізу оператор арқылы жүзеге асырылады:

INSERT INTO САТЫЛЫМДАР_ГЛSELECT *

FROM САТЫЛЫМДАР WHERE Күн = <күн>,

мұнда<күн> — ағымдағы күннің мәні.

Әдеттегі кестеде немесе өзгеретін көрініс үшін негізгі болып табылатын кестеде қолданыстағы жолдар бағандарындағы деректерді өзгерту үшін UPDATE операторы қолданылады. Оператор бір жүгіну кезінде кестеде немесе көріністе барлық жолдардағы деректерді немесе жолдардың бір бөлігін ғана өзгертуге мүмкіндік береді. Оның синтаксисі келесі түрде ұсынылады:

UPDATE <нысан>

SET 1 баған=<1 мән> [,2 баған=<2 мән>...] [WHERE <жағдай>]

Аталған бағандардың әрқайсысын өзгерткенде, сәйкес мән тағайындалады. SELECT операторында көрсетілгендей шартты қанағаттандыратын барлық жазбалар үшін өзгертулер жасалады.

SET ұсынысы орындалатын өзгертулердің тізімін анықтайды: өзгертілетін бағанның аты және теңдік белгісінен кейін бағанның

```
UPDATE САТЫЛЫМДАР SET Баға=Баға*2 WHERE Тауар =
«Ұн»
```

ТАУАРЛАР кестесіндегі оператор қызметінің нәтижесі ретінде Баға өрісіндегі Тауар өрісінің мәні «Ұн» тең болатын барлық жазбаларда екі еселенген болады.

DELETEоператоры нысаннан жазбалар топтары жоюға арналған. Жекелеген ажғдайда бір ғана жазба жойылуы мүмкін.

DELETE операторының форматы:

```
DELETE FROM <нысан>
[WHERE <жағдай>]
```

Іздеу шартына сәйкес келетін барлық жазбалар жойылады. WHERE ұсынысын алып тастаса, барлық жазбалар нысаннан жойылады.

Төменде кантты сату деректері туралы барлық жазбалар

6.5. САҚТАЛАТЫН РӘСІМДЕР ЖӘНЕ ТРИГГЕРЛЕР

6.5.1. Сақталатын рәсімдер мен триггерлердің тілі

Сақталған рәсімдер мен триггерлер дерекқордың метадеректер аймағында сақталған бағдарламалар болып табылады. Олар сервер жағында орындалады, көптеген жағдайларда жүйе ресурстарын үнемдейді және желілік трафикті азайтады.Сақталатын рәсімдерге сақтаулы рәсімдерден, триггерлерден және клиенттік ұсыныстардан жүгіну мүмкін. Триггерлерге тікелей жүгіну мүмкін емес, себебі олар кесте үшін нақты жағдайдың (деректер өзгеруінің) орын алуы кезінде немесе дерекқор оқиғасы жағдайында автоматты түрде шақыртылады. Кестенің әрбір оқиғасы үшін екі фаза орындалады: сәйкесінше оқиғаның пайда болуына дейін (BEFORE) және осы оқиғаның орын алуынан кейін (AFTER).

Сақталған рәсімдерде және триггерлерде деректерді өңдеуге арналған алгоритмдерді сипаттау үшін SQL тілінің кеңейтімі

Бұл кеңейтілім рәсімді SQL (PSQL) немесе сақталатын рәсімдер мен триггерлер тілі деп талады. Тіл тіркеу, тармақтар мен циклдердің әдеттегі операторларын қамтиды. Триггерлерде ерекше мәнмәтінді айнымалылар қолданылуы мүмкін.

Сақталған рәсімдер мен триггерлер тілі классикалық бағдарламалау тілдерінің барлық негізгі құрылымдарын қамтитын қарапайым бағдарламалау тілі болып табылады.Сонымен қатар, ол дерекқор кестелері (INSERT, UPDATE, DELETE және SELECT) деректерді қосу, өзгерту, жою және алу үшін бірнеше өзгертілген операторларды қамтиды. Тілдегі метадеректердің ешбір өзгерісін орындауға болмайды.

Сақталған рәсімдер мен триггерлерде белгілі бір оқиғалар (events) туралы клиенттерге хабарлауға, пайдаланушылық ерекшеліктерді шығаруға мүмкіндік беретін құралдарды пайдалануға болады. Дерекқорын қосу әрекеттерін орындауға жол берілмейді, транзакцияны бастауға, транзакцияларды, құтқару жолдарын жасауға, құтқарушыларға оралуға, транзакцияларды растауға немесе жоюға болмайды. Сақталған рәсім мен триггер сақталатын рәсім тікелей анықталған транзакция мәнмәтінде немесе деректерді өңдеу операциясы дерекқорда орындалады, нәтижесінде триггер автоматты түрде іске қосылады. Егер триггер дерекқордан қосылып, дерекқордан ажыратылған кезде шақырылса, ол үшін транзакция әдепкі бойынша іске қосылады.

Сақталған рәсімдер мен триггерлер жасау синтаксисінде тақырып пен құралды таңдауға болады. *Тақырып* бағдарлама нысанының атауын, жергілікті айнымалылардың сипаттамасын қамтиды.Триггерлер үшін тақырып дерекқор оқиғасын және триггер автоматты түрде іске қосылатын кезеңді анықтайды. Сақталған рәсімнің тақырыбында кіріс және шығыс параметрлерін көрсетуге болады.Сақталған рәсімнің немесе триггердің құралы бұл бағдарламамен орындалатын әрекеттердің сипаттамасын қамтитын ұсыныстар блогы болып табылады.

Операторлар блогы оператордың BEGIN және END жақшаларынан тұрады. Бағдарламаларда өздерінің кез-келген блоктарының екеуі бірізді және бір-біріне салынған болуы мүмкін.

Сақталған рәсімдер мен триггерлердегі деректерді іріктеу, өңдеу және өзгерту үшін барлық әрекеттер оператор блогында орындалады, ол BEGIN және END операторының жақшаларынан тұрады.

үтір) SQL операторы болып табылмайтын SET TERM операторы қолданылады. Бұл жалған оператордың көмегімен триггер немесе сақталған рәсімді жасаудан бұрын триггер немесе сақталған рәсім мәтінінің соңында соңғы таңба болып табылатын нышан көрсетіледі.Сәйкесінше бағдарламалық нысан мәтінін сол SET TERM операторын қолданып сипаттағаннан кейін, терминатордың мәні әдеттегі нұсқаға - үтірмен нүктеге қайтарылады.

Мысалы, белгілі бір триггерді құру кезінде келесі операторды орындау қажет:

```
SET TERM л;

/* ТАУАРЛАР кестесінің бастапқы кілтінің мәнін қалыптастыру */

CREATE TRIGGER TBI_STAFF FOR ТАУАРЛАРBEFORE
INSERT ... [Триггер мәтіні}

END л
```

Блокта операторлар дәйекті түрде орындалады. SQL кеңейтілімдерінде тармақтандыру (IF-THEN-ELSE), цикл (WHILE-DO, FOR-SELECT-DO, FOR EXECUTE STATEMENT), дерекқордағы деректерге жүгіну (деректерді іріктеу, қосу, өзгерту, жою) операторлары орналасады.

Бос орынға рұқсат етілген мәтіндегі кез келген жерде /* және */ таңбалар арасында орналасқан түсініктемелерді енгізуге болады. Осындай бір түсініктеме жолдардың кез келген санын алуы мүмкін. Түсініктемелердің басқа да түрі бар: қатарынан келетін екі минус (-) белгісі. Түсініктеме мәтіні бұл жағдайда ағымдағы жолдың аяғына дейін ғана жалғасады.

Бір жергілікті айнымалыны сипаттау үшін DECLARE VARIABLE операторы қолданылады. Оның жеңілдетілген синтаксисі келесі түрде ұсынылуы мүмкін:

Бір операторда тек бір жергілікті айнымалы мәнді жариялауға болады. Іске қосу және сақтау процедурасында әрбір айнымалы үшін жеке DECLARE VARIABLE операторын пайдаланып, жергілікті айнымалы мәндердің ерікті санын жариялауға болады.Жергілікті айнымалы атауы жергілікті бағдарлама нысанында сақталатын рәсімінің шегінде кіріс және шығыс параметрлерінің жергілікті айнымалы атауларының арасында

Деректер түрі SQL пайдаланылатын деректердің кез келген түрі болуы мүмкін. Деректер түрінің орнына бұрын құрылған доменнің атауын көрсетуге болады.

6.5.2. Триггерлермен жұмыс

Триггер метадеректқор аймағында сақталған және сервер жағында орындалатын бағдарлама болып табылады. Триггерге тікелей кіру мүмкін емес.Бір нақты кестеге (көрініске) қатысты бір немесе бірнеше оқиғалар, немесе деректқор оқиғаларының бірі болғанда, ол автоматты түрде шақырылады. Кестенің оқиғасы пайда болған кезінде шақыртылатын триггер бір кестемен немесе көрінісмен, осы кесте немесе көрініс үшін бір немесе одан да көп оқиғамен (деректерді қосу, өзгерту немесе жою) және осындай оқиғаның бір ғана фазасымен (оқиғаның пайда болуына дейін немесе одан кейін) байланысты. Триггер транзакция контекстінде тиісті оқиғаға себеп болған бағдарлама аясында орындалады. Ерекшеліктер деректқор оқиғаларына жауап беретін триггерлер болып табылады. Кейбіреулер үшін әдепкі транзакция басталады.

Кестеге (көрініске) арналған «оқиға - фаза» қатынасының алты нұсқасы бар:

- жаңа жолды қосуға дейін (BEFORE INSERT);
- жаңа жолды қосқаннан кейін (AFTER INSERT);
- жолды өзгертуге дейін (BEFORE UPDATE);
- жолды өзгерткеннен кейін (AFTER UPDATE);
- жолды жоюға дейін (BEFORE DELETE);
- жолды жойғаннан кейін (AFTER DELETE).

Деректқор оқиғаларына байланысты триггер келесі оқиғаларда шақырылуы мүмкін:

- деректқорға қосылу кезінде (CONNECT); триггер орындалмас бұрын әдепкі транзакция автоматты түрде басталады;
- деректқордан ажыратылған кезде (DISCONNECT); триггер орындалмас бұрын әдепкі транзакция басталады;
- транзакция басталған кезде (TRANSACTION START); триггер бұл транзакция контексінде орындалады;
- транзакция расталған кезде (TRANSACTION COMMIT); триггер бұл транзакция контексінде орындалады;
- транзакция тоқтатылған кезде (TRANSACTION ROLLBACK); триггер транзакция контексінде орындалады.

Бір кесте (көрініс), бір фаза және бір оқиға, сондай-ақ бір фаза және бірнеше оқиғалар үшін автоматты түрде триггерлер жасауға

Егер бір кестеге (көрініске), бір оқиғаның және бір фазаның бірнеше триггерлері болса, онда оларды орындау тізбегін, осы тізбектегі триггерлік орнын көрсетуге болады.

Триггер жасау үшін синтаксисі төменде көрсетілген CREATE TRIGGER операторы пайдаланылады:

CREATE TRIGGER <триггер атауы>

[ACTIVE | INACTIVE}

{ ON <дерекқор оқиғасы>

| FOR {<кесте атауы> | <көрініс атауы^

{BEFORE | AFTER}

<кесте (көрініс) оқиғасы>

[OR <кесте (көрініс) оқиғасы>] . ..

| {BEFORE | AFTER}

< кесте (көрініс) оқиғасы>

[OR < кесте (көрініс) оқиғасы>] . ..

ON {<кесте атауы> | <көрініс атауы>}

Триггербелсенді (ACTIVE) немесебелсендіемес (INACTIVE)болуымүмкін. Егертриггербелсенді (әдепкімән) болса, олкестеніңнемеседерекқордыңсәйкесоқиғасы (лары) келгендеа втоматты түрде шақырылады. Егер триггер белсенді болмаса, триггер пайда болмайды.

Триггер құралының құрылымы:

[<жергілікті айнымалыларды жариялау>]

BEGIN

<оператор>

Жергілікті айнымалы мәндерді триггерлер құралында сипаттағаннан кейін, BEGIN және END операторының жақшаларындағы тұжырымдар блогы керек.

Триггерлер үшін OLD және NEW жаңа контекст ауыспалылығы бар. Осы кілт сөздердің неғұрлым дұрыс атауы - баған атауы префикс.Триггерлерде оның клиенттік бағдарламада өзгеруіне дейін (бұл үшін баған атауының алдында OLD кілт сөзі және нүкте орналастырылады) және өзгеруінен кейін (баған атауының алдында NEW және нүкте) кестенің (көріністің) кез келген баған мәніне жүгінуге болады.

Триггерлердің барлық түрлері үшін мәнмәтін айнымалы OLD (бағанның атауының префиксі) тек оқуға арналған. Оқиға фазасына қарамастан деректерді қосудан туындаған триггерлерде қол жетімді емес.

Оқиғалар кезеңі үшін AFTER кейінгі айнымалы мәндердегі NEW мәнмәтінді айнымалысы да оқуға арналған айнымалы болып табылады. Деректерді жою оқиғасы үшін триггерлерде қол жетімді емес.

OLD. БағанАтауы мәні кез-келген ықтимал өзгертулер жасалғанға дейін болған бағанның күйіне сілтеме жасауға мүмкіндік береді.

NEW. БағанАтауы мәні мүмкін болатын өзгерістер енгізілгеннен кейін бағанның күйіне енуге мүмкіндік береді.

Мысал:

```
CREATE TRIGGER BU_TOVARY FOR TOVARY
ACTIVE

BEFORE UPDATE AS

BEGIN

IF (OLD.TOVAR<>NEW.TOVAR) THEN UPDATE
PRODAJA SET TOVAR=NEW.TOVAR WHERE
TOVAR=OLD.TOVAR;

END
```

Осы мысалда триггер PRODAJAкестесіне TOVARYкесте жазбаларында TOVARбағанының мәні өзгерсе сәйкесінше өзгерістерді енгізеді.

Қолданыстағы триггерді жою үшін келесі оператор қолданылады

DROP TRIGGER <триггер атауы>

Триггерлер дерекқор деректерін өзгерткенде әртүрлі әрекеттерді орындау кезінде пайдалы құрал болып табылады.Триггерлердің негізгі мақсаты жасанды бастапқы кілттің мәндерін қалыптастыру, деректерді өзгертуге қатысты дерекқордың ақпараттық хабарламаларын беру, декларативті деректер тұтастығын сақтаудың арнайы жолдары және дерекқордың кейбір деректерін қосу (өзгерту) кезінде белгілі бір әрекеттерді орындау болып табылады.Триггерлерді дерекқорға қосылу, дерекқордан ажыратылу, транзакцияларды іске қосу, растау немесе жою сияқты дерекқормен байланысты оқиғалар болғанда да қолдануға болады

6.5.3. Сақталатын рәсімдермен жұмыс

Сақталған рәсім, сондай-ақ триггер дерекқордың метадеректер аймағында сақталған және сервер жағында іске қосылған бағдарлама болып табылады.Триггерден айырмашылығы сақталған рәсімдер, триггерлер және клиент бағдарламалары сақталған рәсімдерге жүгіне алады. Рұқсат етілген рекурсия - сақталатын рәсім өзіне сілтеме жасай алады. Сақталған рәсімдер шақыру бағдарламалары сияқты бір транзакция контексінде орындалады.

Сақталған рәсімдердің екі түрі бар - орындалатын сақталған рәсімдер (executed stored procedures) және таңдаудың сақталатын рәсімдері (selected stored procedures).

Орындалатын сақталатын рәсімдер дерекқордағы немесе дерекқормен байланыстырылмаған деректерді өңдеуді орындайды. Бұл рәсімдер енгізу параметрлерін ала алады және шығыс параметрлерін қайтарады.Орындалатын сақталатын рәсімдерге қол жеткізу SQL EXECUTE PROCEDURE операторын орындау кезінде жүзеге асырылады.

Сақталған іріктеу рәсімдері, әдетте, қабылданған жолдардың ерікті санын қайтара отырып, дерекқордан деректерді алуды жүзеге асырады.Таңдау рәсімдері енгізу параметрлерін де ала алады. Әрбір келесі оқу жолының мәні шығыс параметрлеріндегі шақыру бағдарламасына қайтарылады. Бұл рәсімнің орындалуын уақытша тоқтату және таңдалған деректерді шақыру бағдарламасына сақталған рәсімге қайтару үшін SUSPEND операторы

Сақталған рәсімге қол жеткізу SELECT операторын пайдалану арқылы орындалады.

Синтаксистік түрде сақталатын рәсімнің сақтайтын рәсімді жасаудан ешқандай айырмашылығы жоқ. Сақталған рәсімді жасау үшін синтаксисі төменде көрсетілген CREATE PROCEDURE операторы қолданылады:

```
CREATE PROCEDURE <сақталатын_рәсім_атауы>

[AUTHID {OWNER | CALLER}]

[(<кіріс өлшемдерінің тізімі>)]

[RETURNS (<шығыс өлшемдерінің тізімі>)]

AS

[<айнымалыларды жариялау тізімі>]

BEGIN <операторлар блогы>END
```

Сақталған рәсім шақырылатын бағдарламадан енгізу өлшемдерін қабылдай алады. Өлшемдер мәнмен қабылданады, яғни кіріс өлшімдері мәндеріндегі кез келген өзгерістер шақырылған бағдарламада осы өлшемдер мәніне әсер етпейді.Енгізу өлшемдеріне әдепкі қалпы бойынша мән берілуі мүмкін. Әдепкі қалпы бойынша мәндер орнатылатын өлшемдер тізімнің ең соңында орналасуы керек.Егер кіріс өлшемі DEFAULT ұсынысында әдепкі қалпы бойынша мән бар доменге негізделген болса, жаңа әдепкі қалпы бойынша мән домен сипаттамасында көрсетілгенді қайта анықтайды.

Сақталған рәсім шығыс өлшемдерінің ерікті санын шақыру бағдарламасына қайтара алады.

Егер доменнің айнымалы аты рәсімнің жергілікті айнымалы өлшемінің сипаттамасында көрсетілсе, онда осы доменнің барлық сипаттамалары көшіріледі. Сақталған рәсім құралында жергілікті айнымалылардың ерікті саны сипатталуы мүмкін.

Сақталған рәсімдерде кіріс және шығыс өлшемдері де сипатталуы мүмкін. Кіріс өлшемдер тізімі сақталған рәсімнің

Шығу өлшемінің сипаттамасы кіріс өлшемінің сипаттамасына сәйкес келеді.

Триггерлер мен рәсімдердің жергілікті өлшемдері, сақталатын рәсімдерде ғана қолданылатын енгізу және шығару өлшемдері ішкі айнымалылар болып табылады. Ішкі айнымалылардың атаулары пайдаланылатын дерекқор кестелерінің баған атаулары сияқты болуы мүмкін. Бұл атаулардың белгісіздігіне себеп болмайды.Ішкі ауыспаларды SELECT, INSERT, UPDATE немесе DELETE операторларында қолданғанда, оларды кесте бағанының атауларымен шатастырмау үшін ішкі айнымалы мәндердің атаулары алдында әрқашан қос нүкте болуы керек. Барлық басқа жағдайларда кез-келген басқа операторларда ішкі айнымалылардың атаулары әдеттегі жолмен қос нүктесіз жазылады.

Ішкі айнымалының мәнін тағайындау синтаксисі келесідей:

```
<айнымалы атауы> = <мән>;
```

Өрнек ретінде кез келген дұрыс SQL мән болуы мүмкін. EXIT операторы триггердің кез келген нүктесінен немесе сақталған рәсімді түпкілікті END операторына ауысуға, яғни бағдарламаны аяқтауға мүмкіндік береді:

```
EXIT [<белгі>]
```

SUSPEND операторы сақталатын рәсімнің орындалуын уақытша тоқтатады (кестенің көптеген жолдарын таңдайтын SELECT операторын қамтушы, рәсімге жүгіну SELECT операторымен орындалатын рәсім) және шығыс өлшемдерін шақыру бағдарламасына жібереді.

Триггерлер және сақталған рәсімдер EXECUTE PROCEDURE операторын пайдаланған кезде сақталатын рәсімдерді шақырта алады:

```
EXECUTE PROCEDURE <рәсім атауы>

[(<өлшем> [, <өлшем>] . ..)]

[RETURNING_VALUES (<өлшем>[,< өлшем>]...)
```

FORSELECTDOоператорының жұмыс алгоритмі келесі әрекеттерді қамтиды. SELECT операторы орындалады және нәтижелі деректер жинағының әр жолы үшін DO сөзінен кейінгі оператор орындалады.Бұл оператор жиі SUSPEND болып табылады, бұл шығыс өлшемдерін шақыру қосымшасына қайтаруға алып әкеледі.

Әрекеттегі сақталған рәсімді өзгерту үшін ALTER PROCEDURE операторы қолданылады.Әрекеттегі сақталған рәсімді жою үшін DROP PROCEDURE операторы пайдаланылады. Сақталған рәсімдерді жасаудың мысалдарын келтірейік.

Келесі рәсім Тауар_ кіру кіріс өлшемінің мазмұнымен айқындалатын нақты тауарды сатудың барлық деректерін көрсетеді.

```
CREATE PROCEDURE PRODAJA_TOVARA (Тауар_кіруVARCHAR(20))

RETURNS (Күні_шығуDATE, Клиент_шығуVARCHAR(20),

Саны_шығуINTEGER) AS

BEGIN

FOR SELECT Күні, Клиент, СаныFROM Сатылымдар

WHERE Тауар =:Тауар_кіру

INTO: Күні_шығу, :Клиент_шығу,:Саны_шығуDO

SUSPEND;

END
```

Алдымен Тауар өрісі Тауар_кіру өлшеміндегі мәнге тең болатын әрбір кіріс үшін сату кестесінен, сатып алушының атауынан және тұтынылатын тауарлардың санын шығаратын SELECT операторы орындалады. Көрсетілген мәндер шығыс өлшемдерінде жазылады (сәйкесінше Күні_шығу, Клиент_шығу, Саны_шығу). Нәтижелі деректер жиынтығының әрбір жазбасын бергеннен кейін SUSPEND операторы орындалады.Ол шығыс өлшемдерін шақыру бағдарламасына қайтарады және шығыс өлшемдерінің келесі бөлігін шақыру бағдарламасынан сұрағанша рәсімнің орындалуын тоқтатады. Бұл рәсім таңдау рәсімі болып табылады, себебі ол кіріс өлшемдерінің бірнеше мәндерін шақыру бағдарламасына қайтара алады.

Клиенттер_тізім басқа рәсімі тауар бойынша сатып алудың орташа

Егер клиенттің аты бос мән болса, клиенттік атаудың орнына «Деректер жоқ» көрсетіледі.

```
CREATE PROCEDURE Клиенттер_тізім (Тауар_кіруVAR- CHAR(20))

RETURNS (Клиент_кіруVARCHAR(20)) AS DECLARE VARIABLE
Орташа_саныINTEGER;

BEGIN

SELECT AVG(Саны)

FROM Сатылымдар

WHERE Тауар =:Тауар_кіру INTO :Орташа_саны;

FOR

SELECT Клиенттер FROM Сатылымдар

WHERE Саны >: Орташа_саны INTO :Клиент_кіру DO

    BEGIN

    IF (:Клиент_кіру IS NULL) THEN Клиент_кіру =
    «Деректер жоқ»;

    SUSPEND;

    END

END
```

Сақталған рәсімдердің артықшылығы олар бірінші кезекте серверлік жағынан орындалады, бұл көптеген жағдайларда желілік трафикті айтарлықтай төмендетеді, ал екіншіден, жақсы жазылған және тәртіпке салынған сақталған рәсімді көптеген бағдарламаларда сақталатын рәсімдер қолданғаннан кейін көптеген бағдарламалар – сақталатын рәсімдер, триггерлер, клиенттік бағдарламалар қолдана алады.

Сақталған іріктеу рәсімі, әдетте, дерекқордан деректердің жеткілікті санын іріктеу үшін пайдаланылады. Осындай жағдайларда деректерді іріктеу алгоритмі өте күрделі.Мұндай рәсім, әдетте, SELECT операторының декларативті құралы тиісті деректерді кестелерден немесе көріністерден алудың барлық

Алайда, қолданбалы логикамен сақталатын рәсімдерді шамадан тыс жүктеу серверді істен шығаруы мүмкін екендігін естен шығармау қажет, себебі бұл өнімділікті жоғалтуға әкеледі.Бұл мәселе үлкен ақпараттық жүйелерді дамытуда аса маңызды, онда көптеген клиенттер бір мезгілде серверге қол жеткізе алады.Сондықтан көп жағдайда ымыралы шешімдер қабылдау қажет: қосымшаның логикалық бөлігін сервер жағына, ал бір жағынан – клиентке орналастырған жөн.Мұндай клиенттік-серверлік жүйелер логикамен бөлінген жүйелер деп аталады.

6.6. ИНДЕКСТЕРМЕН ЖҰМЫС

Индекс — белгілі бір кестенің көрсетілген бағандарының мәндерін және осы мәндерді қамтушы кестенің жолдарына сілтемелерді қамтитын дерекқор нысаны.

Индекс пайдаланушы немесе жүйе арқылы белгілі бір кесте үшін жасалады, ол көптеген жағдайларда осы кестеде деректерді алу үдерісін жылдамдатуға мүмкіндік береді, ал кейде SELECT операторындағы ORDER BY ұсынысының негізінде пайдаланушы сұрауы бойынша алынған деректердің реттелуін жылдамдатуға мүмкіндік береді.Әрбір индекс жолында индекстің бөлігі болып табылатын бағандардың мәнін және сол баған мәндері бар кестедегі жолға көрсеткішін қамтиды.

Индекстер болған жағдайда, көптеген жағдайларда деректерді индекстеудің болмауына қарағанда тезірек жасауға болады, себебі индекстегі мәндер реттеледі және индекстің өзі салыстырмалы түрде шағын. Мағлұматтардың аз санына ие бағандар үшін индекстер құру қажет емес, мысалы, екі мәнді бағандар үшін, атап айтқанда, бағанда TRUE және FALSE мәндеріне ие болатын логикалық деректер түрін модельдейтін бағандар үшін немесе адамның еркек немесе әйел жынысын енгізу жағдайында. Мұндай индекстер тек сыртқы жадта орын алады және іріктеу әрекеттерін орындау кезінде және деректерді бірізділікке келтіру кезінде өнімділікке ешқандай пайда әкелмейді.

Бастапқы кілттің, бірегей кілттің және сыртқы кілттің шектеулері үшін жүйе автоматты түрде индекстерді құрастырады.

Маңызды ереже - дерекқорды серверден кездейсоқ тоқтатуына әкеліп соқтырған кезде, бастапқы, бірегей немесе сыртқы кілт үшін

Индекс бірегей ретінде жасалуы мүмкін (UNIQUE кілт сөзі). Бұл жағдайда кесте бірегей индексте қамтылған бағандардың мәнімен бірдей екі түрлі жолдардың болуына жол бермейді.

Индекс құрылымында қамтылған бағандардың мәндерін жоғарылату арқылы (ASCENDING - әдепкі қалыптағы мәні бойынша) немесе осы мәндерді (DESCENDING) төмендету арқылы реттелуі мүмкін.Дерекқормен жұмыс барысында кез-келген уақытта индексті белсенді етуі мүмкін (ACTIVE), яғни осы индексте енгізілген кестенің бағандарындағы барлық өзгерістер дерекқордың сәйкесінше кесте жолдарындағы ешбір өзгеріс индекс құрамына әсер етпеген кезде бірден индекстін өзінде, немесе белсенді емес (INACTIVE) болып көрсетіледі.

Әрекеттегі дерекқор кестесіне арналған индексті құру үшін келесі оператор қолданылады:

CREATE [UNIQUE] [ASC[ENDING] | DESC[ENDING]]

INDEX <индекс атауы>ON <кесте>

(<баған> [, <баған>] . ..)

Индекс атауы дерекқордағы барлық индекстердің атаулары арасында, сондай-ақ кесте деңгейіндегі шектеулердің атауы мен кесте деңгейіндегі шектеулер арасында бірегей болуы керек.Бастапқы, бірегей немесе сыртқы кілт шектеулерін енгізген және CONSTRAINT ұсынысындағы шектеу атауын көрсеткен жағдайда жүйе сол атпен индексті құрастырады.

UNIQUE кілт сөзі бірегей индексті құруды анықтайды, бұл индексте индекстің барлық бағандарының бірдей мәндері бар екі жолға ие болмайтынын көрсетеді. Бірегей индекс бөлігі болып табылатын бағандарда бос NULL мәні болмайды.

ASCENDING кілт сөзі (ASC қысқартылған нұсқасы) индекс жазбаларының индекс құрамына жататын бағандар мәндерінің артуы бойынша бірізділікке келтірілетінін білдіреді. Бұл нұсқа әдепкі қалпы бойынша қабылданады.

DESCENDING кілт сөзі (DESC қысқартылымы) индекс жазбаларының индекс бағандары мәндеренің азаюы бойынша бірізділікке келтірілетінін көрсетеді.

CREATE ASCENDING INDEX ON САТЫЛЫМДАРІ_ASC (Тауарлар) CREATE DESCENDING INDEX ON САТЫЛЫМДАРІ_DESC (Тауарлар)

Алғашқы жасау кезінде индекс әдепкі қалпы бойынша белсенді болады, кестеде барлық жаңадан қосылған жолдар немесе негізгі кестенің индекстелген бағандарындағы өзгертілген өзгертулер бірден индекс жағдайына әскер етеді.

Кейбір жағдайларда индексті «өшіріп», оны бірнеше уақыт белсенді емес ету пайдалы болуы мүмкін. Бұл кестедегі топтық операциялар деп аталатын кезде орындалатын уақытты үнемдеуге болады, егер кестедегі жолдан үлкен жолдар саны жазылған немесе кесте өзгертілсе, немесе жолдың үлкен саны жойылса, кесте жойылады.Мұндай операцияны бастау алдында индекс белсенді емес күйге (INACTIVE) ауысады және операция аяқталғаннан кейін ол тағы да белсенді (ACTIVE) болады. Сонымен қатар индекс белсендірілген кезде индекс толығымен қайта құрылады. Жаңадан енгізілген, өзгертілген немесе жойылған жолдар жаңа индекс күйінде ескеріледі.

Индекс жағдайын өзгерту келесі оператор арқылы жүзеге асырылады:

ALTER INDEX <индекс атауы>{ACTIVE | INACTIVE}

ACTIVE кілт сөзі белсенді емес индексті белсенді күйге орнатады, INACTIVE көрсеткіш белсенді емес күйде екенін білдіреді. Индексті белсенді емес күйден белсендіге ауыстырғаннан кейін, жүйе бүкіл индексті толығымен қайта жасайды.

Бұл оператор индекс құрылымын немесе оның тәртібін өзгерту үшін пайдаланылмайды. Егер индекс құрылымына өзгертулер енгізу немесе бірізділікті өзгерту қажет болса, әрекеттегі индексті жою керек (келесі бөлімді қараңыз), содан кейін сол атаумен және қажетті сипаттамалары бар индексті жасаған жөн.

Пайдаланушы құрған индексті жою үшін келесі оператор пайдаланылады

DROP INDEX <индекс атауы>

Сондықтан негізгі, бірегей немесе сыртқы кілт үшін жүйе автоматты түрде жасаған индексті жоюға болмайды. Пайдаланушы

6.7. ГЕНЕРАТОРЛАР

Г е н е р а т о р (generator) — дерекқордың ең қарапайым нысаны. Генератор ДҚ серверінде сақталатын механизм болып табылады, ол бұрынғы генератормен шығарылған мәндермен ешқашан сәйкес келмейтін бірегей мәндерді береді. Ол мәндердің айтарлықтай кең диапазонында бүтін сандарды сақтауға мүмкіндік береді. Ол үшін жадтың 8 байты бөлінеді.Бұл жасанды бастапқы кілттердің мәндерін қалыптастыру үшін қолайлы құрал. Кез келген дерекқор кестесінің әрбір жасанды бастапқы кілті үшін пайдаланушы өзінің жеке генераторын жасайды, онымен осы бастапқы кілт мәндерін қалыптастыру бойынша барлық іс-әрекеттер орындалады.

Негізінде, генераторлар басқа мақсаттар үшін қайталанбайтын бүтін сандардың қатарын алу үшін пайдаланылуы мүмкін.

Генераторды құру үшін келесі оператор қолданылады:

CREATE GENERATOR <генератор атауы>

Генератор атауы дерекқордың барлық генераторлар атауларының арасында бірегей болуы тиіс.

Генераторды құру кезінде оған 0 мәні беріледі. Осы генератордың атауы қолданылатын GEN_ID қызметіне бұдан кейінгі жүгінулер көрсетілген мөлшерге осы мәнді өзгертеді.Әдетте, өсім ретінде бір болып табылады, бірақ нөлден басқа кез келген басқа бүтін сандарды қолдануға болады (синтаксис ережелері бойынша нөлді де қолдануға болады, бірақ бұл сандардың бірегей тізбегін алуға мүмкіндік бермейді).

Генератор мәнін орналастру келесі оператор арқылы жүзеге асырылады:

SET GENERATOR <генератор атауы>ТА <бастапқы мән>

Бірегей мәнді алу үшін келесі қызмет қолданылады:

GEN_ID (ГенераторАтауы, қадам)

DROPGENERATOR<генератор атауы>

Генератор осы генераторға сілтеме жасайтын барлық триггерлер мен сақталған рәсімдер дерекқордан жойылғаннан кейін ғана жойылуы керек.

Төмендегі мысал жаңа жолды қосқанда генератордың қолданылуын суреттейді. Дерекқорда САТЫЛЫМДАР кестесіндегі Нөмір бағаны үшін генератор анықталған жағдайда:

```
CREATE GENERATOR Ген_нөмір;

SET GENERATOR Ген_нөмірТА0;
```

INSERTоператорынан тікелей генераторқа жүгінуді келесі түрде жазуға болады:

```
INSERT INTO PRODAJA

(Нөмір, Күн, Сан, Тауар, Клиент)

VALUES (

GEN_ID (Ген_нөмір, 1),

'01Т05.20137,

100, «Қант»,

«Түймедақ ЖАҚ»)
```

Триггерліпайдаланаотырып, дерекқордажаңажолдыжазуалдындашақырылатыннегізгібағанғабіре геймәндітағайындаудыіскеасыруғаболады:

```
CREATE TRIGGER BI_PRODAJA FOR САТЫЛЫМACTIVE

BEFORE INSERT AS

BEGIN

NEW. Нөмір= GEN_ID (Ген_нөмір, 1)

END
```

Бірегей мәндерді генерациялау үшін сақталатын рәсімді қолдануға болады:

Бұл дегеніміз, бір мезгілде әртүрлі бәсекелес транзакциялардың бір генераторына қол жеткізгенде ешқашан құлыптау қақтығысы болмайды және әрбір қоса жүретін үрдіс бірегей жаңа сандық мәнге ие болады. Мән генераторға сілтеме жасау уақытына байланысты.

6.8. SQL ТІЛІНІҢ БАСЫЛЫМДЫҚТАРЫ

SQL тілі енді өте кең таралымға ие болды және іс жүзінде реляциялық дерекқордың стандартты тіліне айналды. Дерекқрды басқару жүйелерінің барлық ірі әзірлеушілері қазіргі уақытта өз өнімдерін SQL тілін пайдалана отырып жасайды. Бұған әзірлеушілер мен пайдаланушылар үлкен инвестициялар жасайды.Ол қосымшалар архитектурасының бір бөлігі болды, көптеген ірі және беделді ұйымдардың стратегиялық таңдауы болып табылады. Осылайша, SQL пайдаланушыларға, бағдарламалар мен есептеу жүйелеріне ақпаратқа қол жеткізуге мүмкіндік беретін қуатты құрал ретінде ұсынуға болады.

SQL тілі пайдаланушыға келесі мүмкіндіктерді ұсынады:

- әртүрлі дерекқор нысандарын, олардың құрылымын толық сипаттайтын кестелерді жасау;
- деректерді манипуляциялаудың негізгі әрекеттерін орындау: кестелерден деректерді қосу, өзгерту және жою;
- деректерді түрлендіруді орындау өте күрделі және ауқымды түрлерін қоса алғанда, түрлі сұрауларды орындау. SQL тілі барлық жоғарыда аталған мәселелерді пайдаланушы тарапынан ең аз күш-жігермен шешеді және командаларының құрылымы мен синтаксисі өте қарапайым және оқу үшін қол жетімді екенін атап өткен жөн.

SQL тілінің негізгі жетістіктері:

- стандарттылық - бағдарламаларда SQL тілін қолдану халықаралық ұйымдармен стандартталған;
- белгілі бір ДБЖ-дан тәуелсіздік - барлық кең таралғанДБЖSQL тілін қолданады, өйткені реляциялық дерекқор бір ДБЖ-дан екіншісіне аз өңдеу жұмысы арқылы көшіруге болады;
- бір есептеу жүйесінен басқасына көшіру мүмкіндігі;

- SQL реляциялық негізі реляциялық дерекқорлардың тілі болып табылады, сондықтан ол деректерді ұсынудың реляциялық үлгісі кең таралған кезде танымал болды. Реляциялық ДҚ кесте құрылымы жақсы түсініледі, сондықтан SQL тілін үйрену оңай;
- интерактивті сұрауларды жасау қабілеті - интерактивті режимде, күрделі бағдарлама жазбастан қысқа уақыт ішінде сұрау нәтижесін алуға болады;
- SQL тілін дерекқорларға кіруге қажет қосымшаларда қолдану оңай. SQL операторы интерактивті және бағдарламалыққолжетімділік үшін пайдаланылады, сол себепті ДҚ жүгінуді қамтитын бағдарламалардың бөліктері алдымен желіде тексеріліп, содан кейін бағдарламаға біріктірілген болуы мүмкін;
- SQL пайдалану арқылы деректердің әртүрлі көрсетілуін қамтамасыз ете отырып, бір немесе басқа пайдаланушы олардың әр түрлі көріністерін көретін осындай деректер құрылымын елестете алады. Сонымен қатар, ДҚ әртүрлі бөліктеріндегі деректерді бір қарапайым кестеде біріктіріп, ұсынуға болады, яғни бұл ұсыныстарды ДҚ қорғауды жетілдіруге және жеке пайдаланушылардың нақты талаптарына сәйкестендіруге жарамды екенін білдіреді;
- дерекқордың құрылымын динамикалық түрде өзгерту және кеңейту мүмкіндігі - SQL тілі дерекқордың құрылымын басқаруға мүмкіндік береді, осылайша икемділікті қамтамасыз етеді. Дерекқор нысандарын жасауға және жоюға, олардың құрылымын орындау уақытында өзгертуге болады. Бұл дерекқордың доменнің өзгеретін талаптарына сәйкестігі тұрғысынан өте маңызды;
- «клиент-сервер» архитектурасында жұмыс істеу мүмкіндігі – SQL клиент-сервер өзара әрекеттесуді жүзеге асырудың ең жақсы құралдарының бірі, сонымен қатар, ол өзара әрекеттесетін клиенттік жүйе мен ДҚ басқаратын серверлік жүйе арасындағы байланыс ретінде қызмет етеді, әрқайсысына олардың қызметтерін орындайды. SQL тілін құру қажетті теориялық негіздерді дамытуға ғана емес, табысты іске асырылған техникалық шешімдерді дайындауға да ықпал етті. Жаңа нарық – транзакциямен өңдеуді басқару жүйелеріне (OnLine Transaction Processing, OLTP) және жедел талдау өңдеу немесе шешімдерді қабылдауды қолдау жүйелеріне (OnLine Analytical Processing, OLAP) арналған тілдің мамандандырылған жүзеге асырулары пайда бола бастады.

БАҚЫЛАУ СҰРАҚТАРЫ

1. Дерекқормен жұмыс істеу тіліне қойылатын талаптар қандай?
2. SQL тілін іске асыру дегеніміз не?
3. Неліктен SQL тілі әзірлеушілер арасында танымал болды?
4. SQL тілінде қандай санаттар бар?
5. Әрбір тіл санатына қандай командалар жатады?
6. SQL деректерінің қандай түрлері пайдаланылады?
7. Кестелерді жасау кезінде домендер қалай пайдаланылады?
8. Доменде қандай шектеулерді сипаттауға болады?
9. Кестені SQL тілінде жасағанда бастапқы кілт қалай анықталады?
10. Сілтеме тұтастығын шектеу деген не және ол SQL тілінде қалай жасалады?
11. SQL тілінде кестелерді анықтаған кезде байланыс қалай жасалады?
12. SELECTоператорының форматын сипаттап беріңіз.
13. SELECT операторында қандай қызметтерді пайдалануға болады?
14. Сұрау салудың нәтижесін қалай топтауға болады?
15. Сұрау салу нәтижесіне шектеуді қалай салуға болады?
16. Сұрау салу нәтижесін қалай сұрыптауға болады?
17. Кестелердің ішкі қосылымы деп нені атайды?
18. Қандай жағдайларда сұрау салуларда HAVING ұсынысы қолданылады?
19. Шағын сұрау салу деп нені атаймыз?
20. Қандай шағын сұрау салу скалярлық, ал қандайы кестелі деп аталады?
21. Шағын сұрау салуларда не үшін SINGULAR, EXISTS, ALL, SOME ұсыныстары қолданылады?
22. Кестелердің сыртқы байланыстары деп нені атаймыз және сыртқы байланыс ішкі қосылымнан қалай ерекшеленеді?
23. Генератор деп нені атайды?
24. Триггерлер не үшін қолданылады?
25. Қандай оқиғалар кезінде триггерді шақыру әркеті орындалады?
26. Сақталатын рәсімдерді және триггерлерді пайдаланудың артықшылықтары қандай?
27. Триггер мен сақталатын рәсім арасындағы негізгі айырмашылық қандай?
28. Индекстерді қай жағдайларда жасауға тура келеді?
29. SQL арқылы кестеге жаңа жолдарды қалай қосуға болады?
30. SQL арқылы кестелердегі жолдарды қалай жоюға болады?
31. SQL арқылы деректерді кестеде қалай өзгертуге болады?
32. SQL тілін қолданудың артықшылықтары қандай?

Автоматтандырылған ақпараттық жүйе - тұтынушыға ақпаратты сақтау, өңдеу және беру үшін техникалық, бағдарламалық және ұйымдастырушылық құралдардың жиынтығы.

Деректерді біріктіру - жазбаның ішіндегі деректер элементтерінің белгілі бір жиынтығы, ол бір бүтін деп санауға болады. Деректер жиынтығы қарапайым немесе құрылымды (ауқымдар, жазбалар, күрделі сандар және т.б.) ұсынатын деректер элементтерінің белгілі жиынтығы.

Дерекқор әкімшісі - дерекқорды сақтауға уәкілетті тұлғалар немесе адамдар тобы (дерекқордың құрылымы мен мазмұнын өзгерту, пайдаланушының қол жеткізуін белсендіру, барлық пайдаланушыларды қозғайтын басқа әкімшілік қызметтерді орындау).

Деректер мекен-жайы - деректердің орналасуын анықтау.

Адресация - деректерді жадта бөлу және алу тәсілі.

Атрибут - объект туралы ақпараты бар деректер өрісі.

Дерекқор (ДҚ) - белгілі бір домендегі объектілердің күйін және олардың өзара байланысын көрсететін, бірнеше пайдаланушылар пайдаланатын және ең аз резервтеуде сақталатын өзара байланысты деректердің белгілі бір жиынтығы.

Деректер банкі (ДБн) - орталықтандырылған жинақтау және деректерді ұжымдық көп мақсатты пайдалану үшін арнайы ұйымдастырылған деректер, бағдарламалық қамтамасыз ету, тілдік, ұйымдастырушылық және техникалық құралдар жүйесі.

Сыртқы сызба - пайдаланушының тұрғысынан немесе қолданбалы бағдарламадағы деректерді ұсыну.

Ішкі сызба - деректердің физикалық құрылымы.

Деректер - ЭЕМ жадында сақталған немесе ЭЕМ кіру үшін

Логикалық жазба - сыртқы жадпен ақпарат алмасқанда бағдарлама жасақтамасымен жеке бірлік ретінде қабылдайтын деректердің немесе агрегаттардың бірегейлендірілген (аталатын) жиынтығы. Жазба - олардың түріне сәйкес еске салынған өзара қатынастардың табиғатына сәйкес реттелген деректер өрістерінің (элементтерінің) жиынтығы.

Физикалық жазба - бір енгізу-шығару пәрмені бар тұтас жалғыз объект ретінде оқуға немесе жазуға болатын деректер жинағы.

Иерархиялық деректер үлгісі - ДҚ доменінің ұсынуын иерархиялық ағаш түрінде пайдаланады, түйіндері «ата-бала» қатынасына тігінен байланысты. ДҚ навигациясы - бұл құрылымдағы тік және көлденең қозғалыс.

Импорт (жүктеу, download) - утилита (қызмет, пәрмен) Дерекқордан алынған кейбір коммуникативті форматта ұсынылған деректерді қамтитын операциялық жүйенің файлдарын оқуға қызмет ететін ДБЖ.

Инвертирленген файл (тізім) - негізгі файл жазбаларының белгілі бір өрістерінің кілт-мәндерінің тәуелсіз реттелген тізімдері түрінде ұйымдастырылған негізгі мәндерге арналған жазбаларды жылдам және еркін іздеуге арналған файл.

Индекс - жазба мекен-жайын анықтау үшін пайдаланылатын кесте.

Ақпарат - қоршаған әлем туралы нысандар, құбылыстар, үрдістер, оқиғалар туралы білімнің белгісіздігін азайтатын ақпарат. Ақпарат толық, сенімді, уақтылы, дәйекті, барабар болуы керек.

Ақпараттық база - белгілі бір тақырыптық аймақтың жай-күйін көрсететін және ақпараттық жүйе пайдаланатын деректер. Ақпараттық база екі компоненттен тұрады: нақты мәліметтерді жинақтау және осы деректердің сипаттамалары - метадеректер. Деректер сипаттамалардан бөлек, бірақ сонымен бірге, деректер тиісті сипаттама сілтемесіз қолданыла алмайды.

Ақпараттық нысан - нақты әлемде бар немесе бар болған алдын ала ойластырылған аймақтың (нысан, үрдіс, құбылыс немесе оқиға) кейбір мәндерінің сипаттамасы. Ақпараттық нысан - логикалық байланысты ақпарат жинағы болып табылады.

Ақпараттық жүйе - кез келген саланың міндеттеріне ақпаратты жинауды, сақтауды, өңдеуді, шығаруды, беруді қамтамасыз ететін техникалық және бағдарламалық құралдар жиынтығы.

Клиент - ДБЖ провайдері немесе пайдаланушысы, сыртқы немесе ДБЖ-ға қатысты «кіріктірлген» бағдарлама. Клиент бағдарламасы сыртқы интерфейс арқылы ДБЖ компоненттеріне деректермен операциялар жасау үшін пайдаланылатын қосымша ретінде ұйымдастырылады.

Кілт - жазбаның мекенжайын анықтау немесе анықтау үшін пайдаланылатын мән (деректер элементі).

Бастапқы (басты) кілт — мәні жалғыз түрде ғана жазбаны сәйкестендіруші кілт.

Қосымша (сыртқы) кілт - белгілі бір жалпы сипатқа ие жазбалардың кейбір тобын анықтайтын кілт. Деректер жиынтығында бірнеше қосалқы кілттер болуы мүмкін, оларды енгізу қажеттілігі тиісті кілт арқылы жазбаларды іздеу үрдістерін оңтайландыру арқылы практикалық қажеттілікпен анықталады.

Құрамдас кілт - деректерді жинаудың әрбір жазбасын сәйкестендірудің бірегейлігін қамтамасыз ететін бірнеше деректер элементтерінің жинағы.

Сақиналы құрылым – сақина түзеп, соңғы элементі біріншісін көрсетуші дереткердің тізімдік құрылымы.

Бақылау нүктесі - дерекқордың күйін физикалық файлдардағы кэштің ағымдағы жай-күйін үйлестіру (ДҰ жаңарту әрекеттеріндегі жүйелік буфер).

Дерекқорлар тұжырымдамасы - деректерді өңдеудің интеграцияланған ақпараттық технологиясы болып табылады, ол өңдеу бағдарламасының барлық сақталған деректерден оның қажеттілігіне байланысты және осы бағдарламада талап етілетін нысан бойынша бөлу механизміне негізделген.

Коммуникативті формат - дерекқорына (әдетте дерекқор кестелерінің біреуіне) ОЖ деректерін импорттау (жүктеу) кезінде ақпараттық бірліктерді (жазбалар, агрегаттар, өрістер) сенімді түрде тануды қамтамасыз ететін операциялық жүйедегі деректерді ұсынудың тәсілі болып табылады. Коммуникативті форматтағы

Қолжетімділік әдісі - деректермен алмасуды ұйымдастыратын әдіс (тәсіл), әдетте, ОЖ қолдайтын операциялық және сыртқы жады арасында, мысалы, тікелей, дәйекті, индекстік әдістер.

Дерекқор үлгісі - дерекқорды басқарудың логикалық және физикалық аспектілері (деректердің тәуелсіздігі) арасындағы шекараны анықтауға мүмкіндік беретін құралдар жиынтығы; соңғы пайдаланушыларды және бағдарламашыларды деректердің (байланысқа бейімділік) мағынасын жалпы түсіну мүмкіндігі мен құралдарымен қамтамасыз ету; бірыңғай операция ретінде (әр түрлі деректердің жалпы жағдайында) үлкен жиынтық жазбаларда осындай әрекеттерді орындау мүмкіндігін беретін жоғары деңгейлі тілдік ұғымдарды анықтау.

Деректер үлгісі - ресми дерексіз деңгейде объектілер мен қарым-қатынастарды ұсынудың нақты тәсілдерін беретін негізгі құрал.

Логикалық үлгі - информациялық деректер үлгісімен жасалған және нақты ДБЖ-нің дерекқор сипаттамасының тілінде ұсынылған сипаттамасы.

Инфологиялық үлгі (тұжырымдама) - дерекқорын жасауда жұмыс істейтін барлық адамдарға түсінікті табиғи тіл, математикалық өрнектер, кестелер, графиктер және басқа құралдарды қолдану арқылы жасалған, алдын ала ойластырылған аймақтың сипаттамасы.

Физикалық үлгі - ДБЖ сыртқы сақтау құрылғыларындағы деректерді іздеудің орнын және жолын анықтайтын үлгі.

Парадигмалық үлгілер - деректер құрылымдарының деңгейінде өзара байланысқан нысандарды анықтайтын шарттар. Осы тұрғыдан реляциялық, желілік, иерархиялық, объективтік, объектілі-реляциялық, құжаттық және басқа да үлгілер ерекшеленеді.

Деректердің физикалық (логикалық) тәуелсіздігі - өзгертпестен логикалық (физикалық) деректер құрылымын өзгерту мүмкіндігін беретін жүйенің логикалық (физикалық) қасиеті.

Нормаға келтіру - қалыпты формалардың талаптарына сәйкес келетін екі өлшемді кестелер (қатынастар) түрінде күрделі деректер

Деректерді өңдеу деректерді ЭЕМ енгізуді, деректерді кез келген критерий бойынша таңдауды, деректер құрылымын түрлендіруді, ЭЕМ сыртқы жадына деректерді енгізуді, кестелердегі немесе басқа да ыңғайлы нысандағы мәселелерді шешуге байланысты деректерді шығаруды қамтиды.

Қатынас(relation) - кестелердің бірінде (кестенің жолында) кесте, реляциялық ДҚ сақталған немесе деректер сұраулары орындалғанда дерекқорда операцияны орындау кезінде іс жүзінде жасалған деректер агрегаты.

ДҚ пайдаланушысы - ДБЖ деректерді басқару құралдарын пайдаланатын дерекқорға кіретін бағдарлама немесе адам.

Зерттеу аймағы - бұл нақты әлемнің бөлігі, ол туралы мәліметтерді басқаруды ұйымдастыруға және сайып келгенде, автоматтандыру мақсатында зерттелетін ақпараттық жүйеде сақталатын және пайдаланылатын деректер.

Көрініс(View) - бір немесе бірнеше кестелерде сұраудың нәтижелерін (SELECT операторы) қамтитын виртуалды кестені жасау құралы. Соңғы пайдаланушы үшін көрініс дерекқордағы әдеттегі кестеге ұқсайды, оның үстіне SELECT, INSERT, UPDATE және DELETE операторларын орындауға болады. Көріністер іріктеу және деректерді алдын-ала өңдеу үшін пайдаланылады.

Дерекқорды жобалау - дерекқорда сақталған деректерді осы деректермен сипатталған домен объектілерімен байланыстыратын (тіркейтін) домендік үлгілер сияқты өзара сипаттамаларының жүйесін құру рәсімдерінің реттелген үрдісі.

Дерекқорларды бөлу (ДҚБ) - компьютерлік желіде таратылатын логикалық өзара байланысты дерекқорлардың жиынтығы.

Реляциялық алгебра - қатынастарды басқару үшін операциялардың жиынтығын қамтитын алгебра (тіл).

Реляциялық дерекқор - қатынастардан тұратын дерекқор. Мұнда пайдаланушыға қол жетімді барлық ақпарат кестелер түрінде ұйымдастырылады. Әдетте кестелер бірегей атауларға ие және жолдар мен бағандардан тұрады, олардың қиылысындағы мәндер, деректер және деректер бойынша операциялар осы кестелердегі операцияларға дейін азаяды.

Желілік деректер үлісі - CODASYL ұсынған үлгі болып табылады, онда бір жазба бірнеше «баба-ұрпақ» қатынастарына қатыса алатын иерархиялық үлгінің модификациясы.

Деректерді өңдеу жүйесі (ДӨЖ) - деректерді басқару тапсырмаларын орындайтын аппараттық және бағдарламалық жасақтама жиынтығы.

Дерекқорды басқару жүйесі (ДБЖ) - көптеген пайдаланушылармен ДҚ құру, қолдау және бөлісуге арналған тілдік және бағдарламалық құралдар жиынтығы.

Деректер сөздігі - деректердің барлық сипаттамаларының (атаулардың, түрлердің) каталогы болып табылатын кестелердің немесе файлдардың толық жиынтығы. Сондай-ақ, дерекқор әкімшісіне қол жетімді пайдаланушылар, артықшылықтар және т.б. туралы ақпаратты қамтуы мүмкін. Бұл ДБЖ, АДҚ және барлық пайдаланушылар үшін негізгі ақпарат көзі.

Деректер құрылымы - сипаттамалардың ресми сипаты мен тұрақты байланыстарын қысқарту үшін белгілі бір құралдардың мүмкіндіктері мен шектеулерін ескере отырып, деректердің сипаттамасын формальды тілдер арқылы білдіруге бағытталған сипаттар мен қарым-қатынастардың сипаты. Бағдарламалау тұрғысынан деректер құрылымы - жадыдағы құндылықтарды көрсету тәсілі - қоршаған ортаның өлшемі және оны бөлу тәртібі (жолдау-іріктеу рәсімінің сипатын анықтайтын болады).

Желілік деректер құрылымы – желілік сипатына ие және жадыда элементтердің орналасу тәртібіне сәйкес келетін деректер элементтерінің қолдану тәртібі.

Деректерді жазу құрылымы – жеке жазбалар және олардың элементтері деңгейінде деректерді сақтау және оларға қолжетімділікті ұйымдастыру тәсілдерін мақсатты жүзеге асыру (физикалық ортаның ерекшеліктерін есепке алады).

Ақпараттың құрылымы - кешенді композиттік объектілерді ұсынудың нақты үлгісінің сызбалық нысаны және қолданбалы міндеттерді шешуге қатысты анықталған нақты ауданы.

Кесте - реляциялық дерекқорды басқару жүйесіндегі ақпараттың негізгі бірлігі. Ол бір немесе бірнеше ақпараттық бірліктерден (жолдардан) тұрады, олардың әрқайсысы қандай да бір

ДҚ топологиясы - есептеудің әр түрлі түйіндерін қоса алғанда физикалық таратушылар үшін дерекқор компоненттерін тарату сызбасы.

Сақтау нүктесі – транзакцияның барлық жұмысы ДҚ-ға жазылатын уақыт кезеңі. Транзакцияда жұмысқа арналған аралық нүктелер рөліндегі сақтау нүктелерінің қатары қолданылуы мүмкін.

Транзакция - ДҚ бір дәйекті күйден екіншісіне қабылдайтын, бір «оқиға» ретінде ұсынылатын дерекқор деректері бойынша операциялар тізбегі болып табылады.

Триггер - кесте UPDATE, INSERT немесе DELETE операторларының көмегімен өзгертілген кезде автоматты түрде орындалатын рәсімнің арнайы түрі.

Деректерді басқару - деректерді өңдеу жүйесінің табысты жұмыс істеуі үшін қажетті деректермен операциялардың барлық жиынтығы.

Деректерді ұсыну деңгейлері - тұжырымдамалық, ішкі және сыртқы болып табылады. Тұжырымдамалық деңгейде өкілдік тақырыптың перспективасы бойынша деректердің жалпыланған көзқарасын білдіреді. Ішкі қабат - ДҚ жаһандық көрінісі, ол сыртқы сақтау құрылғыларындағы деректерді сақтауды ұйымдастыру үшін ең алдымен қажетті шарттарды анықтайды. Сыртқы қабат пайдаланушылар мен қосымшалардың қажеттіліктерін көрсетеді.

ДБЖ утилиттері – басты компьютеропе рациялық жүйесінің командасымен іске қосылатын және дерекқорға (әдетте деректердің физикалық деңгейіне)қатысты белгілі бір қызметті орындайтын және ұқсас операцияны жүзеге асырушы бағдарлама (әдетте деректердің физикалық деңгейіне) немесе команда (АДҚ ғана қолжетімді ДБЖ өзегінің қызметі).

Файл - операциялық жүйемен қолдау көрсетілетін ақпарат бірлігі. Деректерге қол жеткізу ОЖ-да немесе пайдаланушы бағдарламаларында немесе ДБЖ-нің ішінде, немесе біріктіріліп жүзеге асырылады.

Дерекқор файлы - ДҚ орналастыруға арналған физикалық ОЖ файлы. Осындай файлдағы деректерді басқару ОЖ және ДБЖ бірлесіп жүзеге асырылады.

Сақталған рәсім - сервердің бір орындалу жоспарына жасайтын

Экспорттау (жүктеу) – ДҚ-дан (әдетте кестелердің біреуінен) операциялық жүйенің файлына (файлдарына) ақпарат беру үшін пайдаланылатын, әдетте, коммуникативті форматта ұйымдастырылған ДБЖ қызметтік бағдарламасы (функция, пәрмен).

Деректер бөлшегі - ДБЖ тікелей қол жеткізе алатын және барлық басқа құрылымдар салынған деректердің ең кіші атауы. Әрбір деректер бөлшегі үшін оның түрі анықталуы керек.

Деректермен айла-шарғы жасау тілі - әдетте дерекқорды сұрау және дерекқорды сақтау (деректерді қосу, жою, деректерді жаңарту, ДҚ қорын құру және жою, ДҚ анықтамаларын өзгерту) құралдарын қамтиды.

Деректерді сипаттау тілі - сыртқы көзқарастарды жалпылау болып табылатын ішкі жүйенің деректерін анықтау құралы. Сипаттама деректер үлгісі және олардың қарым-қатынасы, яғни ДҚ қалыптасатын құрылымдар.

Сұрау салуларды құрылымдау тілі (SQL) – реляциялық

Әдебиеттер тізімі

1. Аткинсон М. Манифест систем объектно-ориентированных баз данных / М. Аткинсон, Ф. Бансилон, Д. ДеВитт Д. и др. // СУБД. — 1995. — № 4.

2. Вирт Н. Алгоритмы и структуры данных / Г. Вирт ; пер. с англ. — М.: Мир, 1989.

3. Гроф Дж. SQL: полное руководство / Дж. Грофф, П. Вайнберг ; пер. с англ. — К.: BHV, 2001.

4. Дейт К. Дж. Введение в системы баз данных / К. Дж. Дейт ; пер. с англ. — М.: Вильямс, 2001.

5. Диго С. М. Проектирование и использование баз данных / С. М. Диго. — М.: Финансы и статистика, 1995.

6. Дунаев С. Б. Доступ к базам данных и техника работы в сети. Практические приемы современного программирования / С. Б. Дунаев. — М.: ДИАЛОГ-МИФИ, 1999.

7. Каменнова М. Управление электронными документами: технологии и решения / М. Каменнова // Открытые системы. — 1995. — № 4.

8. Карпова Т. С. Базы данных: модели, разработка, реализация / Т. С. Карпова. — СПб.: Питер, 2001.

9. Ким Вон. Технология объектно-ориентированных баз данных / Вон Ким // Открытые системы. — 1994. — № 4.

10. Когаловский М. Р. Абстракции и модели в системах баз данных / М. Р. Когаловский // Открытые системы. — 1998. — № 4 — 5.

11. Кузнецов С. Д. Основы современных баз данных / С. Д. Кузнецов // www.citfo-rum.ru. 2002.

12. Малкольм Г. Программирование для MicrosoftSQLServer2000 с использованием XML/ Г. Малкольм ; пер. с англ. — М.: Издательскоторговый дом «Русская редакция», 2002.

13. Мартин Дж. Организация баз данных в вычислительных системах / Дж. Мартин. — М.: Мир, 1980.

14. Мартин Дж. Превратите вашу компанию в киберкорпорацию / Дж. Мартин // ComputerworldРоссия. — 1995.

15. Меллинг В. П. Корпоративные информационные архитектуры: и все-таки они меняются / В. П. Меллинг // СУБД. — 1995. — № 2.

16. Озкарахан Э. Машины баз данных и управление базами данных / Э. Озкарахан. — М.: Мир, 1989.

17. Фаронов В. В. Delphi4. Руководство разработчика баз данных / В. В. Фаронов, П. В. Шумаков. — М.: Нолидж, 1999.

18. Фокс Дж. Программное обеспечение и его разработка / Дж. Фокс. — М.: Мир, 1985.

19. Хансен Г. Базы данных. Разработка и управление / Г. Хансен, Дж. Хансен. — М.: Бином, 1999.

20. Цикритзис Д. Модели данных / Д. Цикритзис, Ф. Лоховски. — М.: Финансы и статистика, 1985.

21. Шпеник М. Руководство администратора баз данных MicrosoftSQLServer2000 / М. Шпеник, О. Следж ; пер. с англ. — М.: Вильямс,

Мазмұны

Оқу басылымы

Федорова Галина Николаевна. Дерекқорларды жобалаудың негіздері

Оқу құралы

Редакторы *Е. Н. Соколова, С.Қамшыгер*

Техникалық редакторы *Е. Ф. Коржуева*

Компьютерлік беттеу: *Д. В. Федотов* Корректор *А. П. Сизова*

Бас. № 102116449. 17.02.2016 баспаға қол қойылды. Пішімі 60 х 90/16.

«Baltica» гарнитурасы. № 1 офсеттік қағаз. Офсеттік баспа. Баспа бет. шарт. 14,0.

Таралым 500 дана. Тапсырыс №

«Академия» баспа орталығы» АҚБ. www.academia-moscow.ru129085, Мәскеу, Мир даңғылы, 101В, 1 құр.

Тел./факс: (495) 648-0507, 616-00-29

Идел-Прессте басылды