Ҳайдарова С.

SQL ТИЛИ: ИМКОНИЯТЛАРИ ВА ҚЎЛЛАНИЛИШИ

МУНДАРИЖА

КИРИШ

Маълумки, маълумотлар базаси – маълумотлар тўпламини тақдим этиш ва ташкил этишнинг объектив шакли бўлиб, бунда ушбу маълумотлар ЭҲМ орқали қидириб топилади ва ишлов бериладиган ҳолатда тизимлаштирилган бўлади. Маълумотлар базаси ҳозирги кунда ижтимоий заруратга айланган ва йилдан-йилга кишилик фаолиятининг турли соҳаларида кенг қўлланилмоқда.

Ҳозирги пайтда маълумотлар базасининг энг кенг тарқалган модели бу реляцион моделдир. Иерархик ва тармоқ моделидан фарқли равишда бу моделда алоқалар объект сифатида қаралади, яъни кўпга кўп алоқаларни тасвирлаш учун янги типдаги ёзувлар киритилади.

Реляцион маълумот базалари маълумотларни жадвалларга жойлаш ва жадваллар орасида мос боғлиқликларни, яъни муносабатни (реляцияни) ўрнатишга асосланган. Улар жадваллар орасидаги турли боғлиқликларни ўрнатиш, маълумот киритиш шаклларини яратиш, ҳисобот шаклларини чиқариш, турли сўровлар тузиш имконини беради.

Маълумотларнинг реляцион модели оддий икки ўлчамли жадвал - муносабатларнинг йиғиндисидир. Реляцион маълумотлар базасида сақланувчи маълумотларни қайта ишлаш ва ўқиш учун мулжалланган тил SQL тили бўлиб, компьютер саноатида оммавий тарқалиши жиҳатидан у энг олдинги ўринларда туради. Кейинги бир неча йил ичида SQL маълумотлар базасининг ягона стандарт тили бўлиб қолди.

Маълумотлар базасига мурожаат қилиш жараёнида биз SQL тилидан фойдаланамиз. "Мижоз-Сервер" технологиясига кўра мижоз сўровлари махсус маълумотлар серверларида қайта ишланади, мижозга фақат сўровни қайта ишлаш натижалари қайтарилади. Сервер билан мулоқот қилиш учун эса ягона тил сифатида SQL танланган.

Ҳозирги пайтда ҳамма замонавий реляцион МББТ версиялари ва ҳаттоки нореляцион МББТ версиялари (масалан, Adabas) "Мижоз-Сервер"

технологияси ва SQL тилидан фойдаланадилар. Бу тил маълумотлар базасига сўровларни жўнатиш ва натижалар олиш учун мўлжалланган.

Ҳозирги кунда SQL юздан ортиқ МББТ ларда ишлаяпти. SQL тилининг халқаро стандарти расмий жиҳатдан қабул қилинди ва кейинроқ мукаммаллаштирилди. SQL тили МББТ архитектурасининг муҳим аъзоси бўлиб, у Microsoft компанияси дастур ишлаб чиқаришининг стратегик йўналиши бўлиб хизмат қилади. IBM компаниясининг иккинчи даражали тадқиқот лойиҳасининг бажарилиши натижасида пайдо бўлган SQL тили ҳозирги вақтда муҳим компьютер технологияси ва кучли бозор фактори сифатида кенг тарқалди.

SQL – бу структуралашган сўровлар тилининг (Structured Query Language) қисқартирилган номланиши бўлиб, SQL фойдаланувчининг маълумотлар базаси билан ўзаро алоқасини ташкил этиш учун қўлланилувчи тилдир. SQL фақат реляцион деб номланувчи бир турдаги маълумотлар базаси билан ишлайди.

Бугунги кунга келиб, SQL оддий сўровлар тузувчи восита бўлиб қолмасдан, балки маълумотлар структурасини яратиш, улардаги маълумотларни ўзгартириш, қайта ишлаш, ҳимоялаш, ўқиш ва маълумотлардан биргаликда фойдаланиш, маълумотлар яхлитлигини таъминлаш каби қатор имкониятларга ҳам эгадир.

SQL МББТ билан ўзаро алоқа қилувчи етарлича кучли тилдир. Фойдаланувчи маълумотлар базасидан маълумотларни ўқимоқчи бўлса, у буни МББТ дан SQL ёрдамида сўрайди. МББТ сўровга ишлов беради, талаб қилинган маълумотларни топади ва уни фойдаланувчига узатади.

Ушбу қўлланма SQL тилининг тарихи, имкониятлари ва унинг қўлланилишига оид барча операторларни ўз ичига олади.

Қўлланмада SQL-сўровларни яратиш усуллари хорижий адабиётлардан олинган маълумотлар билан бойитилган ва кўпроқ амалий жиҳатдан ёритилган.

I-боб. SQL тили асослари

1.1. SQL тили ҳақида

Ҳаётда биз ҳар доим маълумотлар базаси билан иш кўрамиз. Қидирув тизими ёрдамида Интернетдан бирор маълумотни олиш учун маълумотлар базасига сўров жўнатиш керак бўлади. Ҳар сафар манзиллар китобида электрон почтангиз манзилини терганингизда ҳам маълумотлар базасига мурожаат қиласиз. Ҳаттоки пластик картангизни банкоматга қўйиб, PIN-кодни ва ҳисобингиздаги қолдиқни текшириш ҳам маълумотлар базаси орқали амалга оширилади. Шундай қилиб, хоҳлайсизми ёки йўқми, маълумотлар базасидан фойдаланишга тўғри келади. Маълумотлар базасига мурожаат қилиш жараёнида SQL тилидан фойдаланамиз, бу тил маълумотлар базасига сўровларни жўнатиш ва натижалар олиш учун мўлжалланган.

Ҳар қандай маълумотлар базаси (МБ) ҳар хил объектларга эга, яъни жадваллар, процедуралар, функциялар, тасаввурлар, кетма кетликлар ва ҳоказо.

"Мижоз-Сервер" технологиясига кўра фойдаланувчи ЭҲМ (Мижоз)лар сўровлари махсус маълумотлар серверларида (Сервер) қайта ишланади, фойдаланувчи ЭҲМларга фақат сўровни қайта ишлаш натижалари қайтарилади. Табиийки, сервер билан мулоқот қилиш учун ягона тил керак ва бундай тил сифатида SQL танланди. Шунинг учун ҳамма замонавий реляцион маълумотлар базасини бош қариш тизимлари (МББТ) версиялари (DB2, Oracle, Ingres, Informix, Sybase, Progress, Rdb) ва ҳаттоки нореляцион МББТ версиялари (масалан, Adabas) "Мижоз-Сервер" технологияси ва SQL тилидан фойдаланадилар.

SQL (Structured Query Language, одатда "сиквел" ёки эс-кю-л дейилади) маъноси *Таркибланган сўровлар тили.*Бу реляцион маълумотлар базаларида ишлашга имкон берадиган тилдир. Бу тил ифодаларининг хусусияти шундан иборатки, улар маълумотларни қайта ишлаш процедураларига эмас, натижаларига йўналтирилгандир.

SQL ўзи маълумотлар қаерда жойлашгани, қандай индекслар ва ҳатто амалларнинг энг самарали кетма кетлигини қўллаш кераклигини аниқлайди, бу деталларни маълумотлар базасига сўровларда кўрсатиш керак эмас. SQL тили фойдаланувчига реляцион маълумотлар базасида сўровларни ташкил қилиш имконини беради.

SQL тилининг ўзи IBM компаниясида 1974 йилда DB2 маълумотлар базасини бошқариш тизими (МББТ) ни яратиш жараёнида ишлаб чиқилган ва кенг кўламда RISC процессорли машиналарда UNIX тизимлар асосида ҳамда мейнфреймларда, суперкомпьютерлар асосида қурилган катта ҳисоблаш тизимларида қўлланилган.

Реляцион тиллар тарихига назар соладиган бўлсак, шуни айтиш мумкинки, алгебраик тиллардан реляцион ҳисобга асосланган тилларга ўтилиши тилларнинг ривожланишида эволюцион қадам ва муҳим туртки бўлди. Реляцион алгебрага яқин бўлган ISBL (Information System Base Language) тили сўровлар тили бўлиб, Питерли(Англия)даги IBM фирмасининг тадқиқот марказида PRTV (Peterlee Relational Test Vehicl) тадқиқот тизимида фойдаланиш учун ишлаб чиқилган. Лекин, бу тилнинг имкониятлари бошқа сўровлар тили билан таққосланганда чекланган, масалан, унда ҳеч қандай агрегат операторлари (ўртача, минимум, максимум ва ҳ.) йўқ, шунингдек, ёзувларни киритиш, ўчириш ва модификация қилиш воситалари мавжуд эмас.

Реляцион ҳисобга асосланган тиллар алгебраик тилларга қараганда юқори даражали тил деб юритилади ва кенг тарқалганлиги билан ажралиб туради, чунки улар процедурага эмас, балки натижа олишга йўналтирилгандир.

SQUARE ва SEQUEL каби сўровлар тилининг яратилиши алгебраик тиллардан реляцион ҳисобга ўтиш имконини берди.

SQUARE тилининг яратилиши IBM нинг Сан-Хоседаги тадқиқот марказида System-R МББТ тизимида фойдаланиш учун биринчи қадам бўлди. Унда реляцион алгебрада бўлмаган қатор имкониятлар ва агрегат

функциялар мавжуд бўлишига қарамай, синтаксисида қуйи индексларнинг ишлатилиши ноқулайлик туғдириб, унчалик эътиборга сазовор бўла олмади. Унинг ривожланган кўриниши сифатида SEQUEL (Structured English Query Language) тили пайдо бўлди. SEQUELда жадвал ва устунларнинг номини калит сўзлар, масалан, SELECT, FROM орқали ифодаланиши SQUARE даги қуйи индекслардан фарқли ўлароқ, анча қулайликлар туғдирди.

1970 йилларнинг охирида ORACLE компанияси томонидан SEQUEL тилининг модификацияланган варианти ишлаб чиқилди ва у SQL деб ном олди. Шу билан бирга мустақил бўлмасдан PL/SQL va Transact-SQL каби ички дастурлаш тилларига инкапсуляция қилинади.

1986 йилда ANSI (American National Standart Institute) SQL тилининг расмий стандартини ишлаб чиқди, 1992 йил бу стандарт кенгайтирилди. SQL стандарти **ANSI** томонидан аниқланган ва ҳозирда **ISO** томонидан қабул қилинган. Лекин коммерциал маълумотлар базалари дастурлари ANSI ни огоҳлантирмасдан SQLни кенгайтирадилар ва фойдали ҳисобланган хоссаларни қўшадилар.

SQL тилининг стандартлари:

- SQL-86
- SQL-89
- SQL-92 – стандартнинг янгиланган варианти
- SQL:1999
- SQL:2003
- SQL:2006
- SQL:2008

Ҳозирги пайтда SQL нинг энг кўп тарқалган стандарти SQL-92 ҳисобланади. Бу тил 30 га яқин операторларга эга бўлиб, баъзи версияларида сал кўпроқ, баъзиларида сал камроқ.

SQL тилида маълумотларни жадвал кўринишда тасвирлашга йўналтирилган амаллар концепцияси кўп бўлмаган (30 дан кам) ифодалардан

иборат компакт тил яратишга имкон берди.

Икки хил SQL мавжуд: **Интерактив** ва **Жойлаштирилган.** Кўп ҳолларда иккала форма бир хил ишлайди, лекин икки хил фойдаланилади:

Интерактив SQL маълумотлар базаси ўзида фаолият кўрсатади ва буюртмачи фойдаланиши учун чиқиш ҳосил қилиш учун ишлатилади. SQL нинг бу формасида команда киритсангиз у даров бажарилади ва дарҳол натижани(агар у мавжуд бўлса) кўришингиз мумкин.

Жойлаштирилган SQL бошқа тилда яратилган дастурга жойлаштирилган SQL командалардан иборат.

1.2. SQL тили буйруқларининг турлари

SQL тили қуйидаги қисмларга бўлинади:[1]

· **DDL** *(*Data Definition Language*)* - *Маълумотларни Таърифлаш Тили* ANSI да схемани таърифлаш тили, объектларни (жадваллар,индекслар, тасаввурлар ва ҳоказо) яратувчи командалардан иборат.

· **DML** (Data Manipulation Language) - *Маълумотларни Ўзгартириш Тили*. Бу ихтиёрий дақиқада жадвалларда қандай қийматлар сақланишини аниқловчи командалар мажмуасидир.

· **DCL** (Data Control Language) - *Маълумотларни Бошқариш Тили* фойдаланувчига маълум объектлар устида маълум таъсир ўтказишга рухсат бериш ёки бермасликни аниқловчи воситалардан иборат.

· **TCL** (Transaction Control Language)-Транзакция*ларни Бошқариш Тили* фойдаланувчига транзакцияни қайта ишлш имконини беради.

· **DQL** (Data Query Language) – *Маълумотларга сўров тили*. Бу энг кўп ишлатиладиган гуруҳ бўлиб, фақат битта SELECT операторидан иборат бўлади. У маълумотлар базасида сўровлар яратиш учун ишлатилади.

· **CCL** (Cursor Control Language) – *Курсорни Бошқариш Тили*.

[1] Д.А. Харитонюк. Базы данных. Негосударственное образовательное учреждение, Октябрьский экономический техникум, 2008, с.105

Маълумотларни Таърифлаш Тили - DDL

Оператор	Описание
CREATE TABLE	Создает новую таблицу в БД
DROP TABLE	Удаляет существующую таблицу из БД
ALTER TABLE	Изменяет структуру таблицы или ограничения таблицы
CREATE VIEW	Создает представление (виртуальную таблицу) соответствующую некоторому SQL запросу
DROP VIEW	Удаляет ранее созданное представление
ALTER VIEW	Изменяет существующее представление
CREATE INDEX	Создает индекс для некоторой таблицы
DROP INDEX	Удаляет существующий индекс

CREAT TABLE - Маълумотлар базасида янги жадвал яратиш.

DROP TABLE - Маълумотлар базасидаги мавжуд жадвални ўчириш.

ALTER TABLE - Маълумотлар базасидаги жадвални ўзгартириш.

CREAT VIEW – SQL-сўровга мос бўлган виртуал жадвални яратиш.

DROP VIEW – Яратилган виртуал жадвални ўчириш.

ALTER VIEW – Яратилган виртуал жадвални ўзгартириш.

CREAT INDEX – Бирор жадвал учун индекс яратиш.

DROP INDEX – Мавжуд индексни ўчириш.

Маълумотларни Ўзгартириш Тили - DML

Оператор	Описание
DELETE	Удаляет одну или несколько записей согласно условиям отбора. Применение оператора согласуется с принципами поддержки ссылочной целостности, поэтому оператор не всегда выполняется корректно, даже если синтаксически записан правильно
INSERT	Вставляет одну или несколько записей, согласно условию отбора, в базовую таблицу
UPDATE	Обновляет значения одного или нескольких полей в одной или нескольких записях, соответствующих условиям отбора

DELETE – Танлаш шартига мос бўлган бир ёки бир неча ёзувни ўчириш.

INSERT – Жадвалга танлаш шартига мос бўлган бир ёки бир неча ёзувни киритиш.

UPDATE - Жадвалга танлаш шартига мос бўлган бир ёки бир неча ёзувни янгилаш.

Маълумотларни Бошқариш Тили - DCL

Оператор	Описание
ALTER DATABASE	Изменение набора основных объектов БД
ALTER DBAREA	Изменение существующей области хранения БД
ALTER PASSWORD	Изменяет пароль для всей базы данных
CREATE DATABASE	Создает новую базу данных и определяет ее основные параметры
CREATE DBAREA	Создает область хранения и делает ее доступной для размещения данных
DROP DATABASE	Удаляет БД (при наличии прав)
DROP DBAREA	Удаляет область хранения если в ней не располагаются активные данные
GRANT	Предоставляет права доступа на действия с объектами БД
REVOKE	Лишает прав доступа к объектам БД или над действиями с объектами БД

ALTER DATABASE - Маълумотлар базасини ўзгартириш.

ALTER DBAREA- Маълумотлар базасини сақлаш соҳасини ўзгартириш.

ALTER PASSWORD - Маълумотлар базасига кириш паролини ўзгартириш.

CREAT DATABASE – Янги базани яратиш.

CREAT DBAREA - Маълумотлар базасини сақлаш соҳасини яратиш.

DROP DATABASE - Маълумотлар базасини ўчириш.

DROP DBAREA - Маълумотлар базасини сақлаш соҳасини ўчириш.

GRANT - Маълумотлар базаси объектларига кириш ва амаллар бажариш ҳуқуқини беради.

REVOKE - Маълумотлар базаси объектларига кириш ва амаллар бажариш ҳуқуқидан маҳрум қилади.

Транзакцияларни Бошқариш Тили - TCL

Оператор	Описание
COMMIT	Завершает транзакцию (комплексную взаимосвязанную обработку информации, объединенную в транзакции)
ROLLBACK	Откат транзакции (отмена изменений, проведенных в ходе выполнения транзакции)
SAVEPOINT	Сохраняет промежуточную точку (состояние) БД, для реализации возможности отката

Транзакция – бу маълумотларни манипуляция қилиш операторларининг кетма-кетлиги бўлиб, у МБ ни бир ҳолатдан иккинчи ҳолатга ўтказади.

COMMIT – транзакцияни тугаллаш.

ROLLBACK – транзакция бажарилишидаги ўзгартиришларни рад этиш.

SAVEPOINT–Рад этилган имкониятларни амалга ошириш учун МБ даги оралиқ нуқта (ҳолат)ни сақлаш.

Маълумотларга сўров тили - DQL

Оператор	**Описание**
SELECT	Оператор, полностью реализующий возможности реляционной алгебры. Позволяет сформировать результирующие отношение, соответствующее запросу

SELECT – Реляцион алгебра имкониятларини амалга оширувчи оператор. Бу оператор сўровга мос бўлган натижани олиш имконини беради.

Курсорни Бошқариш Тили - CCL

Оператор	**Описание**
DECLARE	Определяет курсор для запроса
OPEN	Открывает курсор (Формирует виртуальный НД, соответствующий описанию курсора)
FETCH	Считывает очередную строку из виртуального НД открытого курсора
CLOSE	Закрывает открытый курсор
PREPARE	Готовит оператор SQL к динамическому выполнению
EXECUTE	Выполняет оператор SQL, ранее подготовленный к динамическому выполнению

Курсор SQL тилининг асосий тушунчаси ҳисобланади.

DECLARE – сўров учун курсорни аниқлаш.

OPEN – курсорни очиш.

FETCH – Очилган курсордан кейинги сатрга ўтиш.

CLOSE - Очилган курсорни ёпиш.

PREPARE – SQL операторини динамик бажаришга тайёрлаш.

EXECUTE - Динамик бажаришга тайёрланган SQL операторини бажариш.

1.3. SQL тилида маълумот турлари

SQL тилида қуйидаги маълумотлар турлари ишлатилади:

INTEGER	- бутун сон (одатда 10 тагача қийматли рақам ва ишора).
SMALLINT	- "қисқа бутун" (одатда 5 тагача қийматли рақам ва ишора).
DECIMAL(p,q)	-ўнли сон, p рақам ва ишорадан иборат(0<p< 16).Ўнли нуқтадан сўнг рақамлар сони q орқали берилади (q<p, агар q=0 бўлса ташлаб юборилиши мумкин).
FLOAT	- ҳақиқий сон 15 та қийматли рақам ва бутун даражадан иборат. Даража МББТ типи билан аниқланади (масалан, 75 ёки 307).
CHAR(n)	- узунлиги ўзгармас, n га тенг бўлган символли қатор (0 < n < 256).
VARCHAR(n)	- узунлиги ўзгарувчи, n символдан ошмаган символли қатор (n > 0 ва ҳар хил МББТ ларда ҳар хил, лекин 4096 дан кам эмас).
MONEY	-махсус команда орқали аниқланувчи форматдаги пул.
DATE	-махсус команда орқали аниқланувчи формат-даги сана; сана майдонлари эрамиздан олдин бир неча мингйилликлардан бошланувчи ва эрамиз бешинчи-ўнинчи мингйиллиги билан чекланган ҳақиқий саналарни ўз ичига олади.
TIME	-махсус команда орқали аниқланувчи форматдаги вақт (кўзда тутилган бўйича hh.mm.ss).
DATE TIME	- сана ва вақт комбинацияси.

SQL маълумотларнинг ўзгарувчан-сатрли, сонли, реал вақт, сана ва ҳ. к. турларини киритилишига имкон беради. Сатрли тури белги ва сонлардан ташкил топган ўзгарувчиларни тасвирлаш учун тайинланган. Бунда биринчи белги албатта ҳарф бўлиши керак.

Char -узунлиги 254 байтдан ошмайдиган сатрли ўзгарувчини тасвирлаш учун қўлланилади. Сонли ўзгарувчиларни тасвирлашда NUMBER калитли сўз қўлланилади. -1.0E –100 дан 1.0E+100 гача бўлган доирада 22 рақамга эга бўла оладиган сонларни тасвирлайди.

Сана ва вақтни тасвирлашда Date намунаси қўлланилади.

1.4. SQL тилининг операторлари

SQL операторлари ёрдамида маълумотлар базаси (МБ) дан керакли маълумотларни олиш, уларни янгилаш, ўчириш ва базага маълумотларни киритиш мумкин. Бизга қуйидаги концептуал схема берилган бўлсин.[2]

Концептуал схема деганда МБ нинг мантиқий маънодаги тузилишини тасвирлаш тушунилади. Педагогика институтининг бу реляцион моделини кластер технологияси ёрдамида тасвирласа ҳам бўлади (илова 2).

Шунингдек, бу концептуал схемага мос МБ берилган бўлсин:

[2] Г.Д.Фролов, Э.И.Кузнецов. Элементы информатики.М., Высшая школа, 1989, с.275

БИНО

1	Турон 23
2	Турон 24
3	Уста Бозор 16
4	Вакф чорси 21

ФАКУЛЬТЕТ

М	Математика	2
Ф	Физика	3
Ж	Жисмоний тарбия	4
ФГ	Филология	1

МАШҒУЛОТЛАР

2 2	М
1	ФГ
3	Ф
4	Ж

ГУРУҲ

М	1	1	Акбаров
Ф	2	3	Иномов
ФГ	3	2	Каримова
Ж	2	2	Пулатов

ФАН

Ф-1	Астрономия
Ф-2	Қаттиқ жисм физикаси
М-8	Математик таҳлил
Ф-4	Философия

ИМТИҲОН

М-8	М	1	8.06.14	Акбаров
Ф-2	Ф	4	5.06.14	Иномов
Ф-4	ФГ	3	18.06.14	Зокирова
Ф-1	Ф	5	15.06.14	Иномов

SQL тили таркибига қуйидаги операторлар киритилган:

SELECT (Танлаш) **INSERT (Киритиш)**

DELETE (Ўчириш) **UPDATA (Янгилаш)**

1. SELECT оператори МБда мавжуд бўлган жадваллардан янги жадвалларни шакллантиради. Бу оператор битта ёки бир неча жадваллардан керакли ахборотни олиш имконини беради.

SELECT инструкцияси ёрдамида жадвалдан ахборот олиш учун ҳеч бўлмаганда иккита нарсага эътибор қаратиш керак: нимани олиш керак ва қаердан олиш керак.

Унинг кўриниши қуйидагича бўлади:

SELECT $\alpha_1, \alpha_2, \cdots, \alpha_n$

FROM $r_1, r_2, \ldots, r_m$

WHERE танлаш шарти

бу ерда $\alpha_1, \alpha_2, \cdots, \alpha_n$ - ҳосил қилинадиган жадвалнинг атрибутлари (устунлари) номи; $r_1, r_2, \ldots, r_m$ - жадвалнинг номи; танлаш шарти – сўровни қаноатлантирувчи, $r_1, r_2, \ldots, r_m$ жадвалдаги ёзувларга қўйиладиган шартлар.

Мисол. 1. Сўров: *Ф факультетида ўқитувчи Иномов томонидан қабул қилинадиган барча имтиҳонларнинг калитини танланг.*

Бажариш.

SELECT ФАН_КОДИ, ФАКУЛЬТЕТ_ШИФРИ, ГУРУҲ _НОМЕРИ

FROM ИМТИҲОН

WHERE ИМТИҲОН ОЛУВЧИ – ’ИНОМОВ’&

ФАКУЛЬТЕТ_ШИФРИ=’Ф’

2.INSERT оператори маълумотларни киритиш учун хизмат қилади. Унинг ёрдамида жадвалга битта сатрни қўшиш мумкин.

Унинг кўриниши қуйидагича бўлади:

INSERT INTO жадвал_ номи

VALUES (қийматлар, …);

Биринчи курсга янги гуруҳни киритиш қуйидагича бўлади:

INSERT INTO ГУРУҲ

VALUES (Ф, 6,1, ИНОМОВ)

3.DELETE оператори ёрдамида жадвалдаги кераксиз маълумотларни ўчириш мумкин.

Унинг кўриниши қуйидагича бўлади:

DELETE FROM жадвал_ номи

[WHERE …];

Кейинги курсга ўтилгандан сўнг ГУРУҲ жадвалидан тўртинчи курсларни олиб ташлаш қуйидагича бажарилади:

DELETE ГУРУҲ

WHERE КУРС=4

4. UPDATA оператори ёрдамида жадвалдаги бир ёки бир неча сатрни янгилаш мумкин.

Унинг кўриниши қуйидагича бўлади:

UPDATE жадвал_ номи

SET устун_ номи=қиймат, …

[WHERE …];

ГУРУҲ жадвалидаги курсларнинг сонини биттага ошириш қуйидагича амалга оширилади:

UPDATA ГУРУҲ

SET КУРС=КУРС+1

Бу операторларни билан биз кейинчалик жадвал яратиш билан танишиб бўлгач батафсил кўриб чиқамиз.

1.5. SQL тилида жадвал яратиш ва улар билан ишлаш

SQL тилида жадвал яратиш учун **CREAT TABLE** инструкциясидан фойдаланилади. Бу буйруқ қаторларсиз бўш жадвал яратади.

CREATE TABLE буйруғи жадвал номини ва жадвал ўзини маълум тартибда кўрсатилган устунлар номлари кетма – кетлиги таърифи кўринишида аниқлайди. У маълумотлар типлари ва устунлар ўлчовини аниқлайди. Ҳар бир жадвал жуда бўлмаганда битта устунга эга бўлиши керак.

CREATE TABLE буйруғи синтаксиси:

CREATE TABLE <table-name >

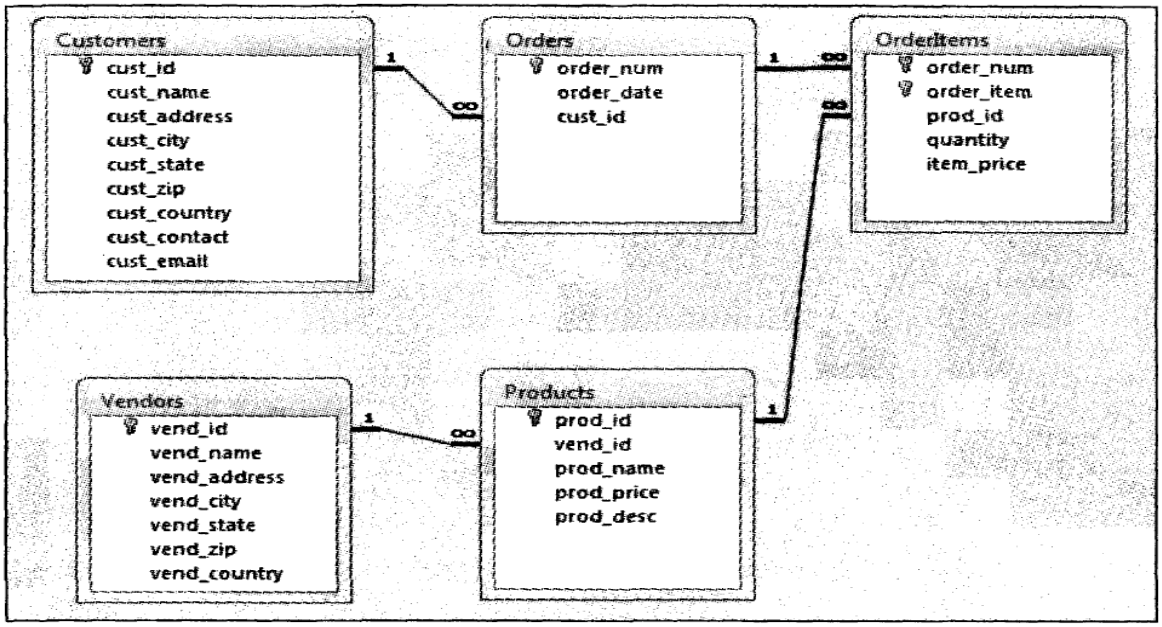
(<column name><data type>[(<size>)],

<column name><data type>[(<size>)], …);

Аргумент қиймати катталиги маълумот турига боғлиқдир. Агар сиз махсус кўрсатмасангиз тизим автоматик қийматни ўрнатади.

Қўлланмада кўп ишлатиладиган жадваллар билан танишиб чиқайлик. Бу жадваллар Products**,** Vendors, Customers, Orders ва OrderItems жадваллари бўлиб, улар орасидаги алоқалар 1-расмда тасвирланган.[3] Ҳар бир жадвал учун бирламчи калитлар аниқланган.

Мисол учун **Products** деб номланган маҳсулотлар жадвалини кўрайлик, бу жадвал маҳсулотлар каталогини ўз ичига олади. Ҳар бир маҳсулот ўзининг уникал, яъни такрорланмайдиган идентификатори (prod_id устуни)га эга ва у vend_id устуни (таъминловчининг уникал идентификатори) билан боғланган. Бу жадвалда prod_id бирламчи калит бўлиб ҳисобланади. Энди 1-расмда келтирилган жадвалларни яратишни кўриб чиқайлик.



1-расм. Жадваллар орасидаги алоқалар

Аввало **Products** жадвалини яратишни кўриб чиқамиз:

```
CREATE TABLE Products
(
   prod_id    char (10)      NOT NULL ,
```

[3] Бен Форта.Освой самостоятельно SQL за 10 минут, 4-е изд.: Пер. с англ.—М .: ООО “И.Д. Вильямс”, 2014. с. 242

```
  vend_id    char (10)      NOT NULL ,
  prod_name  char (255)     NOT NULL ,
  prod_price decimal (8,2)  NOT NULL ,
  prod_desc  text           NULL
) ;
```

Бу ерда

prod_id - маҳсулотнинг уникал идентификатори;

vend_id - таъминловчининг уникал идентификатори;

prod_name – маҳсулотнинг номи;

prod_price – маҳсулотнинг нархи;

prod_desc - маҳсулотнинг тавсифи.

Кўриниб турибдики, бунда **CREATE TABLE** калит сўзидан кейин жадвал номи ёзилади. Жадвални аниқлашда унинг барча устунлари юмалоқ қавсга олинади. Устунлар бир-биридан вергул билан ажратилади.

Келтирилган мисолдаги жадвал бешта устундан иборат бўлиб, ҳар бир устун унинг номини бериш билан аниқланади.Ундан сўнг маълумотлар тури аниқланади.

NULL қиймати устунда ҳеч қандай қиймат мавжуд эмаслигини, яъни устун бўш эканлигини, NOT NULL эса бўш эмаслигини билдиради, хуллас, инструкция нуқта вергул билан тугалланади.

Кейинги жадвал бу буюртмалар жадвали бўлиб, у **Orders** деб номланади. Бу жадвалда мижозларнинг буюртмалари ҳақидаги маълумотлар сақланади. Жадвал учта устундан иборат: буюртманинг уникал номери, муддати ва мижоз идентификатори.

Ҳар бир буюртма уникал номер(order_num устуни)га эга. Буюртмалар мос мижозлар билан cust_id устуни орқали боғланган. Бу жадвалда order_num устуни бирламчи калит, cust_id устуни эса ташқи калит бўлиб ҳисобланади.

Orders жадвалини яратишни кўрайлик:

```
CREATE TABLE  Orders
(
   order_num  int          NOT NULL ,
   order_date  datetime   NOT NULL ,
   cust_id    ch ar(10)     NOT NULL
);
```

Бу ерда

order_num - буюртманинг уникал номери;

order_date - буюртманинг муддати;

cust_id - буюртмани берган мижоз идентификатори.

Қуйида **Vendors** (Таъминловчи), **Customers** (Мижозлар) ва **OrderItems** (Буюртма элементлари) жадвалларини SQL тилида яратиш кўрсатилган.

Vendors жадвалини яратиш:

```
CREATE TABLE  Vendors
(
   vend_id      char(10)      NOT NULL ,
   vend_name    char(50)    NOT NULL ,
   vend_address  char(50)   NULL ,
   vend_city    char(50)      NULL ,
   vend_state   char(5)       NULL ,
   vend_zip     char(10)      NULL ,
   vend_country  char(50)   NULL
);
```

Бу ерда

vend_id – таъминловчининг уникал идентификатори

vend_name – таъминловчининг номи

vend_address – таъминловчининг манзили

vend_city – таъминловчининг шахри

vend_state – таъминловчинингштати

vend_zip – таъминловчининг zip-коди

vend_country – таъминловчининг мамлакати

Customers жадвалини яратиш:

```
CREATE TABLE  Customers
(
 cust_id      char(10)     NOT NULL ,
 cust_name    char(50)  NOT NULL ,
 cust_address char(50)  NULL ,
 cust_city    char(50)      NULL ,
 cust_state   char(5)      NULL ,
 cust_zip     char(10)      NULL ,
 cust_country char(50)   NULL ,
 cust_contact char(50)    NULL ,
 cust_email   char(255)  NULL
);
```

Бу ерда

cust _id –мижознинг уникал идентификатори

cust _name – мижознинг номи

cust _address – мижознинг манзили

cust _city – мижознинг шахри

cust _state – мижознинг штати

cust _zip – мижознинг zip-коди

cust _country – мижознинг мамлакати

cust_contact – мижознинг контакт шахси

cust_email – мижознинг электрон почтаси

OrderItems жадвалини яратиш:

```
CREATE TABLE OrderItems
(
 order_num  int              NOT NULL ,
```

```
 order_item  int            NOT NULL ,
 prod_id    char(10)        NOT NULL ,
 quantity   int             NOT NULL ,
 item_price  decimal(8,2)   NOT NULL
);
```

Бу ерда

order_num – буюртма номери

order_item – буюртма элементи номери

prod_id – маҳсулот идентификатори

quantity – буюртма қилинган маҳсулот сони

item_price – бир бирлик маҳсулотнинг нархи

SQL тили нафақат жадвал маълумотлари билан ишлаш, балки маълумотлар базасида амаллар бажариш, жумладан, жадвални яратиш, ўчириш ва ўзгартириш (янгилаш) имконини беради.

Жадвални ўзгартириш учун **ALTER TABLE** буйруғидан фойдаланилади. Бу инструкция жадвалга янги устунлар қўшиш, устунларни ўчириш, устунлар катталигини ўзгартириш ҳамда чекланишларни қўшиш ва олиб ташлаш имкониятларига эга.

Бу команда ANSI стандарти қисми эмас, шунинг учун ҳар хил МББТ ларда ҳар хил имкониятларга эга. Жадвалга устун қўшиш барча МББТлар қўллаб-қувватлайдиган ягона амал бўлганлиги учун биз уни мисол сифатида кўриб чиқамиз.

Жадвалга устун қўшиш буйруғининг синтаксиси қуйидагича:

ALTER TABLE <table name> ADD <column name>

<data type><size>;

бу ерда table name- жадвалнинг номи, column name– устуннинг номи, data type– қўшиш керак бўлган устундаги маълумотларнинг тури, size – ўлчови.

Масалан, **Vendors** таъминловчи жадвалига vend_phone устунини қўшиш қуйидагича амалга оширилади:

ALTER TABLE Vendors

ADD vend_phone CHAR(20);

Бунга кўра, Vendors жадвалига vend_phone номли устун қўшилади, устундаги маълумотларнинг тури символли бўлиб, унинг узунлиги 20та символдан иборат.

ALTER TABLE буйруғидан эҳтиётлик билан фойдаланиш зарур.Бундай сўровни амалга оширишдан аввал сизда барча маълумотларнинг ва схемаларнинг тўлиқ нусхаси борлигига ишонч ҳосил қилишингиз керак.

Шуни таъкидлаш лозимки, жадвалга ўзгартириш киритилгандан сўнг уни бекор қилиб бўлмайди. Кераксиз устунни қўшгандан кейин уни ўчиришга имконият бўлмаслиги мумкин, масалан, қуйидаги сўров ҳамма МББТ ларда ҳам ишлайвермайди:

ALTER TABLE Vendors

DROP COLUMN vend_phone;

Бунда vend_phone устунини ўчириш талаб қилиняпти.

Баъзи ҳолларда кераксиз жадвалларни ўчиришга тўғри келади. Жадвални ўчириш деганда унинг ёзувларини эмас, балки уни ўзини тўлалигича ўчириш тушунилади.

Жадвални ўчириш буйруғи қуйидаги кўринишга эга:

DROP TABLE< table name >;

Масалан: **DROP TABLE** Vendors;

Бунга кўра Vendors жадвали ўчирилади.

Агар Vendors жадвали бошқа жадваллар билан боғланмаган бўлса, бу буйруқ бажарилади ва Vendors жадвали ўчирилади.

Агар Vendors жадвали бошқа жадваллар билан боғланган бўлса, кўп МББТларда жадваллар орасидаги алоқа бекор қилинмагунча бу буйруқ бажарилмайди. Шунинг учун жадвални ўчиришда хатога йўл қўймаслик учун реляцион қоидаларга амал қилиш тавсия қилинади.

1.6. Жадвалга маълумотларни киритиш

Юқорида биз жадвалларни яратиш тартиби билан танишдик. Энди бу жадвалларни тўлдириш керак бўлади. SQL тилида жадвалга маълумотларни киритиш INSERT буйруғи асосида амалга оширилади.

Унинг кўриниши қуйидагича бўлади:

INSERT жадвал номи ёки

INSERT INTO жадвал номи

INSERT буйруғи битта сатрни киритишига имкон беради, кейинги сатрларнинг киритилиши INSERT буйруғининг қайтарилиши ёрдамида ҳосил бўлади.

INSERT INTO Products

Барча сатрли ўзгарувчилар апострофларга киритилиши лозим.

INSERT INTO синтаксиси қуйидагича бўлади:

INSERT INTO жадвал_номи [(<устунлар рўйхати >)] VALUES (<қийматлар рўйхати >) . Бундай синтаксис жадвалга фақат битта сатр киритиш имконини беради.

Энди INSERT INTO буйруғи ёрдамида **Products** деб номланган маҳсулотлар жадвалига маълумотларни киритайлик.[4] Қулайлик учун 4 та устунни олайлик. Сатрлар сони 9та бўлганлиги сабабли INSERT INTO буйруғи 9 марта такрорланади.

Populate Products table - Products жадвалини тўлдириш

```
INSERT INTO Products(prod_id, vend_id, prod_name, prod_price)
VALUES('BR01', 'BRS01', 'Карамель', 1.29);
INSERT INTO Products(prod_id, vend_id, prod_name, prod_price)
VALUES('BR02', 'BRS01', 'Салат', 0.89  );
INSERT INTO Products(prod_id, vend_id, prod_name, prod_price)
VALUES('BR03', 'BRS01', 'Лўя', 1.09);
```

[4] http ://forta.com/books/0672336073/

```
INSERT INTO Products(prod_id, vend_id, prod_name, prod_price)
VALUES('BNBG01', 'DLL01', 'Қаймоқ', 0.70);
INSERT INTO Products(prod_id, vend_id, prod_name, prod_price)
VALUES('BNBG02', 'DLL01', 'Творог ', 0.80);
INSERT INTO Products(prod_id, vend_id, prod_name, prod_price)
VALUES('BNBG03', 'DLL01', 'Карамель', 1.25);
INSERT INTO Products(prod_id, vend_id, prod_name, prod_price)
VALUES('RGAN01', 'DLL01', 'Ун', 0.65);
INSERT INTO Products(prod_id, vend_id, prod_name, prod_price)
VALUES('RYL01', 'FNG01', 'Қаймоқ', 0.79);
INSERT INTO Products(prod_id, vend_id, prod_name, prod_price)
VALUES('RYL02', 'FNG01', 'Лўя', 1.19);
```

Бунга мос маҳсулотлар жадвалининг кўриниши қуйидагича бўлади:

1-жадвал

prod_id	vend_id	prod_name	prod_price
BR01	BRS01	Карамель	1.29
BR02	BRS01	Салат	0.89
BR03	BRS01	Лўя	1.09
BNBG01	DLL01	Қаймоқ	0.70
BNBG02	DLL01	Творог	0.80
BNBG03	DLL01	Карамель	1.25
RGAN01	DLL01	Ун	0.65
RYL01	FNG01	Қаймоқ	0.79
RYL02	FNG01	Лўя	1.19

Худди шу сингари INSERT INTO буйруғидан фойдаланиб, **Orders** деб номланган буюртмалар жадвалига маълумотлар киритамиз[10]. Бунда сатрлар сони 5та бўлганлиги сабабли INSERT INTO буйруғи 5 марта такрорланади.

Populate Orders table - Orders жадвалини тўлдириш

```
INSERT INTO Orders(order_num, order_date, cust_id)
VALUES(20005, '2012-05-01', '1000000001');
```

```
INSERT INTO Orders(order_num, order_date, cust_id)
VALUES(20006, '2012-01-12', '1000000003');
INSERT INTO Orders(order_num, order_date, cust_id)
VALUES(20007, '2012-01-30', '1000000004');
INSERT INTO Orders(order_num, order_date, cust_id)
VALUES(20008, '2012-02-03', '1000000005');
INSERT INTO Orders(order_num, order_date, cust_id)
VALUES(20009, '2012-02-08', '1000000001');
```

Orders буюртмалар жадвалинингкўриниши қуйидагича бўлади:

2-жадвал

order_num	order_date	cust_id
20005	2012-05-01	1000000001
20006	2012-01-12	1000000003
20007	2012-01-30	1000000004
20008	2012-02-03	1000000005
20009	2012-02-08	1000000001

Қолган 3 та - Customers(Мижозлар),Vendors(Таъминловчилар) ва OrderItems (Буюртма элементлари) жадвалларга ҳам маълумотлар киритамиз.[5]

Populate Customers table - Customers жадвалини тўлдириш

```
INSERT INTO Customers(cust_id, cust_name, cust_address, cust_city, cust_state,
cust_zip, cust_country, cust_contact, cust_email)
VALUES('1000000001', 'Village Toys', '200 Maple Lane', 'Detroit', 'MI', '44444',
'USA', 'John Smith', 'sales@villagetoys.com');
INSERT INTO Customers(cust_id, cust_name, cust_address, cust_city, cust_state,
cust_zip, cust_country, cust_contact)
VALUES('1000000002', 'Kids Place', '333 South Lake Drive', 'Columbus', 'OH',
'43333', 'USA', 'Michelle Green');
```

[5] http ://forta.com/books/0672336073/

```
INSERT INTO Customers(cust_id, cust_name, cust_address, cust_city, cust_state,
cust_zip, cust_country, cust_contact, cust_email)
VALUES('1000000003', 'Fun4All', '1 Sunny Place', 'Muncie', 'IN', '42222', 'USA',
'Jim Jones', 'jjones@fun4all.com');
INSERT INTO Customers(cust_id, cust_name, cust_address, cust_city, cust_state,
cust_zip, cust_country, cust_contact, cust_email)
VALUES('1000000004', 'Fun4All', '829 Riverside Drive', 'Phoenix', 'AZ', '88888',
'USA', 'Denise L. Stephens', 'dstephens@fun4all.com');
INSERT INTO Customers(cust_id, cust_name, cust_address, cust_city, cust_state,
cust_zip, cust_country, cust_contact)
VALUES('1000000005', 'The Toy Store', '4545 53rd Street', 'Chicago', 'IL', '54545',
'USA', 'Kim Howard');
-- ---------------------
-- Populate Vendors table – Vendors жадвалини тўлдириш
-- ---------------------
INSERT INTO Vendors(vend_id, vend_name, vend_address, vend_city,
vend_state, vend_zip, vend_country)
VALUES('BRS01','Bears R Us','123 Main Street','Bear Town','MI','44444', 'USA');
INSERT INTO Vendors(vend_id, vend_name, vend_address, vend_city,
vend_state, vend_zip, vend_country)
VALUES('BRE02','Bear Emporium','500 Park Street','Anytown','OH','44333',
'USA');
INSERT INTO Vendors(vend_id, vend_name, vend_address, vend_city,
vend_state, vend_zip, vend_country)
VALUES('DLL01','Doll House Inc.','555 High Street','Dollsville','CA','99999',
'USA');
INSERT INTO Vendors(vend_id, vend_name, vend_address, vend_city,
vend_state, vend_zip, vend_country)
VALUES('FRB01','Furball Inc.','1000 5th Avenue','New York','NY','11111',
'USA');
```

```
INSERT INTO Vendors(vend_id, vend_name, vend_address, vend_city, vend_state, vend_zip, vend_country)
VALUES('FNG01','Fun and Games','42 Galaxy Road','London', NULL,'N16 6PS', 'England');
INSERT INTO Vendors(vend_id, vend_name, vend_address, vend_city, vend_state, vend_zip, vend_country)
VALUES('JTS01','Jouets et ours','1 Rue Amusement','Paris', NULL,'45678', 'France');
-- ----------------------
```

Populate OrderItems table - OrderItems жадвалини тўлдириш

```
-- ------------------------
INSERT INTO OrderItems(order_num, order_item, prod_id, quantity, item_price)
VALUES(20005, 1, 'BR01', 100, 5.49);
INSERT INTO OrderItems(order_num, order_item, prod_id, quantity, item_price)
VALUES(20005, 2, 'BR03', 100, 10.99);
INSERT INTO OrderItems(order_num, order_item, prod_id, quantity, item_price)
VALUES(20006, 1, 'BR01', 20, 5.99);
INSERT INTO OrderItems(order_num, order_item, prod_id, quantity, item_price)
VALUES(20006, 2, 'BR02', 10, 8.99);
INSERT INTO OrderItems(order_num, order_item, prod_id, quantity, item_price)
VALUES(20006, 3, 'BR03', 10, 11.99);
INSERT INTO OrderItems(order_num, order_item, prod_id, quantity, item_price)
VALUES(20007, 1, 'BR03', 50, 11.49);
INSERT INTO OrderItems(order_num, order_item, prod_id, quantity, item_price)
VALUES(20007, 2, 'BNBG01', 100, 2.99);
INSERT INTO OrderItems(order_num, order_item, prod_id, quantity, item_price)
VALUES(20007, 3, 'BNBG02', 100, 2.99);
INSERT INTO OrderItems(order_num, order_item, prod_id, quantity, item_price)
VALUES(20007, 4, 'BNBG03', 100, 2.99);
INSERT INTO OrderItems(order_num, order_item, prod_id, quantity, item_price)
```

```
VALUES(20007, 5, 'RGAN01', 50, 4.49);
INSERT INTO OrderItems(order_num, order_item, prod_id, quantity, item_price)
VALUES(20008, 1, 'RGAN01', 5, 4.99);
INSERT INTO OrderItems(order_num, order_item, prod_id, quantity, item_price)
VALUES(20008, 2, 'BR03', 5, 11.99);
INSERT INTO OrderItems(order_num, order_item, prod_id, quantity, item_price)
VALUES(20008, 3, 'BNBG01', 10, 3.49);
INSERT INTO OrderItems(order_num, order_item, prod_id, quantity, item_price)
VALUES(20008, 4, 'BNBG02', 10, 3.49);
INSERT INTO OrderItems(order_num, order_item, prod_id, quantity, item_price)
VALUES(20008, 5, 'BNBG03', 10, 3.49);
INSERT INTO OrderItems(order_num, order_item, prod_id, quantity, item_price)
VALUES(20009, 1, 'BNBG01', 250, 2.49);
INSERT INTO OrderItems(order_num, order_item, prod_id, quantity, item_price)
VALUES(20009, 2, 'BNBG02', 250, 2.49);
INSERT INTO OrderItems(order_num, order_item, prod_id, quantity, item_price)
VALUES(20009, 3, 'BNBG03', 250, 2.49);
```

Customers(Мижозлар),Vendors(Таъминловчилар) ва OrderItems (Буюртма элементлари) жадвалларининг кўриниши иловада келтирилган.

Юқорида келтирилган жадваллардан кейинги бобда сўров яратишда, жумладан, сатрларни ва группаларни фильтрлаш, маълумотларни саралаш ва гуруҳлаш жараёнида фойдаланамиз.

Бундан ташқари, бу бобда биз сўров яратишда Laptop номли портатив компьютерлар (ПК) жадвали, *Product номли* маҳсулот жадвали ва *PC* номли жадваллардан фойдаланамиз. Бунда сўров яратишнинг айрим масалалари, жумладан, My SQL да ишлатиладиган INNER JOIN ва LEFT JOIN каби бирлаштириш операторларидан, Transact-SQL да ишлатиладиган TRUNCATE TABLE буйруғидан фойдаланиб сўровлар яратиш назарда тутилган.

II-боб. SQL тилининг қўлланилиши

2.1. SQL тилида сўров яратиш. SELECT инструкцияси

Бизга маълумки, SQL инструкциялари оддий инглиз атамаларидан иборат.Бу атамалар **калит сўзлар** дейилади. SQL нинг ҳар бир инструкцияси бир ёки бир неча калит сўзларни ўз ичига олади. SQL тилининг асосий инструкциялардан бири SELECT инструкцияси бўлиб, у бир ёки бир неча жадваллардан керакли ахборотни олишга мўлжалланган. Бу инструкция ёрдамида сўровлар амалга оширилади.

SELECT оператори SQL тилининг энг кўп қўлланиладиган оператори ҳисобланади. SELECT операторининг синтаксиси қуйидагича:

SELECT [ALL/DISTINCT] <атрибутлар рўйхати>/*
FROM <жадваллар рўйхати >
[WHERE <танлаш шарти >]
[ORDER BY < атрибутлар рўйхати >]
[GROUP BY < атрибутлар рўйхати >]
[HAVING <шарт>]
[UNION< SELECT операторли ифода>]

Бу ерда атрибутлар тушунчаси устунлар, майдонлар каби тушунчаларга эквивалент тушунчадир. Квадрат қавсларда операторни ёзишда қатнашиши шарт бўлмаган элементлар кўрсатилган. ALL калит сўзи натижага шартни қаноатлантирувчи барча сатрлар, шунингдек такрорланувчи сатрлар ҳам киришини билдиради. DISTINCT калит сўзи натижага такрорланувчи сатрлар киритилмаслигини билдиради. Кейин бошланғич жадвалдаги атрибутлар рўйхати кўрсатилади. Бу атрибутлар натижавий жадвалга киритилади. * белгиси натижавий жадвалга бошланғич жадвалнинг барча атрибутлари киритилишини билдиради.

Операторда қатнашиши шарт бўлган сўзлардан FROM калит сўзи бўлиб, бу сўздан кейин танлов бажариладиган жадваллар номи кўрсатилади.

Танлаш ифодасида WHERE калит сўзидан кейин жадвал сатрларини

танлаб олиш шарти кўрсатилади. Бунда натижавий жадвалга WHERE ифодасидаги шарт рост қиймат қабул қиладиган сатрлар киритилади.

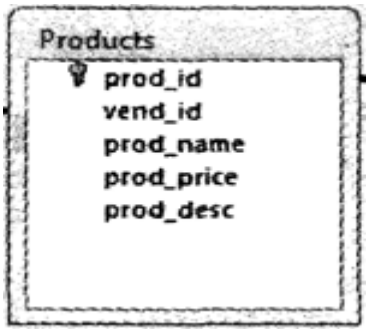
ORDER BY калит сўзи натижавий жадвал сатрларини кўрсатилган устунлар рўйхати бўйича тартиблаш амалини билдиради.

GROUP BY калит сўзидан кейин эса группаланадиган атрибутлар рўйхати кўрсатилади.

HAVING ифодасида ҳар бир группага қўйиладиган шартлар кўрсатилади. (GROUP BY ва HAVING калит сўзлари кейинроқ тушунтирилади)

FROM, WHERE ва ORDER BY калит сўзлари SQL тилининг қолган маълумотларни манипуляциялаш операторларида ҳам шу тарзда ишлатилади.

SELECT инструкцияси билан танишишни оддий мисоллардан бошлаймиз. 1-расмда келтирилган Products жадвалини кўрайлик.



1. Жадвалнинг алоҳида олинган битта устунини ҳосил илиш

Products жадвалидаги prod_name устунини, яъни маҳсулотлар номини ҳосил қилиш сўрови қуйидагича бўлади:

```
SELECT prod_name
FROM Products;
```

Сўров натижаси:

```
prod_name
----------
Карамель
Салат
Лўя
Қаймоқ
```

Творог

Карамель

Ун

Қаймоқ

Лўя

Бунда 6 та маҳсулот бўлишига қарамай 9 та сатр ҳосил бўлди, чунки 3 та маҳсулот номи такрорланяпти.

2. Уникал (такрорланмайдиган) сатрларни ҳосил қилиш

Юқоридаги сўров натижасига эътибор берадиган бўлсак, унда такрорланувчи сатрлар борлигини кўрамиз. Биринчи, учинчи ва тўртинчи сатрлар такрорланувчи бўлиб, мос равишда карамель, лўя ва қаймоқ каби маҳсулотлар номи сўров натижасида такрорланяпти.

Натижага такрорланувчи сатрлар киритилмаслиги учун, яъни сатрлар уникал бўлиши учун DISTINCT калит сўзидан фойдаланамиз:

```
SELECT DISTINCT prod_name
FROM Products;
```

Сўров натижаси эса қуйидагича бўлади:

```
prod_name
------------------
```

Карамель

Салат

Лўя

Қаймоқ

Творог

Ун

Уникал сатрларни ҳосил қилишга доир яна бир мисол:

```
SELECT DISTINCT vend_id
FROM Products;
```

Сўров натижаси қуйидагича бўлади:

```
vend_id
--------------
BRS01
DLL01
FNG01
```

Бу сўров натижаси фақат уникал сатрлардан, яъни 3 та такрорланмайдиган сатрлардан иборат бўлади. Қолган 6 та такрорланувчи сатрлар киритилмаган.

3. Жадвалнинг бир неча устунини ҳосил қилиш

Жадвалнинг бир неча устунини ҳосил қилиш жадвалнинг битта устунини ҳосил қилиш сингари амалга оширилади, лекин бунда SELECT калит сўзидан кейин барча устун номлари вергул орқали кўрсатилиши зарур.

Қуйидаги SELECT инструкцияси Products жадвалидан учта устунни ҳосил қилиш имконини беради, бу устунлар prod_id, prod_name, prod_price номли устунлардир.

```
SELECT prod_id, prod_name, prod_price
FROM Products;
```

Сўров натижаси:

prod_id	prod_name	prod_price
BR01	Карамель	1.29
BR02	Салат	0.89
BR03	Лўя	1.09
BNBG01	Қаймоқ	0.70
BNBG02	Творог	0.80
BNBG03	Карамель	1.25
RGAN01	Ун	0.65
RYL01	Қаймоқ	0.79
RYL02	Лўя	1.19

4. Жадвалнинг барча устунини ҳосил қилиш

Жадвалнинг барча устунини ҳосил қилиш учун SELECT калит сўзидан кейин барча устун номлари кўрсатилиши шарт эмас, унинг ўрнига гуруҳли белги ҳисобланган “юлдузча” (*) белгиси қўйилади:

```
SELECT *
FROM Products;
```

Бу сўров натижаси 1-жадвалда келтирилган, яъни барча тўртта устун маълумотлари келтирилади.

5.Сўров натижаларини чегаралаш

Юқоридаги мисоллардан шуни кўриш мумкинки, SELECT ёрдамида танлаб олинган мезон бўйича барча сатрлар олинади. Шундай ҳоллар бўладики, бизга фақат биринчи сатр ёки берилган сондаги сатрларни олиш керак бўлади. Бунда TOP калит сўзидан фойдаланилади.

Бизга Products жадвалидан биринчи бешта сатрни олиш керак бўлсин. Бу сўров қуйидагича бўлади:

```
SELECT TOP 5  prod_name
FROM Products;
```

Сўров натижаси қуйидагича бўлади:

```
prod_name
------------------------
 Карамель
 Салат
 Лўя
 Қаймоқ
 Творог
```

2.2. Сатрлар бўйича фильтрлаш. WHERE конструкцияси

Маълумотлар базасининг жадвалларида жуда кўп ахборотлар сақланади, камдан-кам ҳолларда жадвалнинг барча сатрларини

ҳосил қилиш эҳтиёжи туғилади. Кўп ҳолларда маълумотларнинг қайсидир қисмини қайта ишлаш ёки ҳисобот тузиш учун олиш керак бўлиб қолади. Бундай ҳолларда танлаш мезони ёки бошқача қилиб айтганда фильтрлаш шарти кўрсатилади.

Сатрларни фильтрлаш учун **WHERE** конструкциясидан фойдаланилади.

SELECT [ALL/DISTINCT] <атрибутлар рўйхати>/*

FROM <жадваллар рўйхати >

[WHERE <танлаш шарти >]

SELECT операторида қатнашиши шарт бўлган сўзлардан FROM сўзи хисобланади. Бу сўздан кейин танлов бажариладиган жадваллар номи кўрсатилади Танлаш ифодасида WHERE калит сўзидан кейин жадвал сатрларини танлаб олиш шарти кўрсатилади. Бунда натижавий жадвалга WHERE ифодасидаги шарт рост қиймат қабул қиладиган сатрлар киритилади.

Бизга Products маҳсулотлар жадвалидан 2 та устунни, бунда prod_price устунида фақат нархи 1.29 га тенг бўлган сатрларни олиш керак бўлсин. Бу сўров қуйидагича бўлади:

SELECT prod_name, prod_price

FROM Products

WHERE prod_price = 1.29;

Сўров натижаси қуйидагича бўлади:

prod_name	prod_price
Карамель	1.29

Юқоридаги мисолда тенгликка текшириш, яъни устунда нархи кўрсатилган қийматга тенг бўлган маҳсулот борлиги аниқланди. SQL да бир қатор шартли (мантиқий) операторлар мавжуд. WHERE ифодасида ишлатиладиган бундай операторлар қуйидаги жадвалда

келтирилган:

Оператор	Текшириш
=	Тенглик
<>	Тенгсизлик
!=	Тенгсизлик
<	Кичик
<=	Кичик ёки тенг
!<	Кичик эмас
>	Катта
>=	Катта ёки тенг
!>	Катта эмас
BETWEEN	Оралиққа(диапазонга) кириш
IS NULL	NULL қиймати

Биз юқорида тенгликка текшириш операторига мисол кўрдик. Энди бошқа операторлар билан танишамиз. Нархи 1 доллардан ошмайдиган маҳсулотлар номини чиқариш керак бўлсин. Бу сўров қуйидагича бўлади:

```
SELECT prod_name, prod_price
FROM Products
WHERE prod_price < 1;
```

Сўров натижаси қуйидагича бўлади:

```
prod_name       prod_price
---------------  --------------
Салат             0.89
Қаймоқ            0.70
Творог            0.80
Ун                0.65
Қаймоқ            0.79
```

Тенгсизликка текшириш операторига мисол кўрайлик. DLL01 фирмаси томонидан тайёрланмаган маҳсулотлар номини чиқариш керак бўлсин. Бу сўров қуйидагича бўлади:

```
SELECT vend_id, prod_name
```

```
FROM Products
WHERE vend_id <> 'DLL01';
```

Сўров натижаси қуйидагича бўлади:

```
vend_id          prod_name
---------------  --------------
BRS01            Карамель
BRS01            Салат
BRS01            Лўя
FNG01            Қаймоқ
FNG01            Лўя
```

Кўпгина МББТларда <> тенгсизлик оператори ўрнига != операторини ишлатса ҳам бўлади:

```
SELECT vend_id, prod_name
FROM Products
WHERE vend_id != 'DLL01';
```

Лекин, Microsoft Accessда <> оператори ишлатилади, != оператори қўллаб-қувватланмайди, уни ишлатишдан аввал ўзингиз ишлатаётган МББТ ҳужжатларига мурожаат қилинг.

Энди BETWEEN операторини қўллаб, нархи 1 дан юқори ва 2 доллардан кам бўлган маҳсулотлар номини чиқарайлик. Бу сўров қуйидагича бўлади:

```
SELECT prod_name, prod_price
FROM Products
WHERE prod_price BETWEEN 1 AND 2;
```

Бу сўров натижаси қуйидагича бўлади:

```
prod_name       prod_price
--------------  --------------
Карамель          1.29
Лўя               1.09
Карамель          1.25
Лўя               1.19
```

BETWEEN конструкцияси true (рост) қийматини қабул қиладиган ифодалар учун диапазон қийматини беради. Бирор фирмага тегишли бўлган **Ходимлар** номли жадвал берилган бўлсин (1-жадвал):

КОДИ	ФАМИЛИЯСИ	ИСМИ	МАОШИ	ЛАВОЗИМИ
10	Расулов	Акбар	1500000	Бош ҳисобчи
25	Каримов	Рустам	900000	Муҳандис
30	Ахмедов	Собир	1000000	Муҳандис
14	Жабборова	Дилдора	1150000	Ҳисобчи
45	Саидов	Мурод	650000	Администратор
51	Зияева	Васила	700000	Менежер
1	Ботиров	Нодир	1800000	Директор
62	Хўжаев	Фарход	500000	Ишчи
54	Обидов	Комил	850000	Менежер
63	Мирзаева	Фарангиз	400000	Ишчи

```
SELECT фамилияси, исми, маоши
FROM ходимлар
WHERE маоши BETWEEN 500000 AND 850000
```

Маоши 500000 билан 850000 оралиғида бўлган ходимлар рўйхатини ҳосил қилиш сўрови натижаси:

ФАМИЛИЯСИ	ИСМИ	МАОШИ
Хўжаев	Фарход	500000
Саидов	Мурод	650000
Зияева	Васила	700000
Обидов	Комил	850000

Бу сўровни таққослаш операторлари ёрдамида ҳам ҳосил қилса бўлади:

```
SELECT фамилияси, исми, маоши
FROM ходимлар
WHERE маоши >= 500000 END маоши <= 850000
```

Маоши 500000 дан катта ва 850000 дан кичик бўлган ходимлар рўйхатини ҳосил қилиш сўрови натижаси:

ФАМИЛИЯСИ	ИСМИ	МАОШИ
Хўжаев	Фарход	500000
Саидов	Мурод	650000
Зияева	Васила	700000
Обидов	Комил	850000

BETWEEN конструкцияли сўровлар қуйидаги кўринишда ҳам бўлиши мумкин:

```
SELECT фамилияси, исми, маоши
FROM ходимлар
WHERE исми BETWEEN 'Мурод' AND 'Собир'
```

Исми 'Мурод' билан бошланадиган ва 'Собир' билан тугайдиган ходимлар рўйхатини ҳосил қилиш натижаси:

ФАМИЛИЯСИ	ИСМИ	МАОШИ
Саидов	Мурод	650000
Ботиров	Нодир	1800000
Каримов	Рустам	900000
Ахмедов	Собир	1000000

Қуйи ва юқори диапазонларни аниқловчи қийматлар реал миқдорлар бўлмаслиги ҳам мумкин, бу жуда қулай, чунки биз ҳар доим ҳам диапазон учун аниқ қийматларни бермаслигимиз мумкин.

```
SELECT фамилияси, исми, маоши
FROM ходимлар
WHERE исми BETWEEN 'Мур' AND 'Соб'
```

Исми 'Мур' ва 'Соб' оралиғида бўлган ходимлар рўйхатини ҳосил қилиш натижаси:

ФАМИЛИЯСИ	ИСМИ	МАОШИ
Саидов	Мурод	650000
Ботиров	Нодир	1800000
Каримов	Рустам	900000
Фозилов	Сардор	820000
Ахмедов	Собир	1000000

Бунда 'Мур' ва 'Соб' қийматлар маълумотлар базасида йўқ. Лекин, исми қуйи диапазондаги 'Мур' ('М', 'Му', 'Мур') ва юқори диапазондаги 'Соб' ('С', 'Со', 'Соб') билан устма-уст тушган барча ходимлар рўйхати танлаш шартига киради. BETWEEN диапазон қиймати қўйилган майдонни ўсиш тартибида саралайди.

Энди IS NULL операторини қўллашга мисол кўрайлик.

Customers мижозлар жадвалида cust_email устунини оладиган бўлсак, бу устунда электрон почта манзили кўрсатилмаган бўлса у NULL қийматга эга бўлади.

```
SELECT cust_name
FROM Customers
WHERE cust_email IS NULL;
```

Натижада электрон манзили кўрсатилмаган мижозлар рўйҳати ҳосил бўлади:

```
cust_name
---------------
Kids Place
The Toy Store
```

2.3. Жадвалга қўшимча маълумотларни киритиш. INSERT ва SELECT INTO операторлари

Маълумотлар базасидаги жадвалга сатрларни қўшиш учун INSERT инструкциясидан фойдаланилади, буни қуйидаги усуллар ёрдамида амалга ошириш мумкин:

- жадвалга битта тўла сатрни қўшиш;
- битта сатрнинг қисмини қўшиш;
- жадвалга бир неча сатрларни қўшиш;
- сўров натижаларини қўшиш.

Келтирилган усулларни кўриб чиқайлик.

Жадвалга тўла сатрни қўшиш

Жадвалга маълумотларни қўшишнинг оддий усули INSERT инструкцияси ёрдамида амалга оширилади, бунинг учун жадвалнинг номи ва янги сатрга киритилиши керак бўлган қийматлар кўрсатилади. Бу инструкция битта сатрни қўшиш имконини беради.

Шуни таъкидлаш лозимки, SQL ни қўллашда баъзи ҳолларда INSERT инструкциясидаги INTO калит сўзи қатнашиши шарт

бўлмаган элемент ҳисобланади. Лекин амалиётлар шуни кўрсатдики, бу калит сўзни талаб этилмаганда ҳам ишлатиш яхши натижалар беради ва хавфсиз ҳисобланади. Шунинг учун биз INSERT INTO дан фойдаланишни тавсия этамиз.

INSERT INTO синтаксиси қуйидагича бўлади:

INSERT INTO жадвал_номи [(<устунлар рўйхати >)] VALUES (<қийматлар рўйхати >).

Мисол кўрайлик. Customers жадвалида бешта мижоз бор эди, энди уникал идентификатори 1000000006 бўлган янги олтинчи мижозни қўшайлик. Бу қуйидагича амалга оширилади:

```
INSERT INTO Customers
VALUES(’1000000006',
        'Toy Land',
        '123 Any Street’,
        'New York',
        'NY',
        '11111',
        'USA',
        NULL,
        NULL);
```

Бу мисолга биноан Customers номли мижозлар жадвалига янги мижоз ҳақида маълумот қўшиляпти. Ҳар бир устунда сақланиши керак бўлган маълумотлар VALUES атрибутида келтирилади. Ҳар бир устун учун қийматлар берилади, агар бирор устун учун мос қиймат бўлмаса, бизнинг мисолимизда бу **cust_contact** ва **cust_email** устунлари бўлиб, уларга **NULL** қиймати берилади. Устунлар жадвалда қай тартибда аниқланган бўлса, шу тартибда тўлдирилади.

Бу кўрсатилган синтаксис анча содда, лекин уни ишончлилигига кафолат йўқ, шунинг учун уни ишлатишни тавсия

этиб бўлмайди, чунки устунлар жойлашиш тартиби жадвал қайта таҳрирланганда ўзгармаслигига кафолат бериб бўлмайди.

Қуйида келтирилган синтаксис бироз мураккаб бўлсада, лекин ишончли ва хавфсиз ҳисобланади.

```
INSERT INTO Customers(cust_id,
                      cust_name,
                      cust_address,
                      cust_city,
                      cust_state,
                      cust_zip,
                      cust_country,
                      cust_contact,
                      cust_email)
VALUES('1000000006',
       'Toy Land',
       '123 Any Street',
       'New York',
       'NY',
       '11111',
       'USA',
       NULL,
       NULL);
```

Бу мисол билан юқоридаги мисол мазмуни бир хил, лекин бу сафар устунлар номи юмалоқ қавсларда жадвал номидан кейин ошкор кўринишда бериляпти. Бунда МББТ ҳар бир устунга VALUES рўйҳатидан олинган қийматни мос қўяди, яъни VALUES рўйҳатидаги биринчи қийматни биринчи устунга, иккинчи қийматни иккинчи устунга ва ҳ. мос қўяди. Бу усулнинг афзаллиги шундаки, жадвалдаги устунлар жойлашуви ўзгарганда ҳам INSERT инструкцияси тўғри ишлайди.

Қуйидаги INSERT инструкцияси ҳам олдинги инструкция сингари тўғри ишлайди, лекин бунда устунлар жойлашиш тартиби аввалгидан фарқ қилади:

```
INSERT INTO Customers(cust_id,
                      cust_contact,
                      cust_email,
                      cust_name,
                      cust_address,
                      cust_city,
                      cust_state,
                      cust_zip,
VALUES('1000000006',
       NULL,
       NULL,
       'Toy Land',
       '123 Any Street',
       'New York',
       'NY',
       '11111';
```

Юқоридагилардан хулоса қилиб шуни айтиш мумкинки, INSERT инструкциясида ҳар доим устунлар рўйҳатини келтиришни тавсия этамиз. Қоидага биноан, INSERT инструкциясини устунлар рўйҳатини ошкор келтирмасдан туриб ишлатиб бўлмайди. Бунга амал қилингандагина сўровни муваффақиятли амалга оширса бўлади.

Жадвалга битта сатрнинг қисмини қўшиш

INSERT инструкциясини ишлатишнинг юқорида тавсия қилинган усулига кўра ҳар бир устун номи ошкор кўрсатилиши керак. Бундай синтаксисни қўллаш маълум устунларни ташлаб

кетиш имконини беради, яъни баъзи устунларга қиймат берилади, баъзиларини эса ташлаб кетиш мумкин.

Қуйидаги мисолни кўрайлик.

```
INSERT INTO Customers(cust_id,
                      cust_name,
                      cust_address,
                      cust_city,
                      cust_state,
                      cust_zip,
                      cust_country)
VALUES('1000000006’,
       'Toy Land’,
       ’123 Any Street',
       'New York',
       'NY',
       '11111',
       'USA')
```

Юқоридаги мисолда иккита **cust_contact** ва **cust_email** устунларига **NULL** қиймати берилган эди, бу эса бу устунларни INSERT инструкциясига киритишга сабаб йўқлигини билдиради. Шунинг учун ҳозирги инструкцияга бу икки устун ва уларга мос қийматлар киритилмади, яъни сатрга тегишли бўлган 9 та устундан 7 таси киритилди, бу эса тўла сатр эмас, балки сатрнинг қисми қўшилганлигини билдиради.

Шуни алоҳида таъкидлаш лозимки, INSERT инструкциясидан устунлар фақат **NULL** қийматига эга бўлганда ёки жимликка кўра қиймат аниқланганда олиб ташланиши мумкин. Агар улардан фарқли қийматга эга бўлган устун олиб ташланишга уриниш бўлса, МББТ хатолик ҳақида маълумот беради ва сатр қўшилмайди.

Жадвалга бир неча сатрларни қўшиш

Юқорида таъкидлаб ўтилганидек, INSERT инструкцияси одатда жадвалга битта сатрни қўшиш имконини беради. Бир неча сатрларни қўшиш учун эса бир неча INSERT инструкциясини бажаришга тўғри келади. Бу қоидадан истисно тариқасида INSERT SELECT инструкцияси ишлатилади. Унга кўра, битта сўров ёрдамида жадвалга бир неча сатрларни қўшиш мумкин.

INSERT SELECT инструкцияси жадвалга SELECT инструкциясини бажариш натижаларини қўшади.

INSERT SELECT инструкциясининг синтаксиси:

INSERT INTO жадвал_номи [(устунлар, ...)]

SELECT устунлар, ... FROM жадвал_номи, ...

[WHERE ...];

Сўров натижаларини қўшиш

Одатда INSERT инструкцияси жадвалга ошкор берилган қийматлардан фойдаланиб сатрни қўшишга хизмат қилади. INSERT инструкциясининг яна бир шакли мавжуд бўлиб, ундан SELECT сўрови натижаларини қўшишда фойдаланиш мумкин. Бу инструкция INSERT SELECT инструкцияси бўлиб, унинг номининг ўзи INSERT ва SELECT инструкцияларини алоҳида қўллаганда бажариладиган натижани англатади.

Фараз қилайлик, Customers номли мижозлар жадвалига бошқа CustNew жадвалидан мижозлар рўйҳатини киритиш керак бўлсин. Бунинг учун CustNew жадвалидан ҳар сафар биттадан сатрни олиб, сўнгра INSERT инструкцияси ёрдамида уларни Customers жадвалига қўшиш керак бўлади. Буни ўрнига биз қуйидагича сўровни бажарамиз:

```
INSERT INTO Customers(cust_id,
                      cust_contact,
                      cust_email,
```

```
                    cust_name,
                    cust_address,
                    cust_city,
                    cust_state,
                    cust_zip,
                    cust_country)
SELECT cust_id,
          cust_contact,
          cust_email,
          cust_name,
          cust_address,
          cust_city,
          cust_state,
          cust_zip,
          cust_country
FROM CustNew;
```

Бу мисолда CustNew жадвалидаги барча маълумотларни Customers жадвалига импорт қилиш учун INSERT SELECT инструкцияси қўлланиляпти. Бунга кўра, аввал SELECT инструкцияси ёрдамида CustNew жадвалидаги қийматлар ҳосил қилинади, сўнгра INSERT инструкцияси ёрдамида Customers жадвалига сатрлар қўшилади.

SELECT инструкциясидаги ҳар бир устун INSERT рўйҳатидаги устунга мос келади. CustNew жадвалида қанча сатр бўлса, INSERT инструкцияси шунча сатрни қўшади.

Шуни таъкидлаш лозимки, бу мисолда оддийлик учун устунлар номи бир хил қилиб олинди, лекин бу шарт эмас, одатда МББТ устунлар номига эътибор қаратмайди, у устун тартибини ҳисобга олади, яъни SELECT инструкциясидаги биринчи устун жадвалнинг биринчи устунини тўлдиришда фойдаланилади.

INSERT SELECT инструкциясидаги WHERE калит сўзи қатнашиши шарт бўлмаган элемент бўлиб, у фақат маълумотларни фильтрлашда ишлатилади.

Жадвалга қўшимча маълумотларни киритишнинг яна бир бошқа усули мавжудки, унда INSERT оператори умуман қўлланилмайди. Бир жадвалдаги маълумотлардан бошқа янги жадвалга нусха олиш учун SELECT INTO инструкциясини ишлатиш мумкин.

INSERT SELECT инструкцияси ёрдамида қўшимча маълумотлар мавжуд жадвалга киритилган бўлса, ундан фарқли равишда SELECT INTO инструкцияси маълумотлардан янги жадвалга нусха олади, шунингдек, МББТ турига қараб, агар жадвал мавжуд бўлса, уни қайтадан ёзади.

SELECT INTO ва INSERT SELECT инструкцияларининг фарқи шундаки, биринчиси маълумотларни экспорт қилса, иккинчиси импорт қилади.

SELECT INTO инструкциясини қўллашга доир мисол кўрайлик. Унга кўра, бу инструкция ёрдамида CustCopy номли янги жадвал яратилади ва унга Customers жадвалидаги барча маълумотлардан нусха олинади:

```
SELECT *
INTO CustCopy
FROM Customers;
```

SELECT * синтаксиси қўлланилганлиги учун Customers жадвалининг барча устунлари CustCopy жадвалида яратилади, агар баъзи устунлардан нусха олиш керак бўлса, у ҳолда * метасимволи ишлатилмайди ва керакли устун номи кўрсатилади.

MariaDB, MySQL, Oracle, PostgreSQL ва SQLite каби МББТ лар бироз бошқача синтаксисни қўллаб-қувватлайди:

```
CREATE TABLE CustCopy AS
```

SELECT * FROM Customers;

SELECT INTO инструкциясини қўллашда қуйидагиларга эътибор қилиш керак:

- SELECT инструкциясининг барча калит сўзларини, жумладан, WHERE ва GROUP BY ни қўллашга ҳам рухсат берилади;

- бир неча жадвал маълумотларини қўшиш учун уларни бирлаштириш мумкин;

- нечта жадвалдан маълумотлар олинишидан қатъий назар, маълумотларни фақат битта жадвалга қўшиш мумкин.

SELECT INTO инструкцияси SQL нинг янги инструкциялари тадқиқотида жадваллардан нусха олишнинг энг яхши воситаси бўлиб ҳисобланади. Нусха олиш билан сиз SQL инструкцияларини реал маълумотлар базаси жадвалларида эмас, балки шу олинган нусха ёрдамида тестдан ўтказиш имкониятига эга бўласиз.

Шундай қилиб, жадвалларга сатрлар қўшишда INSERT, INSERT SELECT ва SELECT INTO инструкциялари қўлланилади.

2.4. Жадвалдаги ёзувларни янгилаш ва ўчириш

Жадвалдаги ёзувларни янгилаш учун UPDATE инструкциясидан фойдаланилади, бунда қуйидаги икки усул мавжуд:

- жадвалнинг маълум бир сатрларини янгилаш;
- жадвалнинг барча сатрларини янгилаш.

UPDATE инструкцияси жуда содда бўлиб, у қуйидаги уч қисмдан иборат:

- янгилаш керак бўлган жадвал номи;
- устун номи ва унинг янги қийматлари;
- янгиланиши керак бўлган сатрлар учун фильтрлаш шартлари.

UPDATE операторининг синтаксиси қуйидагича:

```
UPDATE < жадвал номи >
  SET{устун номи={устун қийматини ҳисоблаш учун ифода
   | NULL
   | DEFAULT},...}
   [ {WHERE <танлаш шарти>}];
```

Агар устун NULL қийматга эга бўлса, уни ошкор кўринишда бериш мумкин, агар жимлик (DEFAULT) ҳолати бўлса, у ҳолда устуннинг қиймати аввалги қиймат билан алмаштирилади. Агар WHERE конструкцияси қатнашмаса, у ҳолда жадвалнинг барча сатрлари янгиланади.

Мисол. Customers жадвалидаги уникал идентификатори 1000000005 бўлган мижозда электрон почта манзили пайдо бўлди, шунинг учун унинг ёзувини янгиловчи сўровни ҳосил қилайлик:

```
UPDATE Customers
SET cust_email =  'kim@thetoystore.com'
WHERE cust id = '1000000005';
```

UPDATE инструкцияси ҳар доим янгиланиши керак бўлган жадвал номи билан бошланади. Бизнинг мисолимизда бу Customers жадвалидир. Сўнгра SET конструкциясидан фойдаланиб cust_email устунининг қиймати янгиланади, яъни электрон почта манзили берилади: SET cust_email = 'kim@thetoystore.com'

UPDATE инструкцияси янгиланиши керак бўлган сатр учун фильтрлаш шартини берувчи WHERE конструкцияси билан тугайди. Демак, жадвалнинг cust_email устунида янги ёзув пайдо бўлади, бу ёзув kim@thetoystore.com кўринишдаги ёзувдир ва у идентификатори 1000000005 бўлган мижознинг электрон почта манзилидир.

Яна мисол кўрайлик. Қуйидаги оператор ёрдамида биз Laptop номли портатив компьютерлар (ПК) жадвалидаги барча ПК-

блокнотларнинг нархини 10 фоизга туширишимиз мумкин:[6]

```
UPDATE Laptop
SET price=price*0.1
```

Фараз қилайлик, ПК-блокнотлардаги хотира ҳажми 10 ГБ дан кам бўлган hd қаттиқ дискларни алмаштириш талаб қилинсин. Бунда янги дискларнинг ҳажми улардаги қурилмаларнинг мавжуд RAM (Random Access Memory- ихтиёрий кириш мумкин бўлган хотира) ҳажмининг ярмига тенг бўлиши керак. Бу масала қуйидагича ечилади:

```
UPDATE Laptop
SET hd=ram/2
WHERE hd<10;
```

Бунда hd ва ram устунлардаги маълумот тури бир хил бўлиши керак.

Устун қийматларини ҳисоблашда сўров ости, яъни сўров ичидаги сўров (ичма-ич жойлашган сўров) лардан ҳам фойдаланиш мумкин. Масалан, барча ПК-блокнотларни ўзларида мавжуд бўлган тезкор процессорлар билан янгилаш талаб қилинсин.

Бу сўров ичидаги сўровни қуйидагича ёзиш мумкин:

```
UPDATE Laptop
 SET speed = (SELECT MAX(speed) FROM Laptop)
```

Энди бир неча устунни янгилашга доир мисол кўрайлик.

Customers жадвалидаги уникал идентификатори 1000000006 бўлган мижозда электрон почта манзили ва унинг контакт шахси пайдо бўлди, шунинг учун унинг ёзувини янгиловчи сўровни ҳосил қилайлик:

```
UPDATE Customers
SET cust_contact = 'Sam Roberts',
cust_email =  'sam@toyland.com'
```

[6] Справка по SQL(DML): Операторы UPDATE и DELETE. http://www.sql-ex.ru/help/select12.php?Lang=0

WHERE cust id = '1000000006';

Бу сўровда битта SET конструкциясида иккита устун қиймати бир-биридан вергул билан ажратилади. Бизнинг мисолимизда бу иккита cust_contact ва cust_email устун қийматлари идентификатори 1000000006 бўлган мижоз учун янгиланяпти.

Устундаги қийматни ўчириш учун унга NULL қийматини бериш керак. Буни қуйидагича бажариш мумкин:

UPDATE Customers

SET cust_email = NULL

WHERE cust id = '1000000005';

Бу ерда cust_email устунидаги қийматни ўчириш учун NULL калит сўзидан фойдаланамиз. Бу бўш сатрни сақлашни билдирмайди. Бўш сатр ҳам ўзича қийматга эга деб ҳисобланади ва ' ' кўринишда ёзилади, NULL эса қийматнинг йўқлигини билдиради.

Жадвалдаги ёзувларни ўчириш учун DELETE инструкциясидан фойдаланилади, бунда қуйидаги икки усул мавжуд:

- жадвалнинг маълум бир сатрларини ўчириш;
- жадвалнинг барча сатрларини ўчириш.

Иккала усулни ҳам кўриб чиқишдан олдин DELETE операторининг синтаксиси билан танишайлик. У қуйидагича бўлади:

DELETE FROM <жадвал номи > [WHERE <танлаш шарти>];

Агар WHERE конструкцияси қатнашмаса, у ҳолда жадвалнинг барча сатрлари ўчирилади.

Бу амални тезроқ бажариш учун Transact-SQL тилида қуйидаги буйруқ мавжуд:

TRUNCATE TABLE <жадвал номи >

Laptop номли портатив компьютерлар (ПК) жадвалидан экран ўлчови 12 дюймдан кичик бўлган барча ПК-блокнотларни ўчириш

талаб қилинсин, у ҳолда бу сўров қуйидагича ёзилади:

```
DELETE FROM Laptop
  WHERE screen < 12;
```

Барча блокнотларни ўчириш учун қуйидаги оператордан фойдаланилади:

```
DELETE FROM Laptop
```

ёки

```
TRUNCATE TABLE Laptop
```

Юқорида UPDATE инструкцияси жуда содда эканлиги айтиб ўтилган эди, DELETE инструкцияси эса ундан ҳам содда инструкция ҳисобланади. Қуйидаги инструкция Customers жадвалидан битта сатрни ўчиради:

```
DELETE FROM Customers
WHERE cust id =  '1000000006';
```

DELETE FROM инструкцияси маълумотлар ўчирилиши керак бўлган жадвал номини кўрсатади. WHERE конструкцияси эса ўчирилиши керак бўлган сатрни фильтрлайди. Бу идентификатори 1000000006 бўлган мижозга тегишли сатрдир. Агар WHERE конструкцияси бўлмаса DELETE FROM инструкцияси Customers жадвалидан барча сатрларни ўчириб ташлар эди.

Баъзи SQL версияларида FROM калит сўзи DELETE инструкциясидан кейин қатнашиши шарт эмас деб ҳисобланади, лекин бу калит сўзни ишлатиш талаб қилинмаганда ҳам уни кўрсатиш яхши амалиёт ҳисобланади.

Яна бир мисол кўрайлик. *Product* маҳсулот жадвалидан *PC* жадвалида унга мос келадиган сатрлари мавжуд бўлмаган маҳсулот моделларини ўчириш талаб қилинсин.

Стандарт синтаксисдан фойдаланиб бу масалани қуйидагича ечиш мумкин:

```
DELETE FROM Product
```

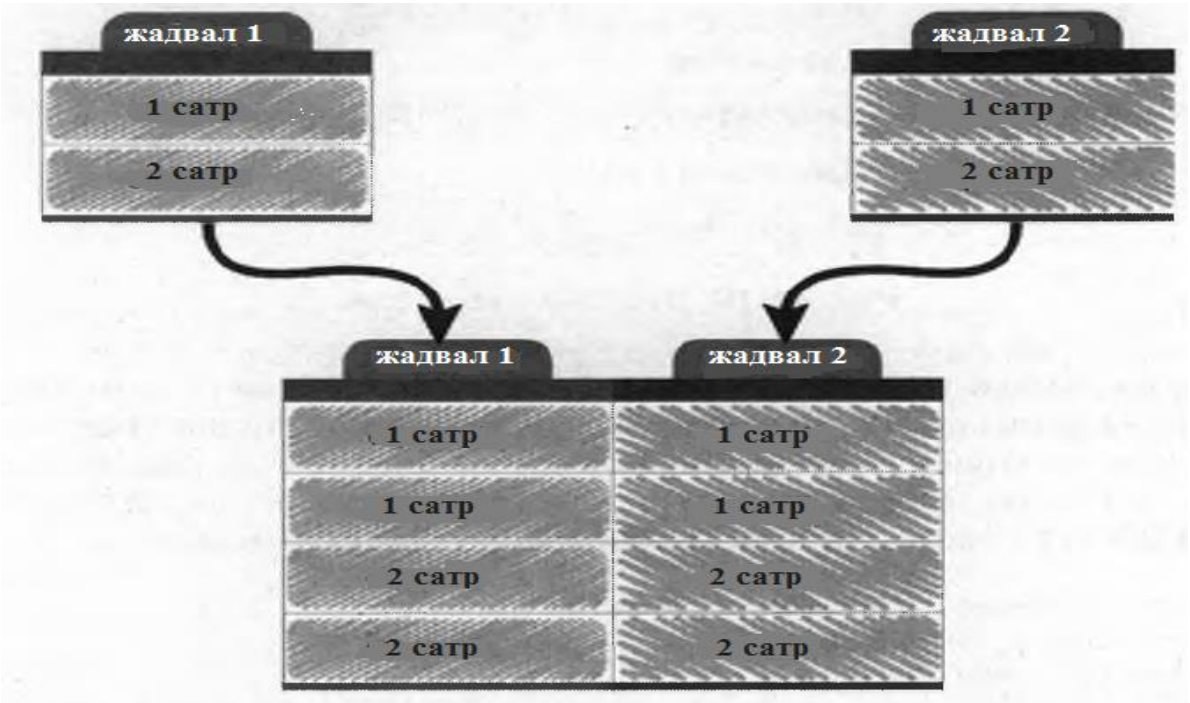
WHERE type='pc' AND model NOT IN (SELECT model FROM PC)

Шуни таъкидлаш лозимки, принтер моделлари ва ПК-блокнотлар ўчириб ташланмаслиги учун type='pc' шарти бу ерда албатта бўлиши керак.

Бу масалани My SQLда қўшимча FROM калит сўзидан фойдаланиб ҳам ечиш мумкин:

```
DELETE FROM Product
  FROM Product pr LEFT JOIN PC ON pr.model=pc.model
  WHERE type='pc' AND pc.model IS NULL
```

Бизга маълумки, My SQLда INNER JOIN оператори ички бирлаштириш оператори бўлиб, бир жадвал элементини бошқа жадвал элементи билан бир-бирига ўхшаш ҳолатда, яъни улар мос бўлса бирлаштиради (2-расм). Иккита жадвални стандарт бирлаштиришда My SQL барча мумкин бўлган сатрлар комбинациясини ажратади[7].



2-расм. Ички бирлаштириш барча мумкин бўлган сатрлар комбинациясини ажратади

[7] Кевин Янк. Простой способ создать сайт на основе базы данных. PHP и MySQL. От новичка к профессионалу. М., ЭКСМО, 5-издание, 2013, с.277

Оддий ҳолни кўрайлик. Ҳар бир жадвал иккитадан ёзувга эга бўлганда бирлаштириш қуйидагича бўлади: 1-жадвалнинг 1-сатри 2-жадвалнинг 1-сатри билан; 1-жадвалнинг 1-сатри 2-жадвалнинг 2-сатри билан; 1-жадвалнинг 2-сатри 2-жадвалнинг 1-сатри билан; 1-жадвалнинг 2-сатри 2-жадвалнинг 2-сатри билан бирлаштирилади. Му SQL натижани олиб, ON шартига қараб қайси сатрни, масалан, устунлари бир-бирига мос бўлган сатрларни қолдириш кераклигини ҳал қилади.

Бизнинг мисолимизда эса бунинг акси, яъни бир жадвал элементи бошқа жадвал элементи билан бир-бирига мос келмаган ҳолатда бирлаштириш талаб қилиняпти, бунда LEFT JOIN чапдан бирлаштирувчи оператордан фойдаланиляпти. Му SQLда LEFT JOIN иккита жадвални чапдан бирлаштириш амалини бажаради. Бу оператор сўров натижаларида жадвалда чапдан жойлашган сатрларни акс эттиради.

Бу ерда ташқи бирлаштиришдан фойдаланиляпти, натижада PC жадвалда қатнашмаган ПК-блокнотлар моделининг *pc.model* устуни NULL қийматга эга бўлади.

Юқорида кўриб ўтилган барча UPDATE ва DELETE инструкциялари WHERE конструкцияси билан биргаликда ишлатилди ва бунинг сабаби бор. Агар DELETE инструкциясини WHERE конструкциясисиз бажарилса жадвалнинг барча маълумотлари ўчиб кетади. Шунинг учун қуйидаги тавсияларга амал қилиш керак бўлади:

► ҳеч қачон UPDATE ва DELETE инструкциялари WHERE конструкциясисиз ишлатманг.

► ҳар бир жадвал бирламчи калитга эга эканлигига ишонч ҳосил қилинг ва агар керак бўлса уни WHERE конструкциясида ҳар сафар ишлатинг.

► WHERE конструкциясини UPDATE ва DELETE инструкциялари билан биргаликда ишлатишдан олдин уни SELECT инструкцияси билан текшириб кўринг ва ёзувларни тўғри фильтрлаётганлигига ишонч ҳосил қилинг, чунки нотўғри шарт ёзиб хатога йўл қўйиш мумкин.

► МББТ бошқа жадваллардаги маълумотлар билан боғлиқ бўлган сатрларни ўчиришга имкон бермаслиги учун ташқи калитлардан фойдаланинг.

► баъзи МББТлар маълумотлар базаси администраторига UPDATE ва DELETE инструкцияларини WHERE конструкциясисиз ишлатилишига тўсиқ бўладиган чекланишларни ўрнатадилар, агар сиз ишлатаётган МББТ бунга имкон берса бемалол WHERE конструкциясисиз ишлатишингиз мумкин.

Шуни ёдда тутингки, SQL да амални бекор қилиш тугмалари йўқ, шунинг учун UPDATE ва DELETE инструкцияларидан эҳтиётлик билан фойдаланиш зарур.

DELETE инструкцияси устунлар номини ва метасимволларни қабул қилмайди. У алоҳида олинган устунларни эмас, балки сатрларни бутунлигича ўчиради. Маълум бир устунни ўчириш учун UPDATE инструкциясидан фойдаланиш керак.

DELETE инструкцияси жадвалдаги алоҳида олинган сатрларни ёки барча сатрларни бир пайтда ўчиради, лекин жадвални ўзини ўчирмайди.

2.5. SQL тилининг агрегат функциялари

Кўп ҳолларда маълумотларни таҳлил қилиш, ҳисоботлар яратиш ва якуний хулоса чиқаришга тўғри келади. Бунинг учун SQL тилида махсус функциялар мавжуд.

Бундай функцияли SQL-сўровларга қуйидаги мисолларни келтириш мумкин;

-Жадвал устунидаги сатрлар сонини аниқлаш;

-устундаги қийматлар йиғиндисини аниқлаш;

-жадвал устунидаги энг катта, энг кичик ва ўртача қийматни аниқлаш.

Бу мисоллар шуни кўрсатадики, фойдаланувчига жадвал бўйича фақат якуний ахборот керак. Бундай ахборотни олишни енгиллаштириш учун SQL тилида бешта махсус функция мавжуд ва улар агрегат функциялар дейилади.

Бу функциялар қуйидаги жадвалда келтирилган:

Функция	Бажарадиган вазифаси
AVG ()	Устундаги қийматларнинг ўртачаси
COUNT ()	Устундаги сатрлар сони
MAX ()	Устундаги энг катта қиймат
MIN ()	Устундаги энг кичик қиймат
SUM ()	Устундаги қийматлар йиғиндиси

AVG () функцияси

AVG () функцияси устундаги қийматларнинг ўртачасини аниқлаш учун мўлжалланган. Бу функцияни маълум олинган сатрлардаги қийматларнинг ўртачасини аниқлашда ҳам қўлласа бўлади. Уни қўллашга доир мисоллар кўрайлик.

1.Products жадвалидаги барча маҳсулотларнинг ўртача нархини аниқлаш талаб этилсин. Бу сўров SQL тилида қуйидаги кўринишда бўлади:

```
SELECT AVG (prod_price) AS avg_price
FROM Products;
```

Натижа қуйидагича бўлади:

```
avg_price
---------------
  0.96111
```

Бу ерда avg_price – псевдоним (тахаллус), яъни ҳисоблашдан кейин ҳосил бўладиган янги майдон номи ва у AS калит сўзидан

кейин келади.

2. DLL01 таъминотчи томонидан тавсия этилган маҳсулотларнинг ўртача нархини аниқлаш керак бўлсин. Бу сўров SQL тилида қуйидаги кўринишда бўлади:

```
SELECT AVG (prod_price) AS avg_price
FROM Products
WHERE vend_id = 'DLL01'
```

Натижа қуйидагича бўлади:

```
avg_price
---------------
 0.8500
```

COUNT () функцияси

COUNT () функцияси сатрлар сонини ҳисоблайди. Унинг ёрдамида жадвалдаги сатрларнинг умумий сонини ёки аниқ бир мезон ёки талабни қаноатлантирувчи сатрлар сонини аниқлаш мумкин.

Бу функциядан 2 хил кўринишда фойдаланиш мумкин:

- COUNT (*) кўринишида, бунда устун қийматлари NULL(бўш) ёки NOT NULL (бўш эмас) лигидан қатъи назар, жадвалдаги сатрлар сони ҳисобланади;

- COUNT (устун) кўринишида, бунда қиймати мавжуд бўлган устунларга мос сатрлар сони аниқланади ва NULL(бўш) қиймати инкор қилинади.

Мисол кўрайлик. Customers жадвалидаги мижозлар сонини аниқлаш талаб этилсин. Бунга мос сўров қуйидагича бўлади:

```
SELECT COUNT (*) AS num_cust
FROM Customers;
```

Натижа:

```
num_cust
--------------
     5
```

Бу мисолда COUNT (*) функцияси барча сатрлар сонини уларнинг қиймати қандай бўлишидан қатъи назар ҳисоблаб беради. Натижа num_cust псевдонимли устун кўринишида берилган.

2. Бу мисолда фақат электрон почта манзилига эга бўлган мижозлар сони ҳисобланади:

```
SELECT COUNT(cust_email) AS num_cust
FROM Customers;
```

Натижа:

```
num_cust
---------------
      3
```

Бу инструкцияда COUNT () функциясидан cust_email устунида нолдан фарқли қийматга эга бўлган сатрлар сонини ҳисоблаш учун фойдаланилган. Бундай сатрлар сони учта, яъни 5 та мижоздан 3 таси электрон почта манзилига эга.

MAX функцияси

MAX функцияси кўрсатилган устундаги энг катта қийматни чиқариб беради. Бунинг учун устуннинг номи кўрсатилиши зарур. Бундай сўров қуйида келтирилган:

```
SELECT MAX(prod_price) AS max_price
FROM Products;
```

Натижа:

```
max_price
---------------
  1.29
```

Бу ерда MAX функцияси Products жадвалидаги энг қиммат маҳсулотнинг нархини чиқариб беряпти.

MIN функцияси

MIN функцияси кўрсатилган устундаги энг кичик қийматни чиқариб беради. Бундай сўров қуйида келтирилган:

```
SELECT MIN(prod_price) AS min_price
FROM Products;
```

Натижа:

```
min_price
---------------
  0.65
```

Бу ерда MIN функцияси Products жадвалидаги энг арзон маҳсулотнинг нархини чиқариб беряпти.

SUM функцияси

SUM функцияси кўрсатилган устундаги қийматлар йиғиндсини чиқариб беради. Бунинг учун устуннинг номи кўрсатилиши зарур.

Мисол кўрайлик. OrderItems жадвалида order item номли буюртма элементлари устуни мавжуд бўлиб, ҳар бир элементга буюртма қилинган маҳсулот сони мос келади.Буюртма номери 20005 бўлган буюртма маҳсулотларининг умумий сони, яъни quantity устунидаги қийматлар йиғиндисини ҳисоблаш сўрови:

```
SELECT SUM(quantity)  AS item_ordered
FROM OrderItems
WHERE order_item = 20005;
```

Натижа:

```
item ordered
----------------
200
```

2. SUM () функциясини ҳисобланувчи майдонлар учун ҳам қўллаш мумкин. Қуйидаги мисолда ҳар бир элемент учун буюртманинг умумий нархи item_price*quantity ифоданинг йиғиндиси орқали аниқланади:

```
SELECT SUM(item_price*quantity)  AS total_price
FROM Order Items
WHERE order_item = 20005;
```

Натижа:

```
total_price
----------------
1648.0000
```

Бу натижа total_price псевдонимли устун кўринишида берилган бўлиб, буюртма номери 20005 бўлган иккита сатрларга мос бўлган item_price ва quantity устунлардаги қийматларни бир-бирига кўпайтириб қўшиш натижасида ҳосил бўляпти: 5.49*100+10.99*100=1648.

Агрегат функцияларни бирлаштириш

Юқорида кўрилган мисолларда фақат битта агрегат функция ишлатилди. Шуни айтиш мумкинки, SELECT инструкцияси бир вақтда бир неча агрегат функциялардан фойдаланиш имконини беради. Мисол:

```
SELECT COUNT (*) AS num_items,
MIN(prod_price)  AS price_min,
MAX(prod_price)  AS price_max,
AVG(prod_price) AS proce_avg
FROM Products;
```

Натижа бўлади:

```
num_items    price_min   price_max    price_avg
------------  ------------ ------------  --------------
     9         0.6500       1.2900         0.8500
```

2.6. SQL тилида маълумотларни саралаш. ORDER BY конструкцияси

Саралаш ва фильтрлаш операторларидан фойдаланиб SQL-сўровларни яратишни амалий нуқтаи назардан кўриб чиқайлик. Юқорида 2.2 – бўлимда келтирилган **Ходимлар** номли 1- жадвални кўрайлик.

Бизга устун номларига кўра тартиблаш ва фамилияси алфавит тартибида жойлаштирилган ходимлар рўйхатини ҳосил қилиш керак бўлсин.

1-жадвал

КОДИ	ФАМИЛИЯСИ	ИСМИ	МАОШИ	ЛАВОЗИМИ
10	Расулов	Акбар	1500000	Бош ҳисобчи
25	Каримов	Рустам	900000	Муҳандис
30	Ахмедов	Собир	1000000	Муҳандис
14	Жабборова	Дилдора	1150000	Ҳисобчи
45	Саидов	Мурод	650000	Администратор
51	Зияева	Васила	700000	Менежер
1	Ботиров	Нодир	1800000	Директор
62	Хўжаев	Фарход	500000	Ишчи
54	Обидов	Комил	850000	Менежер
63	Мирзаева	Фарангиз	400000	Ишчи

Бунинг учун SELECT операторидан, FROM ва ORDER BY калит сўзларидан фойдаланамиз.

Бунда SELECT операторидан кейин тартиблаш керак бўлган устун номлари: фамилияси, исми, коди, лавозими, маоши, FROM калит сўзидан кейин жадвал номи (Ходимлар) ва ORDER BY калит сўзи ёрдамида эса саралаш тартиби аниқланади, бизнинг мисолда саралаш тартиби сифатида фамилияси ўсиш тартибида жойлаштирилган ходимлар рўйхати танлаб олиняпти :

```
SELECT фамилияси, исми, коди, лавозими, маоши
FROM ходимлар
ORDER BY фамилияси
```

Устун номларига кўра тартиблаш ва фамилияси алфавит (ўсиш) тартибида жойлаштирилган ходимлар рўйхати қуйидагича бўлади:

ФАМИЛИЯСИ	ИСМИ	КОДИ	ЛАВОЗИМИ	МАОШИ
Ахмедов	Собир	30	Муҳандис	1000000
Ботиров	Нодир	1	Директор	1800000
Жабборова	Дилдора	14	Ҳисобчи	1150000
Зияева	Васила	51	Менежер	700000
Каримов	Рустам	25	Муҳандис	900000
Мирзаева	Фарангиз	63	Ишчи	400000
Обидов	Комил	54	Менежер	850000
Расулов	Акбар	10	Бош ҳисобчи	1500000
Саидов	Мурод	45	Администратор	650000
Хўжаев	Фарход	62	Ишчи	500000

Энди фамилияси тескари алфавит тартибида жойлаштирилган ходимлар рўйхатини ҳосил қилайлик:

```
SELECT фамилияси, исми, коди, лавозими, маоши
FROM ходимлар
ORDER BY фамилияси DESC
```

Фамилияси тескари алфавит (камайиш) тартибида жойлаштирилган ходимлар рўйхати қуйидагича бўлади:

ФАМИЛИЯСИ	ИСМИ	КОДИ	ЛАВОЗИМИ	МАОШИ
Хўжаев	Фарход	62	Ишчи	500000
Саидов	Мурод	45	Администратор	650000
Расулов	Акбар	10	Бош ҳисобчи	1500000
Обидов	Комил	54	Менежер	850000
Мирзаева	Фарангиз	63	Ишчи	400000
Каримов	Рустам	25	Муҳандис	900000
Зияева	Васила	51	Менежер	700000
Жабборова	Дилдора	14	Ҳисобчи	1150000
Ботиров	Нодир	1	Директор	1800000
Ахмедов	Собир	30	Муҳандис	1000000

Сатрни чиқариш тартибини аниқловчи устун номи танланган устунлар рўйхатида қатнашиши шарт эмас, масалан, маоши устуни SELECT операторида танланган устунлар рўйхатида йўқ, лекин ходимлар рўйхатини уларнинг маошлари тартибланган ҳолда ҳосил қилиш мумкин:

```
SELECT фамилияси, исми, коди, лавозими
FROM ходимлар
ORDER BY маоши
```

Бу сўров натижаси қуйидагича бўлади:

ФАМИЛИЯСИ	ИСМИ	КОДИ	ЛАВОЗИМИ
Мирзаева	Фарангиз	63	Ишчи
Хўжаев	Фарход	62	Ишчи
Саидов	Мурод	45	Администратор
Зияева	Васила	51	Менежер
Обидов	Комил	54	Менежер
Каримов	Рустам	25	Муҳандис
Ахмедов	Собир	30	Муҳандис
Жабборова	Дилдора	14	Ҳисобчи
Расулов	Акбар	10	Бош ҳисобчи
Ботиров	Нодир	1	Директор

2.7. Бирлаштирилган сўровлар

Кўпгина SQL-сўровларда битта ёки бир неча жадваллардан маълумотларни олиш учун битта SELECT инструкциясидан фойдаланилади. SQL, шунингдек, SELECT инструкциясидан кўп марта фойдаланиш натижасида бир нечта сўровларни бажариш ва ягона натижа олиш имконини беради. Бундай сўровлар бирлаштирилган ёки комбинациялашган сўровлар дейилади.

Шуни алоҳида таъкидлаш керакки, битта жадвалдан олинган бирлаштирилган иккита сўров натижаси WHERE конструкциясидаги бир неча шартлар қатнашган битта сўров натижаси билан бир хил бўлади. Бошқача сўз билан айтганда, бир неча шартдан иборат бўлган WHERE конструкциясини ўз ичига олган ихтиёрий SELECT инструкциясини ҳам бирлаштирилган сўров сифатида қараш мумкин.

SQL да сўровлар UNION оператори ёрдамида бирлаштирилади, бу оператор SELECT инструкциясидан кўп марта фойдаланиш орқали ягона натижа олиш имконини беради. Бу оператордан фойдаланиш жуда оддий. Бунинг учун ҳар бир SELECT инструкцияси орасига UNION калит сўзини қўшиш кифоя.

Мисол кўрайлик. Фараз қилайлик, бизга Иллинойс, Индиана ва Мичиган штатидаги барча мижозлар ҳақидаги маълумотларни ўз ичига олган ҳисоботни олиш керак бўлсин. Бу ҳисоботга сиз штати қандай бўлишидан қатъий назар номи Fun4All бўлган мижоз ҳақидаги маълумотларни ҳам қўшмоқчисиз. Бунда албатта бу талабларни бажариш учун WHERE конструкциясидан фойдаланса бўлади, лекин бу ҳолда UNION операторидан фойдаланиш қулайроқ. Айтиб ўтилганидек, UNION оператори SELECT инструкциясидан кўп марта фойдаланишни тақозо қилади, аввал биз уларни алоҳида кўриб чиқиб, сўнгра бирлаштирамиз:

1-сўров қуйидагича бўлади:

```
SELECT cust_name,  cust_contact,  cust_email
```

```
FROM Customers
WHERE cust_ state IN  ( 'IL', 'IN', 'MI') ;
```

Бу SELECT инструкцияси мижознинг штати IN операторида кўрсатилган Иллинойс, Индиана ва Мичиган штатларига тегишли бўлган сатрларни чиқариб беради.

Бу сўров натижаси қуйида келтирилган:

cust_name	cust_contact	cust_email
Village Toys	John Smith	sales@villagetoys.com
Fun4All	Jim Jones	jjones@fun4all.com
The Toy Store	Kim Howard	NULL

2-сўров қуйидагича бўлади:

```
SELECT cust_name,  cust_contact,  cust_email
FROM Customers
WHERE cust_name = 'Fun4All';
```

Бу SELECT инструкцияси номи Fun4All бўлган мижозга тегишли бўлган сатрларни чиқариб беради.

Бу сўров натижаси қуйида келтирилган:

cust_name	cust_contact	cust_email
Fun4All	Jim Jones	jjones@fun4all.com
Fun4All	Denise L. Stephens	dstephens@fun4all.com

Иккала сўровни бирлаштириш учун қуйидагиларни бажариш керак:

```
SELECT cust_name,  cust_contact,  cust_email
FROM Customers
WHERE cust_state IN  ('IL','IN','MI')
UNION
SELECT cust_name,  cust_contact,  cust_email
FROM Customers
WHERE cust_name = 'Fun4All';
```

Бу бирлаштирилган сўров юқорида келтирилган биринчи ва иккинчи

сўровларнинг бири-биридан UNION калит сўзи билан ажратилган SELECT инструкцияларини ўз ичига олади.

Бу бирлаштирувчи сўров натижаси қуйидагича бўлади:

cust_name	cust_contact	cust_email
Fun4All	Jim Jones	jjones@fun4all.com
Fun4All	Denise L. Stephens	dstephens@fun4all.com
Village Toys	John Smith	sales@villagetoys.com
The Toy Store	Kim Howard	NULL

SELECT инструкциялари алоҳида бажарилганда биринчи SELECT инструкцияси натижа сифатида учта сатрни, иккинчи SELECT инструкцияси эса иккита сатрни чиқариб беряпти. Бу икки инструкция UNION оператори ёрдамида бирлаштирилганда натижа сифатида бешта эмас, фақат тўртта сатр чиқариб бериляпти. Натижаларга эътибор берсак, биринчи SELECT инструкцияси натижасидаги иккинчи сатр иккинчи SELECT инструкцияси инструкцияси натижасидаги биринчи сатрда такрорланяпти.

UNION оператори натижалар тўпламидан такрорланувчи сатрларни ўчиради, бу сатр Индиана штатидан бўлган Fun4All номли мижозга тегишли бўлган сатр бўлиб, у иккала SELECT инструкцияси томонидан ҳам ҳосил қилинган эди: Fun4All Jim Jones jjones@fun4all.com. UNION оператори ёрдамида бирлаштирилган сўров натижасида такрорланувчи сатр бўлмайди ва натижа уникал сатрлардан иборат бўлади.

Лекин, хоҳишга кўра UNION операторини ўзгартириш мумкин, агар барча сатрлар, яъни такрорланувчи сатрлар ҳам киритилиши талаб қилинса, у ҳолда UNION ALL операторидан фойдаланиш мумкин. Юқоридаги сўровда UNION ўрнига UNION ALL операторидан фойдаланайлик:

```
SELECT cust_name,  cust_contact,  cust_email
FROM Customers
WHERE cust_state IN  ('IL', 'IN’, 'MI')
UNION ALL
```

```
SELECT cust_name,  cust_contact,  cust_email
FROM Customers
WHERE cust_name = 'Fun4All';
```

Бу сўров натижаси қуйидагича бўлади:

cust_name	cust_contact	cust_email
Village Toys	John Smith	sales@villagetoys.com
Fun4All	Jim Jones	jjones@fun4all.com
The Toy Store	Kim Howard	NULL
Fun4All	Jim Jones	jjones@fun4all.com
Fun4All	Denise L. Stephens	dstephens@fun4all.com

UNION ALL операторидан фойдаланилганда МББТ дубликатларни, яъни такрорланувчи сатрларни ўчирмайди, шунинг учун бу сўров натижасида бешта сатр ҳосил бўлади, улардан бири икки марта такрорланяпти. Демак, хулоса қилиб шуни айтиш мумкинки, UNION оператори натижага такрорланувчи сатрларни киритмайди, UNION ALL оператори эса аксинча уларни киритади.

Энди шу сўровни UNION операторидан эмас, балки WHERE конструкциясидан фойдаланиб яратайлик:

```
SELECT cust_name,  cust_contact,  cust_email
FROM Customers
WHERE cust_state IN  ('IL', 'IN', 'MI')
      OR cust_name = 'Fun4All';
```

UNION оператори ва WHERE конструкциясидан фойдаланиб яратилган бирлаштирувчи сўровлар натижаси бир хил бўлади, лекин бу берилган мисолда уларни таққослаш шуни кўрсатдики, UNION операторини қўллаш WHERE конструкциясига нисбатан кўпроқ ноқулай бўлиши мумкин. Агар фильтрлаш шарти мураккаб бўлса ёки бир неча жадваллардан маълумотни олишга тўғри келса, у ҳолда UNION оператори жараённи сезиларли даражада соддалаштириши мумкин.

SQL стандартида UNION оператори ёрдамида бирлаштириладиган SELECT инструкциялари сонига чекланишлар қўйилмаган, лекин сиз ишлаётган МББТ ҳужжатларига мурожаат қилиб, бундай чекланишлар бор ёки йўқлигига ишонч ҳосил қилсангиз мақсадга мувофиқ бўлади.

Энди бирлаштирилган сўров натижаларини саралашни кўрайлик. Бизга маълумки, SELECT сўрови натижалари ORDER BY конструкцияси ёрдамида сараланади. UNION оператори ёрдамида бирлаштириладиган сўровларда ORDER BY конструкцияси фақат бир марта ишлатилади ва у охирги SELECT инструкциясидан кейин келади.

Мисол. UNION оператори ёрдамида бирлаштириладиган сўров натижасини саралаш талаб қилинсин. Бу сўров қуйидагича бўлади:

```
SELECT cust_name, cust_contact, cust_email
FROM Customers
WHERE cust_state IN ('IL', 'IN', 'MI')
UNION
SELECT cust_name, cust_contact, cust_email
FROM Customers
WHERE cust_name = 'Fun4All'
ORDER BY cust_name, cust_contact;
```

Бу UNION операторида охирги SELECT инструкциясидан кейин фақат битта ORDER BY конструкцияси ишлатиляпти, лекин МББТ бу ORDER BY конструкциясини барча натижаларни саралаш учун қўллайди

Бу сўров натижаси қуйидагича бўлади:

cust_name	cust_contact	cust_email
Fun4All	Denise L. Stephens	dstephens@fun4all.com
Fun4All	Jim Jones	jjones@fun4all.com
The Toy Store	Kim Howard	NULL
Village Toys	John Smith	sales@villagetoys.com

2.8. SQL тилида маълумотларни гуруҳлаш. GROUP BY конструкцияси

Бизга маълумки, SQL тилининг агрегат функциялари статистик кўрсаткичларни олиш учун мўлжалланган. Бу функциялар устундаги сатрлар сони, қийматлар йиғиндисини, ўртача қийматни, шунингдек, энг катта ва энг кичик қийматларни ҳисоблаш имконини беради, бунда ҳисоблашлар айрим устунлар учун бажарилади.

Энди агрегат функциялар жадвалнинг барча маълумотлари ёки WHERE шартини қаноатлантирувчи маълумотлар устида бажарилиши талаб этилсин. Бунга мисол сифатида қуйидаги сўровни олайлик, яъни Products жадвалидаги идентификатори DLL01 бўлган таъминловчи етказиб берадиган маҳсулотлар сонини аниқлаш керак бўлсин.

```
SELECT COUNT (*) AS num_prods
FROM Products;
WHERE vend_id ='DLL01';
```

Натижа:

```
num_prods
---------------
      4
```

Шуни айтиш керакки, WHERE калит сўзи сатрлар бўйича фильтрлашни амалга оширади. Энди шундай савол туғилиши мумкин: ҳар бир таъминловчи нечтадан маҳсулот етказиб беради ёки қайси таъминловчилар фақат биттадан маҳсулот ёки бир неча маҳсулот етказиб беради ?

Айнан шу ҳолларда группалар(гуруҳлар) ишлатилади. Гуруҳлаш барча маълумотларни мантиқий жиҳатдан бўлиш имконини беради, бу эса ўз навбатида ҳар бир гуруҳ учун статистик ҳисобларни бажариш имкониятини беради.

Группалар SELECT инструкциясининг GROUP BY

конструкцияси ёрдамида амалга оширилади. Қуйидаги мисолда ҳар бир таъминловчи нечтадан маҳсулот етказиб беришини аниқлаш сўрови келтирилган:

```
SELECT vend_id, COUNT (*) AS num_prods
FROM Products
GROUP BY vend_id;
```

Натижа қуйидагича бўлади:

```
vend_id          num_prods
-------------    --------------
BRS01                 3
DLL01                 4
FNG01                 2
```

SELECT нинг бу инструкцияси иккита устунни ҳосил қиляпти, биринчи устун таъминловчининг идентификаторини белгиловчи vend_id устун, иккинчи устун COUNT (*) функцияси ёрдамида ҳосил қилинадиган ва ҳисобланувчи майдонларни ўз ичига олган num_prods устунидир. GROUP BY конструкцияси МББТ га маълумотларни саралаш ва уларни vend_id устуни бўйича группалаш буйруғини беради. Натижада num_prods қиймати бир марта бутун жадвал учун эмас, балки vend_id нинг ҳар бир гуруҳланган ёзуви учун бир мартадан ҳисобланади.

Натижалар шуни кўрсатяптики, таъминловчи BRS01 - учта маҳсулот, таъминловчи DLL01 – тўртта, таъминловчи FNG01 иккита маҳсулот етказиб беради. Шуни алоҳида таъкидлаш керакки, GROUP BY конструкцияси МББТ га аввал маълумотларни гуруҳлаш ва кейин эса ҳар бир гуруҳ учун ҳисоблашларни бажариш буйруғини беради (бунда ҳисоблашлар бутун жадвал учун бажарилмайди).

GROUP BY конструкциясини қўллашдан аввал уни ишлатишни баъзи қоидалари билан танишайлик:

-GROUP BY конструкциясидаги устунлар сони ихтиёрий бўлиши мумкин;

-кўпгина SQL версияларида узунлиги ўзгарувчи маълумотга эга бўлган устунларни ишлатиб бўлмайди;

-агар группалаш лозим бўлган устун NULL қийматли сатрга эга бўлса, у ҳолда у алоҳида гуруҳланади, агар бундай сатрлар бир нечта бўлса улар биргаликда гуруҳланади;

-GROUP BY конструкцияси WHERE конструкциясидан кейин ва ORDER BY конструкциясидан олдин туриши керак.

2.9. Группалар бўйича фильтрлаш. HAVING конструкцияси

Биз юқорида WHERE конструкцияси ёрдамида сатрларни фильтрлаш билан танишдик, энди группалар бўйича фильтрлаш қандай амалга оширилишини кўрайлик.

SQL тили GROUP BY конструкцияси ёрдамида нафақат маълумотларни группалаш, балки уларни фильтрлаш имконини ҳам беради, яъни қайси группалар сўров натижасига киритилиши керак, қайсилари киритилмаслиги кераклигини кўрсатиб беради. Масалан, бизга камида иккита буюртма берган мижозлар рўйҳати керак бўлсин. Бундай маълумотларни олиш учун сатрларни эмас, балки группани фильтрлаш зарур бўлади.

Бу ҳолда WHERE калит сўзидан фойдаланиб бўлмайди, чунки WHERE танлаш шарти группаларга эмас, балки сатрларга тегишли, яъни сатрларни фильтрлайди. Қисқа қилиб айтганда, WHERE конструкцияси группа нималигини “билмайди”. Бундай ҳолларда HAVING калит сўзидан фойдаланилади. Уларнинг фарқи шундаки, WHERE сатрларни фильтрласа, HAVING группаларни фильтрлайди. Синтаксиси жиҳатдан улар бир-бирига ўхшаш, HAVING конструкцияси WHERE конструкциясидаги барча операторларни қўллаб-қувватлайди.

Группаларни фильтрлашга доир мисол кўрайлик. Масалан, бизга икки ва ундан ортиқ буюртма берган мижозлар рўйҳати керак бўлсин. Бундай сўров қуйидагича амалга оширилади:

```
SELECT cust_id, COUNT (*) AS orders
FROM Orders
GROUP BY cust_id
HAVING COUNT (*) >=2;
```

Натижа қуйидагича бўлади:

```
  cust_id              orders
---------------      --------------
1000000001              2
```

Мисолдан кўриниб турибдики, бунда WHERE конструкцияси ишламайди, чунки фильтрлаш танлаган сатрлар эмас, балки гуруҳлардаги якуний қийматларга асосланган.

Изоҳ сифатида шуни айтиб ўтиш мумкинки, WHERE конструкцияси маълумотлар группага ажратилмасдан аввал сатрларни фильтрлайди, HAVING эса группалаш амалга оширилгандан кейин группаларни фильтрлайди.

Энди савол туғилиши мумкин: SELECT инструкциясида бир вақтнинг ўзида WHERE ва HAVING конструкцияларидан фойдаланишга эҳтиёж туғиладими? Албатта туғилади. Фараз қилайлик, юқоридаги инструкциядаги фильтрлашни такомиллаштириб, охирги 12 ойда иккита ва ундан ортиқ буюртма берган мижозлар рўйҳатини аниқлаш керак бўлсин.

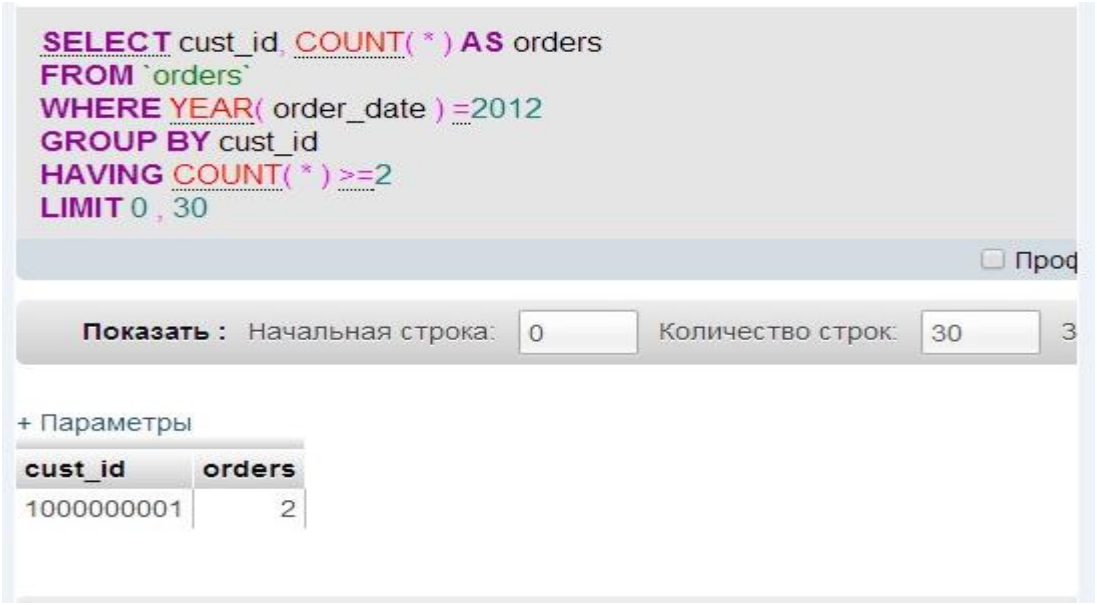
Бундай сўров қуйидагича амалга оширилади:

```
SELECT cust_id, COUNT (*) AS orders
FROM Orders
WHERE YEAR(order_date)=2012
GROUP BY cust_id
HAVING COUNT (*) >=2;
```

Натижа қуйидагича бўлади:

```
  cust_id              orders
---------------      --------------
1000000001               2
```

Бу сўров ва унинг натижаси қуйидаги расмда келтирилган:



Яна бир мисол кўрайлик. Нархи 1 доллардан кам бўлмаган, иккита ва ундан ортиқ маҳсулот етказиб берадиган таъминотчилар рўйҳатини аниқлаш керак бўлсин.

Бундай сўров қуйидагича амалга оширилади:

```
SELECT vend_id, COUNT (*) AS num_prods
FROM Products
WHERE prod_price >=1
GROUP BY vend_id
HAVING COUNT (*) >=2;
```

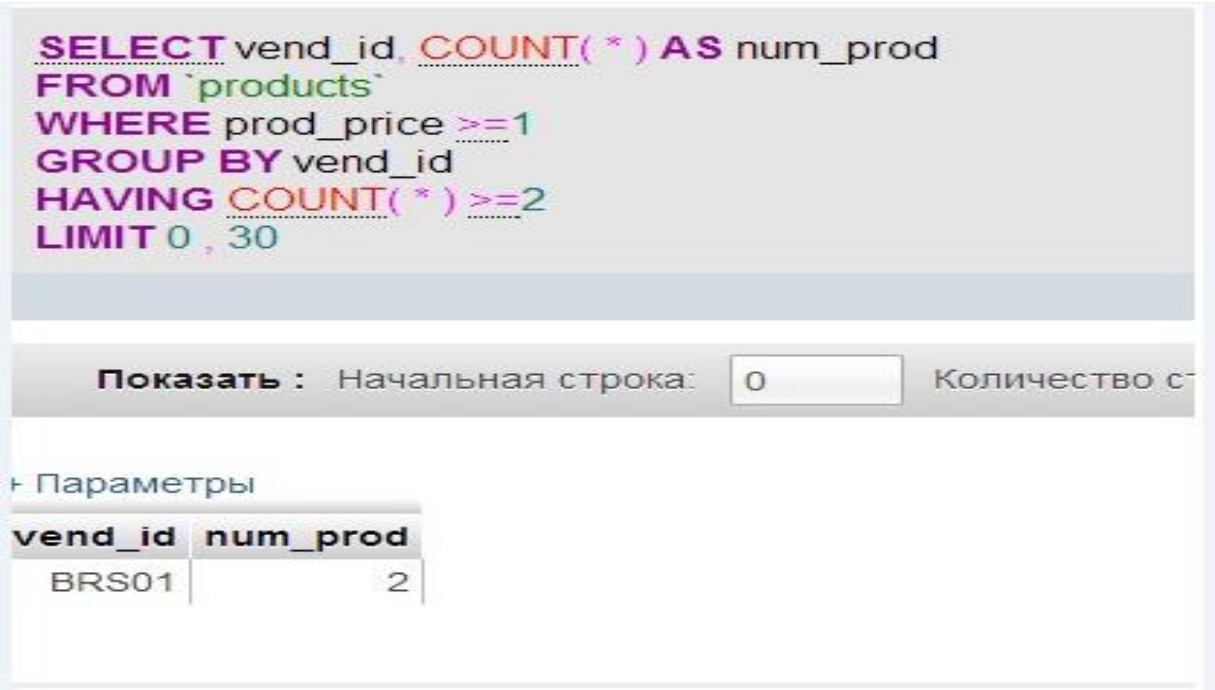
Натижа қуйидагича бўлади:

```
  vend_id              num_prods
---------------      --------------
  BBRS01                   2
```

Бу сўровда WHERE танлаш шарти prod_price устунидаги 1 дан кам бўлмаган қийматга эга бўлган сатрларни фильтрлайди. Сўнгра

маълумотлар vend_id устуни бўйича группаланади. Шундан сўнг HAVING конструкцияси иккитадан кам бўлмаган маҳсулот етказиб берадиган таъминотчилар группасини фильтрлайди.

Бу сўров ва унинг натижаси қуйидаги расмда келтирилган:



SELECT инструкциясидаги барча калит сўзлар маълум тартибда жойлаштирилиши керак. Уларнинг жойлашиш кетма-кетлиги қуйидаги жадвалда келтирилган :

Калит сўзлар	**Вазифаси**
SELECT	Устунлар ёки ифодаларни ҳосил қилиш
FROM	Натижа олинадиган жадвал
[WHERE]	Сатрларни фильтрлаш
[GROUP BY]	Группаларни ҳосил қилиш
[HAVING]	Группаларни фильтрлаш
[ORDER BY]	Натижаларни саралаш тартиби

Квадрат қавсларда қатнашиши шарт бўлмаган калит сўзлар кўрсатилган. Улар жумласига WHERE, GROUP BY, HAVING ва ORDER BY калит сўзлари киради.

2.10. ORDER BY ва GROUP BY конструкцияларидан биргаликда фойдаланиш

SELECT инструкцияси – бу SQL сўровлардир. ORDER BY ва GROUP BY конструкцияларидан биргаликда фойдаланишни намойиш қилишдан аввал маълумотларни гуруҳлашга доир мисол кўрайлик.

Қуйида келтирилган SELECT инструкцияси **OrderItems** жадвалидан учта ва ундан ортиқ маҳсулотни ўз ичига олган барча буюртмаларнинг буюртма номери ва маҳсулот сонини чиқариб беради:

```
SELECT order_num, COUNT (*) AS items
FROM OrderItems
GROUP BY order_num
HAVING COUNT (*) >=3;
```

Натижа қуйидагича бўлади:

order_num	items
20006	3
20007	5
20008	5
20009	3

Энди натижани буюртма қилинган маҳсулот сони бўйича (ўсиш тартибида) саралаш керак бўлсин. Бунинг учун юқоридаги сўровга ORDER BY конструкциясини қўшиш кифоя:

```
SELECT order_num, COUNT (*) AS items
FROM OrderItems
GROUP BY order_num
HAVING COUNT (*) >=3;
ORDER BY items, order_num
```

Изоҳ сифатида шуни айтиш мумкинки, Microsoft Access псевдоним (тахаллус)лар бўйича саралаш имконини бермайди, шунинг учун юқоридаги сўровни Access учун қўллаб бўлмайди. Муаммони ечишнинг йўли бу охирги сатрдаги **ORDER BY** конструкциясидаги **items** устунини ҳисобловчи ифода ёки майдон номи билан алмаштиришдан иборат:

ORDER BY COUNT (*), order_num ёки

ORDER BY 1, order_num

Натижа қуйидагича бўлади:

order_num	items
20006	3
20009	3
20007	5
20008	5

Бу сўровда GROUP BY конструкцияси маълумотларни буюртма номери order_num бўйича группалайди. COUNT (*) функцияси эса ҳар бир буюртмадаги маҳсулотлар сонини аниқлаб беради. HAVING конструкцияси учта ва ундан ортиқ маҳсулотни ўз ичига олган буюртмаларни фильтрлаб беради ва ниҳоят натижа ORDER BY конструкцияси ёрдамида сараланади.

2.11. SELECT инструкциясининг кенгайтирилган имкониятлари. UPDATE CASE буйруғи

Бизга маълумки, кўпгина тилларнинг асосини бошқариш операторлари ташкил этади, улар бошқа операторлар мажмуини бажарилишини назорат қилади. Улар SQL тилида ҳам учрайди ва сўровларни фильтрлашда ҳамда оптималлашда қўлланилади.

SQL да CASE ифодаси мавжуд бўлиб, у if/then шартли операторни ишлатишнинг мантиқий ифодаси ҳисобланади.[8] Бирор

[8] Справка по SQL(DML): Операторы UPDATE и DELETE. http://www.sql-ex.ru/help/select12.php?Lang=0

устуннинг қийматига кўра маълумотларни ўзгартириш, яъни янги устунни тўлдириш керак бўлса CASE ифодасидан фойдаланилади.

Айтайлик, Laptop портатив компьютерлар жадвалидаги ҳажми 20 ГБ ли қаттиқ дискларни хотираси 128 Мб дан кичик бўлган портатив компьютер блокнотларига, 40 ГБ ли қаттиқ дискларни эса қолган портатив компьютер блокнотларига ўрнатиш керак бўлсин. Бу сўров қуйидагича бўлади:

```
UPDATE Laptop
  SET hd = CASE WHEN ram<128 THEN 20 ELSE 40 END
```

UPDATE операторининг синтаксисини эслатиб ўтамиз:

```
UPDATE < жадвал номи >
   SET{устун номи={устун қийматини ҳисоблаш учун ифода
    | NULL
    | DEFAULT},...}
    [ {WHERE <танлаш шарти>}];
```

Бизнинг мисолимизда бу синтаксис қуйидагича бўлади:

```
UPDATE < жадвал номи >
   SET{устун номи={устун қийматини ҳисоблаш учун ифода}
```

Қолган NULL, DEFAULT, WHERE каби операторлар қатнашиши шарт бўлмаган операторлардир.

Демак, бу синтаксисга кўра устун номи бу hd қаттиқ дисклар устуни бўлиб, унинг қийматини ҳисоблаш учун CASE WHEN ram<128 THEN 20 ELSE 40 END ифодасидан фойдаланиляпти. Бунда янги устун тўлдирилади ва ҳосил бўлади.

Энди юқоридаги сўровни бироз ўзгартирайлик. Laptop жадвалидаги ҳажми 40 ГБ ли қаттиқ дискларни хотираси 128 Мб дан катта бўлган портатив компьютер блокнотларига, 20 ГБ ли қаттиқ дискларни эса қолган портатив компьютер блокнотларига ўрнатиш керак бўлсин. Бу сўров қуйидагича бўлади:

```
UPDATE Laptop
SET hd = CASE WHEN ram>128 THEN 40 ELSE 20 END
```

2.12. SQL – сўров ичидаги сўров

Шуни таъкидлаш лозимки, юқорида келтирилган барча сўровлар оддий сўровлар бўлиб, уларда маълумотлар маълум жадваллардан алоҳида олинган инструкциялар ёрдамида ҳосил қилинди. SQL тилида шунингдек **сўров ости**, яъни сўров ичидаги сўров ёки ичма-ич жойлашган сўровларни ҳам яратиш мумкин. Айрим ҳолларда у пастки сўров деб ҳам юритилади. Сўров ости бу бир неча сўровларни битта ягона инструкцияга бирлаштириш демакдир.

Нима учун бундай сўровларни ишлатишга зарурат туғилишини тушунтириш учун мисолларга мурожаат қиламиз. Биз фойдаланаётган маълумотлар базаси жадваллари реляцион жадваллардир(иловага қаранг). Базада буюртмалар иккита жадвалда сақланади. Orders жадвалида буюртма номери, мижоз идентификатори ва буюртма муддати кўрсатилган. Буюртманинг айрим элементлари эса OrderItems жадвалида сақланади.

Orders жадвалида мижозлар ҳақидаги маълумотлар сақланмайди, унда фақат мижознинг идентификатори сақланади. Мижозлар ҳақидаги маълумотлар Customers жадвалида сақланади.

Энди фараз қилайлик, бизга идентификатори RGAN01 бўлган маҳсулотни буюртма қилган барча мижозлар рўйҳатини ҳосил қилиш керак бўлсин. Бунинг учун қуйидагиларни бажариш зарур:

1) идентификатори RGAN01 бўлган маҳсулотни ўз ичига олган барча буюртмалар номерини чиқариш;

2) олдинги қадамда идентификатори RGAN01 бўлган маҳсулотни буюртма қилган барча мижозларнинг идентификаторини олиш;

3) олдинги қадамда идентификатори олинган барча мижозлар ҳақидаги маълумотни чиқариш.

Санаб ўтилган қадамларнинг ҳар бирини алоҳида сўров кўринишида бажариш мумкин. Бунда бир неча SELECT инструкциясидан фойдаланилади, яъни битта SELECT инструкциясидан олинган натижалардан кейинги SELECT инструкциясининг WHERE конструкциясини тўлдиришда фойдаланилади. Лекин, бу учала сўровни битта ягона инструкцияга бирлаштириш мумкин.

Биринчи SELECT инструкцияси prod_id устунида идентификатори RGAN01 бўлган маҳсулотни ўз ичига олган order_num устунидаги барча буюртмалар номерини чиқариб беради:

```
SELECT order_num
FROM OrderItems
WHERE prod_id='RGAN01';
```

Натижа бу маҳсулотни ўз ичига олган буюртмалар номери иккита эканлигини кўрсатади:

```
order_num
----------
  20007
  20008
```

Кейинги қадам 20007 ва 20008 буюртмалар билан боғлиқ бўлган мижозларнинг идентификаторини олишдан иборат. IN операторидан фойдаланиб қуйидаги SELECT инструкциясини ҳосил қиламиз:

```
SELECT cust_id
FROM Orders
WHERE order_num IN (20007,20008);
```

Натижа қуйидагича бўлади:

```
cust_id
----------
1000000004
1000000005
```

Энди биринчи сўровни сўров остига айлантириб, бу иккала сўровни бирлаштирамиз:

```
SELECT cust_id
FROM Orders
WHERE order_num IN   (SELECT order_num
                      FROM OrderItems
                      WHERE prod_id='RGAN01');
```

Натижа яна юқоридаги натижа билан бир хил бўлади:

```
cust_id
----------
1000000004
1000000005
```

Сўров ичидаги сўровларда аввал ички сўров бажарилади, юқоридаги мисолга эътибор берсак, МББТ икки амални бажаради.

У аввал қуйидаги сўров остини (ички сўровни) бажаради:

```
SELECT order_num FROM OrderItems WHERE prod_id='RGAN01'
```

Натижада иккита 20007 ва 20008 буюртма номерлари чиқариб берилади. Кейин бу икки қиймат ташқи сўровдаги WHERE конструкциясидаги IN операторига узатилади.

Энди ташқи сўров қуйидаги кўринишда бўлади:

```
SELECT cust_id FROM Orders WHERE order_num IN ( 20007,20008)
```

Энди 3-қадамда 2-қадамда идентификатори олинган барча мижозлар ҳақидаги маълумотни чиқаришдан иборат:

```
SELECT cust_name, cust_contact
FROM Customers
WHERE cust_id IN ('1000000004', '1000000005');
```

Бунда ҳам '1000000004' ва '1000000005' идентификаторларни кўрсатмасдан, уларни ўрнига WHERE конструкциясини ташқи сўровга айлантириш мумкин.

Энди учала сўровни бирлашмаси бўлган ягона сўров қуйидаги кўринишда бўлади:

```
SELECT cust_name, cust_contact
FROM Customers
WHERE cust_id IN (SELECT cust_id
                  FROM Orders
                  WHERE order_num IN (SELECT order_num
                                      FROM OrderItems
                                      WHERE prod_id=
                                          'RGAN01'));
```

Натижа эса қуйидагича бўлади:

cust_name	cust_contact
Fun4All	Denise L. Stephans
The Toy Store	Kim Howard

Бундай сўровни амалга ошириш учун МББТ учта SELECT инструкциясини ишлаб чиқиши керак эди. Ягона бирлаштирилган сўровлар орқали эса бу масалани тез ҳал қилиш мумкин.

Юқоридаги сўров ичидаги сўровда аввал қуйи даражадаги сўров, сўнгра ўрта даражадаги ва кейин юқори даражадаги сўров бажарилади. Натижани юқори даражадаги сўров чиқариб беради.

Бундай сўров остиларидан фойдаланиб SQL нинг кучли инструкцияларини яратиш мумкин, бунда бўйсунувчи сўровлар сони чегараланмаган, лекин, шуни айтиш мумкинки, сўров ости даражалари ошган сари компьютернинг амаллар бажариш тезлиги камайишини сезиш мумкин.

2.13. Сўров остидан ҳисобланувчи майдон сифатида фойдаланиш

Сўров остидан фойдаланишнинг яна бир усули ҳисобланувчи майдонларни яратишдан иборат. Фараз қилайлик, Customers (мижозлар) жадвалидаги ҳар бир мижоз томонидан қилинган буюртмаларнинг умумий сонини аниқлаш талаб этилсин. Буюртмалар Orders жадвалида мижозларнинг мос идентификатори билан биргаликда сақланади.

Бу сўровни амалга ошириш учун қуйидагиларни бажариш керак:

1) Customers жадвалидан мижозлар рўйҳатини ҳосил қилиш;
2) Orders жадвалидан ҳар бир танланган мижоз томонидан қилинган буюртмалар сонини ҳисоблаш.

Аввалги бўлимлардан бизга маълумки, жадвалдаги сатрлар сонини ҳисоблаш SELECT COUNT (*) инструкцияси ёрдамида бажарилади. Маълум мижознинг идентификаторини фильтрлаш учун эса WHERE конструкциясидан фойдаланиб, фақат шу мижознинг буюртмалари сонини ҳисоблаш мумкин. Масалан, қуйидаги сўров ёрдамида идентификатори 1000000001 бўлган мижоз томонидан қилинган буюртмалар сонини ҳисоблаш мумкин:

```
SELECT COUNT(*) AS orders
FROM Orders
WHERE cust_ id = '1000000001';
```

COUNT(*) функцияси орқали ҳар бир мижоз учун якуний ахборотни олиш учун COUNT(*) ифодадан сўров ости сифатида фойдаланамиз. Қуйидаги мисолни кўрайлик:

```
SELECT cust_name,
       cust_state,
       (SELECT COUNT(*)
       FROM Orders
      WHERE Orders.cust_id = Customers.cust_id)  AS orders
FROM Customers
```

ORDER BY cust_ name;

Бу SELECT инструкцияси Customers жадвалидан ҳар бир мижоз учун учта: cust_name, cust_state ва orders каби устунларни ҳосил қилади. Orders майдони ҳисобланувчи майдон бўлиб, у юмалоқ қавс ичидаги сўров остининг бажарилиши натижасида шакллантирилади.

Сўров ости ҳар бир танланган мижоз учун бир мартадан бажарилади. Келтирилган мисолда сўров ости беш марта бажариляпти, чунки бешта мижознинг номи ҳосил қилинган. Сўров остидаги WHERE конструкцияси аввалги WHERE конструкцияларидан бироз фарқ қилади, чунки унда устунларнинг тўлиқ номларидан фойдаланилади. WHERE конструкцияси МББТ дан Orders жадвалидаги cust_id қийматини Customers жадвалидан ҳосил қилинаётган қиймат билан солиштиришни талаб қилади:

WHERE Orders.cust_id = Customers.cust_id

Бу синтаксисда жадвал номи билан устун номи нуқта билан ажратилган, устун номларида ноаниқлик пайдо бўлган ҳолларда ҳар сафар бу синтаксис қўлланилиши керак. Бизнинг мисолимизда иккита cust_id устуни бор: биттаси Customers жадвалида, иккинчиси Orders жадвалида.

Сўров натижаси қуйидагича бўлади:

cust_ name	cust_ state	orders
Fun4All	IN	1
Fun4All	AZ	1
Kids Place	OH	0
The Toy Store	IL	1
Village Toys	MI	2

Қуйидаги SELECT инструкциясида устунлар номи тўлиқ берилмаган ва МББТ Orders жадвалидаги cust_id майдонини ўзини ўзи билан солиштириляпти деб ҳисоблайди:

SELECT COUNT(*) FROM Orders WHERE cust_id = cust_id

Қуйидаги сўров Orders жадвалидаги умумий буюртмалар

сонини ҳисоблаб беради:

```
SELECT cust_name,
       cust_state,
      (SELECT COUNT(*)
       FROM Orders
      WHERE cust_id = cust_id) AS orders
FROM Customers
ORDER BY cust_name;
```

Бу ҳолда сўров натижаси қуйидагича бўлади:

cust_ name	cust_ state	orders
Fun4All	IN	5
Fun4All	AZ	5
Kids Place	OH	5
The Toy Store	IL	5
Village Toys	MI	5

Бу эса бизга кераксиз бўлган натижа, бизга Customers жадвалидаги ҳар бир мижоз қилган буюртмаларнинг умумий сонини аниқлаш керак эди.

Сўров остилар юқорида келтирилган кўринишдаги SELECT инструкцияларини яратишда жуда фойдали ҳисобланади, лекин шуни назарда тутиш керакки, унда устунлар номи тўғри кўрсатилган бўлиши керак. Акс ҳолда МББТ сизнинг хоҳишингизни тўғри идрок эта олмай нотўғри натижаларни бериши мумкин. Юқоридаги мисоллардан устунлар номини тўлиқ ва аниқ кўрсатиш муҳимлигига ишонч ҳосил қилиш мумкин.

Баъзи ҳолларда устун номларидаги ноаниқлик хатолик тўғрисидаги хабарнинг пайдо бўлишига олиб келиши мумкин. Сўров остилар ҳар доим ҳам оптимал ечим бермайди. Шунинг учун SELECT инструкциясида бир неча жадваллар иштирок этса, устунлар номини тўлиқ ва аниқ кўрсатиш яхши амалиёт ҳисобланади ва ҳар хил ноаниқликларнинг олдини олиш имконини беради.

III-боб. SQL тилининг мижоз-сервер технологиясида қўлланилиши

3.1. Маълумотларни тақсимланган қайта ишлаш

Жамоа бўлиб фойдаланиладиган маълумотларга бўлган талаб охирги вақтда ортиб бормоқда. Бу эса маълумотларни тақсимланган қайта ишлаш тизимларига бўлган эътиборни кучайишига олиб келади.

Маълумотларни тақсимланган қайта ишлаш деганда иловаларни бир неча ҳудудий тақсимланган компьютерларда қайта ишлаш тушунилади, яъни тақсимланган маълумотлар базаси тушунилади (3-расм).

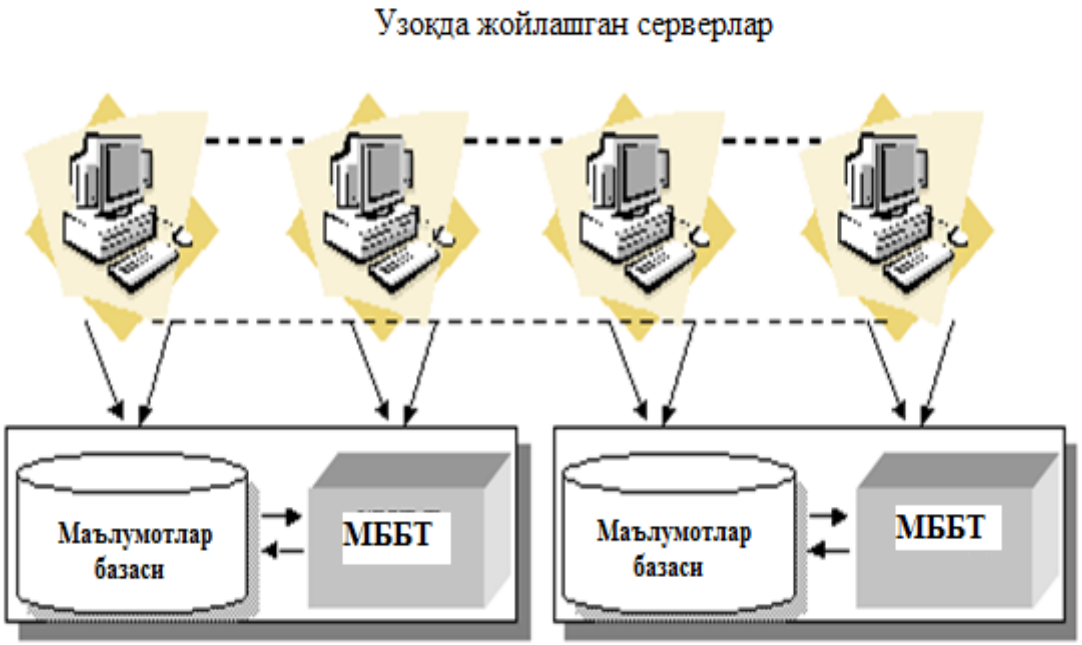
Тақсимланган маълумотлар базаси деганда кўпгина муаллифлар барча маълумотлар фақат узоқдаги серверда жойлашган бўлиб, бир-биридан масофада жойлашган мижоз компьютерлар бу базадан фойдаланадилар деб тушунадилар, бу нотўғри.

Тўғрироғи, тақсимланган маълумотлар базаси бир неча компьютерларда жойлашган бўлади, яъни, файлнинг бир қисми битта компьютерда, яна бир қисми иккинчи компьютерда ва қолган қисмлари эса бошқа компьютерларда жойлашган бўлади, бу компьютерлар бир-биридан масофада жойлашган компьютер тармоғини ташкил қилади.

Маълумотлар базасининг ҳар бир қисми МББТ бошқаруви остида ишлайди, у МБ фрагментларига кириш имконини беради.

Фойдаланувчиларнинг тақсимланган маълумотлар базаси билан ўзаро алоқалари **локал ва глобал иловалар** орқали амалга оширилади. Локал иловалар фойдаланувчига ўзининг локал маълумотлари билан ишлаш имконини беради ва бунда улар бошқа фрагментларга кирмайдилар.

Глобал иловалар эса фойдаланувчига тармоқнинг бошқа компьютерида жойлашган маълумотлар базасининг фрагментлари билан ишлаш имконини беради. Бунга мисол қилиб, Интернет тармоғини олиш мумкин: дунё бўйича турли компьютерларда маълумотлар киритилади ва сақланади, ихтиёрий фойдаланувчи эса улар қаерда жойлашишидан қатъий назар уларга кириш ва фойдаланиш имконига эга.



3- расм. Тақсимланган маълумотлар базаси

Маълумотларни тақсимланган қайта ишлаш технологияси иккита тамойилга асосланади. Биринчи тамойил “файл-сервер”, иккинчиси эса “мижоз-сервер” деб номланади.

Маълумотларни тақсимланган қайта ишлашнинг биринчи тамойили қуйидагича амалга оширилади (4-расм).

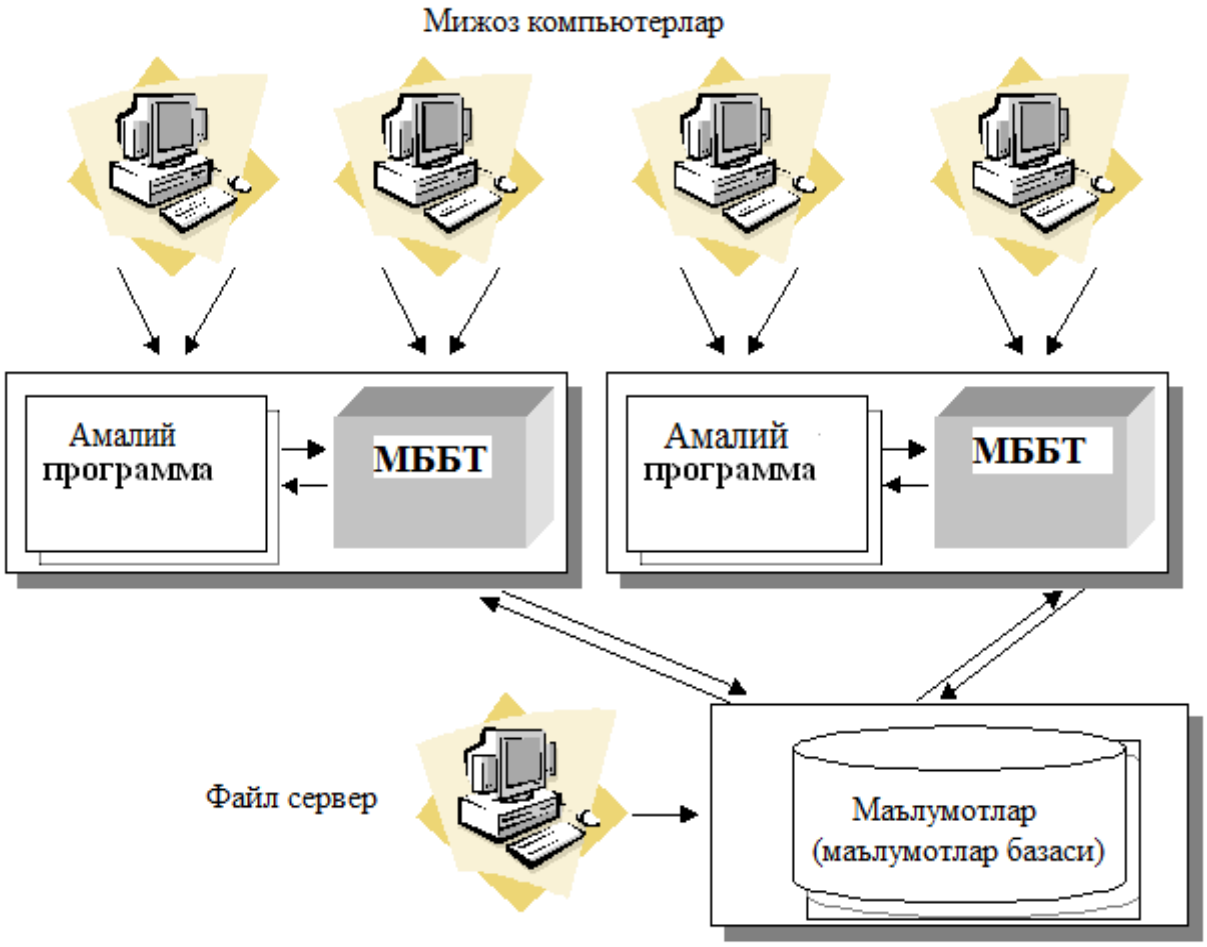
Тармоқда файл-сервер деб номланувчи бош компьютер бўлади.

Сервер–бу тармоққа мўлжалланган МББТни, маълумотлар бошқарувини маълумотлар базаси атамасида амалга оширувчи ва файллар сервери ёки файл-сервер (File Server) деб номланувчи қисмининг иш фаолиятини амалга оширувчи машина.

Сервер ахборот (файллар, маълумотлар базалари) ва қурилма ресурсларидан (принтерлар, модемлар) биргаликда фойдаланиш имконини беради.

Фойдаланувчи билан сервернинг ўзаро алоқасини таъминловчи тармоқ ОТ икки қисмдан ташкил топади: биринчи (асосий) қисми файл-серверда, иккинчиси (қобиқ) тармоқ компьютерларида (ишчи станциялар) ўрнатилади. Қобиқ, иш станциялари билан сервер орасидаги ўзаро алоқани (инфақат

маълумотларни сақлаш жойи сифатида ишлатилади, уларни қайта ишлаш эса фойдаланувчи компьютерида (ишчи станция) амалга оширилади.



4-расм. Файл-сервер архитектураси

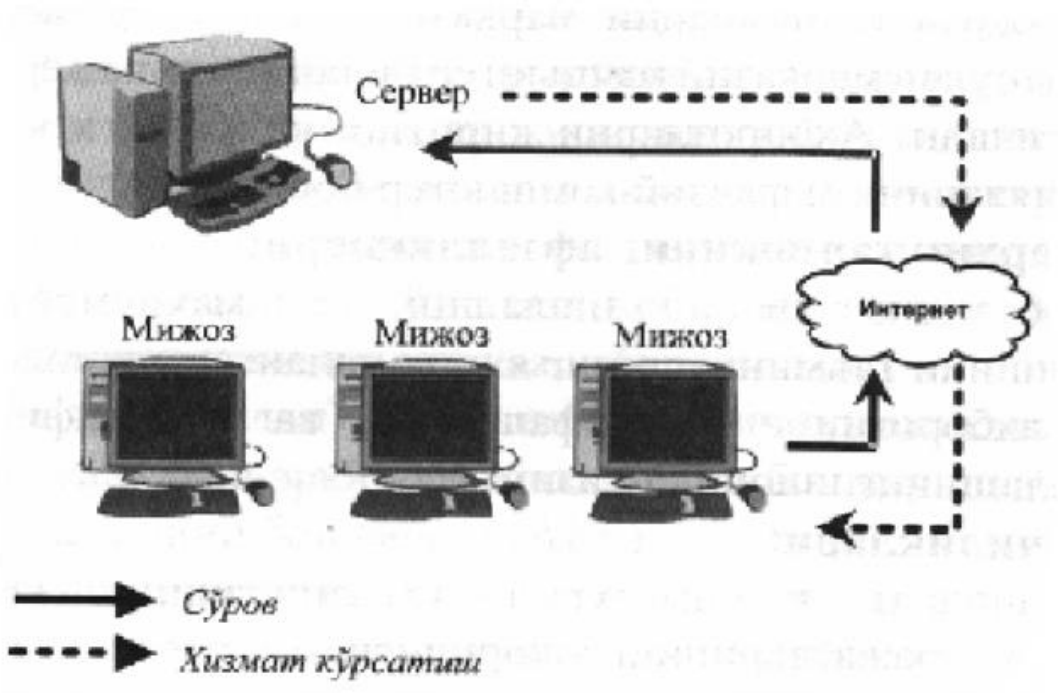
Файл-сервер архитектурасида ишлайдиган МББТ лар сирасига dBase и Microsoft Accessларни киритиш мумкин.

3.2. Мижоз-сервер технологияси

Маълумотларни тақсимланган қайта ишлашнинг иккинчи тамойили мижоз-сервер технологиясига асосланади. Шуни таъкидлаш лозимки, Интернетда узоқлашган компьютерлар билан ишлаш учун мижоз-сервер технологияси қўлланилади. Бунда фойдаланувчи бевосита ишлаётган компьютер (ишчи станция) **мижоз,** асосий маълумотлар ва ресурслар жойлаштан узоқлашган компьютер эса **сервер** деб қаралади.

Мижоз-сервер тизимининг ишлаш тамойили 5-расмда келтирилган. Бу технологияга таяниб, Интернет ресурсларига бемалол кириб, улардан

фойдаланиш имконияти пайдо бўлди. Бундай технологияни қўллаш жуда оддий. Керак бўлган маълумот ёки ресурсга кириш учун мижоз дастур ишга туширилади ва у керакли маълумот ва ресурсларни аниқлаштиради. Сўнгра бу дастур компьютер тармоғи орқали ресурс ва маълумотларни бошқарувчи сервер дастур билан боғланади.



5-расм. Мижоз-сервер тизимининг ишлаш тамойили

Мижоз ва сервер орасидаги мулоқотни **қайдномалар** (протоколлар) амалга оширади. Мижоз дастур мижоз ва сервер учун бир хил бўлган амалий дастур қайдномасига ўтказади ва уни узатишни таъминловчи қайдномалар орқали серверга узатади. Сервер эса мижоз сўровини қабул қилиб, мос қайднома орқали тегишли маълумот ва ресурсларни мос қайдномаси асосида мижоз компьютерга жўнатади.

WWW билан боғлиқ бўлган масалаларда ҳам кўпинча иккита сўз: мижоз ва сервер кўп ишлатилади. Мижоз-сервер технологияси WWW да ҳам кенг фойданилади. Сервер дастури Интернетнинг ҳар бир хост компьютерларидан олинган ҳужжатларни бошқариш учун хизмат қилади.

WWW серверлари Интернет хост компьютерларидан (узоқдаги компьютер) олинган WWW ҳужжатларига кириш имконини беради.

Мижоз дастури WWW ҳужжатларини кўриш учун, сервер дастури эса Интернетнинг ҳар бир хост компьютерларидан олинган ҳужжатларни бошқариш учун хизмат қилади. WWW мижозлари унда ишлаш учун интерфейсдан фойдаланади, яъни талабномалар юборади, маълумотлар қабул қилади ва ҳужжатларни қараб чиқади.

Мижоз-сервер технологияси турли платформаларда ишлайдиган амалиёт тизимларда ҳам кенг қўлланилиб келмоқда.

Мижоз-сервер ҳисоблаш моделининг муҳим хусусиятларидан бири бу мижоз ва сервер ўртасида амалий масалаларни тақсимлашдир. Мижоз учун ҳам, сервер учун ҳам асосий дастурий таъминот операцион тизимдир. Мижоз ва серверниг аппарат платформаси ва операцион тизимлари бир-биридан фарқ қилиши мумкин.

Агар сервер ва мижоз бир хил коммуникация протоколидан фойдаланса ва бир хил иловаларни қўллаб-қувватласа, у ҳолда бу фарқнинг ҳеч қандай аҳамияти бўлмайди.

Мижоз ва серверниг ўзаро мулоқоти коммуникацияли дастурий таъминот орқали амалга оширилади. Бундай дастурий таъминотга мисол қилиб TCP/IP протоколини, OSI протоколини ҳамда SNA каби турли фирма архитектураларини келтириш мумкин.

“Мижоз-сервер” иловаларининг умумий тузилишига кўра бажариладиган ишлар мижоз ва сервер ўртасида турлича тақсимланиши мумкин, унинг тақсимланиши маълумотлар базасидаги ахборотнинг табиати ва кўринишига боғлиқ.

Мижоз ва сервер ўртасида иловаларни тақсимланишни реляцион маълумотлар базаси мисолида кўриб чиқамиз. Бу муҳитда сервер маълумотлар базаси сервери бўлиб ҳисобланади. Мижоз ва сервер ўртасидаги ўзаро алоқа **транзакция** кўринишида амалга оширилади, бунда мижоз серверга сўров жўнатади ва ундан жавоб олади. Транзакция – бу маълумотларни манипуляция қилиш операторларининг кетма-кетлиги бўлиб, у МБ ни бир ҳолатдан иккинчи ҳолатга ўтказади.

Сервер маълумотлар базасини бошқаришга жавоб беради. Мижоз машиналарида маълумотлар базасидан фойдаланиш имконини берувчи турли иловалар жойлашган. Иловалар сервер фойдаланувчи талабини таҳлил қилади ва МБ сўровларни ҳосил қилади. Мижоз ва серверни махсус дастурий таъминот боғлайди, унга кўра мижоз сўровларни бажаради ва маълумотлар базасига кириш имконига эга бўлади. Бунинг учун махсус SQL сўровлар тили қўлланилади, яъни тармоқ орқали иловалар серверидан МБ серверига фақат сўров матни жўнатилади.

Мижоз–сервер технологиясини қўллаш тармоққа бирлаштирилган компютерларга асосланади. Мижоз–сервер архитектурасида компютерлардан бири махсус бошқарув функцияларини бажаради, яъни тармоқ сервери бўлади. Мизож сервер технологияси бўйича ишлайдиган севернинг ўзини илова – сервер деб аталади.

Мижоз – сервер архитектураси функцияларни фойдаланувчи дастури (мижоз деб аталувчи) ва сервер функцияларига ажратади. Мижоз–дастур МБ сақланаётган серверга SQL (Structured Query Language) таркибланган сўровлар тилидаги сўровни жўнатади. SQL реляцион МБ ларнинг халқаро стандарти ҳисобланади. Масофадаги сервер сўровни қабул қилади ва фаол бўлган SQL-серверга беради.

SQL-сервер – бу масофадаги МБни бошқарувчи махсус дастур. SQL-сервер сўровни талқин қилади, бажаради, сўров натижаларини расмийлаштиради ва мижоз –дастурига узатади. Бу жараёнда мижоз компютерининг ресурслари сўровни бажаришда иштирок этмайди. Мижоз компютер фақат сервер МБ га сўров жўнатади ва натижани қабул қилади. Кейин натижани зарур шаклда талқин қилади ва фойдаланувчига тасвирлаб беради. Мижоз–дастурга сўровни бажариш натижаси жўнатилади, яъни тармоқ орқали фақат мижозга зарур бўлган маълумотлар жўнатилади. Натижада тармоқ юкламаси пасаяди. Қолаверса, сўров МБ сақланаётган серверда бажарилади ва катта ҳажмли маълумотлар пакетини тармоқ орқали жўнатиш зарур бўлмайди.

Бундан ташқари, SQL-сервер агар имкони бўлса минимал вақт ва сарф-харажатларда бажарилиши учун сўровни оптималлаштиради. Буларнинг барчаси тизимни тезлигини оширади ва сўров натижасини кутиш вақтини қисқартиради.

Сервер томонидан сўровлар бажарилганда маълумотларни хавфсизлик даражаси анча юқори бўлади. Чунки маълумотларни бутунлик қоидаси сервердаги маълумотлар базасида аниқланади ва ушбу маълумотлар базасидан фойдаланувчи барча дастурлар учун ягона ҳисобланади. Шу тарзда бутунликни қўллаб-қувватлашда қарама-қарши қоидаларнинг вужудга келишишининг олди олинади.

SQL–серверлар томонидан қўллаб-қувватланадиган ишончли транзакциялар аппарати битта маълумотни бир вақтда турли фойдаланувчилар томонидан ўзгартирилишига йўл қўймайди ва МБ да авариялы тугаган ўзгаришларни бекор қилиш имконияти мавжуд.

Мижоз – сервер архитектураси қуйидагича қурилган:

• маълумотлар базаси файллар тўплами шаклида махсус компьютер, яъни тармоқ серверининг қаттиқ дискида жойлашади;

• МББТ ҳам тармоқ серверида жойлашади;

• мижоз компьютерларидан иборат бўлган локал тармоқ мавжуд бўлиб, ҳар бир компьютерда МБ билан ишловчи мижоз–дастури ўрнатилган;

• ҳар бир мижоз компьютерда фойдаланувчилар дастурни ишга тушириш имконига эга. Дастурнинг фойдаланувчи интерфейси ёрдамида фойдаланувчилар маълумотларни танлаш/янгилаш учун сервердаги МБ га мурожаат қилади. Мурожаат қилиш учун махсус SQL сўровлар тили ишлатилади, яъни серверга фақат сўров матни жўнатилади;

• МББТда сервердаги МБ нинг физик структураси ҳақидаги маълумотлар сақланади;

• МББТ сервердаги маълумотларга мурожаатни қайд қилади ва серверда маълумотларни қайта ишлаш амаллари бажарилади ва натижа мижоз

компьютерига юборилади. Шу тарзда МББТ натижаларни мижоз–дастурига юборади;

• дастур эса ўзидаги фойдаланувчи интерфейси ёрдамида натижаларни тасвирлаб беради.

Бажариладиган функциялар сервер ва мижоз ўртасида қандай тақсимланганини кўриб чиқамиз:

Мижоз дастури функциялари:

- сўровларни серверга жўнатиш;

- сервердан олинган сўров натижаларини талқин қилиш;

-натижаларни бирор шаклда фойдаланучига кўрсатиш (фойдаланувчи интерфейси).

Сервер функциялари:

- мижоз–дастурлардан сўровларни қабул қилиш;

- сўровларни талқин қилиш;

- МБ га сўровларни оптималлаштириш ва бажариш;

- натижаларни мижоз-дастурга жўнатиш;

- хавфсизлик тизимини ва мурожаатни чегаралаш;

- МБ бутунлигини бошқариш;

- кўп фойдаланувчили иш режими барқарорлигини таъминлаш.

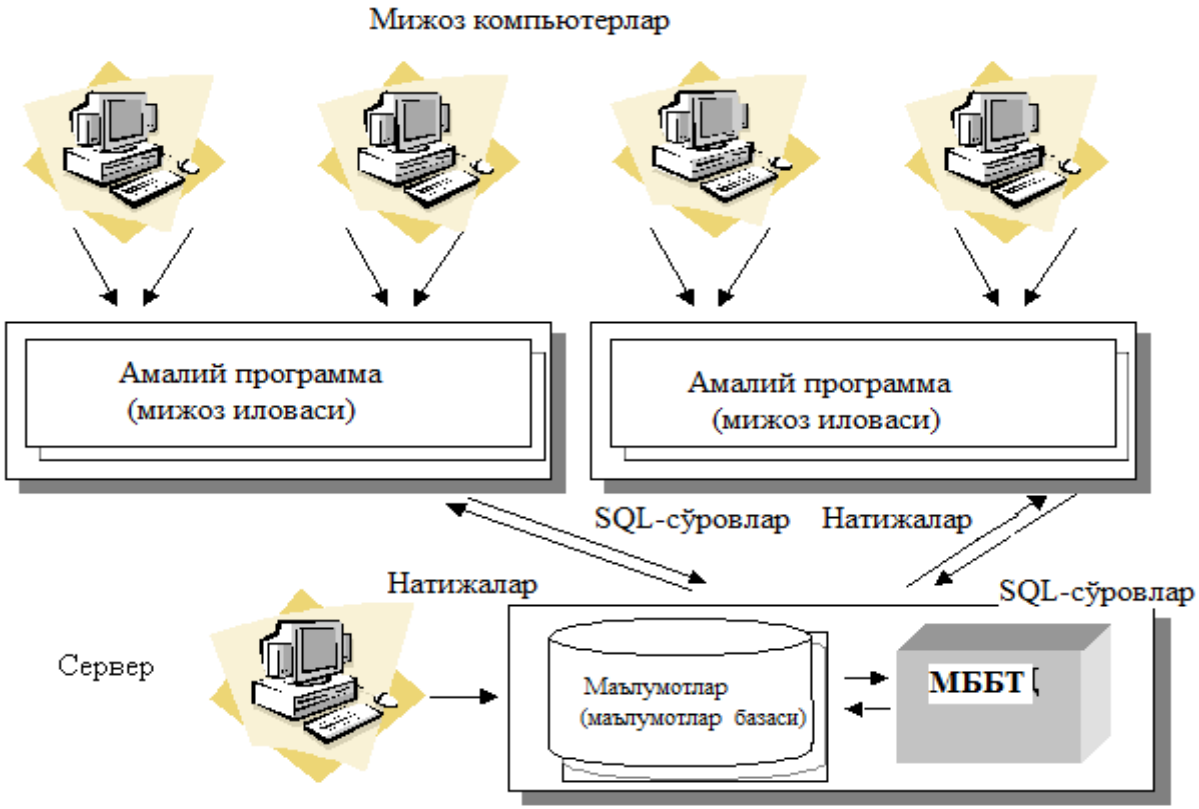
Қоидага кўра, SQL–серверга битта ходим ёки ходимлар гуруҳи, яъни SQL–сервер администратори томонидан хизмат кўрсатилади. Улар маълумотлар базасини физик характеристкаларини бошқаради, МБнинг турли компонентларини ростлайди ва қайта аниқлайди, оптималлаштиради, янги МБлар яратади, мавжуд МБларни ўзгартиради ҳамда турли фойдаланувчилар ваколатларини белгилайди.

Мижоз маълумотларни қайта ишлаш жараёнида сервердан мураккаб амалларни бажариш, файлларни ўқиш, маълумотлар базасида маълумотларни қидириш ва ҳ.к.лар бўйича сўровни шакллантириши мумкин (6-расм).

Мижоз – компьютер тармоғининг масаласи, ишчи станцияси ёки фойдаланувчиси. Марказий машинада (серверда) жамоа бўлиб

фойдаланиладиган марказлаштирилган маълумотлар базаси сақланади деб тасаввур қилинади. Тармоқдаги қолган барча машиналар ишчи станция (мижоз) вазифасини бажаради. Улар ёрдамида тизимдан фойдаланувчиларнинг марказий маълумотлар базасидан рухсат этилган фойдаланиши амалга ошириб берилади.

Фойдаланувчиларнинг сўровларига мос равишда маълумотлар базаси файллари ишчи станцияларга узатилади. Улар асосан шу станцияларда қайта ишланади. Иш станцияси фойдаланувчи сўровларини қайта ишлашда қаноатлантирадиган даражадаги реактивликни таъминлаш учун зарур бўлган ресурсга эга бўлиши шарт.



6-расм. Мижоз-сервер архитектураси

Мижоз-сервер тамойили доирасида сервер нафақат дастур ва маълумотларни сақлаш жойи, балки ҳисоблаш муҳити сифатида ҳам ишлатилади. Қаралаётган моделда дастурий таъминот ўзаро боғланган иккита дастурдан ташкил топади: файл-сервер ва мижоз-фойдаланувчи дастурлари. Дастур–мижоз сўровни шакллантиради ва уни ҳамма учун

рухсати бўлган компьютерга ўрнатилган файл-серверга (дастурлар) жўнатади.

Маълумотларни қайта ишлаш ҳамма фойдаланадиган қувватли компьютерда бажарилади, компьютер-мижозда эса тегишли баённома ёрдамида бажарилган сўров натижалари акс эттирилади.

Ҳар қандай МБ ҳар хил объектларга эга, яъни жадваллар, процедуралар, функциялар, тасаввурлар, кетма кетликлар ва ҳоказоларга. Мижоз-сервер технологиясига кўра фойдаланувчи ЭҲМ (Мижоз)лар сўровлари махсус маълумотлар серверларида (Сервер) қайта ишланади, фойдаланувчи ЭҲМ ларга фақат сўровни қайта ишлаш натижалари қайтарилади.

Табиийки, сервер билан мулоқот қилиш учун ягона тил керак ва бундай тил сифатида SQL танланди. Шунинг учун ҳамма замонавий реляцион МББТ версиялари (DB2, Oracle, Ingres, Informix, Sybase, Progress, Rdb) ва ҳаттоки нореляцион МББТ версиялари (масалан, Adabas) мижоз-сервер технологияси ва SQL тилидан фойдаланадилар.

SQL-сервер–махсус дастур бўлиб, у узоқда жойлашган маълумотлар базасини бошқаради. *SQL-сервер* сўровларни қабул қилиш, унинг МБ да бажарилиши ва унинг натижасини мижозга юбориш каби вазифаларни амалга оширади.

Бунда мижоз компьютер ресурсларидан фойдаланилмайди, мижоз компьютер фақат МБ серверига сўров жўнатади. Узоқдаги сервер сўровни қабул қилади ва уни МБ нинг *SQL-сервер*ига йўналтиради ва мижоз компьютер натижани олади.

3.3. Мижоз-сервер технологиясининг дастурий таъминоти

“Мижоз-сервер” туридаги маълумотлар базасини бошқариш тизимлари (МББТ) га қуйидаги машхур МББТ ларни киритиш мумкин:

- IBM DB2;
- Microsoft SQL Server (Microsoft SQL Server Express);
- MariaDB
- MySQL;

► Oracle (Oracle Express);

► PostgreSQL.

Кўпгина МББТ лар ўзининг хусусий мижоз утилиталари билан ишлаб чиқарилади. Улардан баъзиларини кўриб чиқамиз.

I. **Apache Open Office Base** — бу маълумотлар базаси билан ишлашга мўлжалланган ва Javaга асосланган очиқ мижоз иловасидир. Сўровларни SQL тилида ёзиш мумкин. Бунинг учун:

1. **Apache OpenOffice Base** да ўз базангизни очинг.

2. Ойнанинг чап қисмида жойлашган **Database** панелида **Queries** бўлимини танланг.

3. Query Design ойнасини ҳосил қилиш учун **Tasks** панелида **Create Query In SQL** тугмасини босинг.

4. Катта матн майдонида ўз SQL-сўровингизни киритинг.

5. SQL-сўровни бажариш учун **Run Query** тугмасини босинг (унда ҳужжатлар ва яшил белги тасвирланган). Шунингдек, **<F5>** тугмасини босиш мумкин ёки **Edit** менюсидаги **Run Query** буйруғини танланг.

II. IBM DB2 – бу IBM компаниясида ишлаб чиқарилган кучли юқори самарали МББТ ҳисобланади. У SQL-сўровларни бажаришда фойдаланиладиган мижоз асбоблари мажмуи билан биргаликда ишлаб чиқарилади. Қуйида **Control Center Java**-утилитаси учун мўлжалланган утилитани кўрайлик:

1. **Control Center** ни ишга туширинг.

2. Object View панели ойнасининг чап қисмида барча мавжуд маълумотлар базасининг рўйхати келтирилади. Раскройте раздел **AH Databases** бўлимини очинг ва керакли маълумотлар базасини танланг.

3. **Query** ойнасини очиш учун ўз маълумотлар базангизнинг номи устида сичқончанинг ўнг тугмасини босинг ва va **Query** бўлимини ёки Selected менюсидаги **Query** бўлимини танланг.

4. Юқоридаги майдонга SQL инструкциясини киритинг.

5. Сўровни ишга тушириш учун **Execute** тугмасини босинг.

6. Қуйидаги ойнада сўров натижасини жадвал кўринишида кўриш учун **Query Results** ни босинг.

III. MariaDB шахсий мижоз утилитасига эга эмас, шунинг учун MySQL нинг мижоз утилиталаридан фойдаланилади.

IV. MySQL (ўқилиши „Май эс кю эл“). Реляцион маълумотлар базасини бошқариш системаси. Турли операцион системаларда қўлланишга мослаштирилган Open-Source - маҳсулот бўлиб, кўпчилик динамик web-саҳифалар учун асос ҳисобланади.

MySQL Швециядаги MySQL AB фирмаси томонидан яратилган бўлиб, 2008-йилнинг февралида Sun Microsystems томонидан сотиб олинган. Эндиликда дастур кодини яратишда ушбу фирма жавобгар. General Public License ҳимоясидадир. MySQL AB/Sun фирмаси дастур кодига муаллифлик ҳуқуқига эгалиги сабабли ушбу дастурий таъминотнинг коммерциал тури ҳам бор.

MySQL номининг келиб чиқиши аниқ эмас. 1996-йилдан бери кўплаб кутубхона ва tool (восита)лар номи олдида My префикси қўйила бошланган.

Баъзи маълумотларга кўра эса фирмани ташкил қилганлардан бири деб ҳисобланувчи Монти Уидениус (Monti Uidenius) нинг қизи исмидан ва SQL бирикмасидан ташкил топган.

Unix, MacOS X ва Linixdan ташқари, MySQL Windows, OS/2 ва i5/OS (аввалги OS/400) учун ҳам мослаштирилган.

Windows учун эса баъзида айрим чекланишлар учрайди. MySQL кўпинча web-сервер учун маълумотларни сақлашда қўлланилади, web-сервер **Apache** ва **PHP** билан биргаликда ишлатилади.

MySQL ҳозирда 6.000.000 дан ортиқ фойдаланувчилар ва кунига 35.000 дан ортиқ download қилинадиган дунёнинг энг машҳур маълумотларни бошқариш тизимидир.

MySQL билан икки усулда ишлаш мумкин. Бу МББТ mysql деб аталадиган буйруқлар сатри утилитаси билан бирга етказиб берилади. Бу

ихтиёрий SQL инструкциясини бажаришда қўлланиладиган ва сўровлар яратишнинг матнли воситасидир.

Бундан ташқари, ишлаб чиқарувчилар MySQL Workbench интерфаол утилитасини муомалага чиқардилар. MySQL ни ўрганишда у билан ишлашни тавсия қилинади.

mysql утилитасидан фойдаланиш учун қуйидаги амалларни бажариш керак:

1. Бу утилитани ишга тушириш учун **mysql** ни киритинг, хавфсизликни таъминлаш учун u ва p параметрларни, яъни фойдаланувчи номи ва паролни киритиш керак.

2. mysql> таклифидан сўнг ишчи маълумотлар базасининг номи киритилади: USE база_данных.

3. mysql > таклифидан сўнг ўз SQL-сўровингизни киритинг ва ҳар бир инструкция нуқта вергул (;) билан тугашини текширинг. Натижалар экранда намоён бўлади.

4. Мавжуд буйруқлар рўйҳатини олиш учун \h ни ёки ахборот ҳолатини билиш учун \ s ни киритинг.

5. mysql утилитасидан чиқиш учун \q ни киритинг.

MySQL Workbench утилитасидан фойдаланиш учун қуйидагиларни бажаринг:

1. Утилитани ишга туширинг.

2. Чап устунда MySQL MB сига мавжуд бўлган уланишлар кўрсатилади. Хоҳлаган уланишни очиш учун унинг устида сичқонча тугмасини босинг ёки керакли маълумотлар базаси рўйҳатда бўлмаса New Connection буйруғини танланг.

3. Маълумотлар базасига улангандан сўнг панелли ойна пайдо бўлади. Object Browser панелининг чап қисмида мавжуд маълумотлар базаси келтирилган, марказдаги катта матнли панел SQL инструкцияларини киритишга мўлжалланган, сўров натижалари ва хабарлар қуйи панелда келтирилади.

4. Янги SQL ойнасини очиш учун + SQL тугмасини босинг.

5. Сўровни киритгандан сўнг уни бажариш учун Execute тугмасини босинг. Натижалар қуйида келтирилади.

V. Microsoft SQL Server (шу жумладан, **Microsoft SQL Server Express** ҳам)

Microsoft SQL Server **SQL Server Management Studio** номли кучли утилитага эга. У кўпгина масалаларни ечиш имконини беради, жумладан, маълумотлар базасини бошқариш, хавфсизликни таъминлаш, ҳисобот яратиш ва бошқалар. У шунингдек SQL-сўровлар яратиш учун қулай муҳитни таъмилайди. Бунинг учун:

1. SQL Server Management Studio ни ишга туширинг.

2. Сервер ҳақидаги ва ўзингиз ҳақидаги маълумотларни киритинг.

3. Бир неча панелларга бўлинган **SQL Server Management Studio** ойнаси пайдо бўлади. Object Explorer панели ойнасининг чап қисмида барча мавжуд маълумотлар базасининг рўйхати уларнинг параметрлари билан келтирилади. Ойнанинг асосий қисмини SQL инструкцияларини киритишга мўлжалланган катта матн майдони эгаллаган.

4. Пастки панелнинг чап қисмида очилувчи рўйхат бўлиб, унда фақат мавжуд бўлган маълумотлар базаси кўрсатилган. Агар керак бўлса рўйхатдан бошқа маълумотлар базасини танланг.

5. Катта матнли ойнада ўз SQL-сўровингизни киритинг ва уни бажариш учун **Execute Query** тугмасини босинг.

VI. Oracle да жуда кўп бошқарувчи ва мижоз утилиталари мавжуд. SQL ни ўрганишда **Oracle SQL Developer** утилитасидан фойдаланишни тавсия этамиз.

У билан ишлаш учун қуйидагиларни бажариш керак:

1. **Oracle SQL Developer** ни ишга туширинг..

2. Бирор маълумотлар базаси билан ишлашдан аввал унга уланиш керак. Бунинг учун ойнанинг чап қисмидаги Connections панелидаги буйруқлардан фойдаланиш мумкин.

3. Маълумотлар базасига улангандан сўнг SQL инструкцияларини киритиш учун **SQL Worksheet в окне Query Builder** ёрлиғидан фойдаланиш мумкин.

4. SQL- сўровингизни амалга ошитиш учун **Execute** тугмасини босинг. Сўров натижалари қуйи панелда тасвирланади.

VII. Postgre икки усулда ишлаш мумкин. Бу МББТ psql деб аталадиган буйруқлар сатри утилитаси билан бирга етказиб берилади. Бу ихтиёрий SQL инструкциясини бажаришда қўлланиладиган ва сўровлар яратишнинг матнли воситасидир. Бундан ташқари, **pgAdmin** интерфаол утилитаси мавжуд бўлиб, у асосан маъмурий мақсадларга мўлжалланган, лекин SQL инструкцияларини тестлашда ҳам қўллаш мумкин.

psql утилитасидан фойдаланиш учун қуйидаги амалларни бажариш керак:

1. Бу утилитани ишга тушириш учун **psql** ни киритинг, муайян маълумотлар базасини юклаш учун psql буйруқлар сатрида уни номи киритилади: база_данных.

2. Ўз SQL-сўровингизни киритинг ва ҳар бир инструкция нуқта вергул (;) билан тугашини текширинг. Натижалар экранда намоён бўлади.

3. Мавжуд буйруқлар рўйҳатини олиш учун \ ? ни киритинг.

4. SQL бўйича маълумот олиш учун \ h ни киритинг.

5. psql утилитасидан чиқиш учун \ q ни киритинг.

3.4. MySQL маълумотлар базаси сервери

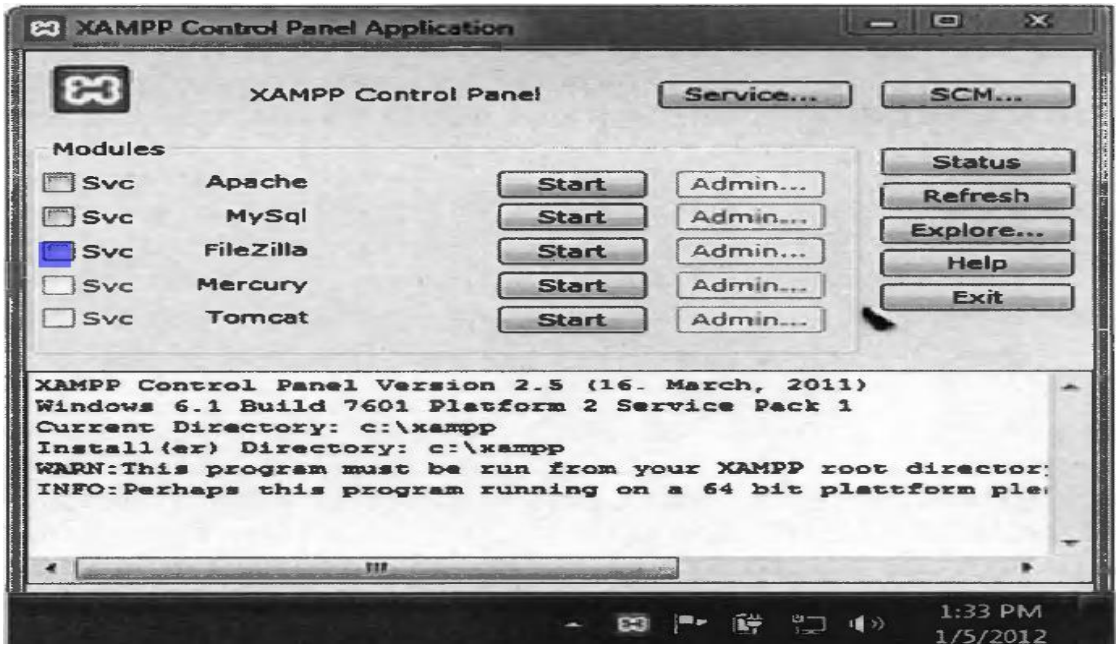
MySQL маълумотлар базаси (МБ) сервери – бу катта ҳажмдаги ахборотларни структураланган кўринишда сақлаш ва PHP скрипт тили ёрдамида унга кириш имконини берувчи дастур ҳисобланади. PHP сервер томонида ишлайдиган сервер тили ҳисобланади. MySQL МБ сервери мижоз дастурлари сўровларига жавоб беради. PHP скрипт кўринишида MySQL учун хусусий мижоз иловасини яратишдан аввал биз **phpMyAdmin** мижоз иловасидан фойдаланайлик.

MySQL маълумотлар базаси серверига PHP тилида ёзилган **phpMyAdmin** иловаси ёрдамида уланиш мумкин. У SQL сўровларни киритиш ва натижалар олиш имконини беради.

MySQL маълумотлар базаси серверини бошқариш учун SQL тили буйруқларидан фойдаланилади. Бу реляцион МБ ларида ишлашга имкон берадиган тилдир. Демак, MySQL - маълумотлар базаси сервери, SQL тили ёрдамида сиз бу маълумотлар базаси билан мулоқот қиласиз.

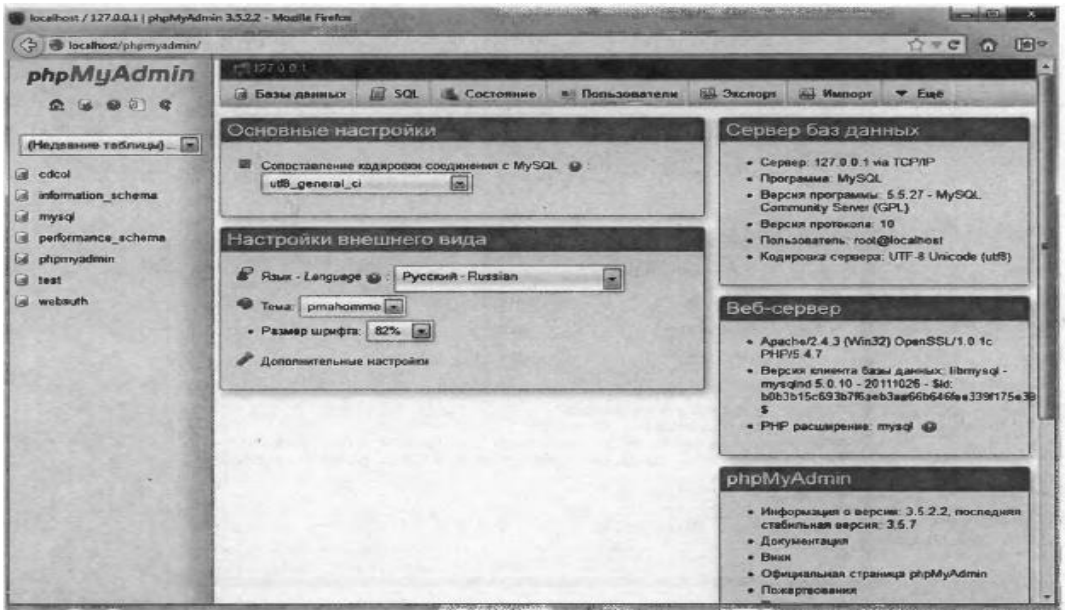
MySQL маълумотлар базаси сервери иши билан танишиш учун бир неча буйруқларни киритиш зарур, ундан аввал PHP тилида ёзилган **phpMyAdmin** иловасини ҳосил қиламиз. Бу илова XAMPP ва MAMP дастурий пакетларнинг таркибига киради. Бу пакетлар ичида Apache, PHP ва MySQL нинг нусхалари бўлиб, **phpMyAdmin** иловасидан фойдаланиш учун аввало XAMPP ёки MAMP ёки Denver ни ўрнатиш зарур.

XAMPP ўрнатилгандан сўнг система қайта юкланганда XAMPP нинг бошқариш панели пайдо бўлади (7-расм).



7-расм. XAMPP нинг бошқариш панели

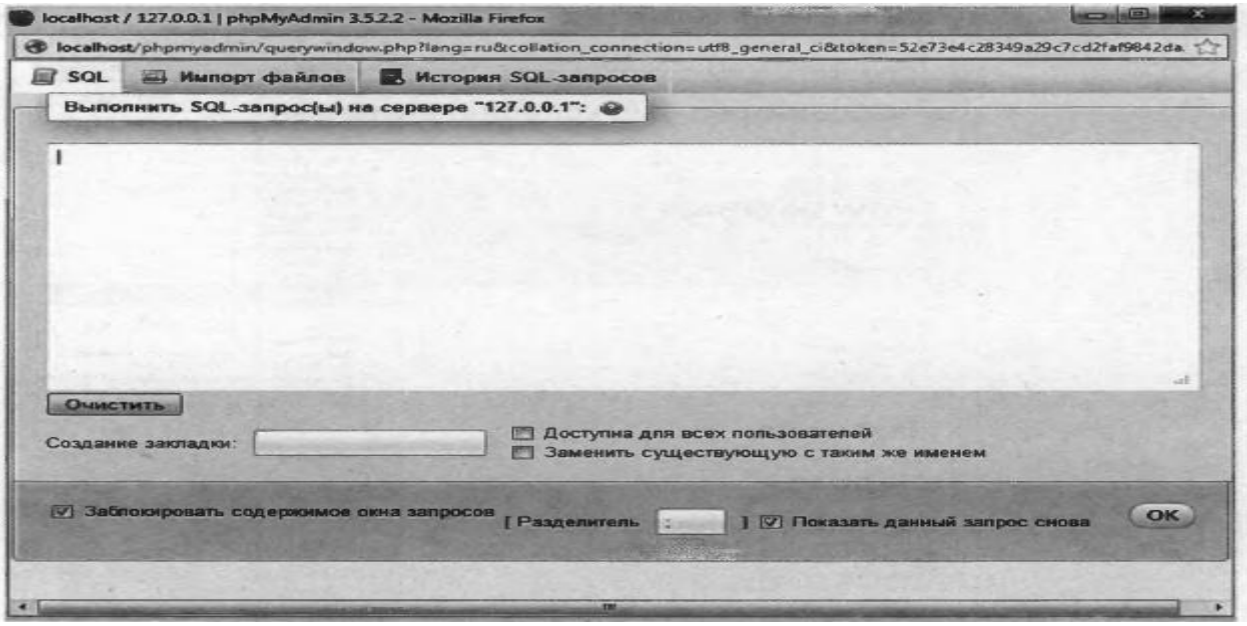
phpMyAdmin иловасига кириш учун MySQL ёзуви рўпарасидаги **Admin** тугмаси босилади. Шундай қилиб, натижада браузерда **phpMyAdmin** саҳифаси очилади (8-расм).



8- расм. phpMyAdmin саҳифаси

phpMyAdmin логотипининг тагида нишонлар жойлашган, чапдан иккинчи нишон устида сичқонча тугмасини босиб (9-расм), SQL-сўровларни киритиш учун 10-расмдаги ойнани ҳосил қиламиз.

9- расм. SQL- сўровлар ойнасини очиш учун нишон

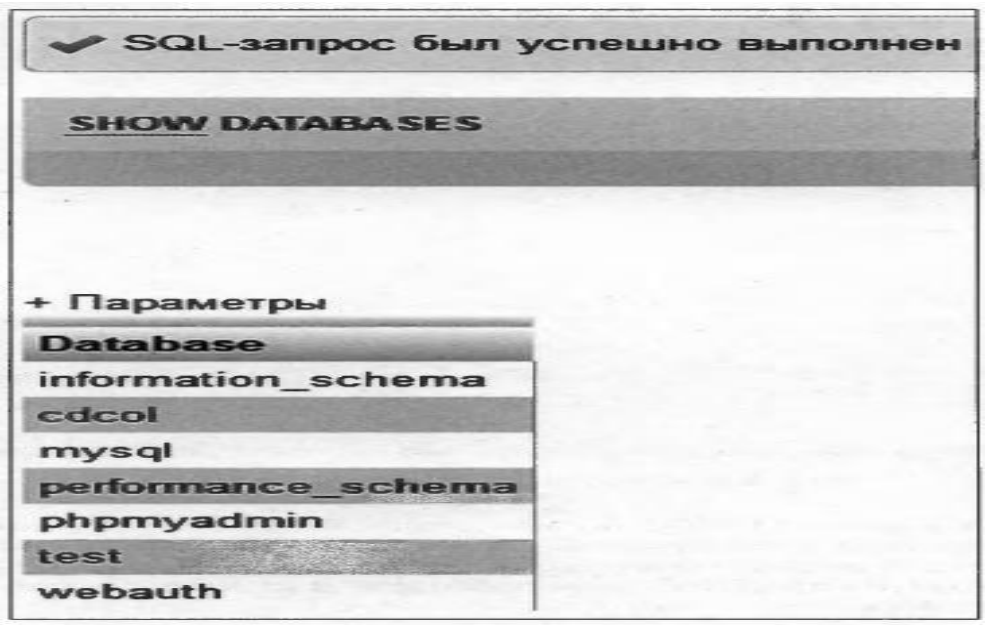


10-расм. SQL- сўровни киритиш ойнаси

MySQL маълумотлар базаси сервери бир неча маълумотлар базасини бошқариши мумкин. Серверга улангандан сўнг биринчи навбатда сиз ишлаш керак бўлган маълумотлар базасини танлашингиз керак. Бунинг учун жорий сервердаги базалар рўйҳатини олиш зарур.

SQL- сўровни киритиш ойнасида қуйидаги буйруқни киритинг: SHOW DATABASES ва OK тугмасини босинг.

Бу сўров MySQL серверда мавжуд бўлган маълумотлар базаси рўйҳатини кўрсатади. Натижада phpMyAdmin нинг асосий ойнасида сўров натижаси ҳосил бўлади (11-расм).



11-расм. phpMyAdmin нинг асосий ойнасида сўров натижаси

Унга кўра **information_schema** ва **mysql** номли базалар мавжуд эканлигини кўрамиз. MySQL биринчи базадан серверда қолган базаларни ҳисобга олишда фойдаланади, агар сиз катта лойиҳалар билан ишлашни режалаштирмаётган бўлсангиз уни шундайлигича қолдирганингиз маъқул.

Иккинчи базада MySQL фойдаланувчилар, уларнинг пароли ва ваколати тўғрисидаги маълумотларни сақлайди.

test маълумотлар базаси – бу шунчаки шаблон, у бошида MySQL билан биргаликда берилади, уни бемалол ўчириб ташлашингиз мумкин, чунки сиз ўз хусусий маълумотлар базангиз билан ишлайсиз.

Яна бир MySQL буйруғини кўрайлик: DROP (Ўчириш)

DROP DATABASE *test*

Бу буйруқни SQL- сўровни киритиш ойнасида киритиб, OK тугмасини боссангиз, phpMyAdmin DROP DATABASE буйруғи ўчирилганлиги ҳақидаги хабарни чиқаради. Шундай қилиб, phpMyAdmin га ўрнатилган хавфсизлик воситаси жуда муҳим сўровларни ишга туширишга тўсқинлик қилади.

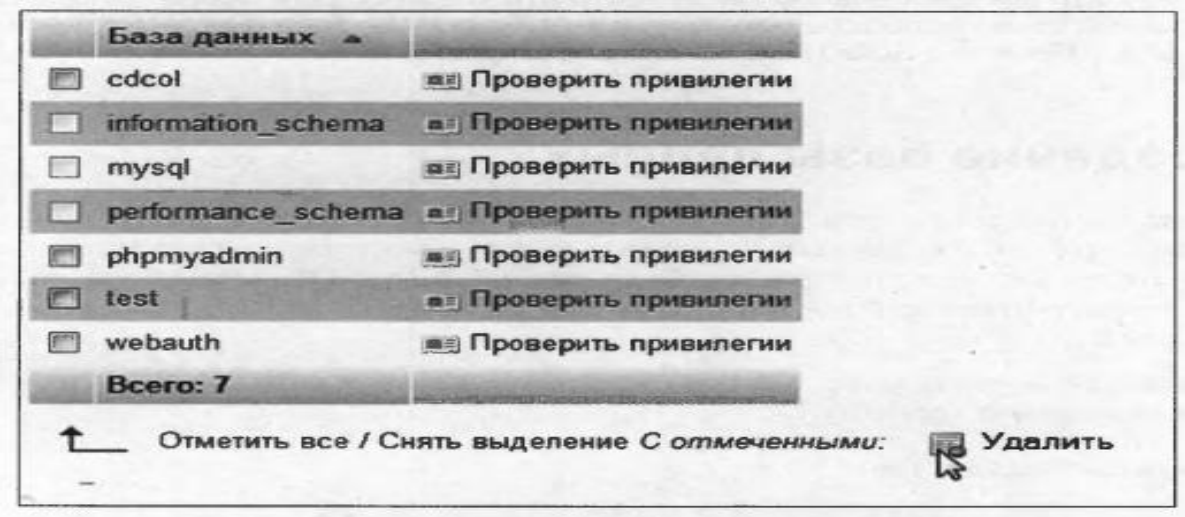
phpMyAdmin даги маълумотлар базасини ўчириш учун phpMyAdminning асосий ойнасидаги **Базы данных** (юқорида чапдан охирги) ёрлиғига ўтинг, серверингиздаги базалар рўйҳатини кўрасиз.

Сиз ўчирмоқчи бўлган база (бизнинг ҳолда бу *test)* олдидаги байроқчани белгилаб, пастдан ўнгдаги **Удалить** тугмасини босинг(12-расм).

phpMyAdmin иловаси сиз ҳақиқатан ҳам маълумотлар базасини ўчиришингизни тасдиқлашни сўрайди. Ижобий жавоб бўлганда сизнинг илтимосингизни бажаради, phpMyAdmin эса бу амал муваффақиятли бажарганлиги тўғрисидаги хабарни чиқаради.

MySQL да DROP DATABASE га ўхшаш хавфли буйруқлар мавжуд бўлиб, агар эҳтиётлик билан иш кўрмасангиз, барча маълумотлар базасини ўчириб ташлашингиз мумкин.

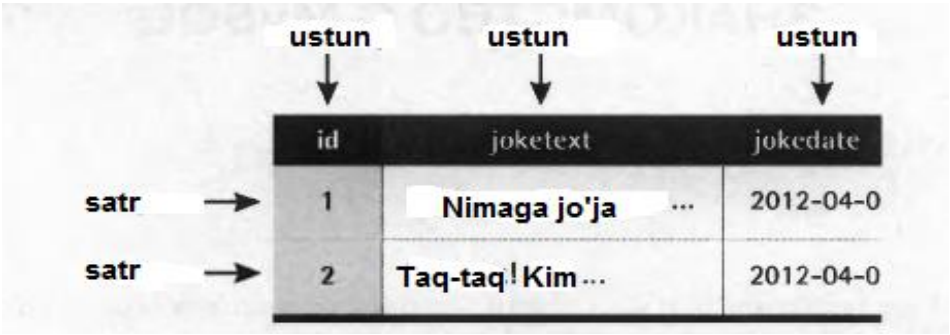
phpMyAdmin ҳар доим ҳам сизни хатолардан ҳимоя қилиб, огоҳлантирувчи хабарларни чиқармаслиги мумкин.



12-расм. phpMyAdmin да маълумотлар базасини ўчириш тугмаси ишончли ҳимояланган (яширинган)

3.5. MySQL да маълумотлар базасини яратиш

Энди маълумотлар базасида ахборот қандай сақланишини батафсил кўриб чиқайлик. Маълумотлар базаси ўз ичига битта ёки бир нечта жадвалларни олади. Ишни joke (ҳазил) номли жадвални яратишдан бошлаймиз. Маълумотлар базасидаги ихтиёрий жадвал бир неча **устун** ёки **майдон**дан иборат бўлади. joke жадвали учта устундан иборат бўлсин: **id, joketext, jokedate** (13-расм). Биринчи устун – **id** устуни ҳар бир ҳазилнинг уникал (такрорланмайдиган) номери, иккинчи устун – **joketext** устунида ҳазиллар матни сақланади, учинчи устун – **jokedate** устунида ҳазилларни маълумотлар базасига киритилган санаси сақланади.



13-расм. Ҳазиллар матнини ўз ичига олган маълумотлар базаси жадвали

Бу жадвални яратиш учун қуйидаги буйруқларни бажариш керак:

```
CREATE TABLE joke  (
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    joketext TEXT,
    jokedate DATE NOT NULL
 )  DEFAULT CHARACTER SET utf8 ENGINE=InnoDB
```

Ҳар бир сатрни алоҳида кўриб чиқайлик. Биринчи сатр joke номли жадвални яратади.Очилувчи юмалоқ қавс жадвалдаги устунлар рўйҳатини бошланишини билдиради. Иккинчи сатр id INT NOT NULL AUTO_INCREMENT PRIMARY KEY, бизга тури integer (INT) бўлган бутун сонларни сақлаш учун **id** устуни кераклигини билдиради. Сатрнинг қолган қисмларида устуннинг қўшимча характеристикалари келтирилади:

• жадвалда сатр яратилганда, унга мос устун бўш бўлмаслиги керак (NOT NULL);

• ёзув қўша туриб устуннинг қиймати кўрсатилмаса, у ҳолда MySQL автоматик равишда жадвалдаги энг катта қийматни бирга ошириб сон танлайди (AUTO_lNCREMENT);

• жадвалдаги бу устун уникал (такрорланмайдиган) идентификатор ролини ўйнагани учун ундаги барча ёзувлар уникал бўлиши керак (PRIMARY KEY).

joketext TEXT - учинчи сатр (TEXT) матндан иборат **joketext** устуни кераклигини билдиради.

jokedate DATE NOT NULL

Тўртинчи сатр охирги **jokedate** устунини билдиради, унда (DATE) сана маълумотлари сақланади, у ҳам тўлдирилган бўлиши керак, яъни бўш бўлмаслиги керак.

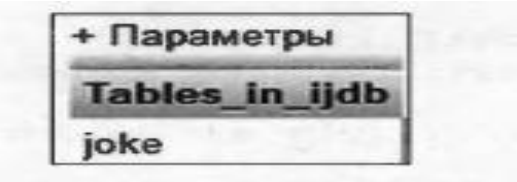
) DEFAULT CHARACTER SET utf8

Бу сатрдаги ёпилувчи юмалоқ қавс жадвалда устунлар рўйҳати тугаганлигини билдиради. DEFAULT CHARACTER SET utf8 MySQL га бу жадвалдаги матн UTF-8 да кодланган бўлиши кераклигини хабар беради. Бу веб-саҳифаларни кодлашнинг энг кенг тарқалган усули ҳисобланади.

ENGINE=InnoDB

сатрдан MySQL жадвал яратишда маълумотларни сақлашда қайси форматдан фойдаланиши кераклиги аниқлайди, InnoDB формати ҳозирги кунда сайт яратишда маълумотлар базаси учун энг яхши формат ҳисобланади, агар бу формат кўрсатилмаса MySQL эски MyISAM форматини беради.

Бу жадвалнинг ҳақиқатан ҳам яратилганлигига ишонч ҳосил қилиш учун SQL- сўровни киритиш ойнасида қуйидаги буйруқни киритамиз: **SHOW TABLES** ва OK тугмасини босамиз. phpMyAdmin иловаси 14-расмдаги натижани чиқаради. Бу сизнинг **ijdb** номли базангиздаги барча жадваллар рўйҳати, унда фақат битта биз ҳозиргина яратган joke жадвали бор.



14-расм. Мавжуд маълумотлар базасидаги жадваллар рўйҳати

Жадвални яратишни кўриб чиқдик, энди уни тўлдириш, яъни маълумотлар киритиш керак, ундан аввал уни ўчириш ҳақида тўхталайлик. Бу вазифа ҳам маълумотлар базасини ўчириш каби бўлади, лекин МБ ни DROP DATABASE буйруғи билан ўчирмоқчи бўлганда phpMyAdmin хавфли сўров ҳақида огоҳлантирган эди, бу сафар жадвални ўчиришда бундай ҳол бўлмайди, шунинг учун бундай сўровни беришдан олдин янги жадвал яратишга тайёр бўлиш керак. Жадвални ўчириш учун қуйидаги буйруқ берилади:

DROP TABLE жадвал номи ёки DROP TABLE joke

Энди жадвалга маълумотлар киритишни кўрайлик. Бунинг учун INSERT буйруғидан фойдаланилади. У икки кўринишда бўлади:

INSERT INTO жадвал номи SET

1- устун номи = 1- устун қиймати,

2- устун номи = 2- устун қиймати,

…

ёки

INSERT INTO жадвал номи

(1- устун номи, 2- устун номи, ...)

VALUES (1- устун қиймати, 2- устун қиймати, ...)

Жадвалга ҳазил қўшиш учун хоҳлаган буйруқдан фойдаланиш мумкин.

INSERT INTO joke SET

joketext = " Нима учун жўжа йўлни кесиб ўтди? Йўлнинг нариги томонига ўтиш учун!",

jokedate = "2012-04-01"

ёки

INSERT INTO joke

(joketext, jokedate) VALUES (

" Нима учун жўжа йўлни кесиб ўтди? Йўлнинг нариги томонига ўтиш учун!",

"2012-04-01")

Энди жадвалдан маълумотларни чиқаришни кўрайлик. Бунинг учун SELECT фойдаланилади. Қуйидаги буйруқ **joke** жадвалида сақланадиган барча маълумотларни чиқариб беради:

SELECT * FROM joke

Бу сўровни “joke дан ҳаммасини танланг (ёки олинг)” деб таржима қилса бўлади. Бу сўров натижаси қуйидагича бўлади:

id	joketext	jokedate
1	Nima uchun jo'ja yo'lni kesib o'tdi? Yo'lning narigi tomoniga o'tish uchun!	2012-04-01

15-расм. phpMyAdmin joke жадвалидаги барча маълумотларни чиқариб беради

MySQL да фақат **id** ва **jokedate** устунларини чиқариш сўрови қуйидагича бўлади:

SELECT id, jokedate

FROM joke

Сўров натижаси қуйидагича:

id	jokedate
1	2012-04-01

Натижалар сонини, масалан, joke жадвалидаги ҳазиллар сонини билиш учун COUNT функциясидан фойдаланилади:

SELECT COUNT(*)

FROM joke

Сўров натижаси қуйидагича:

COUNT(*)
1

Демак, joke жадвалида фақат битта ҳазил бор.

3.6. SQL-ифодалар синфи

SQL тили бир неча алоҳида бўлакларга бўлинган:[9]

- маълумотлар схемасини бошқарувчи SQL-ифодалар;
- маълумотлар билан ишлашга мўлжалланган SQL-ифодалар;
- транзакцияларни бошқарувчи SQL-ифодалар.

Маълумотлар схемасини бошқарувчи SQL-ифодалар маълумотлар базасида сақланадиган маълумотларнинг структурасини аниқлашга мўлжалланган. Масалан, маълумотлар базасидаги янги жадвал маълумотлар схемасини бошқарувчи SQL-ифода **creat table** (жадвал яратиш) ёрдамида яратилади, бу жадвални маълумотлар билан тўлдириш учун эса маълумотлар билан ишлашга мўлжалланган SQL-ифода **insert (киритиш)** керак бўлади.

Бу иккала ифодани мисолда кўриб чиқайлик. Қуйида **corporation** (корпорация) номли жадвални яратувчи маълумотлар схемасини бошқарувчи SQL-ифодани келтирамиз:

```
CREATE TABLE corporation
 (corp_id SMALLINT,
  name VARCHAR(30),
  CONSTRAINT pk_corporation PRIMARY KEY (corp_id)
 );
```

Бу ифода иккита **corp_id** ва **name** устунларидан иборат жадвални яратади, бу ерда **corp_id** устуни бирламчи калит сифатида аниқланган ва унинг тури SMALLINT- "қисқа бутун" кўринишда бўлиб, одатда 5 тагача қийматли рақам ва ишорадан иборат бўлади. **name** устуни тури эса VARCHAR(30) - узунлиги ўзгарувчи, 30 та символдан ошмаган символли қатор.

Охирги сатрда жадвал учун чекланишлар (constraint) келтирилган, унда МБ серверига қайси устун ёки устунлар бирламчи калит (primary key)

[9] Алан Бьюли. Изучаем SQL. Вводный курс для разработчиков и администраторов БД. Санкт-Петербург–Москва, Символ ®, 2007

сифатида аниқланганлиги ҳақида хабар берилади, бизнинг мисолимизда бу чекланиш **corp_id** устунига қўйилган ва у **pk_corporation** деб номланган.

Энди маълумотлар билан ишлашга мўлжалланган SQL-ифодани, яъни **corporation** жадвалига ёзув киритувчи ифодани келтирамиз:

```
INSERT INTO corporation (corp_id, name)
VALUES (27, 'Acme Paper Corporation');
```

Бу ифода **corporation** жадвалига корпорация идентификатори 27 ва номи Acme Paper Corporation бўлган сатрни киритади.

Энди ҳозиргина киритилган маълумотларни ҳосил қилиш учун SELECT (танлаш) ифодасидан фойдаланамиз. Бу SQL-сўровни mysql > таклифидан сўнг киритамиз:

```
mysql< SELECT name
    > FROM corporation
    > WHERE corp_id = 27;
```

Сўров натижаси қуйидагича бўлади:

```
+-----------------------------------------+
| name                                    |
+-----------------------------------------+
| Acme Paper Corporation                  |
+-----------------------------------------+
```

Энди транзакцияларни бошқарувчи SQL-ифодалар билан танишайлик. Баъзи ҳолларда бир вақтнинг ўзида параллел равишда бир неча SQL-сўровларни амалга оширишга тўғри келади. Бир неча фойдаланувчи бир хил маълумотдан фойдаланишига тўғри келганда яна бир элемент зарур бўлади. Бунда транзакциялардан фойдаланилади.

Транзакция инглизча transaction сўзидан олинган бўлиб, операция, манипуляция, амаллар бажариш каби маъноларни англатади. Юқорида таъкидлаб ўтилганидек, транзакция – бу маълумотларни манипуляция қилиш (қайта ишлаш) операторларининг кетма-кетлиги бўлиб, у МБ ни бир ҳолатдан иккинчи ҳолатга ўтказади (1.2 бўлимга қаранг).

Транзакция параллелик ҳолатидаги бир неча SQL-ифодаларни гуруҳлаш механизми ҳисобланади. Улар бир вақтда амалга оширилиши керак бўлган сўровлар тўплами устида мураккаб амалларни бажариш имконини беради.

Фараз қилайлик, сизнинг дуконингиз маълумотлар базасида иккита жадвал бўлиб, биринчи жадвалда сотувга қўйилган маҳсулотлар рўйҳати ва уларнинг сони, иккинчи жадвалда эса сайтингизга ташриф буюрганлар томонидан қилинган буюртмалар келтирилган бўлсин. Буюртма расмийлаштирилгандан сўнг маҳсулотлар тўғрисидаги маълумотларни янгилаш, шунингдек буюртма қилинган маҳсулотларни белгилаш ва корзина қутисига ташлаш керак бўлади, бу амаллар кетма-кетлиги тўғри бажарилмаса сайтингизга ташриф буюрган бошқа фойдаланувчи ахборотлар қарама-қаршилигига дуч келади ва бу ҳолат сизнинг дуконингиз обрўсига салбий таъсир кўрсатиши мумкин.

Яна бир мисол. Мижоз омонат ҳисобидаги 500 долларини жорий ҳисобга ўтказмоқчи бўлсин. Бу транзакцияни бошлаш START TRANSACTION буйруғи орқали амалга оширилади. Сўнгра керакли SQL-сўровлар жўнатилади ва у муваффақиятли бажарилса COMMIT буйруғи орқали тугалланади.

Агар бу пуллар биринчи ҳисобдан муваффақиятли ечиб олиниб, техник сабабларга кўра иккинчи ҳисобга ўтказилмаган бўлса, у ҳолда мижознинг норозилигига сабаб бўлади ва у ўзининг 500 долларини қайтариб беришни талаб қилади. Бу турдаги хатоликларнинг олдини олиш мақсадида транзакцияни ROLLBACK буйруғи орқали қайтаришга тўғри келади. Сиз транзакцияни бажариш жараёнида фикрингиздан қайтиб, бу жўнатилган сўровлар гуруҳини бажариш керак эмас деган қарорга келиш имкониятига ҳам эгасиз, бу ҳолда ҳам шу буйруқдан фойдаланиш мумкин.

ROLLBACK (қайтариш) буйруғи сервердан транзакция бошланишидан киритилган барча ўзгаришларни рад этишни сўрайди.

Бу жараён қуйидаги кўринишда бўлади:

START TRANSACTION;

```
/* Етарли қолдиқни таъминланган ҳолда биринчи ҳисобдан пуллар ечиб олинсин */
UPDATE account SET avail_balance = avail_balance   500
WHERE account_id = 9988
    AND avail_balance > 500;
 IF <Олдинги ифода орқали битта сатр ўзгартирилди> THEN
     /* Пуллар кейинги ҳисобга ўтказилсин */
     UPDATE account SET avail_balance = avail_balance + 500
        WHERE account_id = 9989;
     IF < Олдинги ифода орқали битта сатр ўзгартирилди > THEN
    /* Ҳаммаси бажарилди, барча ўзгаришлар инобатга олинсин */
    COMMIT;
    ELSE
       /* транзакция бажарилмаса ундаги барча ўзгаришлар рад этилсин */
      ROLLBACK;
    END IF;
ELSE
  /* Ҳисобдаги маблағ етарли эмас ёки ҳисобни янгилашда хатога йўл қўйилди */
   ROLLBACK;
END IF;
```

Бу кодли кўриниш PL/SQL ёки Transact SQL каби процедурага мўлжалланган бирор дастурлаш тилига ўхшаши мумкин, лекин у ҳеч қайси тилни такрорламайди ва у псевдокодда ёзилган.

Бу кодли фрагмент транзакцияни ишга тушириш билан бошланади (START TRANSACTION). Сўнгра ҳисоб рақами идентификатори 9988 бўлган биринчи ҳисобдаги 500 долларни ечиб, ҳисоб рақами идентификатори 9989 бўлган иккинчи ҳисобга ўтказиш сўралади. Агар бу амалга ошса, транзакция инобатга олинади, акс ҳолда транзакцияни қайтариш амалга оширилади, яъни транзакция бошидан киритилган ўзгаришлар рад этилади.

Дастур транзакция ёрдамида беш юз долларни омонат ҳисобида қолишини ёки ҳеч қандай ўғриликлар содир этмай жорий ҳисобга ўтказилишини кафолатлайди. Бунда сервернинг асосий вазифаларидан бири нормал иш ҳолатига қайтишдан олдин сервер ўчирилган ҳолдаги барча тугалланмаган транзакцияларни аниқлаш ва уларни қайтаришдан иборат.

Демак, юқоридаги мисоллардан хулоса қилиб шуни айтиш мумкинки, бир вақтнинг ўзида параллелик ҳолатидаги бир неча SQL-ифодаларни хатоларсиз якунлашда транзакциялар муҳим аҳамиятга эга.

ХУЛОСА

SQL тили ҳозирги вақтда муҳим компьютер технологияси ва кучли бозор фактори сифатида кенг тарқалди. SQL тили МББТ архитектурасининг муҳим аъзоси бўлиб, у ҳозирги кунда юздан ортиқ МББТ ларда ишлаяпти.

Шуни алоҳида таъкидлаш керакки, SQL инструкцияларини ёзиш учун маълумотлар базаси структурасини яхши билиш керак. Қайси ахборот қайси жадвалда сақланади, бир жадвал бошқа жадвал билан қандай боғланган ва маълумотлар сатрларга қандай тақсимланганлигини билмасдан туриб самарали SQL-кодни ёзиб бўлмайди.

Компьютер билан боғлиқ ва компьютер ёрдамида жуда тез амалга ошириш мумкин бўлган шундай масалалар туркуми мавжудки, уларга ҳар куни ва ҳар қадамда дуч келамиз. Бундай масалалар туркуми маълумотлар базаси бўлиб, уларни лойиҳалаш, ҳосил қилиш, маълум бир тизимга келтириш, маълумотларни тўплаш, ташкил этилган базадан керакли маълумотларни қидириб топиш билан шуғулланувчи дастур эса маълумотлар базасини бошқариш тизимларидир.

Бизнинг кундалик турмушимизда учрайдиган нарсаларнинг барчаси қайсидир маълумотлар базасида қайд қилинади. Маълумотлар базаси билан ишлай олиш компьютерда ишлашнинг муҳим омилларидан ҳисобланади. Фойдаланувчи маълумотлар базасидан маълумотларни ўқимоқчи бўлса, у буни МББТ дан SQL ёрдамида сўрайди. МББТ сўровга ишлов беради, талаб қилинган маълумотларни топади ва уни фойдаланувчига узатади.

SQL маълумотлар базасининг энг кенг тарқалган тили ҳисобланади. Сиз иловалар ишлаб чиқасизми, маълумотлар базаси администратори, веб-дизайнер ёки Microsoft Office пакетининг фойдаланувчиси бўласизми, бу муҳим эмас. SQL ни амалий жиҳатдан яхши билиш сизга маълумотлар базаси билан мулоқот қилишга ва керакли сўровларга ундан жавоб олишга имкон беради.

SQL мавзуси хорижий адабиётларда етарлича ёритилган. Уларда SQL тилини тез ва осон ўрганиш учун услубий кўрсатмалар, амалий мисоллар келтирилган. Шу мақсадда қўлланмада мазкур тилни тез ўрганиш учун улардан олинган маълумотлардан фойдаланилди.

Қўлланмада оддий сўровлардан бошлаб аста-секин мураккаб сўровлар, сўров ичидаги сўровларни ташкил қилиш масалаларига ўтилганлиги сизга SQL ни тезда ўзлаштириб олишга ёрдам беради.

Кўпгина МББТлар учун SQL инструкциялари бир хил, айрим ҳолларда фарқ қилиши мумкин, бунинг учун http ://forta.com/books/0672336073/ сайтидан фойдаланишингиз тавсия этилади, унда бу фарқлар изоҳ сифатида келтирилган ва асосий эътибор Apache OpenOffice Base, IBM DB2, MS Access, MS SQL Server, MariaDB, MySQL, Oracle, PostgreSQL, SQLite каби МББТ ларга қаратилган.

Назария бу яхши нарса, лекин унинг тушунчаларини амалиётда қўллаш учун SQL тилининг яна бир неча қирраларини ўрганишга тўғри келади.

Қўлланмада келтирилган жадвалларни тўлиқ деб бўлмайди. Буюртмаларни рўйҳатга олиш реал тизими бошқа маълумотлар тўпламини, масалан, тўлов реквизитлари, етказиб бериладиган маҳсулотларнинг назорат номерлари ва шу каби бошқа кўпгина маълумотларни ҳам ўзида сақлаши мумкин.

Лекин, шунга қарамасдан бу жадваллар орқали маълумотлар базасини структуралашни кўргазмали равишда кўрсатишга эришилди. Олинган билимларни фойдаланувчи ўз шахсий базасини яратишда қўллаши мумкин.

Мазкур қўлланмада SQL тилининг сўровларни ҳосил қилишга мўлжалланган SELECT инструкцияси синтаксиси ва унда қатнашадиган барча калит сўзларни қўллаш юзасидан услубий кўрсатмалар берилган.

Қўлланма хорижий адабиётлардан олинган маълумотлар билан бойитилган ва кўпроқ амалий жиҳатдан ёритилган. Ундан барча таълим муассасалари ўқитувчилари, талабалари ва мустақил изланувчилар фойдаланишлари мумкин деб ҳисоблаймиз.

Адабиётлар

1.Э.С.Бабаджанов.Маълумотлар базалари фанидан маърузалар матни.ТАТУ Нукус филиали, 84 б.

2. С.С.Ғуломов, Б.А.Бегалов, Т.И.Сарсатская. Маълумотлар ва билимлар базаси. Ўқув қўлланма – Тошкент.: ТДИУ, 2005 , 164 б.

3.Э.С.Бабаджанов.SQL тили ва унинг таркиби SQL тили тарихи. https://referat.arxiv.uz/index.php?do=files&op=download&fileid=63504

4. Б.А.Бегалов. Введение в базы данных. Электронное учебное пособие. Программисты: А.Бобожонов, У.Муслимов. Ташкент. ТГЭУ, 2004

5. Бен Форта.Освой самостоятельно SQL за 10 минут, 4-е изд.: Пер. с англ.—М .: ООО “И.Д. Вильямс”, 2014. — 288 с .

6. Алан Бьюли. Изучаем SQL. Вводный курс для разработчиков и администраторов БД. Санкт-Петербург–Москва, Символ ®, 2007

7. Базы данных. Курс лекций по дисциплине: Методические указания по дисциплине базы данных для бакалаврской подготовки студентов по направлению 230100 «Информатика и вычислительная техника»/ А.А. Иптышев, Д.И. Морозов, А.А Городилов, М.А. Иптышев, Красноярск.: СФУ, 2007, 146с

8. Д.А. Харитонюк. Базы данных. Курс лекций с примерами и заданиями для самостоятельной работы студентов специальности «Автоматизированные системы обработки информации и управления (по отраслям)» / Негосударственное образовательное учреждение, Октябрьский экономический техникум, 2008

9. Г.Д.Фролов, Э.И.Кузнецов. Элементы информатики.М., Высшая школа, 1989, 304 с.

10. http ://forta.com/books/0672336073/

11. http ://www.williamspublishing.com/Books/978-5-8459-1858-1.html

12. Кузнецов С.Д. Введение в системы управления базами данных //СУБД. - 1995. - №1,2,3,4, 1996. - №1,2,3,4,5.

13. Кузнецов С.Д. Стандарты языка реляционных баз данных SQL: краткий обзор //СУБД. - 1996. - №2. - С.6-36.

14. Справка по SQL(DML): Операторы UPDATE и DELETE. http://www.sql-ex.ru/help/select12.php?Lang=0

15. SQL | Заявление о ситуации. SQL | CaseStatement - GeeksforGeeks https://www.geeksforgeeks.org/sql-case-statement/

16. SQL CASE worksinpracticallyall SQL-Databases - Modern SQL https://modern-sql.com/feature/case

17. М.В.Петрушин. Справочное руководство по MySQL 4.0. Изд.65, 2006

18. Кевин Янк. Простой способ создать сайт на основе базы данных. PHP и MySQL. От новичка к профессионалу. М., ЭКСМО, 5-издание, 2013, 384с.

19. С.Винкоп. Использование Microsoft SQL Server 7.0. Специальное издание.: пер. с англ. К.; М.; СПб.: Издательский дом «Вильямс», 1999

20.А.В.Фролов, Г.В.Фролов. Базы данных в Интернете. Практическое руководство по созданию Web-приложений с базами данных. Изд. 2-ое, испр. М.: Издательско-торговый дом «Русская Редакция», 2000, 448с.

21. Леон Аткинсон. MySQL. Библиотека профессионала.: Пер. с англ. — М.: Издательский дом "Вильямс", 2002, 624 с.

22.Аллен Г. Тейлор. SQL для чайников. 7-е издание. www.dialektika.com

23. Люк Веллинг, Лора Томсон. Разработка веб-приложений с помощью PHP и MYSQL. Библиотека разработчика. 4-е издание. www.williamspublishing.com

24. Кристиан Уэнц. PHP и MySQL Карманный справочник www.williamspublishing.com

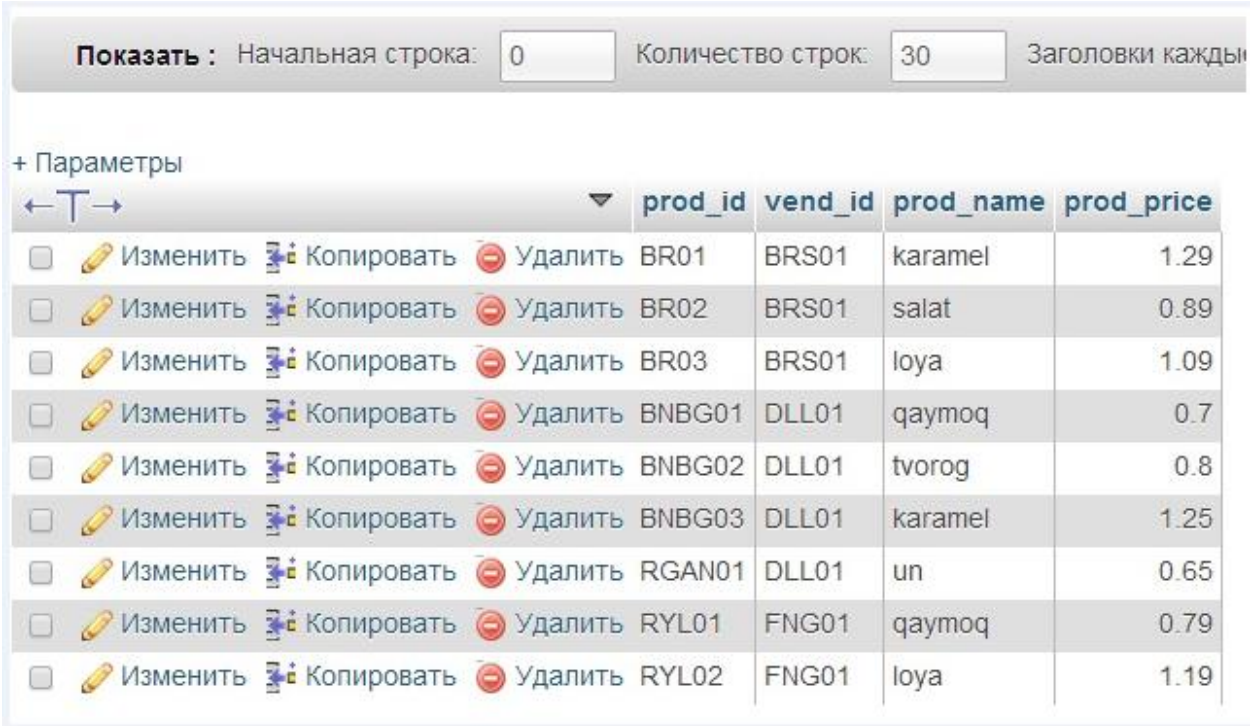
25. Т.Коннолли, К.Бегг, А.Страчан. Базы данных: проектирование, реализация и сопровождение. Теория и практика. 2-е издание. www.williamspublishing.com

Илова 1

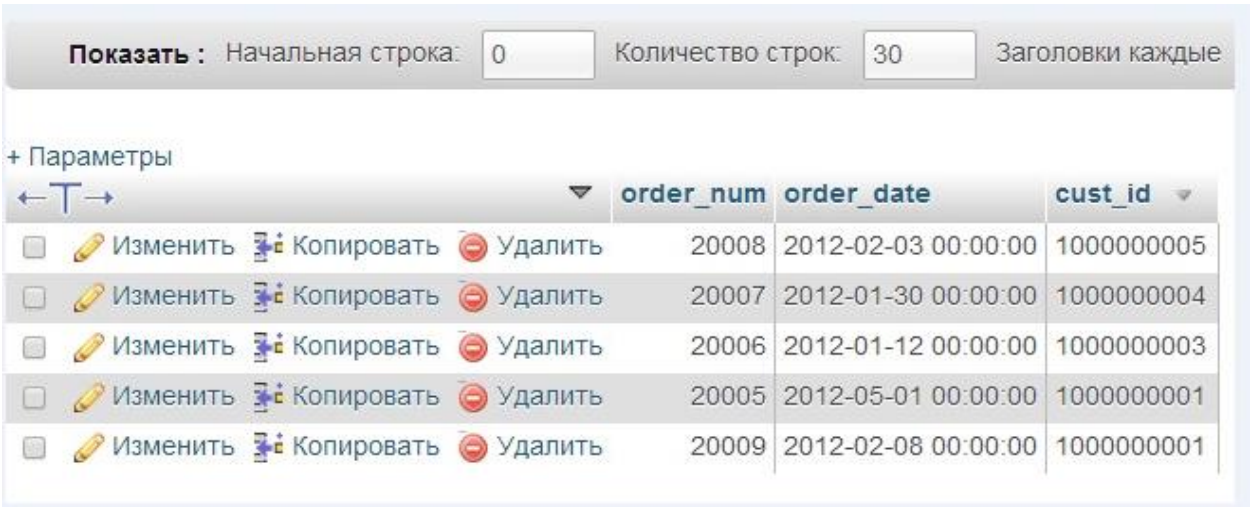
OrderItems жадвали

order_num	order_item	prod_id	quantity	item_price
20005	1	BR01	100	5.49
20005	2	BR03	100	10.99
20006	1	BR01	20	5.99
20006	2	BR02	10	8.99
20006	3	BR03	10	11.99
20007	1	BR03	50	11.49
20007	2	BNBG01	100	2.99
20007	3	BNBG02	100	2.99
20007	4	BNBG03	100	2.99
20007	5	RGAN01	50	4.49
20008	1	RGAN01	5	4.99
20008	2	BR03	5	11.99
20008	3	BNBG01	10	3.49
20008	4	BNBG02	10	3.49
20008	5	BNBG03	10	3.49
20009	1	BNBG01	250	2.49
20009	2	BNBG02	250	2.49
20009	3	BNBG03	250	2.49

Products жадвали

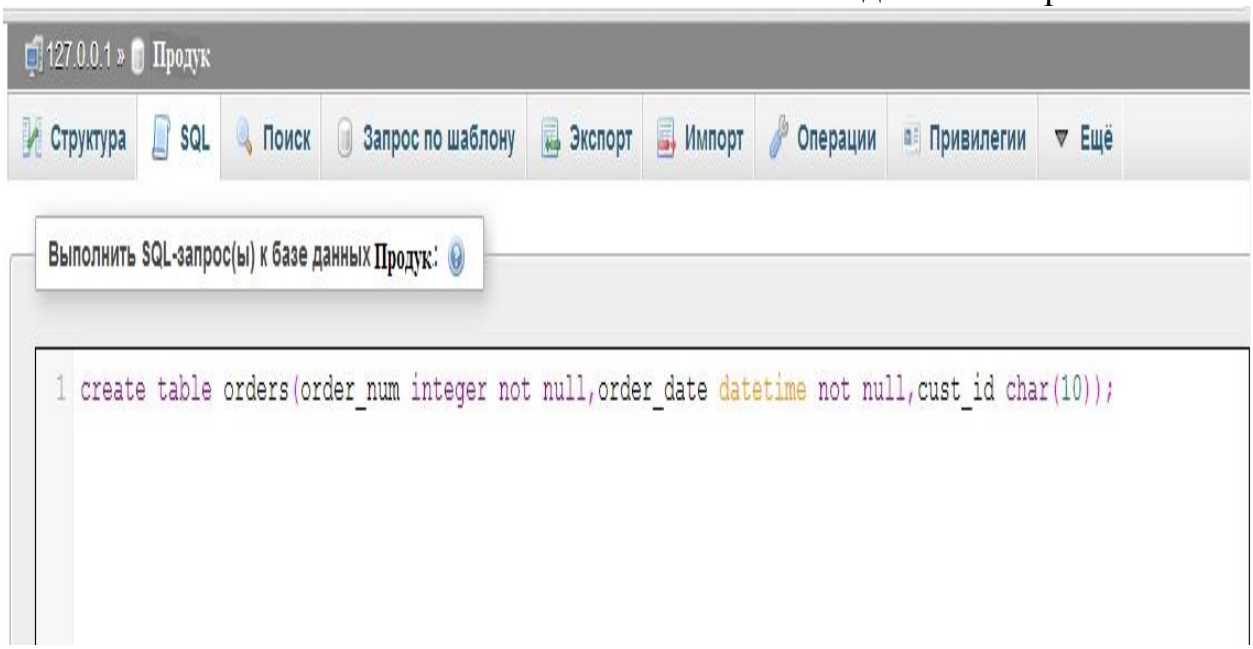


Orsers жадвали

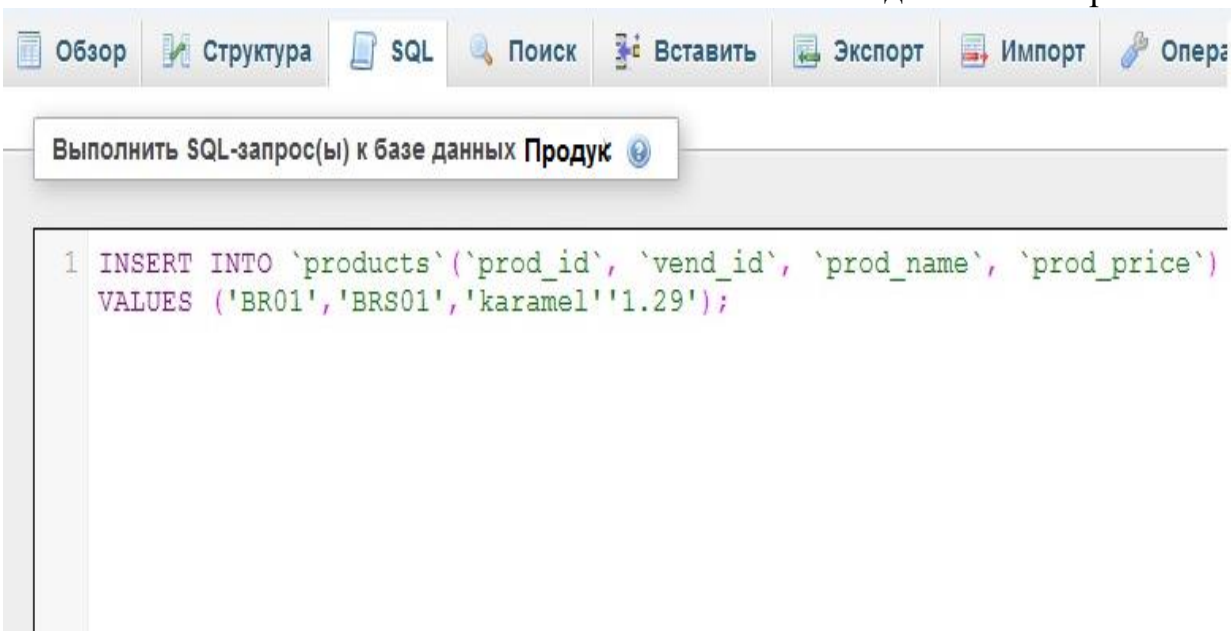


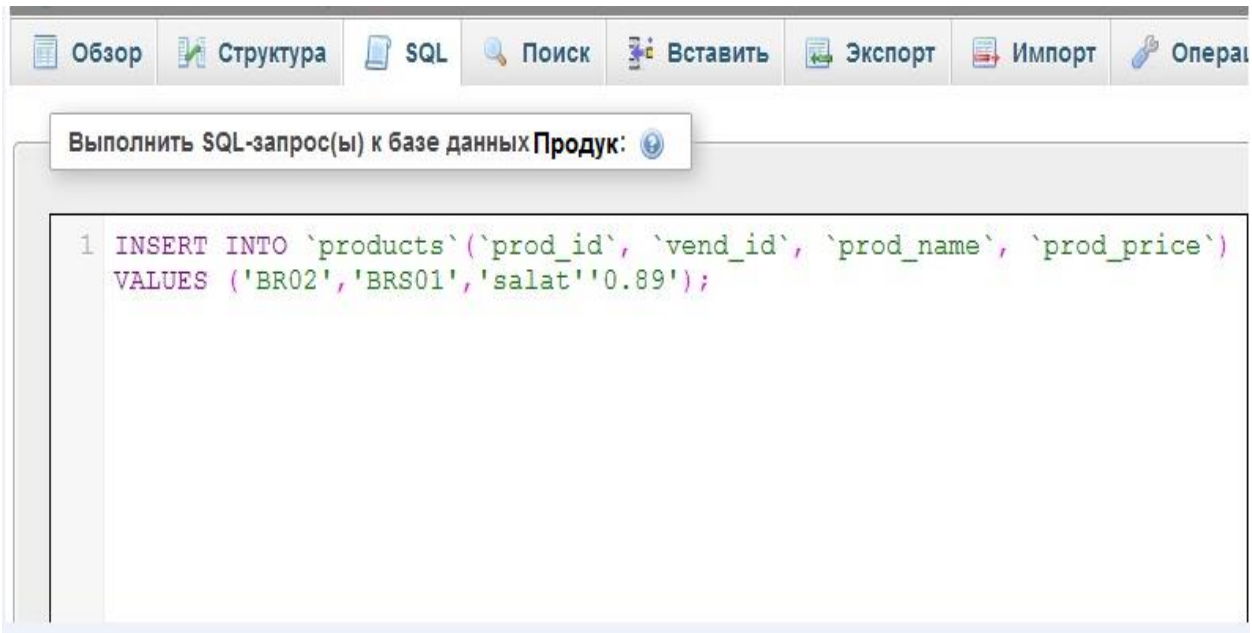
Products жадвалини яратиш

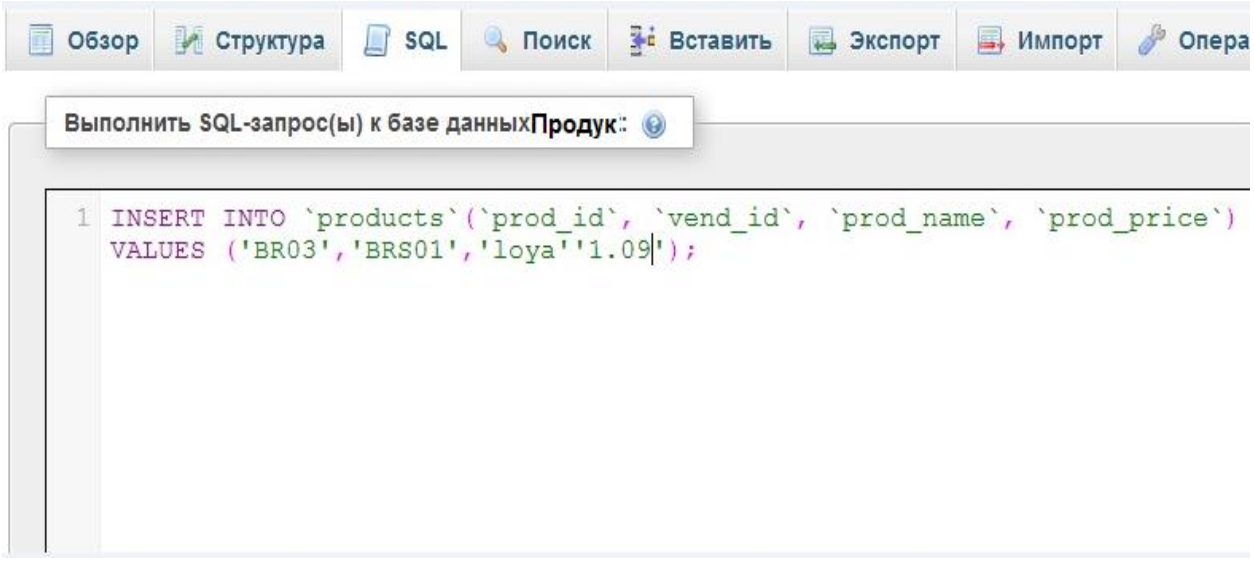
Orsers жадвалини яратиш



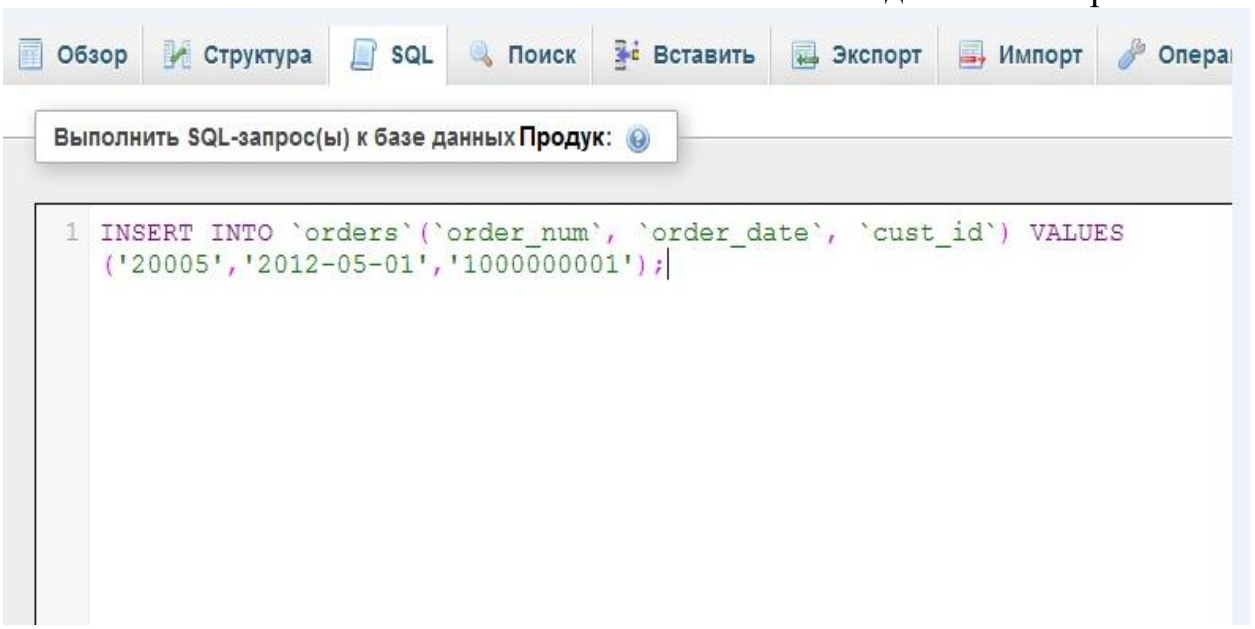
Products жадвалини киритиш

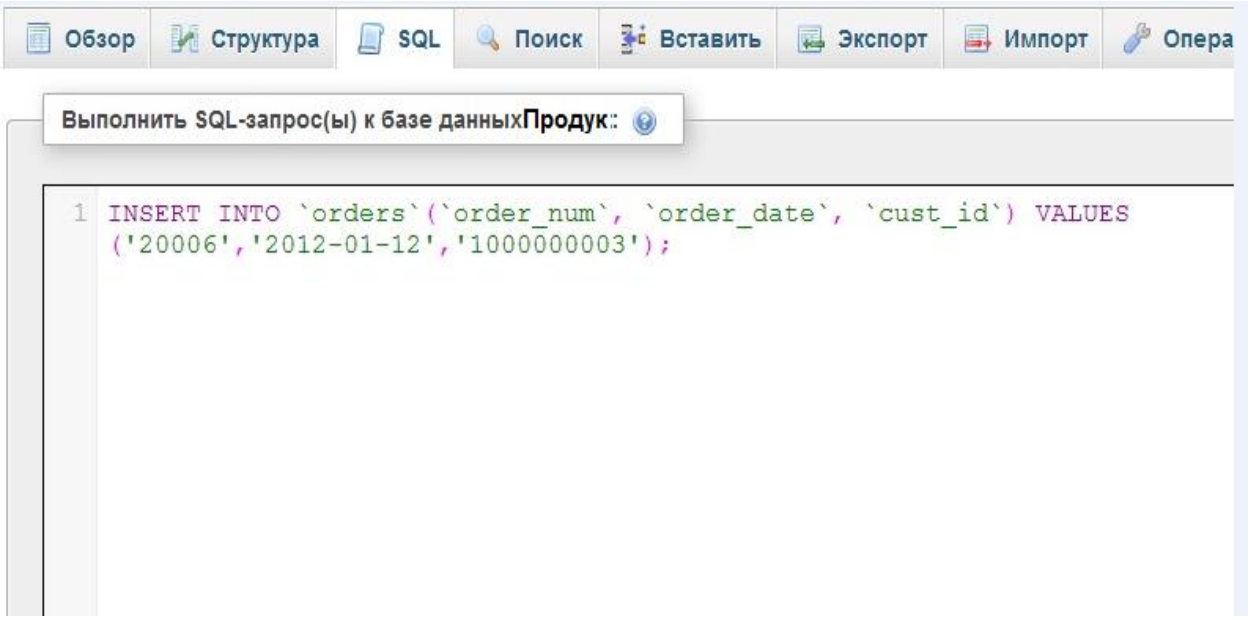


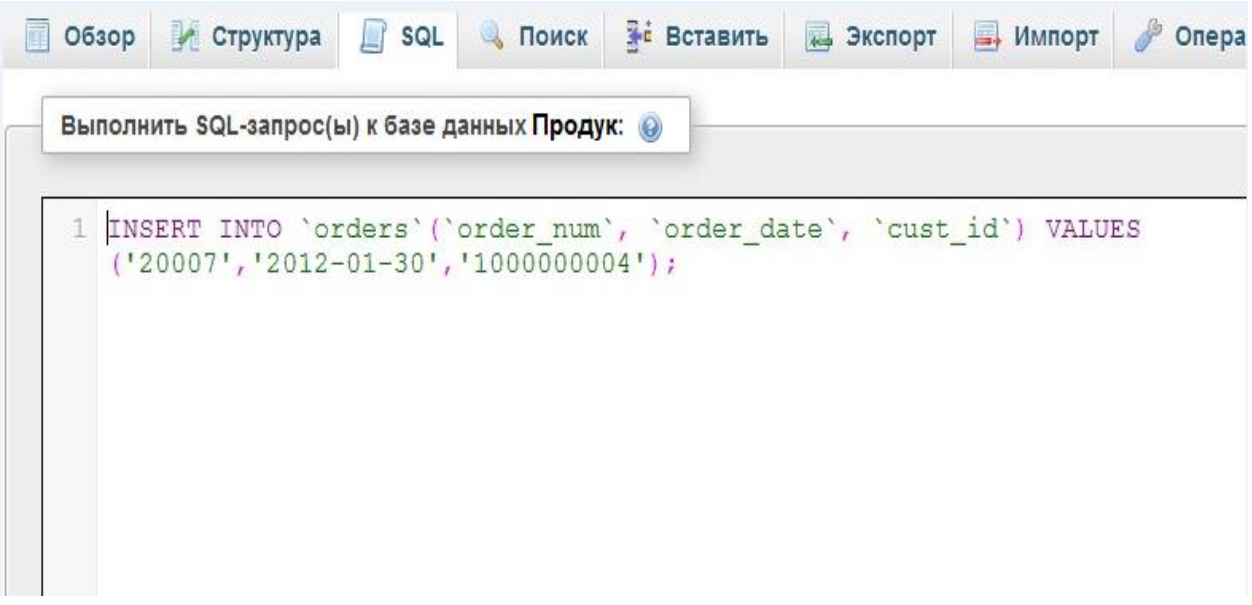




Orsers жадвалини киритиш







Customers жадвали

cust_id	cust_name	cust_address	cust_city	cust_ state	cust_ zip	cust_ country	cust_contact	cust_email
1000000001	Village Toys	200 Maple Lane	Detroit	MI	44444	USA	John Smith	sales@villagetoys.com
1000000002	Kids Place	333 South Lake Drive	Columbus	OH	43333	USA	Michelle Green	
1000000003	Fun4All	1 Sunny Place	Muncie	IN	42222	USA	Jim Jones	jjones@fun4all.com
1000000004	Fun4All	829 Riverside Drive	Phoenix	AZ	88888	USA	Denise L. Stephens	dstephens@fun4all.com
1000000005	The Toy Store	4545 53rd Street	Chicago	IL	54545	USA	Kim Howard	

Vendors жадвали

vend_id	vend_name	vend_address	vend_city	vend_state	vend_zip	vend_country
BRS01	Bears R Us	123 Main Street	Bear Town	MI	44444	USA
BRE02	Bear Emporium	500 Park Street	Anytown	OH	44333	USA
DLL01	Doll House Inc.	555 High Street	Dollsville	CA	99999	USA
FRB01	Furball Inc.	1000 5th Avenue	New York	NY	11111	USA
FNG01	Fun and Games	42 Galaxy Road	London	NULL	N16 6PS	England
JTS01	Jouets et ours	1 Rue Amusement	Paris	NULL	45678	France

Илова 2

Педагогика институти ахборот моделининг реляцион концептуал схемаси