Ma'lumotlar ombori(Database) nima ?

• DBMS nima ?

• DBMS ni qandey turlari bor ?

• RDBMS nima ?

• DBMS bilan RDBMS ni nima farqi bor ?

• SQL nima ?

• Query nima ?

• SQL ni qandey elementlari bor ?

• DDL nima ?

• DML nima ?

• TCL nima ?

• DQL nima ?

• Client Server model nima ?

• Postgresql uchun qandey client larni bilasiz ?

**Ma'lumotlar ombori (Database) nima?**

Ma'lumotlar ombori - bu ma'lumotlarni saqlash, boshqarish va ularni osonlik bilan qidirish uchun mo'ljallangan tizimdir. Ma'lumotlar odatda strukturali bo'lib, jadval shaklida saqlanadi.

**DBMS nima?**

DBMS (Database Management System) - bu ma'lumotlar omborlarini yaratish, boshqarish va ularga kirish imkonini beruvchi dasturiy ta'minot. U ma'lumotlarni qo'shish, o'chirish, yangilash va qidirish kabi operatsiyalarni bajarishga imkon beradi.

**DBMS ni qandey turlari bor?**

DBMS ni turli xil turlari mavjud:

1. **Hierarchical DBMS** - Ma'lumotlar daraxt strukturasida saqlanadi.
2. **Network DBMS** - Ma'lumotlar graflar shaklida tashkil etilgan.
3. **Relational DBMS (RDBMS)** - Ma'lumotlar jadval (relations) shaklida saqlanadi.
4. **Object-oriented DBMS (OODBMS)** - Ma'lumotlar obyektlar shaklida saqlanadi.
5. **NoSQL DBMS** - Ma'lumotlar omborlari strukturaviy bo'lmagan ma'lumotlarni boshqarish uchun mo'ljallangan (document, key-value, column-family, graph).

**RDBMS nima?**

RDBMS (Relational Database Management System) - bu ma'lumotlarni jadval shaklida tashkil etish va boshqarish uchun mo'ljallangan DBMS. Har bir jadval bir-biri bilan bog'langan bo'lib, ma'lumotlar orasidagi munosabatlarni aniqlashga imkon beradi.

**DBMS bilan RDBMS ni nima farqi bor?**

1. **Ma'lumotlar tuzilishi**: DBMS har qanday shakldagi ma'lumotlarni boshqarishi mumkin, RDBMS esa ma'lumotlarni faqat jadval shaklida saqlaydi.
2. **ACID xususiyatlari**: RDBMS ACID (Atomicity, Consistency, Isolation, Durability) xususiyatlarini qo'llab-quvvatlaydi, DBMS esa har doim ham bu xususiyatlarni qo'llab-quvvatlamaydi.
3. **O'zgartirish va qidirish imkoniyatlari**: RDBMS murakkab SQL sorovlarini bajarishga imkon beradi, DBMS esa bunday imkoniyatlarga ega bo'lmasligi mumkin.

**SQL nima?**

SQL (Structured Query Language) - bu ma'lumotlar omborlarini boshqarish va ulardan ma'lumotlarni olish uchun ishlatiladigan standart dasturlash tili.

**Query nima?**

Query (so'rov) - bu ma'lumotlar omboridan ma'lumot olish yoki o'zgartirish uchun yozilgan buyruq. SQL so'rovlar ma'lumotlarni tanlash (SELECT), qo'shish (INSERT), o'zgartirish (UPDATE) yoki o'chirish (DELETE) kabi operatsiyalarni bajarishi mumkin.

**SQL ni qandey elementlari bor?**

SQL elementlari quyidagilarni o'z ichiga oladi:

1. **DDL (Data Definition Language)** - Ma'lumotlar tuzilmasini aniqlash uchun ishlatiladi (CREATE, ALTER, DROP).
2. **DML (Data Manipulation Language)** - Ma'lumotlarni boshqarish uchun ishlatiladi (SELECT, INSERT, UPDATE, DELETE).
3. **DQL (Data Query Language)** - Ma'lumotlarni qidirish uchun ishlatiladi (SELECT).
4. **TCL (Transaction Control Language)** - Transaksiyalarni boshqarish uchun ishlatiladi (COMMIT, ROLLBACK, SAVEPOINT).
5. **DCL (Data Control Language)** - Huquqlarni boshqarish uchun ishlatiladi (GRANT, REVOKE).

**DDL nima?**

DDL (Data Definition Language) - bu ma'lumotlar tuzilmasini aniqlash va boshqarish uchun ishlatiladigan buyruqlar to'plamidir. Masalan, CREATE TABLE, ALTER TABLE, DROP TABLE.

**DML nima?**

DML (Data Manipulation Language) - bu ma'lumotlarni qo'shish, o'zgartirish va o'chirish uchun ishlatiladigan buyruqlar to'plamidir. Masalan, INSERT, UPDATE, DELETE.

**TCL nima?**

TCL (Transaction Control Language) - bu transaksiyalarni boshqarish uchun ishlatiladigan buyruqlar to'plamidir. Masalan, COMMIT, ROLLBACK, SAVEPOINT.

**DQL nima?**

DQL (Data Query Language) - bu ma'lumotlarni qidirish va olish uchun ishlatiladigan buyruq. Asosiy DQL buyruqi SELECT bo'lib, ma'lumotlarni jadvallardan tanlash uchun ishlatiladi.

**Client Server model nima?**

Client-Server modeli - bu tarmoq modeli bo'lib, unda mijoz (client) dasturi serverga so'rov yuboradi va server ushbu so'rovga javob beradi. Mijoz-server modeli ma'lumotlar omborlari, veb-xizmatlar va boshqa ko'plab tarmoq dasturlarida ishlatiladi.

**Postgresql uchun qandey clientlarni bilasiz?**

PostgreSQL uchun bir qancha mijoz dasturlar mavjud:

1. **psql** - Bu PostgreSQL ning komandali satr interfeysi.
2. **pgAdmin** - Bu PostgreSQL uchun grafik foydalanuvchi interfeysi (GUI).
3. **DBeaver** - Bu ko'p platformali ma'lumotlar ombori boshqaruv dasturi.
4. **DataGrip** - Bu JetBrains tomonidan ishlab chiqilgan professional SQL editor.
5. **Navicat** - Bu ma'lumotlar omborini boshqarish va rivojlantirish uchun ishlatiladigan GUI vositasi.

Bu vositalar PostgreSQL bilan ishlashni osonlashtiradi va samaradorlikni oshiradi.

4o

Table nima ?

• upsert nima ?

• PSQL dagi qandey muhim command larni bilasiz ?

• Constraint nima ?

• PostgreSQL da qanday constraint lar bor ?

• foreign key constraint nima ?

• unique constraint nima ?

• check constraint nima ?

• not null constraint nima ?

• primary key constraint nima ?

**Table nima?**

Table (jadval) - bu ma'lumotlarni saqlash uchun tuzilgan struktura bo'lib, satrlar va ustunlardan iborat. Har bir ustun ma'lum bir turdagi ma'lumotlarni saqlaydi (masalan, integer, text), har bir satr esa bitta yozuvni ifodalaydi.

**Upsert nima?**

Upsert - bu mavjud bo'lmagan yozuvlarni qo'shish va mavjud yozuvlarni yangilashni o'z ichiga olgan operatsiya. PostgreSQLda INSERT ... ON CONFLICT buyrug'i bilan amalga oshiriladi.

**PSQL dagi qandey muhim command larni bilasiz?**

PSQL (PostgreSQL interaktiv terminali)da ba'zi muhim buyruqlar:

* \c [database\_name] - Ma'lumotlar bazasiga ulanish.
* \dt - Jadvallar ro'yxatini ko'rsatish.
* \d [table\_name] - Jadval tuzilmasini ko'rsatish.
* \q - PSQLdan chiqish.
* \i [file.sql] - SQL faylini bajarish.
* \l - Ma'lumotlar bazalari ro'yxatini ko'rsatish.

**Constraint nima?**

Constraint (cheklov) - bu jadvaldagi ma'lumotlarning yaxlitligi va to'g'riligini ta'minlash uchun qo'llaniladigan qoidalar. Bu qoidalar ma'lumotlarning noto'g'ri kiritilishiga yo'l qo'ymaydi.

**PostgreSQL da qanday constraint lar bor?**

PostgreSQLda quyidagi constraintlar mavjud:

* Primary Key
* Foreign Key
* Unique
* Check
* Not Null

**Foreign Key constraint nima?**

Foreign Key constraint - bu bir jadvaldagi ustun (yoki ustunlar to'plami) boshqa bir jadvaldagi primary key ustun(lar)i bilan bog'langanligini ko'rsatadigan cheklov. Bu jadval o'rtasidagi munosabatlarni aniqlaydi va ma'lumotlarning yaxlitligini ta'minlaydi.

**Unique constraint nima?**

Unique constraint - bu ustundagi yoki ustunlar to'plamidagi barcha qiymatlarning yagona va takrorlanmasligini ta'minlaydi. Bu, masalan, email manzillar yoki foydalanuvchi nomlari kabi qiymatlarning takrorlanmasligini ta'minlashda ishlatiladi.

**Check constraint nima?**

Check constraint - bu ma'lumotlarning belgilangan shartlarga muvofiqligini ta'minlaydigan cheklov. Masalan, yosh ustunining qiymati 18 dan katta bo'lishi kerakligi haqida shart qo'yish mumkin.

**Not Null constraint nima?**

Not Null constraint - bu ustundagi qiymatning hech qachon NULL bo'lmasligini ta'minlaydigan cheklov. Bu, masalan, foydalanuvchi ismi kabi qiymatlarning har doim kiritilishi kerakligini ta'minlashda ishlatiladi.

**Primary Key constraint nima?**

Primary Key constraint - bu jadvaldagi har bir satrni yagona identifikator bilan aniqlaydigan ustun yoki ustunlar to'plamidir. Primary key har doim yagona va takrorlanmas bo'lishi kerak va Not Null constraint bilan birgalikda ishlatiladi.

4o

PostgreSQL da qandey raqamli tip lar bor ?

• timestamp va timestamptz nima farqi bor ?

• Qanday usullarda uuid yaratsa bo'ladi ?

• hstore qayday ma'lumot turi xisoblanadi ?

• json da -> va ->> farqlari nima ?

• qanday usullar da custom data tip yaratish mumkin ?

• domain bilan type ni nima fariq bor ?

• sequence nima ?

• generated as identity nima ?

• temp yoki temporary table qanday table ?

• copy table qanday table ?

ChatGPT

**PostgreSQL da qandey raqamli tiplari bor?**

PostgreSQLda bir nechta raqamli turlar mavjud:

* smallint: 2 baytli integer, qiymatlari -32,768 dan 32,767 gacha.
* integer (yoki int): 4 baytli integer, qiymatlari -2,147,483,648 dan 2,147,483,647 gacha.
* bigint: 8 baytli integer, qiymatlari -9,223,372,036,854,775,808 dan 9,223,372,036,854,775,807 gacha.
* decimal (yoki numeric): Cheksiz aniqlikdagi raqamlar.
* real: 4 baytli suzuvchi nuqtali raqam, ya'ni "float" turidagi raqam.
* double precision: 8 baytli suzuvchi nuqtali raqam.
* serial: Avtomatik ravishda o'sib boradigan integer (32 bit).
* bigserial: Avtomatik ravishda o'sib boradigan big integer (64 bit).

**Timestamp va Timestamptz nima farqi bor?**

* timestamp: Bu tur UTCga nisbatan vaqtni saqlaydi, lekin vaqt zonasini saqlamaydi.
* timestamptz (timestamp with time zone): Bu tur vaqtni UTC formatida saqlaydi va vaqt zonasini hisobga oladi. Kiritilgan vaqt zonasi avtomatik ravishda UTC ga aylantiriladi va saqlanadi.

**Qanday usullarda UUID yaratsa bo'ladi?**

PostgreSQLda UUID yaratish uchun quyidagi usullar mavjud:

* gen\_random\_uuid(): Bu funksiya pgcrypto kengaytmasidan foydalanadi va tasodifiy UUID yaratadi.
* uuid-ossp kengaytmasi yordamida bir nechta variantlar mavjud:
  + uuid\_generate\_v1(): MAC-manzil va joriy vaqtga asoslangan UUID yaratadi.
  + uuid\_generate\_v4(): Tasodifiy UUID yaratadi.

sql

Copy code

-- gen\_random\_uuid() ni ishlatish uchun:

SELECT gen\_random\_uuid();

-- uuid-ossp kengaytmasidan foydalanish uchun:

CREATE EXTENSION IF NOT EXISTS "uuid-ossp";

SELECT uuid\_generate\_v4();

**Hstore qayday ma'lumot turi hisoblanadi?**

hstore - bu PostgreSQLdagi kalit-qiymat (key-value) juftlarini saqlash uchun ishlatiladigan ma'lumot turi. U JSONga o'xshaydi, lekin faqat kalit-qiymat juftlarini saqlaydi va nesting (ichma-ich struktura) ni qo'llab-quvvatlamaydi.

**JSONda -> va ->> farqlari nima?**

* ->: Bu operator JSON obyekti yoki massiv ichidan kalit yoki indeks bilan qiymatni olish uchun ishlatiladi, natija JSON turida bo'ladi.
* ->>: Bu operator JSON obyektidan yoki massivdan kalit yoki indeks bilan qiymatni olish uchun ishlatiladi, natija esa tekst turida bo'ladi.

sql

Copy code

-- -> va ->> farqi

SELECT '{"name": "John", "age": 30}'::jsonb -> 'name'; -- natija: "John" (jsonb)

SELECT '{"name": "John", "age": 30}'::jsonb ->> 'name'; -- natija: John (text)

**Qanday usullar da custom data type yaratish mumkin?**

PostgreSQLda maxsus ma'lumot turlarini yaratish usullari:

* CREATE TYPE: Murakkab ma'lumot turini yaratish uchun ishlatiladi.
* CREATE DOMAIN: Oldindan aniqlangan ma'lumot turiga asoslangan yangi ma'lumot turini yaratish uchun ishlatiladi.

sql

Copy code

-- Custom type yaratish

CREATE TYPE address AS (

street VARCHAR(50),

city VARCHAR(50),

zipcode VARCHAR(10)

);

-- Custom domain yaratish

CREATE DOMAIN us\_postal\_code AS TEXT

CHECK (VALUE ~ '^\d{5}$');

**Domain bilan Type ni nima farqi bor?**

* Domain: Bu oldindan aniqlangan ma'lumot turiga asoslangan yangi ma'lumot turi. Bu validatsiya va default qiymatlar kabi qo'shimcha cheklovlarni qo'shish imkonini beradi.
* Type: Bu ma'lumotlarning yangi, murakkab turlarini yaratish imkonini beradi. U bir nechta oddiy ma'lumot turlarini birlashtirish va bitta obyektda saqlash uchun ishlatiladi.

**Sequence nima?**

Sequence - bu avtomatik ravishda o'sib boradigan raqamlarni yaratish uchun ishlatiladigan obyekt. U odatda primary key ustunlarini avtomatik ravishda to'ldirish uchun ishlatiladi.

sql

Copy code

-- Sequence yaratish

CREATE SEQUENCE serial\_seq

START 1

INCREMENT BY 1;

-- Sequence ni ishlatish

SELECT nextval('serial\_seq');

**Generated as identity nima?**

Generated as identity - bu serial tipidagi ustunlar o'rniga foydalaniladigan yangi standart. Bu ustunlar avtomatik ravishda o'sib boradi va primary key sifatida ishlatiladi.

sql

Copy code

CREATE TABLE test (

id INT GENERATED ALWAYS AS IDENTITY,

name TEXT

);

**Temp yoki Temporary table qanday table?**

Temporary (yoki temp) jadval - bu ma'lumotlar bazasidagi vaqtinchalik jadval bo'lib, faqat sessiya davomida mavjud bo'ladi. Sessiya tugagach, jadval avtomatik ravishda o'chiriladi.

sql

Copy code

CREATE TEMP TABLE temp\_table (

id SERIAL,

name TEXT

);

**Copy table qanday table?**

Copy table (nusxa jadvali) - bu mavjud jadvalning nusxasini yaratish uchun ishlatiladi. U mavjud jadvaldagi barcha ma'lumotlar va tuzilmani yangi jadvalga ko'chiradi.

sql

Copy code

-- Copy table yaratish

CREATE TABLE new\_table AS TABLE existing\_table;

Yuqoridagi ko'rsatmalar PostgreSQL ma'lumotlar bazasini boshqarish va undan samarali foydalanishga yordam beradi.

1/2

4o

• column alies nima ? • order by clause ni tushuntiring ? • distinct nima ? • between operaatori nima uchun kerak ? • like va ilike operatorlarini faqri qanday • all, any, exists vazifalari, nima maqsadlar da ishlatiladi ?

ChatGPT

**Column Alias nima?**

**Column alias** - bu natija jadvalida ustun nomini vaqtinchalik o'zgartirish uchun ishlatiladigan nom. Alias asosan natijalarni o'qishni osonlashtirish yoki ko'rsatiladigan ustun nomlarini qisqartirish uchun ishlatiladi. Aliaslar AS kalit so'zi bilan yaratiladi, lekin bu kalit so'zdan foydalanish shart emas.

sql

Copy code

SELECT column\_name AS alias\_name FROM table\_name;

-- yoki

SELECT column\_name alias\_name FROM table\_name;

Masalan:

sql

Copy code

SELECT first\_name AS name FROM employees;

**ORDER BY Clause ni tushuntiring?**

**ORDER BY clause** natijalarni belgilangan ustunlar bo'yicha tartiblash uchun ishlatiladi. Bu ASC (oshish tartibida) yoki DESC (kamayish tartibida) kalit so'zlari yordamida amalga oshiriladi.

sql

Copy code

SELECT column1, column2 FROM table\_name

ORDER BY column1 [ASC|DESC], column2 [ASC|DESC];

Masalan:

sql

Copy code

SELECT first\_name, last\_name FROM employees

ORDER BY last\_name ASC, first\_name DESC;

**DISTINCT nima?**

**DISTINCT** kalit so'zi natijadagi dublikat satrlarni olib tashlash uchun ishlatiladi, ya'ni faqat noyob satrlar qaytariladi.

sql

Copy code

SELECT DISTINCT column1, column2 FROM table\_name;

Masalan:

sql

Copy code

SELECT DISTINCT department FROM employees;

**BETWEEN operatori nima uchun kerak?**

**BETWEEN** operatori berilgan qiymatlar diapazonidagi qiymatlarni filtrlash uchun ishlatiladi. Diapazonning boshlanishi va tugash qiymatlari kiritiladi.

sql

Copy code

SELECT column\_name FROM table\_name

WHERE column\_name BETWEEN value1 AND value2;

Masalan:

sql

Copy code

SELECT \* FROM employees

WHERE salary BETWEEN 50000 AND 100000;

**LIKE va ILIKE operatorlarini farqi qanday?**

* **LIKE** operatori: Matnli ma'lumotlarni moslashtirishda ishlatiladi va katta-kichik harflarni farqlaydi.
* **ILIKE** operatori: LIKE operatoriga o'xshaydi, lekin katta-kichik harflarni farqlamaydi. Bu operator PostgreSQLga xosdir.

sql

Copy code

SELECT \* FROM employees

WHERE name LIKE 'J%'; -- Katta-kichik harflarni farqlaydi

SELECT \* FROM employees

WHERE name ILIKE 'j%'; -- Katta-kichik harflarni farqlamaydi

**ALL, ANY, EXISTS vazifalari, nima maqsadlarda ishlatiladi?**

* **ALL**: Subquery (ichki so'rov) natijalaridagi barcha qiymatlar bilan solishtirish uchun ishlatiladi.

sql

Copy code

SELECT \* FROM employees

WHERE salary > ALL (SELECT salary FROM employees WHERE department = 'HR');

* **ANY**: Subquery natijalaridagi bir yoki bir nechta qiymatlar bilan solishtirish uchun ishlatiladi.

sql

Copy code

SELECT \* FROM employees

WHERE salary > ANY (SELECT salary FROM employees WHERE department = 'HR');

* **EXISTS**: Subquery natijalari mavjudligini tekshirish uchun ishlatiladi. Agar subqueryda hech bo'lmaganda bitta satr mavjud bo'lsa, EXISTS TRUE qiymatini qaytaradi.

sql

Copy code

SELECT \* FROM employees e

WHERE EXISTS (SELECT 1 FROM departments d WHERE e.department\_id = d.id);

Bu operatorlar va klauzalar SQL so'rovlarini kuchliroq va moslashuvchan qilishga yordam beradi.

4o

• CONCAT\_WS funksiyasini vazifasi nima ? • to\_date funksiyasini vazifasi nima va qandey ishlaydi ? • like va ilike ni farqi nima ? • similar to ning vazifasi nima ? • posix regular expression ning vazifasi nima ?

ChatGPT

**CONCAT\_WS funksiyasini vazifasi nima?**

**CONCAT\_WS** funksiyasi PostgreSQLda bir nechta satrlarni bitta satrga birlashtirish uchun ishlatiladi, bu holda har bir satr orasida belgilangan ajratuvchi (separator) qo'shiladi. WS "With Separator" (Ajratuvchi bilan) degan ma'noni anglatadi.

sql

Copy code

SELECT CONCAT\_WS(separator, string1, string2, ...);

Masalan:

sql

Copy code

SELECT CONCAT\_WS(', ', 'John', 'Doe', 'Developer');

-- Natija: 'John, Doe, Developer'

**TO\_DATE funksiyasini vazifasi nima va qandey ishlaydi?**

**TO\_DATE** funksiyasi PostgreSQLda matnli satrni sanaga aylantirish uchun ishlatiladi. Bu funksiya matn formatini va natijaviy sanani qanday shakllantirishni aniqlaydigan format modelini talab qiladi.

sql

Copy code

TO\_DATE(text, format);

Masalan:

sql

Copy code

SELECT TO\_DATE('2023-06-13', 'YYYY-MM-DD');

-- Natija: 2023-06-13

Bu yerda text bu sanani ifodalovchi matnli satr va format esa matnni sanaga aylantirish uchun ishlatiladigan format modelidir.

**LIKE va ILIKE ni farqi nima?**

* **LIKE** operatori matnli ma'lumotlarni moslashtirish uchun ishlatiladi va katta-kichik harflarni farqlaydi (case-sensitive).
* **ILIKE** operatori LIKE operatoriga o'xshaydi, lekin katta-kichik harflarni farqlamaydi (case-insensitive). Bu operator PostgreSQLga xosdir.

sql

Copy code

-- LIKE operatori misoli:

SELECT \* FROM employees

WHERE name LIKE 'J%'; -- Faqat 'J' bilan boshlanuvchi ismlar (katta 'J' harfi)

-- ILIKE operatori misoli:

SELECT \* FROM employees

WHERE name ILIKE 'j%'; -- 'J' yoki 'j' bilan boshlanuvchi ismlar

**SIMILAR TO ning vazifasi nima?**

**SIMILAR TO** operatori PostgreSQLda murakkab matnli ma'lumotlarni moslashtirish uchun ishlatiladi va regular expression (oddiy ifoda) sintaksisini qo'llab-quvvatlaydi. U LIKE va POSIX regular expression operatorlari orasidagi xususiyatlarga ega.

sql

Copy code

column SIMILAR TO pattern;

Masalan:

sql

Copy code

SELECT \* FROM employees

WHERE name SIMILAR TO '(John|Jane)%';

Bu yerda (John|Jane)% shabloni "John" yoki "Jane" bilan boshlanuvchi ismlarni moslashtiradi.

**POSIX Regular Expression ning vazifasi nima?**

**POSIX Regular Expression** PostgreSQLda matnli ma'lumotlarni moslashtirish uchun ishlatiladigan kuchliroq vositadir. U ko'proq moslashtirish qoidalarini qo'llab-quvvatlaydi.

* ~: Match regular expression (oddiy ifoda bilan mos kelish).
* ~\*: Case-insensitive match (katta-kichik harflarni farqlamaslik).
* !~: Do not match regular expression (oddiy ifoda bilan mos kelmaslik).
* !~\*: Case-insensitive do not match (katta-kichik harflarni farqlamaslik).

sql

Copy code

-- ~ operatori misoli:

SELECT \* FROM employees

WHERE name ~ '^J.\*'; -- 'J' bilan boshlanuvchi ismlar (katta harflarni farqlaydi)

-- ~\* operatori misoli:

SELECT \* FROM employees

WHERE name ~\* '^j.\*'; -- 'J' yoki 'j' bilan boshlanuvchi ismlar (katta-kichik harflarni farqlamaslik)

POSIX regular expressions yordamida murakkab moslashtirish qoidalarini yozish mumkin, masalan, belgilar to'plamlari, ko'p belgilar, yoki belgilar guruhi.

4o

• GROUP BY Statement qachon ishlatiladi(nima uchun kerak) ?

• GROUP BY Statement sintaksisi qanday ?

• Having clause qachon ishlatiladi ?

• Having clause sintaksisi qanday ?

• Having clause va Where clause qanday farqi bor ?

• Union Statement ning vazifasi nima, nima uchun ishlatiladi ?

• Union Statement sintaksisi qanday ?

• Intersect Statement nima ?

• Intersect Statement ni sintaksisi qanday ?

• Except Statement nima ?

• Except Statement ni sintaksisi qanday ?

• View nima ?

• Virtual table nima ?

• Standart view qanday ishlaydi ?

• Materialized view nima ?

• Materialized view qanday ishlaydi ?

• Qanday qilib Materialized view dagi ma'lumotlarni yangilash mumkin ?

**GROUP BY Statement**

**GROUP BY** statement ma'lumotlarni guruhlash va agregat funksiyalar bilan birgalikda ishlatish uchun kerak bo'ladi. Bu statement yordamida bir xil qiymatga ega satrlarni bitta guruh sifatida ko'rib chiqish va ularga agregat funksiyalarni qo'llash mumkin.

**GROUP BY Statement Sintaksisi**

sql

Copy code

SELECT column1, AGGREGATE\_FUNCTION(column2)

FROM table\_name

GROUP BY column1;

Masalan:

sql

Copy code

SELECT department, COUNT(\*)

FROM employees

GROUP BY department;

**HAVING Clause**

**HAVING** clause GROUP BY statement bilan birgalikda qo'llaniladi va guruhlangan natijalarni filtrlash uchun ishlatiladi. HAVING guruhlangan natijalarga qo'llanadigan shartlarni belgilaydi, xuddi WHERE individual satrlarga qo'llanilganidek.

**HAVING Clause Sintaksisi**

sql

Copy code

SELECT column1, AGGREGATE\_FUNCTION(column2)

FROM table\_name

GROUP BY column1

HAVING condition;

Masalan:

sql

Copy code

SELECT department, COUNT(\*)

FROM employees

GROUP BY department

HAVING COUNT(\*) > 10;

**HAVING Clause va WHERE Clause Farqi**

* **WHERE** clause: Ma'lumotlarni guruhlashdan oldin individual satrlarni filtrlash uchun ishlatiladi.
* **HAVING** clause: Ma'lumotlar guruhlangandan keyin guruhlangan natijalarni filtrlash uchun ishlatiladi.

Masalan:

sql

Copy code

SELECT department, COUNT(\*)

FROM employees

WHERE salary > 50000 -- Satrlarga qo'llanadi

GROUP BY department

HAVING COUNT(\*) > 10; -- Guruhlarga qo'llanadi

**UNION Statement**

**UNION** statement ikki yoki undan ortiq SELECT so'rov natijalarini birlashtirish uchun ishlatiladi. Bu statement takrorlanuvchi satrlarni olib tashlaydi.

**UNION Statement Sintaksisi**

sql

Copy code

SELECT column1, column2 FROM table1

UNION

SELECT column1, column2 FROM table2;

Masalan:

sql

Copy code

SELECT name, department FROM employees

UNION

SELECT name, department FROM contractors;

**INTERSECT Statement**

**INTERSECT** statement ikki yoki undan ortiq SELECT so'rov natijalarining umumiy satrlarini qaytaradi.

**INTERSECT Statement Sintaksisi**

sql

Copy code

SELECT column1, column2 FROM table1

INTERSECT

SELECT column1, column2 FROM table2;

Masalan:

sql

Copy code

SELECT name, department FROM employees

INTERSECT

SELECT name, department FROM contractors;

**EXCEPT Statement**

**EXCEPT** statement birinchi SELECT so'rov natijalarida mavjud, lekin ikkinchi SELECT so'rov natijalarida mavjud bo'lmagan satrlarni qaytaradi.

**EXCEPT Statement Sintaksisi**

sql

Copy code

SELECT column1, column2 FROM table1

EXCEPT

SELECT column1, column2 FROM table2;

Masalan:

sql

Copy code

SELECT name, department FROM employees

EXCEPT

SELECT name, department FROM contractors;

**View**

**View** - bu saqlangan so'rov natijalari asosida yaratilgan virtual jadval. Viewlar ma'lumotlarni oson ko'rish va boshqarish uchun ishlatiladi.

**Virtual Table**

**Virtual table** - bu fizik jihatdan mavjud bo'lmagan, lekin ma'lumotlar bazasi so'rovi asosida yaratilib, ko'rib chiqilishi mumkin bo'lgan jadval.

**Standart View Qanday Ishlaydi?**

Standart viewlar asl jadvallarga asoslangan so'rovlarni ifodalaydi va ulardagi ma'lumotlar o'zgartirilganda avtomatik ravishda yangilanadi. Ular ma'lumotlar bazasida real jadvallarni yaratmasdan, natijalarni ko'rishga imkon beradi.

sql

Copy code

CREATE VIEW view\_name AS

SELECT column1, column2

FROM table\_name

WHERE condition;

**Materialized View**

**Materialized View** - bu view natijalari saqlanadigan va indekslanadigan fizik jadval. Bu viewlar ko'pincha og'ir so'rovlarni tezroq bajarish uchun ishlatiladi.

**Materialized View Qanday Ishlaydi?**

Materialized viewlar bir marta yaratilgandan so'ng, natijalar fizik jadval sifatida saqlanadi. Ma'lumotlar bazasidagi o'zgarishlar materialized viewda avtomatik yangilanmaydi.

sql

Copy code

CREATE MATERIALIZED VIEW view\_name AS

SELECT column1, column2

FROM table\_name

WHERE condition;

**Qanday qilib Materialized Viewdagi Ma'lumotlarni Yangilash Mumkin?**

Materialized viewni yangilash uchun REFRESH MATERIALIZED VIEW buyrug'i ishlatiladi.

sql

Copy code

REFRESH MATERIALIZED VIEW view\_name;

Bu buyruq materialized viewni qayta hisoblash va yangilashni amalga oshiradi.

4o

• PL/pgSQL nima ?

• Nima uchun PL/pgSQL foydalanish kerak ?

• Dollar-Quoted string constant nima ?

• Anonymous block nima ?

• rowtype qanday o'zgaruvchi ?

• record qanday o'zgaruvchi ?

• Qanday qilib o'zgaruvchilarni constant qilib qo'ysa bo'ladi ?

• IF statement sintaksisi qanday ?

• CASE statement sintaksisi qanday ?

• PL/pgSQL da qanday usullarda sikl hosil qilsa boladi ?

**PL/pgSQL nima?**

**PL/pgSQL** (Procedural Language/PostgreSQL Structured Query Language) - bu PostgreSQL ma'lumotlar bazasida foydalanish uchun yaratilgan procedural til. U SQL buyruqlarini procedural til konstruksiyalari bilan birlashtirish imkonini beradi, shu bilan murakkab ma'lumotlar bazasi amallarini bajarish osonlashadi.

**Nima uchun PL/pgSQL foydalanish kerak?**

PL/pgSQL-dan foydalanish bir necha sabablar tufayli foydalidir:

* **Murakkab mantiqni amalga oshirish:** PL/pgSQL procedural konstruksiyalarni, shuningdek, looping, shartli iboralar va o'zgaruvchilarni qo'llab-quvvatlaydi.
* **Ishlashni yaxshilash:** Mantiq ma'lumotlar bazasi darajasida amalga oshirilishi tufayli, PL/pgSQL yordamida jarayonlarni optimallashtirish va natijalarni tezlashtirish mumkin.
* **Qayta foydalanish va parvarish qilish:** Protsedural kodni funksiya yoki protsedura sifatida ajratish va qayta ishlatish osonroq bo'ladi.

**Dollar-Quoted String Constant nima?**

**Dollar-Quoted String Constant** PL/pgSQLda uzoq matnli satrlarni, xususan, ko'p qatorli matnlarni yozishda foydalaniladi. Bu usulda satrlar $tag$ va $tag$ orasiga joylashtiriladi, bu yerda tag ixtiyoriy identifikator bo'lib, ularni ajratish uchun ishlatiladi.

sql

Copy code

DO $$

BEGIN

RAISE NOTICE 'Hello, this is a dollar-quoted string!';

END $$;

**Anonymous Block nima?**

**Anonymous Block** PL/pgSQL-da aniqlanmagan protsedura yoki funksiyadir, ya'ni uni saqlab qo'yish shart emas va darhol bajariladi. Bu odatda DO statement bilan amalga oshiriladi.

sql

Copy code

DO $$

BEGIN

-- PL/pgSQL kod

RAISE NOTICE 'This is an anonymous block!';

END $$;

**rowtype qanday o'zgaruvchi?**

**%ROWTYPE** o'zgaruvchisi jadvallar yoki viewlarning tuzilishini aks ettiruvchi o'zgaruvchi bo'lib, jadvallar yoki viewlarning barcha ustunlari va ularning turlarini saqlaydi.

sql

Copy code

DECLARE

employee\_record employees%ROWTYPE;

BEGIN

SELECT \* INTO employee\_record FROM employees WHERE employee\_id = 1;

-- employee\_record o'zgaruvchisi employees jadvalidagi ustunlarni saqlaydi

END;

**record qanday o'zgaruvchi?**

**RECORD** o'zgaruvchisi tuzilishini oldindan bilmagan satrlarni saqlash uchun ishlatiladi. Bu dinamik va ko'pburchakli o'zgaruvchilar bo'lib, ma'lumotlar strukturasini oldindan bilmasdan ishlatish imkonini beradi.

sql

Copy code

DECLARE

employee\_record RECORD;

BEGIN

SELECT \* INTO employee\_record FROM employees WHERE employee\_id = 1;

-- employee\_record o'zgaruvchisi dinamik struktura bo'lib, employees jadvalidagi ma'lumotlarni saqlaydi

END;

**Qanday qilib o'zgaruvchilarni constant qilib qo'ysa bo'ladi?**

PL/pgSQLda o'zgaruvchini CONSTANT deb belgilash orqali o'zgarmas qilishingiz mumkin. Bunday o'zgaruvchilarning qiymati bir marta tayinlanadi va o'zgartirilishi mumkin emas.

sql

Copy code

DECLARE

max\_salary CONSTANT NUMERIC := 100000;

BEGIN

-- max\_salary o'zgarmas o'zgaruvchi, qiymati 100000

END;

**IF statement sintaksisi qanday?**

**IF** statement shartli mantiqni amalga oshirish uchun ishlatiladi.

sql

Copy code

IF condition THEN

-- kod

ELSIF another\_condition THEN

-- boshqa kod

ELSE

-- boshqa kod

END IF;

Masalan:

sql

Copy code

DECLARE

salary NUMERIC := 50000;

BEGIN

IF salary > 60000 THEN

RAISE NOTICE 'High salary';

ELSIF salary > 40000 THEN

RAISE NOTICE 'Average salary';

ELSE

RAISE NOTICE 'Low salary';

END IF;

END;

**CASE statement sintaksisi qanday?**

**CASE** statement bir nechta shartlarni tekshirish va ularga mos keluvchi kod bloklarini bajarish uchun ishlatiladi.

sql

Copy code

CASE

WHEN condition1 THEN

-- kod

WHEN condition2 THEN

-- boshqa kod

ELSE

-- boshqa kod

END CASE;

Masalan:

sql

Copy code

DECLARE

grade CHAR := 'B';

BEGIN

CASE grade

WHEN 'A' THEN

RAISE NOTICE 'Excellent';

WHEN 'B' THEN

RAISE NOTICE 'Good';

WHEN 'C' THEN

RAISE NOTICE 'Fair';

ELSE

RAISE NOTICE 'Poor';

END CASE;

END;

**PL/pgSQL da qanday usullarda sikl hosil qilsa bo'ladi?**

PL/pgSQLda bir nechta sikl (loop) konstruksiyalari mavjud:

* **LOOP**: Shart qo'yilmagan cheksiz sikl.
* **WHILE**: Shartga asoslangan sikl.
* **FOR**: Ro'yxat yoki interval asosida iteratsiya qiluvchi sikl.

**LOOP**

sql

Copy code

LOOP

-- kod

EXIT WHEN condition; -- shart

END LOOP;

Masalan:

sql

Copy code

DECLARE

counter INT := 0;

BEGIN

LOOP

EXIT WHEN counter > 10;

counter := counter + 1;

RAISE NOTICE 'Counter: %', counter;

END LOOP;

END;

**WHILE**

sql

Copy code

WHILE condition LOOP

-- kod

END LOOP;

Masalan:

sql

Copy code

DECLARE

counter INT := 0;

BEGIN

WHILE counter <= 10 LOOP

RAISE NOTICE 'Counter: %', counter;

counter := counter + 1;

END LOOP;

END;

**FOR**

* **Numeric FOR loop**:

sql

Copy code

FOR counter IN [REVERSE] start..end LOOP

-- kod

END LOOP;

Masalan:

sql

Copy code

BEGIN

FOR counter IN 1..10 LOOP

RAISE NOTICE 'Counter: %', counter;

END LOOP;

END;

* **FOREACH loop** (array yoki dynamic cursor asosida):

sql

Copy code

FOREACH element IN ARRAY array\_name LOOP

-- kod

END LOOP;

Masalan:

sql

Copy code

DECLARE

num\_array INT[] := ARRAY[1, 2, 3, 4, 5];

element INT;

BEGIN

FOREACH element IN ARRAY num\_array LOOP

RAISE NOTICE 'Element: %', element;

END LOOP;

END;

Bu PL/pgSQL dasturlash tilidagi asosiy konstruksiyalar va ularning ishlatilishi bo'yicha qo'llanmalar.

• RAISE statement ning maqsadi sintaksisi ?

• RAISE statement da nechta level bor ?

• RAISE statement da default level qanday ?

• Exception handling qanday amalga oshiriladi PL/pgSQL da ?

• function nima ?

• procedure nima ?

• in, out, inout parameter lar farqilari ?

• Function dan qanday qilib table qaytarish mumin ?

• Trigger nima ?

**RAISE Statement**

**RAISE** statement PL/pgSQLda xabarlarni ko'rsatish, xatolarni qayta ishlash va jurnal (log) yozuvlarini qo'shish uchun ishlatiladi.

**RAISE Statement Sintaksisi**

sql

Copy code

RAISE level 'format' [, expression, ...];

**RAISE Statement darajalari**

1. **DEBUG**
2. **LOG**
3. **INFO**
4. **NOTICE**
5. **WARNING**
6. **EXCEPTION**

**RAISE Statement da Default Level**

NOTICE darajasi default hisoblanadi.

Masalan:

sql

Copy code

BEGIN

RAISE NOTICE 'This is a notice message';

RAISE WARNING 'This is a warning message';

RAISE EXCEPTION 'This is an exception message';

END;

**Exception Handling**

PL/pgSQLda xatolarni qayta ishlash EXCEPTION bloklari orqali amalga oshiriladi.

**Sintaksisi**

sql

Copy code

BEGIN

-- kod

EXCEPTION

WHEN exception\_name THEN

-- xatolarni qayta ishlash

WHEN OTHERS THEN

-- boshqa xatolarni qayta ishlash

END;

Masalan:

sql

Copy code

BEGIN

-- Biror kod

RAISE EXCEPTION 'An error occurred';

EXCEPTION

WHEN others THEN

RAISE NOTICE 'Caught an exception';

END;

**Function nima?**

**Function** PL/pgSQLda bir yoki bir nechta amallarni bajarish uchun ishlatiladigan qayta foydalaniladigan kod blokidir. Funksiya natija qaytaradi.

**Function Sintaksisi**

sql

Copy code

CREATE OR REPLACE FUNCTION function\_name(parameters)

RETURNS return\_type AS $$

DECLARE

-- o'zgaruvchilar

BEGIN

-- kod

RETURN result;

END;

$$ LANGUAGE plpgsql;

Masalan:

sql

Copy code

CREATE OR REPLACE FUNCTION get\_employee\_salary(emp\_id INT)

RETURNS NUMERIC AS $$

DECLARE

salary NUMERIC;

BEGIN

SELECT salary INTO salary FROM employees WHERE employee\_id = emp\_id;

RETURN salary;

END;

$$ LANGUAGE plpgsql;

**Procedure nima?**

**Procedure** ham PL/pgSQLda qayta foydalaniladigan kod blokidir, lekin u natija qaytarmaydi. Protseduralar odatda biror-bir jarayonni bajarish uchun ishlatiladi.

**Procedure Sintaksisi**

sql

Copy code

CREATE OR REPLACE PROCEDURE procedure\_name(parameters)

LANGUAGE plpgsql AS $$

BEGIN

-- kod

END;

$$;

Masalan:

sql

Copy code

CREATE OR REPLACE PROCEDURE update\_salary(emp\_id INT, new\_salary NUMERIC)

LANGUAGE plpgsql AS $$

BEGIN

UPDATE employees SET salary = new\_salary WHERE employee\_id = emp\_id;

END;

$$;

**IN, OUT, INOUT Parametrlari Farqlari**

* **IN**: Kiruvchi parametr, funktsiyaga/protseduraga kiritiladi va o'zgarmaydi.
* **OUT**: Chiquvchi parametr, funktsiyadan/protseduradan qiymat qaytarish uchun ishlatiladi.
* **INOUT**: Kiruvchi va chiquvchi parametr, funktsiyaga/protseduraga kiritiladi va o'zgartirilishi mumkin.

Masalan:

sql

Copy code

CREATE OR REPLACE PROCEDURE process\_salary(IN emp\_id INT, OUT new\_salary NUMERIC)

LANGUAGE plpgsql AS $$

BEGIN

SELECT salary \* 1.1 INTO new\_salary FROM employees WHERE employee\_id = emp\_id;

END;

$$;

**Function dan Table Qaytarish**

PL/pgSQLda funksiya jadvallarni qaytarishi mumkin.

**Sintaksisi**

sql

Copy code

CREATE OR REPLACE FUNCTION function\_name(parameters)

RETURNS TABLE (column1 datatype, column2 datatype, ...)

AS $$

BEGIN

RETURN QUERY

SELECT column1, column2, ...

FROM table\_name

WHERE condition;

END;

$$ LANGUAGE plpgsql;

Masalan:

sql

Copy code

CREATE OR REPLACE FUNCTION get\_department\_employees(dept\_id INT)

RETURNS TABLE (employee\_id INT, employee\_name TEXT) AS $$

BEGIN

RETURN QUERY

SELECT employee\_id, employee\_name

FROM employees

WHERE department\_id = dept\_id;

END;

$$ LANGUAGE plpgsql;

**Trigger nima?**

**Trigger** - bu jadvalga qo'shilgan, yangilangan yoki o'chirilgan ma'lumotlarga javoban avtomatik ravishda bajariladigan protsedura.

**Trigger Sintaksisi**

1. **Trigger Function** yaratish:

sql

Copy code

CREATE OR REPLACE FUNCTION trigger\_function\_name()

RETURNS TRIGGER AS $$

BEGIN

-- kod

RETURN NEW; -- yoki RETURN OLD;

END;

$$ LANGUAGE plpgsql;

1. **Trigger** yaratish:

sql

Copy code

CREATE TRIGGER trigger\_name

{BEFORE | AFTER} {INSERT | UPDATE | DELETE}

ON table\_name

FOR EACH ROW

EXECUTE FUNCTION trigger\_function\_name();

Masalan:

1. Trigger funksiyasini yaratish:

sql

Copy code

CREATE OR REPLACE FUNCTION audit\_log()

RETURNS TRIGGER AS $$

BEGIN

INSERT INTO audit(employee\_id, action, action\_time)

VALUES (NEW.employee\_id, TG\_OP, now());

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

1. Trigger yaratish:

sql

Copy code

CREATE TRIGGER employee\_audit

AFTER INSERT OR UPDATE OR DELETE

ON employees

FOR EACH ROW

EXECUTE FUNCTION audit\_log();

Bu asosiy PL/pgSQL tushunchalari va sintaksislari, ular yordamida siz ma'lumotlar bazasida murakkab logik amallarni amalga oshirishingiz mumkin.

4o

• ACID nima ?

• Transaction nima ?

• rollback ning vazifasi nima ?

• savepoint ning vazifasi nima ?

• Table ni join qilish nima degani ?

• Postgresql da necha xil join bor ?

• Natural join qanday join ?

• index nima uhcun ishlatiladi ?

• unique index bilan index ni farqi nima ?

**ACID nima?**

**ACID** bu ma'lumotlar bazasi tranzaksiyalari uchun muhim bo'lgan to'rtta xususiyatni ifodalovchi qisqartmadir:

1. **Atomicity (Atomiklik)**: Tranzaksiya yoki butunlay bajariladi yoki umuman bajarilmaydi. Agar tranzaksiya davomida xatolik yuzaga kelsa, barcha o'zgartirishlar bekor qilinadi.
2. **Consistency (Konsistentlik)**: Tranzaksiya ma'lumotlar bazasini bir konsistent holatdan boshqa bir konsistent holatga o'tkazishi kerak.
3. **Isolation (Izolyatsiya)**: Parallel tranzaksiyalar bir-biriga ta'sir qilmasligi kerak, ya'ni har bir tranzaksiya o'z holatida izolyatsiya qilinadi.
4. **Durability (Barqarorlik)**: Tranzaksiya muvaffaqiyatli tugagandan so'ng, uning natijalari doimiy ravishda saqlanadi va hatto tizim nosozliklarida ham saqlanadi.

**Transaction nima?**

**Transaction** - bu ma'lumotlar bazasida birgalikda bajarilishi kerak bo'lgan bir yoki bir nechta SQL operatsiyalarining birligi. Tranzaksiya ma'lumotlar bazasining yaxlitligini saqlashga yordam beradi.

**Rollback ning vazifasi nima?**

**ROLLBACK** - bu tranzaksiyani bekor qilish va barcha o'zgartirishlarni bekor qilish uchun ishlatiladigan buyruq. Agar tranzaksiya davomida xatolik yuzaga kelsa yoki tranzaksiyani bekor qilish zarurati tug'ilsa, ROLLBACK yordamida barcha o'zgartirishlar orqaga qaytariladi.

**Savepoint ning vazifasi nima?**

**SAVEPOINT** - bu tranzaksiya ichida vaqtinchalik belgi yoki nuqta yaratadi. Bu nuqtadan tranzaksiyaning ma'lum qismi bekor qilinishi mumkin, bu esa tranzaksiyalarning aniq qismlarini boshqarish imkonini beradi.

**Sintaksisi**

sql

Copy code

SAVEPOINT savepoint\_name;

-- Biror kod

ROLLBACK TO SAVEPOINT savepoint\_name;

**Table ni join qilish nima degani?**

**Join qilish** - bu ikki yoki undan ortiq jadvallarni ularning umumiy ustunlariga asoslanib birlashtirish jarayonidir. Join amaliyoti yordamida jadvallardagi ma'lumotlar birgalikda ko'rib chiqiladi.

**PostgreSQL da necha xil join bor?**

PostgreSQLda bir necha turdagi joinlar mavjud:

1. **INNER JOIN**
2. **LEFT (OUTER) JOIN**
3. **RIGHT (OUTER) JOIN**
4. **FULL (OUTER) JOIN**
5. **CROSS JOIN**
6. **NATURAL JOIN**

**Natural Join qanday join?**

**NATURAL JOIN** - bu ikki jadvalni ularning umumiy ustunlari asosida avtomatik ravishda birlashtiruvchi join turi. Bunda umumiy ustunlar aniqlanadi va ularga asoslangan holda jadvallar birlashtiriladi.

Masalan:

sql

Copy code

SELECT \* FROM table1

NATURAL JOIN table2;

**Index nima uchun ishlatiladi?**

**Index** - bu jadvallardagi ma'lumotlarga tezroq kirishni ta'minlash uchun yaratilgan ma'lumotlar strukturasi. Indekslar yordamida ma'lumotlar bazasida qidiruv va filtr operatsiyalari tezroq amalga oshiriladi.

**Unique Index bilan Index ni farqi nima?**

* **Index**: Har qanday ustun yoki ustunlar to'plamiga qo'llanishi mumkin va ma'lumotlar bazasidagi qidiruvlarni tezlashtiradi.
* **Unique Index**: Oddiy indeks kabi ishlaydi, lekin u faqat takrorlanmas qiymatlar uchun ishlatiladi, ya'ni ushbu indeks o'rnatilgan ustunda takrorlanmas qiymatlar bo'lishini ta'minlaydi.

Masalan, UNIQUE cheklovi bilan yaratilgan indeks:

sql

Copy code

CREATE UNIQUE INDEX index\_name ON table\_name(column\_name);

Bu indeks faqat takrorlanmas qiymatlar saqlanishini ta'minlaydi, shu bilan birga qidiruv operatsiyalarini tezlashtiradi.

• Postgresql da ROLE qanday tushuncha ?

• Role yaratish sintaksisi qanday ?

• grant ning vazifasi nima ?

• revoke ning vazifasi qanday ?

• Tablespace nima ?

• Qanday qilib custom tablespace yaratish mumkin ?

• Schema nima ?

**PostgreSQL da ROLE qanday tushuncha?**

**ROLE** PostgreSQLda foydalanuvchilar va guruhlarni boshqarish uchun ishlatiladigan tushuncha. Role foydalanuvchilar yoki boshqa rollar guruhini ifodalashi mumkin. Rollar yordamida ma'lumotlar bazasiga kirish huquqlarini boshqarish va taqsimlash osonlashadi.

**Role yaratish sintaksisi qanday?**

Role yaratish uchun CREATE ROLE buyruğidan foydalaniladi.

**Sintaksisi**

sql

Copy code

CREATE ROLE role\_name;

Masalan, readonly nomli role yaratish uchun:

sql

Copy code

CREATE ROLE readonly;

**GRANT ning vazifasi nima?**

**GRANT** buyruği biror role yoki foydalanuvchiga muayyan huquqlarni taqdim etish uchun ishlatiladi. Bu huquqlar jadvallarga, sxemalarga, funksiyalarga va boshqa ma'lumotlar bazasi obyektlariga taalluqli bo'lishi mumkin.

**Sintaksisi**

sql

Copy code

GRANT privilege ON object TO role\_name;

Masalan, readonly roliga employees jadvalida SELECT huquqini berish uchun:

sql

Copy code

GRANT SELECT ON employees TO readonly;

**REVOKE ning vazifasi qanday?**

**REVOKE** buyruği biror role yoki foydalanuvchidan muayyan huquqlarni olib tashlash uchun ishlatiladi.

**Sintaksisi**

sql

Copy code

REVOKE privilege ON object FROM role\_name;

Masalan, readonly rolidan employees jadvalida SELECT huquqini olib tashlash uchun:

sql

Copy code

REVOKE SELECT ON employees FROM readonly;

**Tablespace nima?**

**Tablespace** - bu ma'lumotlar bazasidagi obyektlar (masalan, jadvallar va indekslar) saqlanadigan diskda mantiqiy joylashuv. Tablespace yordamida ma'lumotlarni diskda boshqarish va saqlash optimallashtiriladi.

**Qanday qilib custom tablespace yaratish mumkin?**

Custom tablespace yaratish uchun CREATE TABLESPACE buyruğidan foydalaniladi.

**Sintaksisi**

sql

Copy code

CREATE TABLESPACE tablespace\_name LOCATION 'directory\_path';

Masalan, /mnt/data katalogida custom\_space nomli tablespace yaratish uchun:

sql

Copy code

CREATE TABLESPACE custom\_space LOCATION '/mnt/data';

**Schema nima?**

**Schema** - bu ma'lumotlar bazasidagi obyektlarni (jadvallar, ko'rinishlar, funksiya va boshqalar) guruhlash va mantiqiy tashkil qilish uchun ishlatiladigan tushuncha. Har bir ma'lumotlar bazasida bir nechta sxemalar bo'lishi mumkin, va har bir sxemada ma'lumotlar bazasi obyektlari bo'lishi mumkin.

Masalan, public - bu PostgreSQLda default sxema.

**Schema yaratish sintaksisi**

sql

Copy code

CREATE SCHEMA schema\_name;

Masalan, hr nomli sxema yaratish uchun:

sql

Copy code

CREATE SCHEMA hr;

**Xulosa**

* **ROLE**: PostgreSQLda foydalanuvchilar va guruhlarni boshqarish uchun ishlatiladi.
* **Role yaratish**: CREATE ROLE role\_name;
* **GRANT**: Huquqlarni taqdim etadi, masalan, GRANT SELECT ON table TO role\_name;
* **REVOKE**: Huquqlarni olib tashlaydi, masalan, REVOKE SELECT ON table FROM role\_name;
* **Tablespace**: Ma'lumotlarni diskda mantiqiy tashkil qilish uchun joy.
* **Tablespace yaratish**: CREATE TABLESPACE tablespace\_name LOCATION 'directory\_path';
* **Schema**: Ma'lumotlar bazasi obyektlarini guruhlash uchun ishlatiladi.
* **Schema yaratish**: CREATE SCHEMA schema\_name;

Bu tushunchalar va buyruqlar PostgreSQL ma'lumotlar bazasini boshqarishda va optimallashtirishda muhim ahamiyatga ega.

4o