4-ma'ruza: Multimedia MB so‘rovlari

Reja:

1. SQL tili haqida tushuncha
2. SQL tilida ma‘lumotlar turlari
3. Jadvallar yaratish va ularga cheklovlar kiritish
4. Maydon qiymatlarini tekshirish (CHECK cheklovi)
5. Oddiy SELECT so‘rovlar va mantiqiy operatorlar
6. SQL turlari va strukturasi

SQL tili haqida tushuncha

SQL (Structured Query Language) strukturalashgan so‘rov tili ma’nosini bildirib, u relyatsion ma‘lumotlar bazasi bilan ishlash imkonini yaratib beradigan tildir.

Ma’lumki, relyatsion modelning tarixi (va bilvosita SQL tarixi ham) 1970 yil Ye,F.Koddni (bu paytda u IBM korporatsiyasining San Xosedagi tadqiqot markazida ishlagan) maqolasi chiqqan davrdan boshlanadi. 1974 yil shu laboratoriyada ishlovchi D. Chemberlen "Structured English Query Language" yoki SEQUEL deb nomlangan tilni e’lon qiladi. 1976 yil bu tilning qayta ishlangan SEQUEL/2 versiyasi yaratildi va u rasmiy ravishda SQL deb atalgan. Xozirgi kunda SQL qisqartmasini ba’zilar "sikvel" deb talffuz etadi. Biroq rasmiy ravishda u "es-kyu-el" deb o‘qilishi kerak.

SQL tili relyatsion algebra paydo bo‘lgandan keyin paydo bo‘ldi va uning birinchi prototipi IBM Research kompaniyasi tomonidan 70 yillar oxirida yaratilgan. Bu til birinchi IBM System R nomli MBBT tarkibiga kiritilgan. Keyinchalik bu til ko‘pgina tijorat MBBT tarkibida qo‘llanilgan va keng tarqalganligi sababli vaqt o‘tishi bilan relyatsion MBBT larda ma‘lumotlar ustida amallar bajaruvchi tillarning norasmiy standarti bo‘lib qoldi. SQL tilining birinchi ramiy standarti 1989 yil qabul qilingan. Ko‘pgina MBBT lar ushbu standartni qo‘llab – quvvatlaydi. Biroq ma‘lumotlar bazasi bilan bog‘liq axborot texnologiyalarining rivojlanishi va ba’zi talablarning paydo bo‘lishi birinchi SQL standartini qayta ishlash va kengaytirishni taqoza etdi.

1992 yil oxirida SQL tilining yangi xalqaro standarti (SQL/92 yoki SQL2) qabul qilindi unda ham ba’zi kamchiliklar aniqlangan, biroq shunga qaramasdan SQL/89 ga nisbatan aniq va to‘liqroq xisoblanadi. Xozirgi paytda ko‘pgina MBBT ishlab chiqaruvchilar o‘z maxsulotlarini SQL2 standartini qanoatlantiradigan qilib o‘zgartirdilar.

1999 yil SQL3 deb atalgan yangi standart paydo bo‘ldi. Agar SQL1 va SQL2 standartlari biri –biridan miqdor jixati bilan farq qilgan bo‘lsa, SQL3 standarti sifat jixatlari bilan farqlanadi. SQL3 ga murakkab strukturaga ega ma‘lumotlar tipini ishlatish imkonini beradigan yangi ma‘lumotlar tipi kiritilgan. Bu tipni ob’ektga mo‘ljallanganlik darajasi yuqori xisoblanadi. SQL tilini tula qonli an’anaviy dasturlash tillari tarkibiga kiritib bo‘lmaydi. Chunki unda dastur bajarilishini boshqaruvchi va boshqa ko‘pgina an’anaviy operattorlar yo‘q. Unda faqat ma‘lumotlar bazasida saqlanayotgan ma‘lumotilarga murojaat qiluvchi operatorlar mavjud.

SQL tili foydalanuvchi relyatsion ma‘lumotlar bazasi bilan muloqat qilishi uchun mo‘ljallangan bo‘lib, quyidagi 3 ta qismdan iborat:

• DDL (Data Definition Language) – ma‘lumotlarni aniqlash tili. Ma‘lumotlar bazasini (jadvallarini, indekslarini va x.k.) yaratish va uning sxemasini taxrirlash uchun mo‘ljallangan..

• DCL (Data Control Language) – ma‘lumotlarni boshqarish tili. Foydalanuvchilarning ma‘lumotlar bazasi ob’ektlariga murojatini chegaralash operatorlaridan iborat.

• DML (Data Manipulation Language) – ma‘lumotlarni qayta ishlash tili. Ma‘lumotlar bazasi jadvallariga o‘zgartirishlar kiritish uchun mo‘ljallangan.

Ma‘lumotlar bazasi bilan ishlovchi ixtiyoriy til foydalanuvchiga quyidagi imkoniyatlarni yaratishi lozim:
• strukturasini to‘la tavsiflagan xolda ma‘lumotlar bazasini va jadvallarini yaratish;
• ma‘lumotlar ustida manipulyatsiya amallarini bajarish, masalan, jadvallardan ma‘lumotlarni kiritish, taxrirlash, va o‘chirish;
• oddiy va murakkab so‘rovlarni bajarish.
Bundan tashqari, ma‘lumotlar bazasi bilan ishlovchi til yuqoridagi amallarni bajarish uchun foydalanuvchilardan kam urinishlarini talab qilishi, hamda komandalarining sintaksisi va tuzilishi o‘zganish uchun ososn va tushunarli bo‘lishi kerak. Nixoyat bu til universal bo‘lishi kerak. Bu bir MBBT dan boshqasiga o‘tganda komandalarni bir xil strukturasi va sintaksisidan foydalanishni ta’xminlaydi. SQL tili bu talablarni barchasini qanoatlantiradi.

SQL tilida ma‘lumotlar turlari

SQL tilida ma‘lumotlarning quyidagi asosiy turlari ishlatilib, ularning formatlari har xil MBBT lar uchun farq qilishi mumkin:

Jadvallar yaratish va ularga cheklovlar kiritish

Jadvallarni yaratish. Jadvallar CREATE TABLE buyrug‘i bilan yaratiladi. Bu buyruq qatorlarsiz bo‘sh jadval yaratadi. U jadval nomini, ma’lum tartibda ko‘rsatilgan ustunlar nomlari ketma – ketligi, ma‘lumotlar turlari va ustunlar o‘lchovini aniqlaydi.
CREATE TABLE buyrug‘ining umumiy yozilishi:
CREATE TABLE
([()],
[(Jadval yaratishda va ular ustida ish yuritishda quyidagi 2 ta jadvaldan iborat ma‘lumotlar bazasini misol sifatida qaraymiz.
Sotuvchilar (Salepeople):
SNum – xar bir sotuvchi unikal nomeri,
SName – sotuvchi nomi,
City – sotuvchi adresi (shaxri),
Comm – sotuvchilarning o‘nli shakldagi komission foydasi.

Buyurtmachilar (Customers):

CNum – xar bir buyurtmachi unikal nomeri;
CName – buyurtmachi nomi;
City – buyurtmachi adresi (shaxri);
Rating – buyurtmachining boshqalardan ustunlik darajasini ko‘rsatuvchi kod;
SNum – shu buyurtmachiga tayinlangan sotuvchi nomeri.
Misol uchun sotuvchilar jadvalini yaratish:
CREATE TABLE Salepeople
(SNum integer, SName char(10), City char(10), Comm decimal);

Cheklovlarni kiritish. Jadvalni yaratayotganda (yoki uni o‘zgartirayotganda), maydonlarga kiritilayotgan qiymatlarga cheklovlar o‘rnatish mumkin. Bu holda SQL cheklovlarga to‘g‘ri kelmaydigan hamma qiymatlarni rad etadi.
Maydonga bo‘sh (NULL) qiymatlar kiritilishi oldini olish uchun CREATE TABLE buyrug‘ida NOT NULL cheklovi ishlatiladi. Masalan, birlamchi kalitlar xech qachon bo‘sh bo‘lmasliklari kerak, shuning uchun Salepeople jadvalini quyidagicha yaratish mumkin:
CREATE TABLE Salepeople
(Snum integer NOT NULL,
Sname char(10), city char(10), comm decimal);
Ko‘p hollarda ustunga kiritilgan qiymatlar bir biridan farq qilishi kerak bo‘ladi. Agar ustunga UNIQUE cheklovi o‘rnatilsa, unda ustunga qiymat kiritishga urinish rad etiladi. Bu cheklov bo‘sh bo‘lmaydigan (NOT NULL) deb e’lon qilingan maydonlarga qo‘llaniladi. Masalan:
CREATE TABLE Salepeople
(SNum integer NOT NULL UNIQUE,

Sname char(10), city char(10), comm decimal);
Jadval cheklovi UNIQUE maydonlar guruhiga ham o‘rnatilishi mumkin. Bu bir necha maydonlar qiymatlari kombinatsiyasi unikalligini ta’minlaydi.

Maydon qiymatlarini tekshirish (CHECK cheklovi)

CHECK cheklovi jadvalga kiritilayotgan ma‘lumot qabul qilinishidan oldin mos kelishi lozim bo‘lgan shart kiritishga imkon beradi. CHECK cheklovi CHECK kalit so‘zi ko‘rsatilgan maydondan foydalanuvchi shartli ifodadan iboratdir. Misol uchun Salepeople jadvali Comm ustuniga kiritilayotgan qiymat 1 dan kichik bo‘lsin.

```
CREATE TABLE Salepeople
( SNum integer NOT NULL PRIMARY KEY,
SName char(10) NOT NULL UNIQUE,
City char(10),
Comm decimal CHECK ( Comm < 1 ));
```

CHECK cheklovidan maydonga ma’lum qiymatlarini kiritishdan himoya qilib, xatolar oldini olish uchun foydalanish mumkin. Masalan, mahsulotni sotish shaxobchalariga ega bo‘lgan shaharlar faqat London, Barselona, San Xose va Nyu York bo‘lsin.

```
CREATE TABLE Salepeople
(SNum integer NOT NULL PRIMARY KEY,
SName char(10) NOT NULL UNIQUE,
City char(10) CHECK (City IN ('London','New York','San Jose', 'Barselona')), Comm decimal CHECK ( Comm < 1 ));
```

CHECK jadval cheklovi sifatida kelishi mumkin. Bu shartga bir necha maydon kiritishga imkon beradi. Masalan:

```
CREATE TABLE Salepeople
( SNum integer NOT NULL PRIMARY KEY,
SName char(10) NOT NULL UNIQUE,
City char(10),
Comm decimal,
CHECK (Somm < .15 OR City = 'Barcelona'));
```

Ko‘zda tutilgan qiymatlarni o‘rnatish

Biror bir maydon uchun qiymat ko‘rsatmagan holda jadvalga satr qo‘shilish kerak bo‘lsa, SQL bunday maydonga kiritish uchun ko‘zda tutilgan qiymatga ega bo‘lishi kerak, aks holda buyruq rad etiladi. Eng umumiy ko‘zda tutilgan qiymat NULL qiymatdir. CREATE TABLE buyrug‘ida ko‘zda tutilgan qiymat DEFAULT operatori orqali, ustun cheklovi sifatida ko‘rsatiladi. Masalan:

```
CREATE TABLE Salepeople
( SNum integer NOT NULL PRIMARY KEY,
SName char(10) NOT NULL UNIQUE,
City char(10) DEFAULT 'New York',
Comm decimal CHECK ( Comm < 1 ));
```

Maydonlar qiymatlarini kiritish, o‘chirish va o‘zgartirish

Jadvallarni o‘chirish. Faqat bo‘sh jadvalni o‘chirish mumkin. Jadvalni o‘chirish buyrug‘i quyidagi ko‘rinishga ega:
DROP TABLE ; Masalan: DROP TABLE Salepeople;
Jadvalni yaratilgandan so‘ng o‘zgartirish. Jadvalni o‘zgartirish uchun ALTER TABLE buyrug‘idan foydalaniladi. Bu buyruqda jadvalga yangi ustunlar qo‘shish, ustunlarni o‘chirish, ustunlar kattaligini o‘zgartirish, hamda cheklovlarni qo‘shish va olib tashlash imkoniyatlariga ega.
Jadvalga ustun qo‘shish buyrug‘i:

```
ALTER TABLE ADD
;
```

Masalan:

```
ALTER TABLE Salepeople ADD Phone CHAR(7);
```

Qiymatlarni kiritish. Hamma satrlar SQLda INSERT buyrug‘i yordamida kiritiladi. INSERT quyidagi formatga ega:
INSERT INTO

[(column [,column] ...)]
VALUES ([,] ...);
Masalan, sotuvchilar jadvaliga yangi satr kiritish uchun quyidagi buyruqdan foydalanish mumkin:
INSERT INTO Salepeople VALUES (11, 'Peel', 'London', .12);
Ustun nomlarini ko‘rsatish ham mumkin, masalan:
INSERT INTO Salepeople (Sname, Comm, SNum)
VALUES ('Peel', .12, 11);
Bu yerda e’tibor berilsa City ustuni tashlab yuborilgan, chunki unga ko‘zda tutilgan qiymat kiritiladi.
Satrlarni o‘chirish. Satrlarni jadvaldan DELETE buyrug‘i bilan o‘chirish mumkin. U alohida qiymatlarni emas faqat satrlarni o‘chiradi. DELETE quyidagi formatga ega:
DELETE FROM
[WHERE search-condition];
Masalan, sotuvchilar jadvalidagi hamma satrlarni o‘chirish uchun, quyidagi shartni kiritish mumkin: DELETE FROM Salepeople;
Ma’lum satrlarni o‘chirish uchun shartlardan foydalaniladi. Masalan, jadvaldan Axelrod sotuvchini o‘chirish uchun uning nomerini shartda berish kerak:
DELETE FROM Salepeople WHERE SNum = 13;
Maydon qiymatlarini o‘zgartirish. O‘zgartirish UPDATE buyrug‘i yordamida bajariladi. Bu buyruqda UPDATE ifodasidan so‘ng jadval nomi va SET ifodasidan so‘ng ma’lum ustun uchun o‘zgartirish ko‘rsatiladi. UPDATE ikki formatga ega. Ulardan birinchisi:
UPDATE

SET column = expression [, column = expression] ...
[WHERE search-condition]
bu yerda expression - bu ustun | ifoda | konstanta | o‘zgaruvchi.
Ikkinchi variant:
UPDATE

SET column = expression, ...
[FROM table-list]
[WHERE search-condition]
Masalan, hamma buyurtmachilar bahosini 200 ga o‘zgartirish mumkin:
UPDATE Customers SET Rating = 200;
Ma’lum satrlarni o‘zgartirish uchun DELETE dagi kabi shartlardan foydalanish kerak. Masalan, Peel (SNum=11) sotuvchining hamma buyurtmachilari uchun bir xil o‘zgartirish quyidagicha kiritiladi:
UPDATE Customers SET Rating = 200 WHERE SNum = 11;
SET vergul bilan ajratilgan ixtiyoriy sondagi ustunlarga qiymat tayinlashi mumkin. Masalan:
UPDATE Salepeople SET SName ='Gibson', City='Boston', Comm=.10
WHERE SNum = 14;
UPDATE buyrug‘ining SET jumlasida ifodalarni ham ishlatish mumkin. Masalan: UPDATE Salepeople SET Comm = Comm * 2;

Oddiy SELECT so‘rovlar va mantiqiy operatorlar

SELECT operatori MB jadvallaridan natijaviy to‘plam olish uchun mo‘ljallangandir. SELECT operatori yordamida MBga so‘rov beriladi va u foydalanuvchiga ma‘lumotlarning natijaviy to‘plamini qaytaradi. Bu ma‘lumotlar jadval shaklida qaytariladi. Bu jadval keyingi SELECT operatori tomonidan yana qayta ishlanishi ham mumkin.
SELECT operatori SQL quyidagi ko‘rinishga ega:
SELECT [ALL]
FROM jadval
WHERE izlash sharti
GROUP BY ustunlar
HAVING izlash sharti
ORDER BY tartiblash spesifikatori

Masalan, OFFICES jadvalidagi hamma yozuvlarni qaytaruvchi sodda so‘rov quyidagicha yoziladi. SELECT * FROM OFFICES
Misol: Hamma xizmatchilarning nomlari, ofislari va ishga olish sanalari ro‘yxatini hosil qilish.
SELECT NAME, REP_OFFICE, HIRE_DATE FROM SALESREPS
SELECT operatori WHERE sharti berilgan shart asosida kerakli ma‘lumotlarni qaytarish uchun xizmat qiladi. Masalan, sotuvlarda haqiqiy hajmi rejadan oshgan ofislarni ko‘rsatish kerak.
SELECT CITY, SALES, TARGET FROM OFFICES
WHERE SALES > TARGET
Nomeri 105 ga teng bo‘lgan xizmatchi nomi, haqiqiy va rejadagi sotuvlar hajmini ko‘rsatish:
SELECT SALES, NAME, QUOTA FROM SALESREPS
WHERE EMPL_NUM = 105
Agar izlash sharti TRUE bo‘lsa qator natijaviy to‘plamga qo‘shiladi, agar izlash sharti FALSE bo‘lsa, qator natijaviy to‘plamga qo‘shilmaydi, agar NULL bo‘lsa ham natijaviy to‘plamdan chiqariladi. O‘z ma’nosiga ko‘ra WHARE, kerakli yozuvlarni qoldiruvchi filtr sifatida ishlatiladi.

Mantiqiy operatorlar

BETWEEN va IN operatorlari. BETWEEN operatori - bu qiymatlar diapazoniga tegishlilikni tekshirishdir. Misol: Narxi har xil diapazonga mos keluvchi buyurtmalarni topish.
SELECT ORDER_NUM, AMOUNT FROM ORDERS
WHERE AMOUNT BETWEEN 20.000 AND 29.999
NOT ifodasi shartni teskarisiga o‘giradi, yani tegishli emas ma’nosini bildiradi. NOT ifodasi yordamida berilgan diapazonga tegishlilikni tekshirish mumkin, masalan: sotuvlar haqiqiy hajmlari rejaning 80 dan 120 protsentgacha bo‘lgan oraliqqa tushmaydigan xizmatchilar ro‘yxatini chiqarish.
SELECT NAME, SALES, QUOTA FROM SALESREPS
WHERE SALES NOT BETWEEN (0.8 * QUOTA) AND (1.2 * QUOTA)
IN operatori to‘plamga tegishlilikni tekshiradi. Masalan, to‘rtta aniq xizmatchilar tomonidan olingan hamma buyurtmalarni aniqlash.
SELECT ORDER_NUM, REP, AMOUNT FROM ORDERS
WHERE REP IN (107, 109, 101, 103)
NOT IN yordamida diapazonga "tegishli emaslikni " tekshirish mumkin.
LIKE operatori. Quyidagicha '%' shablonli LIKE operatorini qaraymiz:
SELECT COMPANY, CREDIT_LIMIT FROM CUSTOMERS
WHERE COMPANY LIKE '%n'
Bu xolda LIKE '%n' operatori 'n' harfiga tugaydigan hamma yozuvlarni ko‘rsatadi, agar % shabloni birinchi kelsa:
SELECT COMPANY, CREDIT_LIMIT FROM CUSTOMERS
WHERE COMPANY LIKE '%gan'
Agar faqat bitta simvol ixtiyoriy bo‘lsa '_' shabloni qo‘llaniladi. Masalan:
SELECT COMPANY, CREDIT_LIMIT FROM CUSTOMERS
WHERE COMPANY LIKE 'Ap_lsin'
Yozuvlarni tartiblash, ORDER BY ifodasi. SELECT operatori tarkibida natijaviy yozuvlarni tartiblangan holda taqdim etish uchun ORDER BY ifodai ko‘zda tutilgan. Masalan, agar o‘quvchilar ro‘yxatini alfavit tartibida yoki tovarlar narxini kamayish tartibida chiqarish zarur bo‘lsa, u holda bu ifodadan foydalanish kerak bo‘ladi.
Quyidagi misolni ko‘ramiz: Har bir offis uchun sotuvlar haqiqiy hajmlarini regionlar nomlari, har bir regionda esa shaharlar nomlari bo‘yicha alfavit tartibida chiqarish.
SELECT CITY, REGION, SALES FROM OFFICES
ORDER BY REGION, CITY
Masalan: Sotuvlari haqiqiy xajmlari kamayish tartibida bo‘lgan offislar ro‘yxatini chiqarish.
SELECT CITY, REGION, SALES FROM OFFICES
ORDER BY SALES DESC
Sotuvlar hajmlarini DESC predikatini qo‘llab kamayish tartibida chiqaramiz. O‘sish tartibida chiqarish uchun ASC predikati qo‘‘llanadi.Bu predikat ko‘zda tutilgan bo‘lib, uni ko‘rsatish shart emas.

Bir necha jadvallar bilan ishlash

Jadvallarni jamlashtirish. Jamlashtirish relyatsion ma‘lumotlar bazasi operatsiyalaridan biri bo‘lib, jadvallar orasidagi aloqani belgilaydi va ulardan ma‘lumotni bitta buyruq yordamida ajratishga imkon beradi. Jamlashda jadvallar FROM buyrug‘idan so‘ng ro‘yxat sifatida tasvirlanadi. So‘rov predikati ixtiyoriy jadval ixtiyoriy ustuniga tegishli bo‘lishi mumkin. Jamlashning eng soddasi bu dekart ko‘paytmasidir, uni quyidagicha bajarish mumkin:

SELECT Customers.*, Salepeople.* FROM Salepeople, Customers;

Lekin bu yerda hosil bo‘lgan jadval keraksiz ma‘lumotlarga ega. Keraksiz satrlarni olib tashlash uchun WHERE jumlasidan foydalaniladi.

Masalan: berilgan shahardagi sotuvchilar va buyurtmachilar ixtiyoriy kombinatsiyasini ko‘rish uchun quyidagini kiritish lozim:

SELECT Customers.CName, Salepeople.SName, Salepeople.City
FROM Salepeople, Customers
WHERE Salepeople.City = Customers.City;

Jamlashda SQL bir necha jadval satrlari kombinatsiyasini predikatlar bo‘yicha solishtirishdir. Misol: har bir sotuvchiga mos keluvchi buyurtmachilar ro‘yxatini chiqarish:

SELECT Customers.CName, Salepeople.SName
FROM Customers, Salepeople
WHERE Salepeople.SNum = Customers.SNum;

Sodda joylashtirilgan ostki so‘rovlar.

SQL yordamida so‘rovlarni bir birining ichiga joylashtirish ham mumkin. Odatda ichki so‘rov qiymat hosil qiladi va bu qiymat tashqi predikat tomonidan tekshirilib, to‘g‘ri yoki noto‘g‘riligi tekshiriladi.

Misol: bizga sotuvchi nomi ma’lum: Motika, lekin biz SNUM maydoni qiymatini bilmaymiz va buyurtmachilar jadvalidan hamma buyurtmalarni ajratib olmoqchimiz. Buni quyidagicha amalga oshirish mumkin:

SELECT * FROM Orders WHERE SNum =
(SELECT SNum FROM Salepeople
WHERE SName = 'Motika’);

Agar ostki so‘rovda IN operatoridan foydalanilsa, ixtiyoriy sondagi satrlar hosil qilish mumkin. Misol: Londondagi sotuvchilar uchun hamma buyurtmalarni ko‘rsatish.

SELECT * FROM Orders WHERE SNum IN
(SELECT SNum FROM Salepeople WHERE City = 'London');

Bu natijani jamlanma orqali ham hosil qilish mumkin. Lekin odatda ostki so‘rovli so‘rovlar tezroq bajariladi. Ostki so‘rovlarni HAVING izlash sharti ichida ishlatish ham mumkin. Bu ostki so‘rovlar agar ko‘p qiymatlar qaytarmasa agregat funksiyalaridan yoki GROUP BY yoki HAVING operatorlaridan foydalanishi mumkin. Misol:

SELECT Rating, COUNT (DISTINCT CNum) FROM Customers
GROUP BY Rating
HAVING Rating >(SELECT AVG (Rating) FROM Customers
WHERE City = 'San Jose');

Bu buyruq San Jose dagi baholari o‘rtachadan yuqori bo‘lgan buyurtmachilarni aniqlaydi.

UNION ifodasidan foydalanish. UNION ifodasi bir yoki bir necha so‘rovlar natijasini birlashtirishga imkon beradi.

Misol: Londonda joylashgan hamma sotuvchilar va buyurtmachilarni bitta jadvalda chiqarish.

SELECT SNum, SName FROM Salepeople WHERE City = 'London'
UNION
SELECT CNum, CName FROM Customers WHERE City = 'London';

So‘rovlarda funksiyalar

Agregat funksiyalar qo‘llanishi.

Agregat (yoki STATIK) funksiyalar sonli yoki hisoblanuvchi ustunlar bilan ishlaydi. Agregat funksiya argumenti butun ustun bo‘lib, bitta qiymat qaytaradi. Bu funksiyalarga quyidagilar kiradi:

• SUM() – ustundagi hamma qiymatlar summasini hisoblash.
• AVG() – ustundagi hamma qiymatlar o‘rtachasi qiymatini hisoblash.
• MIN() – ustundagi hamma qiymatlar eng kichigini aniqlash.
• MAX() – ustundagi hamma qiymatlar eng kattasini aniqlash.
• COUNT() – ustundagi qiymati sonini aniqlash.
• COUNT(*) – so‘rov natijasi jadvalidagi satrlar sonini aniqlash.

Agregatlash argumenti bo‘lib ustun nomidan tashqari ixtiyoriy matematik ifoda xizmat qilishi ham mumkin. Misol: Sotuv kompaniyada reja bajarilishining o‘rtacha protsentini aniqlash.
SELECT AVG(100 * (SALES/QUOTA)) FROM SALESREPS
Masalan, sotuv kompaniyasida sotuvlar xajmini chiqarish.
SELECT SUM(QUOTA), SUM(SALES) FROM SALESREPS
AVG() agregatlash funksiyasiga yana bir sodda misolni ko‘ramiz. Masalan: "ACI" ishlab chiqaruvchi mollari o‘rtacha narxini hisoblash.
SELECT AVG(PRICE) FROM PRODUCTS
WHERE MFR_ID = 'ACI'
Ekstremumlarni topishda MIN() va MAX() funksiyalari sonli ustunlar, sanalar va satrli o‘zgaruvchilar bilan ishlaydi. Eng sodda qo‘llanish sonlar bilan ishlash. Masalan, eng ko‘p va kam sotuvlar rejadagi hajmini chiqarish.
SELECT MIN (QUOTA), MAX (QUOTA) FROM SALESREPS
Masalan, buyurtmalardan eng oldin berilgan so‘rov sanasini topish.
SELECT MIN(ORDER_DATE) FROM ORDERS
MBdagi yozuvlar sonini sanash uchun COUNT() qo‘llaniladi. Bu funksiya son qiymat qaytaradi. Masalan: kompaniya mijozlari sonini chiqarish.
SELECT COUNT(CUST_NUM) FROM CUSTOMERS
COUNT(*) funksiyasi satrlar sonini hisoblaydi. Misol
SELECT COUNT(*) FROM ORDERS
Agregat funksiyalar jadval uchun natijaviy satr hosil ham qiladi. Masalan: Buyurtma o‘rtacha narxini chiqarish.
SELECT AVG(AMOUNT) FROM ORDERS

SQL turlari va strukturasi

Yuqorida ta’kidlab o‘tilganidek SQL tili tarkibida xisoblash jarayonini boshqarish imkonini beruvchi IF ... THEN ...ELSE, GO TO, DO ... WHILE kabi buyruqlar mavjud emas. Bunday masalalar dasturiy yo‘l bilan (dasturlash tili yoki masalalarni boshqarish tili) yoki interaktiv xolda (foydalanuvchining so‘rovlari asosida) amalga oshiriladi. Imkoniyati cheklanganligi sababli (xisoblash jarayonini boshqarish imkoniyati) SQL tili 2 ta usulda qo‘llanilish mumkin. Birinchi usulda interaktiv ishlash nazarda tutiladi. Bunda foydalanuvchi SQL operatorlarini terminaldan beradi. Ikkinchi usulda protsedurali tildagi dasturga SQL tili operatorlari kiritiladi.
Interaktiv rejimda ma‘lumotlar bazasi bilan ishlashda foydalanuvchi muloqat rejimida ishlaydi, ya’ni SQL tilidagi so‘rovni kiritadi va natijani oladi, yani so‘rovni kiritadi va natijag ega bo‘ladi va x.k.
Kiritilgan SQL rejimida boshqa dasturlash tillarida yaratigan dastur tarkibiga kiritiladi. Bu ma‘lumotlar bazasi bilan boshqa algoritmik tillarda yaratilgan amaliy dasturlar orqali ishlashni ta’minlaydi. Biroq bu yerda qo‘shimcha dasturiy vosita kerak bo‘ladi. U dasturlash tili bilan SQL operatorlari o‘rtasidagi interfeysni ta’minlab beradi.

Atamalar

SQL til yordamida ma‘lumotlar bazasiga beriladigan so‘rov deganda joriy buyruq tavsiflagan va ma‘lumotlar bazasini boshqarish tizimi tomonidan bajarish uchun mo‘ljallangan ma‘lumotlar bazasi ustida bajariladigan buyruq tushuniladi. So‘rov SQL tili opreatorlari yordamida yaratiladi. Operatorlar so‘zlar deb ataluvchi aloxida ma’noli qismlardan tashkil topadi. Operatorlar sintaksisi SQL tilining standartida belgilab berilgan.
SQL tili relyatsion ma‘lumotlar bazasi bilan ishlashiga qaramasdan "munosabat" atamasi o‘rniga "jadval" atamasi, "kortej" va "atribut" atamalari o‘rniga "satr" va "ustun" atamalari ishlatiladi.

SQL strukturasi va operatorlari

SQL tili 1 jadvalda tasvirlangan bo‘limlardan iborat.
1 jadval. Ma‘lumotlarni aniqlash operatorlari DDL(Data Definition Language – ma‘lumotlarni aniqlash tili)
2 jadval. Ma‘lumotlarni manipulyatsiyalash operatorlari Data Manipulation Language (DMP)
3 jadval. Data Query Language (DQL) so‘rov tili
4 jadval. Tranzaksiyalarni boshqarish operatorlari
5. Jadval. Ma‘lumotlarni boshqarish operatorlari • DCL (Data Control Language) – ma‘lumotlarni boshqarish
Ko‘pincha ustunlardagi qiymatlarning maksimal, minimal va o‘rtacha qiymatlarini xisoblashga to‘g‘ri keladi. Masalan, o‘rtacha ballni xioblash zarurati paydo bo‘ladi. Bunday xisoblashlarni bajarish uchun SQL tilida maxsus agregat

funksiyalari mavjud:
MIN – ustundagi minimal qiymat;
MAX – ustundagi maksimal qiymat;
SUM – ustundagi qiymatlar yig‘indisi;
AVG – ustundagi qiymatlarning o‘rtachasi;
COUNT – ustundagi NULL dan farqli bo‘lgan qiymatlar miqdori.
Quyidagi so‘rov talabalar imtixonlarda olgan ballarining o‘rtachasini aniqlaydi.
SELECT AVG(mark) FROM mark_st

SQL tili Oddiy so‘rovlari.

Sozdayte zapros s imenem «Sam?y prostoy zapros», kotor?y v?chislyaet znachenie v?rajeniya «2+2» i v?daet teku? ee vremya, ne ispolzuya pri etom tablis? ili drugie zapros? baz? dann?x.
SELECT 2+2;

SELECT ma‘lumotlarni tanlash operatori

Ma‘lumotlarni tanlash SELECT operatori yordamida bajariladi. Bu SQL tilining eng ko‘p qo‘llaniladigan operatori xisoblanadi. SELECT operatorini sintaksisi quyidagicha:
SELECT [ALL/DISTINCT] /*
FROM
[WHERE]
[ORDER BY < atributlar ro‘yxati >]
[GROUP BY < atributlar ro‘yxati >]
[HAVING]
[UNION< SELECT operatorli ifoda>]

Kvadrat qavslrda operatorni yozishda qatnashishi shart bo‘lmagan elementlar ko‘rsatilgan. ALL kalit so‘zi natijaga shartni qanoatlantiruvchi barcha satrlar , shuningdek takrorlanuvchi satrlar ham kirishini bildiradi. DISTINCT kalit so‘zi natijaga takrorlanuvchi satrlar kiritilmasligini bildiradi. Keyin bolang‘ich jadvaldagi atributlar ro‘yxati ko‘rsatiladi. Bu atributlar natijaviy jadvalga kiritiladi. * simvoli natijaviy jadvalga boshlang‘ich jadvalning barcha atributlari kiritilishini bildiradi.
Operatorda qatnashishi shart bo‘lgan so‘zlardan FROM so‘zi xisoblanadi. Bu so‘zdan keyin tanlov bajariladigan jadvallar nomi ko‘rsatiladi.
Tanlash ifodasida WHERE kalit so‘zidan keyin jadval satrlarini tanlab olish sharti ko‘rsatiladi. Bunda natijaviy jadvalga WHERE ifodasidagi shart rost qiymat qabul qiladigan satrlar kiritiladi.
ORDER BY kalit so‘zi natijaviy jadval satrlarini ko‘rsatilgan ustunlar ro‘yxati bo‘yicha tartiblash amalini bildiradi.
GROUP BY kalit so‘zidan keyin gruppalanadigan atributlar ro‘yxati ko‘rsatiladi.
HAVING ifodasida har bir gruppaga qo‘yiladigan shartlar ko‘rsatiladi. (GROUP BY va HAVING kalit so‘zlari keyinroq tushuntiriladi)
FROM, WHERE va ORDER BY kalit so‘zlari SQL tilining qolgan ma‘lumotlarni manipulyatsiyalash operatorlarida ham shu tarzda ishlatiladi.
So‘rovlar yaratishni aniq misol uchun ko‘rib o‘tamiz (1-rasm)
Barcha studentlar ro‘yxatini tanlash va tasvirlash.
SELECT *
FROM student
yoki
SELECT id_st, surname
FROM student
Agar ushbu so‘rovga ORDER BY surname ifodasi qo‘shilsa, uxolda ro‘yxat familiya bo‘yicha tartiblanadi. Jimlikka ko‘ra tartiblash o‘sish bo‘yicha bajariladi. Agar kamayish bo‘yicha tartiblash kerak bo‘lsa, u xolda oxirgi ifodadagi atribut nomidan keyin DESC so‘zi qo‘shiladi..
«1» kodli student olgan baxolar ro‘yxatini tanlab olish va tasvirlash
SELECT id_st, mark
FROM mark_st

Where id_st = 1

Ekzamenlarda kamida bitta 2 yoki 3 baxo olgan studentlar kodini tanlab olish

WHERE so‘zidan keyin solishtirish amallarini (<, >, =, <> va x.k.) va mantiqiy operatorlar qatnashgan ifodalarni joylashtirish mumkin.

```
SELECT id_st, mark
FROM mark_st
WHERE ( MARK >= 2 ) AND ( MARK <= 3 )
```

SQL tilida shart ifodalarini tuzish uchun solishtirish va mantiqiy operatorlardan tashqari yana bir qator maxsus operatorlar qo‘llaniladi. Bu operatorlar dasturlash tillarida mavjud emas. Bu operatolar:

IN – biror qiymatlar to‘plamiga tegishliligini tekshirish;
BETWEEN – biror qiymatlar diapozoniga tegishliligini tekshirish;
LIKE – namuna bilan mosligini tekshirish;
IS NULL – qiymat mavjudmasligini tekshirish.

IN operatori biror qiymatlar to‘plamiga tegishlilikni tekshirish uchun ishlatiladi.

Quyidagi so‘rov oxigi keltirilgan misoldagi so‘rov natijalarini beradi (ekzamenlarda kamida bitta 2 yoki 3 baho olgan studentlarni identifikatorini tanlab oladi)

```
SELECT id_st, mark
FROM mark_st
WHERE mark IN (2,3)
```

Xuddi natijani BETWEEN operatoridan foydalanib olish mumkin:

```
SELECT id_st, mark
FROM mark_st
WHERE mark BETWEEN 2 AND 3
```

Familiyalari A xarfi bilan boshlanuvchi studentlar ro‘yxatini tanlab olish

Bunday xolatda LIKE operatoridan foydalanish qulay xisoblanadi. LIKE operatori faqat simvolli maydonlar uchun qo‘llaniladi va maydon qiymati operatorda ko‘rsatilgan namunaga mosligini tekshirish imkonini yaratadi. Namuna quyidagi maxsus simvollardan tashkil topadi:

_ (tagiga chizish belgisi) – bitta ixtiyoriy simvolni bildiradi;
% (foiz belgisi) – ixtiyoriy miqdordagi simvollar ketma – ketligini bildiradi.

```
SELECT id_st, surname
FROM student
WHERE surname LIKE 'A%'
```

Ko‘pincha ustunlardagi minimal, maksimal yoki o‘rtacha qiymatlarni xisoblashga to‘g‘ri keladi. Masalan, keltiligan jadvallarda o‘rtacha baxoni xisoblash mumkin. Bunday xisoblashni bajarish uchun SQL tilida maxsus agregat funksiyalari mavjud:

MIN() – ustundagi minimal qiymat;
MAX() – ustundagi maksimalqiymat;
SUM() – ustundagi qiymatlar summasi;
AVG() – ustundagi qiymatlarni o‘rtachasi;
COUNT() – ustundagi NULL dan farqli qiymatlar miqdori.

Quyidagi so‘rov studentlar olgan baxolarning o‘rtachasini xisoblaydi.

```
SELECT AVG(mark)
FROM mark_st
```

Albatta, agregat funksiyalarini WHERE so‘zi bilan birgalikda qo‘llash mumkin. Quyidagi so‘rov 100 kodli studentning ekzamenlarda olgan baxolarining o‘rtachasini xisoblaydi:

```
SELECT AVG(mark)
FROM mark_st
WHERE id_st = 100
```

Navbatdagi so‘rov 10 kodli studentning o‘rtacha baxosini xisoblaydi:

```
SELECT AVG(mark)
FROM mark_st
WHERE id_ex = 10
```

SQL tili ko‘rib o‘tilgan mexanizmiga qo‘shimcha ravishda agregat funksiyalarni to‘la jadval uchun emas, balki gruppalangan qiymatlar uchun qo‘llash imkonini beruvchi vosita mavjud. Buning uchun SQL da maxsus GROUP BY konstruksiyasi mavjud bo‘lib unda ko‘rsatilgan ustun qiymatlari bo‘yicha gruppalash amalga oshiriladi. Masalan, har bir studentni ekzamenlarda olgan baxolarining o‘rtachasini aniqlash mumkin. Buning uchun quyidagi so‘rov beriladi:

```
SELECT id_st, AVG(mark)
FROM mark_st
GROUP BY id_st
```

Bu imkoniyat ham odatdagidek WHERE so‘zi bilan birgalikda qo‘llanilishi mumkin. Bu so‘rovni bajarishda MBBT avval jadvaldan WHERE ifodasidagi shartni qanoatlantiruvchi satrlarni tanlab oladi, keyin tanlangan satrlarni gruppalashva agregatlash amalini bajaradi.

Quyidagi so‘rov har bir studentning 100 kodli ekzamendan olgan baxolarining o‘rtachasini aniqlaydi.

```
SELECT id_st, AVG(mark)
FROM mark_st
WHERE id_ex = 100
GROUP BY id_st
```

Ko‘rib turganimizdek gruppalash bittadan ko‘p maydon bo‘yicha bajarilishi mumkin. GROUP BY seksiyasiga ega bo‘lgan so‘rovlar uchun quyidagi muxim cheklanish mavjud: bunday so‘rovlar natijasi tarkibiga gruppalash bajarilgan ustunlar va agregatlash natijasiga ega bo‘lgan ustunlar kiradi.

Biror xulosaga kelishdan oldin SQL tilining barcha imkoniyatlarini ko‘rib chiqish kerak. Masalan, ixtiyoriy tekstni so‘rov tarkibiga kiritish mumkin. Bunga misol keltiramiz:

```
SELECT 'O‘rtacha ball =’, AVG(mark)
FROM mark_st
WHERE id_ex = 10
```

Ushbu so‘rov natijasida foydalanuvchi faqat oddiy sonlarni emas, balki tekstni ham ko‘radi.

Bir necha jadvallardan ma‘lumotlarni tanlash uchun SQL ni qo‘llash

Sh paytgacha faqat bitta jadvaldan ma‘lumotlarni tanlash miollari keltirildi. Relyatsion amallarga mos bo‘lgan bir nechcha jadvallardan ma‘lumotlar tanlab olishni ham bajarish mumkin. Bir nechta jadvallardan ma‘lumotlarni tanlab olishga to‘liq misollar keltirish imkoni yo‘q. Bunga doir ba’zi misollarni ko‘rib o‘tamiz.

Qoidaga ko‘ra, ma‘lumotlar tanlab olinadigan jadvallar u yoki bu shaklda bir biri bilan bog‘langan. Masalan, birga ko‘p va x.k.

1 rasmdagi ER-diagrammaga qarang. Bu miolda bog‘langan jadvallar mavjud. student, mark_st va exam_st jadvallarini ko‘rib chiqamiz.

mark_st jadvali id_ex maydoni bo‘yicha exam_st jadvali bilan bog‘langan.

mark_st jadvali id_st maydoni bo‘yicha student jadvali bilan bog‘langan.

Masalan, studentlarni ro‘yxatini ular ekzamenlardan olgan baxolari bilan birgalikda tanlash zarur bo‘lsin. Buning uchun quyidagi so‘rov beriladi:

```
SELECT student.surname, mark_st.id_ex, mark_st.mark
FROM student, mark_st
WHERE student.id_st = mark_st.id_st
```

Keltirilgan ko‘p jadvalli so‘rov bir jadvalli so‘rovdan quyidagilar bilan farq qiladi.

1. FROM seksiyasida ikkita jadvl ko‘rsatilgan.

2. jadvallr soni bitta ko‘p, shuning uchun ko‘rsatilgan maydonlar nomining bir qiymatliligi yo‘qoladi. Masalan, ko‘p xollarda maydonni FROM da ko‘rsatilgan jadvallar ro‘yxatidagi qaysi jadvaldan olish noma’lum bo‘lib qoladi. Maydon nomlarining ko‘p qiymatliligini bataraf etish uchun maydon nomida perefiks - jadval nomi qo‘shimcha qilinadi. Jadval nomi maydon nomidan nuqta bilan ajratiladi.

3. WHERE ifodasida jadvallarni birlashtirish sharti ko‘rsatiladi.

Ko‘rinib turganidek jadval nomidan iborat prefiksdan foydalanish so‘rovni murakkablashtiradi. Bunday

murakkablikni bartaraf etish uchun psevdonim ishlatiladi. Yuqoridagi so‘rovni quyidagicha yozish mumkin:

```
SELECT E.surname, M.id_ex, M.mark
FROM student E, mark_st M
WHERE E.id_st = M. id_st
```

INSERT ma‘lumotlarni kiritish operatori:

```
INSERT INTO jadval_nomi [() ] VALUES ()
```

Bunday sintaksis jadvalga faqat bitta star kiritish imkonini beradi. Agar satrdagi barcha ustunlarga qiymat kiritilayotgan bo‘lsa, so‘rovda barcha ustunlar nomini ko‘rsatish zarur emas.

Masalan, BOOKS jadvaliga yangi kitob ma‘lumotlari kiritiladi

```
INSERT INTO BOOKS ( ISBN, TITL, AUTOR, COAUTOR, YEARIZD, PAGES)
VALUES ("5-88782-290-2","Apparatn?e sredstva IBM PC.Ensiklopediya",
"Guk M. ", "",2000, 816)
```

Bu kitob avtori faqat bitta va soavtor (hammuallif) mavjud emas, biroq ustunlar ro‘yxatida COAUTOR ustuni ham ko‘rsatilgan. Shuning uchun VALUES bo‘limida bu ustunga mos qiymatni ko‘rsatish zarur. Misolda bu maydon uchun bo‘sh satr ("") ko‘rsatilgan. Bu soavtor yo‘qligini bildiradi. Shuningdek bu yerda aniqlangmagan NULL qiymatini ko‘rsatish ham mukin edi.

Satrdagi barcha ustunlarga qiymat kiritishda ustunlar ro‘yxatini ko‘rsatish zarur emas. Bunda faqat qiymatlar ro‘yxatini ko‘rsatish yetarli bo‘ladi. Bunday xolda operator ko‘rinish quyidagicha shaklda bo‘ladi:

```
INSERT INTO BOOKS VALUES ("5-88782-290-2",
"Apparatn?e sredstva IBM PC. Ensiklopediya"."Guk M.",".2000.816)
```

Misolda keltirilgan ikkiala operator ham bir xil amalni bajaradi.

Shuningdek to‘liq miqdorda bo‘lmagan qiymatlarni ko‘rsatish mumkin. Ya’ni iymatlar qatorida soavtorni ko‘rsatmaslik mumkin, chunki jeoriy kitobda soavtor yo‘q. Biroq bunda qiymat kiritiladigan ustun nomlarini quyidagicha shaklda ko‘rsatish kerak bo‘ladi:

```
INSERT INTO BOOKS ( ISBN, TITL, AUTOR, YEARIZD,PAGES)
VALUES ("5-88782-290-2"."Apparatn?e sredstva IBM PC.
Ensiklopediya". Guk M.".2000,816)
```

Bu xolda COAUTOR ustuniga NULL qiymati yoziladi.

Agar jadvalni yaratishda ustun yoki atributga majburiy qiymat (NOT NULL) belgisi qo‘yilgan bo‘lsa, u xolda INSERT operatorida joriy ustunning har bir satriga kiritiladigan qiymatg ko‘rsatilishi kerak. Shuning uchun, agar jadvalning hamma ustuni majburiy qiymat li bo‘lsa, u xolda har bir yangi kiitladigan satrda barcha ustun uchun qiymat mavjud bo‘lishi kerak va bunda ustunlar ro‘yxatini ko‘rsatish shar emas. Aks xolda jadvalda kamida bitta maburiy qiymatli bo‘lmagan ustun bo‘lsa, u xolda albatta ustunlar ro‘yxatini ko‘rsatish shart bo‘ladi.

Qiymatlar ro‘yxatida maxsus funksiyalar va ifodalar ko‘rsatilish ham mumkin. Bunda ushbu funksiyalarning qiymatlari ma‘lumotlarni kiritish momentida xisoblangan bo‘lishi zarur.

Ma‘lumotlarni kiritish operatori birdaniga bir necha satrlarni kiritish imkoniga ham ega. Bunda qiymatlar satri boshqa bir jadvaldan tanlab olinadi. Masalan studentlar xaqidagi jadval mavjud bo‘lsin. Unda studentlarning familiyasi, adresi, uy telefoni va tug‘ilgan sanasi ko‘rsatilgan bo‘lsin. U xolda bitta operator yordamida ularni bibliotekaning kitobxonlariga aylantirish mumkin:

```
INSERT INTO READER (FIO_studenta, Adres, Telefon, Data_rojd)
SELECT (FIO_studenta, Adres, Telefon, Den_rojd)
FROM STUDENT
```

DELETE o‘chirish operatori:

Ma‘lumotlarni o‘chirish operatori jadvaldan shartni qanoatlantiruvchi bir yoki bir neta satrlarni o‘chirishi mumkin.

```
DELETE FROM jadval _nomi [WHERE tanlash_sharti]
```

Agar satrlarni tanlash sharti ko‘rsatilmasa, u xolda jadvaldagi barcha satrlar o‘chiriladi. Natijada ma‘lumotlarga ega

bo‘lmagan bo‘sh bo‘lgan jadval xosil bo‘ladi.
Agar jadvaldan oldingi sessiya natijalarini o‘chirish kerak bo‘lsa, u xolda R1 jadvalidagi barcha satrlar o‘chiriladi:

DELETE FROM R1

WHERE qismidagi shart ifodasi xuddi SELECT operatoridagi filtrlash shartiga o‘xshash bo‘ladi. Bu shart jadvaldan qaysi satrlar o‘chirilishi kerakligini aniqlaydi.
Masalan, student Mironova A.V. o‘chirilmaslii kerak bo‘lsa, quyidagi so‘rov beriladi:

DELETE FROM R2 WHERE FIO = "Mironov A.V."

WHERE qismida biror so‘rov ko‘rsatilishi mumkin. Masalan, agar jadvaldan o‘zlashtirmagan studentlarni o‘chirish kerak bo‘lsin. Oliy ta’lim qonuniga ko‘ra oxirgi sessiyada ikkita va undan ortiq fandan ikki baxo olgan student o‘zlashtirmagan xisoblanadi. U xolda tanlab olish sharti ikkita va undan ko‘p ikki baxo olgan studentlarni va ikkita undan ko‘p ekzamenlarni topshirmagan studentlarni aniqlashi kerak. Bunday studentlarni aniqlash uchun R1 jadvalidan 2 baxoli va baxo ko‘rsatilmagan satrlar tanlab olinishi, keyin olingan natija FIO ustuni bo‘yicha gruppalanishi kerak. Keyin har bir gruppadagi satrlar soni aniqlanadi (bu har bir studentning olgan ikki baxolari bilan topshirmagan ekzamenlar sonini bildiradi) va ikkitadan ko‘p satrga ega ustunlar tanlab olinadi. Endi ushbu murakkab bo‘lgan konstruksiyani SQL tilida yozamiz va sodda ko‘rinishga ega bo‘lishini ko‘ramiz.

DELETE FROM R2 WHERE R2.FIO IN (SELECT R1.FIO FROM R1
WHERE Otsenka = 2 OR Otsenka IS NULL GROOP BY R1.FIO HAVING COUNT(*)
>= 2
DELETE operatsiyasini bajarishda unda qatnashgan qism so‘rovda satrlar o‘chiriladigan jadval ko‘rsatilmasligi kerak. Ma‘lumotlarni manipulyatsiyalash operatsiyalarining barchasi ma‘lumotlar bazasining butunligi tushunchasi bilan bog‘langan. Manipulyatsiyalash amallari sintaktik jixatdan to‘g‘ri bo‘lsada butunlik talablari tufayli bajarilmasligi mmkin.

UPDATE ma‘lumotlarni yangilash operatsiyasi

o‘zgarish yuz berganda va mos xolda bu o‘zgarishni ma‘lumotlar bazasida akslantirish uchun ishlatiladi.

UPDATE jadval _nomi SET ustun_nomi = yangi_qiymat [WHERE tanlash_sharti]

Bu yerda ham WHERE qismi DELETE operatoridagi kabi ko‘rsatilishi shart emas. U DELETE operatoridagi kabi bir xil vazifani bajaradi va o‘zgartirish amali bajariladigan satlarni tanlash imkonini beradi. Agar tanlash sharti (WHERE qismi) ko‘rsatilmagan bo‘lsa, u xolda o‘zgartirish amali jadvlning barcha satrlari uchun bajariladi.
Masalan, student Stepanova K. Ye. ma‘lumotlar bazasi fanidan "2" baho oldi, keyin uni "3" bahoga qayta topshirgan bo‘lsin. Bu xolatga mos xolda R1 jadvalini o‘zgartirish amali quyidagi operator bilan amalga oshiriladi:

UPDATE R1
SET R1.Otsenka = 3
WHERE R1.FIO = "Stepanova K.Ye." AND R1.Dissiplina = "Baz? dann?x"

Qanday xolatlarda bir nechta satrlarni o‘zgartirish zarurati paydo bo‘ladi? Bu kam uchaydigan masala emas. Masalan, agar guruxlar jadvalidagi guruxlarni kursini bittaga oshirish zarr bo‘lsa quyidagi o‘zgartirish amali bajarishi mumkin. Guruxlar jadvali quyidagicha sxemaga ega bo‘lsin:

R4 = < Gurux, Kurs>

UPDATE R4
SET R4.Kypc = R4.Kypc + 1

Ma‘lumotlarni turi. Ma‘lumotlar bazasini yaratish

MYSQL tilida jadvaldagi maydonlarni quyidagi tiplari mavjud [2]:
Ma‘lumotlarni aniqlash

SQL ning DDL (Data Definition Language) ma‘lumotlarni aniqlash tili ma‘lumotlar bazasining sxema, domenlar, jadvallar, tasvirlar va indekslar kabi ob’ektlarini yaratish va o‘chirish uchun ishlatiladi.
Quyida SQL da ma‘lumotlarni aniqlash tilining asosiy operatorlari nomi keltirilgan.

CREATE SCHEMA DROP SCHEMA
CREATE DOMAIN ALTER DOMAIN DROP DOMAIN
CREATE TABLE ALTER TABLE DROP TABLE
CREATE VIEW DROP VIEW
Bu operatorlar konseptual sxemaning tarkibiga kiruvchi strukturalarni yaratish, taxrirlash va o‘chirish uchun ishlatiladi. Ba’zi MBBT larda quyidagi 2 ta operator xam mavjud:

CREATE INDEX DROP INDEX

Jadvallarni yaratish

Jadvallar CREATE TABLE komandasi bilan yaratiladi. Bu komanda qatorlarsiz bo‘sh jadval yaratadi. CREATE TABLE komandasi jadval nomini va jadvalning o‘zini ko‘rsatilgan tartibdagi ustunlar ketma – ketligi ko‘rinishida aniqlaydi. Unda har bir ustundagi ma‘lumotlarning tiplari va ustunlar o‘lchovi ko‘rsatiladi. Har bir jadval juda bo‘lmaganda bitta ustunga ega bo‘lishi kerak.
CREATE TABLE komandasi sintaksisi:

CREATE TABLE
([()],
[()], ...);

Argument qiymati kattaligi ma‘lumot turiga bog‘liqdir. Agar siz maxsus ko‘rsatmasangiz, tizim avtomatik qiymatni o‘rnatadi.
Misol uchun sotuvchilar jadvalini yaratishni ko‘rib chiqamiz:
CREATE TABLE Salepeople
(SNum integer,
SName char (10),
City char (10),
Comm decimal);

Jadvallarni o‘chirish

Jadvalni o‘chirish imkoniga ega bo‘lish uchun, jadval egasi (Ya’ni yaratuvchisi) bo‘lishingiz kerak. Faqat bo‘sh jadvalni o‘chirish mumkin. Qatorlarga ega bo‘lgan, to‘ldirilgan jadvalni o‘chirish mumkin emas, Ya’ni jadval o‘chirishdan oldin tozalangan bo‘lishi kerak. Jadvalni o‘chirish komandasi quyidagi ko‘rinishga ega:
DROP TABLE < table name >;
Masalan: DROP TABLE Salepeople;

Jadvalni o‘zgartirish

Jadvalni o‘zgartirish uchun ALTER TABLE komandasidan foydalaniladi. Bu komanda jadvalga Yangi ustunlar qo‘shish, ustunlarni o‘chirish, ustunlar kattaligini o‘zgartirish,hamda cheklanishlarni qo‘shish va olib tashlash imkoniyatlariga ega. Bu komanda ANSI standarti qismi emas, shuning uchun har xil tizimlarda har xil imkoniyatlarga ega.
Jadvalga ustun qo‘shish uchun komandaning tipik sintaksisi:
ALTER TABLE

ADD
;

Masalan:
ALTER TABLE Salepeople ADD Phone CHAR(7);

Jadvallar uchun cheklanishlar
Cheklanishlarni aniqlash.
Ko‘p xollarda ustunga kiritilgan qiymatlar bir biridan farq qilishi kerak. Agar ustun uchun UNIQUE cheklanishi o‘rnatilsa, bu ustungsha mavjud qiymatni kiritishga urinish rad etilapdi. Bu cheklanish bo‘sh bo‘lmaydigan (NOT NULL) debe’lon qilingan maydonlarga qo‘llanishi mumkin.
Masalan:
CREATE TABLE Salepeople
(SNum integer NOT NULL UNIQUE,
SName char (10),
City char (10),
Comm decimal);
Unikalligi talab qilinadigan maydonlar(birlamchi kalitlardan tashqari) kandidat kalitlar yoki unikal kalitlar deyiladi.
Jadval cheklanishi UNIQUE maydonlar guruxiga o‘rnatilishi mumkin. Bu bir necha maydonlar qiymatlari kombinatsiyasi unikalligini ta’minlaydi. Bizning ma‘lumotlart bazamizda har bir buyurtmachi bitta sotuvchiga biriktirilgan. Ya’ni Buyurtmachilar jadvalida buyurtmachi nomeri (cnum) va sotuvchi nomeri (snum) kombinatsiyasi unikal bo‘lishi kerak. Bu cheklanishni UNIQUE (cnum, snum) yordamida, Customers jadvalini yaratishda kiritish mumkin. Bu ustunlar uchun NOT NULL cheklanishini kiritish zarurdir.
Birlamchi kalitlar cheklanishlari.
SQL birlamchi kalitlarni to‘g‘ridan to‘g‘ri birlamchi kalit (PRIMARY KEY) cheklanishi orqali ta’riflaydi. PRIMARY KEY jadvalni yoki ustunlarni cheklashi mumkin. Bu cheklanish UNIQUE cheklanishi kabi ishlaydi, jadval uchun faqat bitta birlamchi kalit (ixtiyoriy sondagi ustunlar uchun) aniqlanishi mumkin bo‘lgan xoldan tashqari. Birlamchi kalitlar NULL qiymatga ega bo‘lishi mumkin emas.
Misol:
CREATE TABLE Salepeople
(SNum integer NOT NULL PRIMARY KEY,
SName char (10),
City char (10),
Comm decimal);
Maydon qiymatlarini tekshirish (CHECK cheklanishi).
CHECK cheklanishi jadvalga kiritilayotgan ma‘lumot qabul qilinishidan oldin mos kelishi lozim bo‘lgan shart kiritishga imkon beradi. CHECK cheklanishi CHECK kalit so‘zi ko‘rsatilgan maydondan foydalanuvchi predikat ifodalaridan iboratdir.
Misol: Salepeople jadvali Comm ustuniga kiritilayotgan qiymat 1 dan kichik bo‘lish sharti.
CREATE TABLE Salepeople
(SNum integer NOT NULL PRIMARY KEY,
SName char(10) NOT NULL UNIQUE,
City char(10),
Comm decimal CHECK (Comm < 1));

Ko‘zda tutilgan qiymatlarni (poumolchaniyu) o‘rnatish

Biror bir maydon uchun qiymat ko‘rsatmagan xolda jadvalga satr qo‘shsangiz, SQL bunday maydonga kiritish uchun ko‘zda tutilgan qiymatga ega bo‘lishi kerak, aks xolda komanda rad etiladi. Eng umumiy ko‘zda tutilgan qiymat NULL qiymatdir. CREATE TABLE komandasida ko‘zda tutilgan qiymat DEFAULT operatori orqali, ustun cheklanishi sifatida ko‘rsatiladi. Masalan:
CREATE TABLE Salepeople
(SNum integer NOT NULL PRIMARY KEY,
SName char(10) NOT NULL UNIQUE,
City char(10) DEFAULT 'New York',
Comm decimal CHECK (Comm < 1));

Ma‘lumotlar yaxlitligini ta’minlash

Jadval bir maydonidagi hamma qiymatlar boshqa jadval maydonida aks etsa, birinchi maydon ikkinchisiga ilova qiladi deyiladi. Bu ikki maydon orasidagi bog‘liqlikni ko‘rsatadi. Masalan, buyurtmachilar jadvalida har bir buyurtmachi, sotuvchilar jadvalida o‘ziga biriktirilgan sotuvchiga ilova qiluvchi SNum maydoniga ega. Bir maydon ikkinchisiga ilova qilsa tashqi kalit, u ilova qilayotgan maydon ajdod kalit deyiladi. Buyurtmachilar jadvalidagi SNum maydoni tashqi kalit, sotuvchilar jadvalidagi SNum - ajdod kalitdir.

Tashqi kalit bitta maydondan iborat bo‘lishi shart emas. Birlamchi kalit kabi, tashqi kalit bitta modul sifatida qayta ishlanuvchi bir necha maydonlarga ega bo‘lishi musmkin. Maydon tashqi kalit bo‘lsa ilova qitlayotgan jadval bilan ma’lum usulda bog‘liqdir. Tashqi kalit har bir qiymati (satri), ajdod kalitning bitta va faqat bitta qiymatiga(satriga) ilova qilishi kerak. Bu xolda tizim ilovali yaxlit xolatda deyiladi

Shu bilan birga ajdod kalit qiymati tashqi kalit bir necha qiymatlariga ilova qilishi mumkin.

Cheklanish FOREIGN KEY.

SQL ilovali yaxlitlikni FOREIGN KEY yordamida ta’minlaydi. Tashqi kalit vazifasi ajdod kalitda ko‘rsatilmagan qiymatlarni tashqi kalit maydonlariga kiritmaslikdir. FOREIGN KEY cheklanishi sintaksisi:

FOREIGN KEY REFERENCES

[]

Birinchi ro‘yxat komanda tomonidan o‘zgartiriluvchi ustunlar ro‘yxatidir. Pktable - bu ajdod kalitli jadval. Ikkinchi ustunlar ro‘yxati bu ajdod kalitni tashkil qiluvchi ustunlardir.

Misol uchun Sotuvchilar jadvaliga ilova qiluvchi tashqi kalit sifatida e’lon qilingan SNum maydoniga ega bo‘lgan Buyurtmachilar jadvalini yaratamiz:

```
CREATE TABLE Customers
( CNum integer NOT NULL PRIMARY KEY,
CName char(10),
City char(10),
SNum integer,
FOREIGN KEY (SNum) REFERENCES Salepeople (SNum) );
```

4-mavzuga doir savollar:

1. SELECT tanlash operatorining sintaksisi
2. IN, BETWEEN, LIKE, IS NULL operatorlari. Ularni ta’riflang va misollar keltiring.
3. Agregat funksiyalari.
4. SELECT operatorida yordamida bir nechta jadvallardan ma‘lumotlarni tanshlash
5. SQL tilining INSERT ma‘lumotlarni operatorlari.
6. SQL tilida ma‘lumotlarning tiplari.
7. SQL DDL ma‘lumotlarni aniqlash tili
8. SQL tilida jadvallar yaratish operatori