Podstawa

Tworzenie bazy danych

```
CREATE ROLE car_portal_role LOGIN;

CREATE DATABASE car_portal
    WITH
    ENCODING 'UTF8'
    LC_COLLATE 'en_US.UTF-8'
    LC_CTYPE 'en_US.UTF-8'
    OWNER car_portal_role;
```

Tabele

`CREATE TABLE`

Mają 4 różne typy. Są dedykowane do konkretnych zadań. Jest możliwość klonowania zadań.

Materializacja wyników przez `SELECT`.

Tabele trwałe - zwykłe tabele, cykl od utworzenia do usunięcia

```
CREATE TABLE employees (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100),
    position VARCHAR(100)
);
```

Tabele tymczasowe - tabele, cykl życia to sesja użytkownika.

```
CREATE TEMPORARY TABLE temp_employees (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100),
    position VARCHAR(100)
);
```

Tabele bez logowanie *unlogged* - szybsze niż tabele trwałe, dane nie są zapisywane do plików WAL. Nie są odporne na awarie i nie mogą być replikowane na węzeł podrzędny

```
CREATE UNLOGGED TABLE my_unlogged_table (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100)
);
```

Tabele podrzędne - tabela, która dziedziczy jedną lub więcej tabel. Dziedzizenie jest często używane z wykluczeniem ograniczeń (constraint exclusion) w celu fizycznego partycjonowania danych na dysku twardym.W PostgreSQL można utworzyć tabelę nadrzędną (partycjonowaną) i następnie utworzyć tabele podrzędne.

```
-- Tabela nadrzędna (partycjonowana):
CREATE TABLE sales (
    sale_id SERIAL PRIMARY KEY,
    sale_date DATE,
    amount DECIMAL
) PARTITION BY RANGE (sale_date);

-- Tabele podrzędne
CREATE TABLE sales_2020 PARTITION OF sales
    FOR VALUES FROM ('2020-01-01') TO ('2020-12-31');

CREATE TABLE sales_2021 PARTITION OF sales
    FOR VALUES FROM ('2021-01-01') TO ('2021-12-31');
```

Podstawa

Logowanie do postgreSQL

```
sudo -i -u user_name
```

Wylogowanie z postgreSQL

```
exit
```

Wejście do psql

```
psql

psql -U user_name -d nazwa_bazy -h host -p port

psql -U użytkownik -d baza_danych -h 192.168.1.100 -p 5432
```

Host domyślnie **localhost**

Port domyślnie **5432**

Nie trzeba wykorzystywać wszystkich parametrów jeśli są domyślne

Polecenia w terminalu (nie w psql)

Usuwanie istniejącej bazy danych PostgreSQL

```
dropdb nazwa_bazy;
```

Tworzenie nowej bazy danych PostgreSQL

```
createdb nazwa_bazy;
```

Usuwanie usera

```

```

```
dropuser name
```

Tworzenie usera

```
createuser name
```

Tworzy ponownie indeksy

```
reindexdb
```

PostgreSQL `psql` Commands

General Commands

Command	Description
\bind [PARAM]...	Ustawienie parametrów zapytania
\copyright	Wyświetlenie warunków użytkowania i dystrybucji PostgreSQL
\crosstabview [COLUMNS]	Wykonanie zapytania i wyświetlenie wyników w formie tabeli przestawnej
\errverbose	Wyświetlenie ostatniego komunikatu o błędzie z maksymalną szczegółowością
\g [(OPTIONS)] [FILE]	Wykonaj zapytanie (i wyślij wynik do pliku lub |pipe); \g bez argumentów jest równoważne średnikowi
\gdesc	Opisanie wyniku zapytania bez jego wykonania
\gexec	Wykonanie zapytania, a następnie wykonanie każdego wyniku z tego zapytania
\gset [PREFIX]	Wykonanie zapytania i zapisanie wyniku w zmiennych psql
\gx [(OPTIONS)] [FILE]	Jak \g, ale wymusza rozszerzony tryb wyjściowy
\q	Zakończenie sesji psql
\watch [[i=]SEC] [c=N]	Wykonanie zapytania co SEC sekund, do N razy

Help Commands

Command	Description
\? [commands]	Wyświetlenie pomocy dla komend backslash, pokazuje wszystkie komendy po \

Command	Description
\? options	Wyświetlenie pomocy dla opcji wiersza poleceń psql
\? variables	Wyświetlenie pomocy dla zmiennych specjalnych
\h [NAME]	Pomoc dotycząca składni komend SQL, * dla wszystkich komend

Query Buffer Commands

Command	Description
\e [FILE] [LINE]	Edytowanie bufora zapytania (lub pliku) za pomocą zewnętrznego edytora
\ef [FUNCNAME [LINE]]	Edytowanie definicji funkcji za pomocą zewnętrznego edytora
\ev [VIEWNAME [LINE]]	Edytowanie definicji widoku za pomocą zewnętrznego edytora
\p	Wyświetlenie zawartości bufora zapytania
\r	Zresetowanie (wyczyszczenie) bufora zapytania
\s [FILE]	Wyświetlenie historii lub zapisanie jej do pliku
\w FILE	Zapisanie bufora zapytania do pliku

Input/Output Commands

Command	Description
\copy ...	Wykonanie komendy SQL COPY z danymi przesyłanymi do klienta
\echo [-n] [STRING]	Zapisanie ciągu znaków do standardowego wyjścia (-n dla braku nowej linii)
\i FILE	Wykonanie komend z pliku
\ir FILE	Jak \i, ale w odniesieniu do lokalizacji bieżącego skryptu
\o [FILE]	Wysłanie wyników zapytania do pliku lub \p
\qecho [-n] [STRING]	Zapisanie ciągu znaków do strumienia \o (-n dla braku nowej linii)
\warn [-n] [STRING]	Zapisanie ciągu znaków do standardowego błędu (-n dla braku nowej linii)

Conditional Commands

Command	Description
\if EXPR	Rozpoczęcie bloku warunkowego
\elif EXPR	Alternatywa w bieżącym bloku warunkowym
\else	Ostatnia alternatywa w bieżącym bloku warunkowym
\endif	Zakończenie bloku warunkowego

Informational Commands

Command	Description
\d[S+]	Wyświetl tabelę, widok, sekwencję
\d[S+] NAME	Opis tabeli, widoku, sekwencji lub indeksu
\da[S] [PATTERN]	Wyświetl agregaty
\dA[+] [PATTERN]	Wyświetl metody dostępu
\dAc[+] [AMPTRN [TYPEPTRN]]	Wyświetl klasy operatorów
\dAf[+] [AMPTRN [TYPEPTRN]]	Wyświetl rodziny operatorów
\dAo[+] [AMPTRN [OPFPTRN]]	Wyświetl operatory rodzin operatorów
\dAp[+] [AMPTRN [OPFPTRN]]	Wyświetl funkcje wspierające rodziny operatorów
\db[+] [PATTERN]	Wyświetl przestrzenie tabel
\dc[S+] [PATTERN]	Wyświetl konwersje
\dconfig[+] [PATTERN]	Wyświetl parametry konfiguracyjneameters
\dC[+] [PATTERN]	Wyświetl rzutowania
\dd[S] [PATTERN]	Wyświetl opisy obiektów, które nie są wyświetlane w innych miejscach
\dD[S+] [PATTERN]	Wyświetl domeny
\ddp [PATTERN]	Wyświetl domyślne uprawnienia
\dE[S+] [PATTERN]	Wyświetl tabele zewnętrzne
\des[+] [PATTERN]	Wyświetl serwery zewnętrzne
\det[+] [PATTERN]	Wyświetl tabele zewnętrzne
\deu[+] [PATTERN]	Wyświetl mapowania użytkowników
\dew[+] [PATTERN]	Wyświetl opakowania danych zewnętrznych
\df[anptw][S+] [FUNCPTRN [TYPEPTRN ...]]	Wyświetl funkcje (agregatowe, normalne, procedury, wyzwalacze, okna).
\dF[+] [PATTERN]	Wyświetl konfiguracje wyszukiwania tekstu
\dFd[+] [PATTERN]	Wyświetl słowniki wyszukiwania tekstu
\dFp[+] [PATTERN]	Wyświetl analizatory wyszukiwania tekstu
\dFt[+] [PATTERN]	Wyświetl szablony wyszukiwania tekstu
\dg[S+] [PATTERN]	Wyświetl role.
\di[S+] [PATTERN]	Wyświetl indeksy

Command	Description
\dl[+]	Wyświetl obiekty duże, takie jak w \lo_list
\dL[S+] [PATTERN]	Wyświetl języki proceduralne
\dm[S+] [PATTERN]	Wyświetl widoki materializowane
\dn[S+] [PATTERN]	Wyświetl schematy
\do[S+] [OPPTRN [TYPEPTRN [TYPEPTRN]]]	Wyświetl operatory
\dO[S+] [PATTERN]	Wy swietl zestawienia
\dp[S] [PATTERN]	Wyświetl uprawnienia dostępu do tabel, widoków i sekwencji
\dP[itn+] [PATTERN]	Wyświetl [tylko indeksy/tabele] zagnieżdżone w partycjach [n=nested]
\drds [ROLEPTRN [DBPTRN]]	Wyświetl ustawienia ról dla bazy danych
\drg[S] [PATTERN]	Wyświetl przyznane uprawnienia do ról
\dRp[+] [PATTERN]	Wyświetl publikacje replikacji
\dRs[+] [PATTERN]	Wyświetl subskrypcje replikacji
\ds[S+] [PATTERN]	Wyświetl sekwencje
\dt[S+] [PATTERN]	Wyświetl tabele
\dT[S+] [PATTERN]	Wyświetl typy danych
\du[S+] [PATTERN]	Wyświetl role
\dv[S+] [PATTERN]	Wyświetl widoki
\dx[+] [PATTERN]	Wyświetl rozszerzenia
\dX [PATTERN]	Wyświetl statystyki rozszerzone
\dy[+] [PATTERN]	Wyświetl wyzwalacze zdarzeń triggers
\l[+] [PATTERN]	Wyświetl bazy danych
\sf[+] FUNCNAME	Wyświetl definicję funkcji
\sv[+] VIEWNAME	Wyświetl definicję widoku
\z[S] [PATTERN]	To samo, co \dp (uprawnienia dostępu).

Large Objects Commands

Command	Description
\lo_export LOBOID FILE	Zapisz duży obiekt do pliku
\lo_import FILE [COMMENT]	Wczytaj duży obiekt z pliku

Command	Description
\lo_list[+]	Wyświetl duże obiekty
\lo_unlink LOBOID	Usuń duży obiekt

Formatting Commands

Command	Description
\a	Przełącz między wyjściem w formacie niezsynchronizowanym i zsynchronizowanym
\C [STRING]	Ustaw tytuł tabeli lub usuń, jeśli brak
\f [STRING]	Ustaw lub wyświetl separator pól w wyjściu niezsynchronizowanym
\H	Przełącz tryb wyjścia w formacie HTML
\pset [NAME [VALUE]]	Ustaw opcje wyjścia tabeli (np. granice, kolumny, format CSV).
\t [on|off]	Pokaż tylko wiersze (domyślnie wyłączone).
\T [STRING]	Ustaw atrybuty znacznika HTML <table>, lub wyłącz je, jeśli nie są ustawione
\x [on|off|auto]	Przełącz rozszerzone wyjście. (domyślnie wyłączone)

Connection Commands

Command	Description
\c[onnect] {[DBNAME|- USER|- HOST|- PORT|-] | conninfo}	Połączenie z nową bazą danych (domyślnie "postgres")
\conninfo	Wyświetlenie informacji o bieżącym połączeniu
\encoding [ENCODING]	Pokazanie lub ustawienie kodowania klienta
\password [USERNAME]	Zmiana hasła użytkownika w sposób bezpieczny

Operating System Commands

Command	Description
\cd [DIR]	Zmiana bieżącego katalogu roboczego
\getenv PSQLVAR ENVVAR	Pobranie zmiennej środowiskowej
\setenv NAME [VALUE]	Ustawienie lub usunięcie zmiennej środowiskowej
\timing [on|off]	Przełączanie wyświetlania czasu wykonania komend (obecnie wyłączone)
! [COMMAND]	Wykonanie polecenia w powłoce lub uruchomienie interaktywnej powłoki

Variable Commands

Command	Description
\prompt [TEXT] NAME	Monitoruj użytkownika o ustawienie zmiennej wewnętrznej
\set [NAME [VALUE]]	Ustaw zmienną wewnętrzną lub wypisz wszystkie, jeśli nie ma parametrów
\unset NAME	usuń zmienną wewnętrzną

```
\set nazwa_wartosci 'wartosc' -- tworzenie zmiennej
SELECT * FROM users WHERE username = :'nazwa_wartosci'; -- użycie w
zapytaniu
```

Aliases

```
\alias ll \! ls -l

\alias mytables \dt public.*
```

alias **ll** - wyświela nam wszystkie pliki

alias **mytables** - wyświetla nam wszystkie tabele w schemacie

Typy danych

Typy numeryczne

Name	Storage Size	Description	Range
smallint	2 bytes	small-range integer	-32768 to +32767
integer	4 bytes	typical choice for integer	-2147483648 to +2147483647
bigint	8 bytes	large-range integer	-9223372036854775808 to +9223372036854775807
decimal	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
numeric	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
real	4 bytes	variable-precision, inexact	6 decimal digits precision
double precision	8 bytes	variable-precision, inexact	15 decimal digits precision
smallserial	2 bytes	small autoincrementing integer	1 to 32767
serial	4 bytes	autoincrementing integer	1 to 2147483647
bigserial	8 bytes	large autoincrementing integer	1 to 9223372036854775807

Typy monetarne (pieniądze)

Name	Storage Size	Description	Range
money	8 bytes	currency amount	-92233720368547758.08 to +92233720368547758.07

Typy tekstowe

Name	Description

Name	Description
character varying(n), varchar(n)	variable-length with limit
character(n), char(n), bpchar(n)	fixed-length, blank-padded
bpchar	variable unlimited length, blank-trimmed
text	variable unlimited length

Typy binarne

Name	Storage Size	Description
bytea	1 or 4 bytes plus the actual binary string	variable-length binary string

Typy daty/czasu

Name	Storage Size	Description	Low Value	High Value	Resolution
timestamp [(p)] [without time zone]	8 bytes	both date and time (no time zone)	4713 BC	294276 AD	1 microsecond
timestamp [(p)] with time zone	8 bytes	both date and time, with time zone	4713 BC	294276 AD	1 microsecond
date	4 bytes	date (no time of day)	4713 BC	5874897 AD	1 day
time [(p)] [without time zone]	8 bytes	time of day (no date)	00:00:00	24:00:00	1 microsecond
time [(p)] with time zone	12 bytes	time of day (no date), with time zone	00:00:00+1559	24:00:00-1559	1 microsecond
interval [fields] [(p)]	16 bytes	time interval	-178000000 years	178000000 years	1 microsecond

Typy bool'owskie

Name	Storage Size	Description
boolean	1 byte	state of true or false

Typy enumeryczne

```
CREATE TYPE mood AS ENUM ('sad', 'ok', 'happy');
CREATE TABLE person (
    name text,
    current_mood mood
);
INSERT INTO person VALUES ('Moe', 'happy');
SELECT * FROM person WHERE current_mood = 'happy';
 name | current_mood
------+--------------
 Moe  | happy
```

Typy geometryczne

Name	Storage Size	Description	Representation
point	16 bytes	Point on a plane	(x,y)
line	24 bytes	Infinite line	{A,B,C}
lseg	32 bytes	Finite line segment	[(x1,y1),(x2,y2)]
box	32 bytes	Rectangular box	(x1,y1),(x2,y2)
path	16+16n bytes	Closed path (similar to polygon)	((x1,y1),...)
path	16+16n bytes	Open path	[(x1,y1),...]
polygon	40+16n bytes	Polygon (similar to closed path)	((x1,y1),...)
circle	24 bytes	Circle	<(x,y),r> (center point and radius)

Typy adresów internetowych

Name	Storage Size	Description
cidr	7 or 19 bytes	IPv4 and IPv6 networks
inet	7 or 19 bytes	IPv4 and IPv6 hosts and networks
macaddr	6 bytes	MAC addresses
macaddr8	8 bytes	MAC addresses (EUI-64 format)

Typy pozostałe

- Typy wyszukiwania tekstowego (tsvector, tsquery)
- Typ UUID
- Typ XML
- Typ JSON (string, number, boolean, null)

- Typ tablicowy

```
CREATE TABLE sal_emp (
    name            text,
    pay_by_quarter  integer[],
    schedule        text[][]
);
```

Schematy

Tworzenie, usuwanie schematu

```
CREATE SCHEMA myschema;
DROP SCHEMA myschema;
```

Aby uzyskać dostęp do obiektów w schemacie, należy wpisać kwalifikowaną nazwę: `schema.table` lub bardziej ogólną `database.schema.table`

Tworzenie tabeli na schemacie

```
CREATE TABLE myschema.mytable (
 ...
);
```

Aby usunąć schemat z wszystkimi obiektami w nim zawartymi

```
DROP SCHEMA myschema CASCADE;
```

Aby utworzyć schemat należący do kogoś innego

```
CREATE SCHEMA schema_name AUTHORIZATION user_name;
```

Wykorzystanie schematów:

- kontrola autoryzacji
- organizacja obiektów bazy danych
- utrzymywanie zewnętrznego kodu SQL

Utworzenie schematu dla takiej samej roli (2), w (1) można zmienić na inną rolę

```
CREATE SCHEMA car_portal_app AUTHORIZATION car_portal_app

CREATE SCHEMA AUTHORIZATION car_portal_app
```

Widoki

Widok to nazwana kwerenda lub osłona wokół instrukcji *SELECT*

Widoki są podstawowym elementem relacyjnych baz danych, które można porównać do metod w klasach UML

Jesteśmy wstanie dodawać, usuwać wiersze widoku.

Widoki upraszczają zapytania i zwiększania modularność kodu

Korzyśći widoków:

- uproszczenie złożonych zapytań
- poprawa wydajności dzięki buforowaniu wyników
- redukcja ilości kodu SQL
- integracja z językami obiektowymi dzięki widokom aktualizowanym
- zastosowanie mechanizmów autoryzacji na poziomie wiersza
- łatwe prowadzenie zmian bez koniecznosci wdrażania nowego oprogramowania
- implementacja warstwy abstrakcji między bazą danych a aplikacjami wysokopoziomowymi

Różnice od procedów składowanych

ich zależności są utrzymywane w bazie danych. są bardziej wydajne. modyfikacja widoku może być zabroniona z powodu efektu kaskadowych

Tworzenie widoku

```
CREATE [OR REPLACE] [TEMP | TEMPORARY] [RECURSIVE] VIEW nazwa_widoku
AS
```

Rodzaje widoków

- tymczasowe : automatycznie usuwane na końcu sesji usera;

```
CREATE TEMP VIEW temp_user_view AS
SELECT
    id,
    username,
    email
FROM
    users
WHERE
    active=true;
```

- rekurencyjne obsługują rekurencje podobnie jak funckje w jezykach wysokiego poziomu. dane hierarchiczne

```
CREATE RECURSIVE VIEW employee_hierarchy AS
-- Poziom bazowy: pracownicy bez przełożonych
SELECT id AS employee_id, name AS employee_name, manager_id
FROM employees
WHERE manager_id IS NULL
UNION ALL
-- Poziom rekurencyjny: pracownicy podlegli przełożonym
SELECT e.id AS employee_id, e.name AS employee_name, e.manager_id
FROM employees e
JOIN employee_hierarchy eh ON e.manager_id = eh.employee_id;
```

- Aktualizowane widoki (materializowane) - przechowuja fizycznie dane w tabelach i mogą być okresowo odświeżane. Używane w dużych i czesto wykonywanych zapytan

```
CREATE MATERIALIZED VIEW sales_summary AS
SELECT product_id, SUM(quantity) AS total_sales
FROM sales
GROUP BY product_id;

...

REFRESH MATERIALIZED VIEW nazwa_widoku;
```

Aby usunąć widok z zależnościami

```
DROP VIEW nazwa_widoku_zaleznego;
DROP VIEW nazwa_widoku_pierwotnego;


DROP VIEW nazwa_widoku CASCADE  -- usuwa odrazu widok pierwotny
```

widoki materializowane mają możliwość automatycznego aktualizoanie, co oznacza ze mozna robic INSERT UPDATE DELEE

warunki dla aktualizowanych widoków:

- oparty na tabeli
- brak: distinct, with, group by, offset, having, limit, union, except, intersect

Widok edytowalny - pozwala edycje danych

```

```

```
CREATE VIEW editable_view_customers AS
SELECT id,username,email,balance,is_actibe
FROM accounts
WHERE is_active = TRUE AND balance >0
WITH LOCAL CHECK OPTION;
```

Widok tylko do odczytu

```
CREATE VIEW readonly_view_customers AS
SELECT DISTINCT id,username,email,balance,is_actibe
FROM accounts
WHERE balance =0;
```

INDEKSY

obiekty fizyczne przyspieszające odczyt danych

- optymalizacja wydajnosci zapytan
- weryfikacja ograniczeń tj. UNIQUE

indeksy można tworzyć na tabeli lub na widoku

Indeks unikalny CREATE UNIQUE INDEX nazwa ON nazwa_tabeli

`index scan` możemy uzyc dzieki indeksowi aby zoptymalizowac wyszukiwanie zamiast wykonywac pelne skanowanie tabeli

plan wykonania `EXPLAIN` select powoduje wyswietlenie `query plan` w terminalu

TYPY INDEKSÓW

- unikalny Indeks zapewniający unikalność wartości w kolumnach. Zapewnienie unikalności danych w kolumnach, często używane dla kolumn PRIMARY KEY i UNIQUE.

```
CREATE UNIQUE INDEX idx_unique_example ON tabela_name (kolumna_name);
```

- B-tree (domyślny) drzewo zbilansowane, gdy operacje porównawcze. Zapytania z operatorami porównania (np. =, <, >, <=, >=, BETWEEN, IN). Idealny do indeksowania liczb, dat i tekstów.

```
CREATE INDEX idx_btree_example ON tabela_name (kolumna_name);
```

- Hash Index - Indeks oparty na funkcji skrótu (hash). do obsługi równości,choć b-tree zwykle spełnia te potrzeby, ale nie są transakcyjnie bezpieczne i nie są replikowane w węzłach slave podczas replikacji.

```
CREATE INDEX idx_hash_example ON tabela_name USING HASH (kolumna_name);
```

- GIN (generalized inverted index)-Indeks odwrócony, używany do złożonych struktur danych (tablice, JSONB). , przydatny gdy wiele wartosci zlozonych musi byc mapowanych na jeden wiersz np. tablice, json. Zastosowanie tablice, wyszukiwanie pełnotekstowe.

```
CREATE INDEX idx_gin_example ON tabela_name USING GIN (kolumna_name);
```

- GiST (generalized search tree) - dla zrównoważonych struktur drzewiastych. Zastosowanie: typy geometryczne, wyszukiwanie pełnotekstowe, wyszukiwanie przestrzenne

```
CREATE INDEX idx_gist_example ON tabela_name USING GiST (kolumna_name);
```

- BRIN (Block Range Index) - przydatny do duzych tabel, gdzie dane uporzadkowane. Indeks na zakresach bloków danych. Jest wolniejszy od b-tree ale jest mniejszu. Zastosowanie: przechowywanie danych na dużą skalę.

```
CREATE INDEX idx_brin_example ON tabela_name USING BRIN (kolumna_name);
```

- SP-GiST -Indeks podzielony przestrzennie, przeznaczony do danych przestrzennych. Indeksowanie danych, które są podzielone przestrzennie (np. punkty w przestrzeni). Idealny dla danych o rozproszonej strukturze.

```
CREATE INDEX idx_spgist_example ON tabela_name USING SPGiST (kolumna_name);
```

- Partial Index -Indeks na wybranych danych, spełniających określony warunek. Indeksowanie tylko części danych w tabeli, które spełniają określony warunek (np. tylko wartości większe niż 100).

```
CREATE INDEX idx_partial_example ON tabela_name (kolumna_name) WHERE
kolumna_name > 100;
```

- Expression Index- Indeks tworzony na wyrażeniu lub funkcji.Przyspiesza zapytania, które operują na funkcjach lub wyrażeniach (np. LOWER(my_column)), a nie na samej kolumnie.

```
CREATE INDEX idx_expression_example ON tabela_name (LOWER(kolumna_name));
```

- Foreign Key Index- Indeks tworzony dla kolumny klucza obcego.Używany do poprawy wydajności operacji JOIN oraz zapewnienia integralności referencyjnej między tabelami.

```
CREATE INDEX idx_fk_example ON tabela_name (kolumna_name);
```

Funkcje tekstowe

Funkcja/Operator	Opis	Przykład użycia
`text || text → text`	Łączy dwa ciągi znaków.	`'Post' || 'greSQL' → PostgreSQL`
`text || anynonarray → text`	Łączy ciąg znaków z innym typem danych, który zostaje przekonwertowany na tekst.	`'Value: ' || 42 → Value: 42`
`btrim ( string text [, characters text ] ) → text`	Usuwa najdłuższy ciąg zawierający tylko znaki z `characters` (domyślnie spacje) z początku i końca ciągu.	`btrim('xyxtrimyyx', 'xyz') → trim`
`text IS [NOT] [form] NORMALIZED → boolean`	Sprawdza, czy ciąg jest w określonym formacie normalizacji Unicode.	`U&'\0061\0308bc' IS NFD NORMALIZED → t`
`bit_length ( text ) → integer`	Zwraca liczbę bitów w ciągu (8 razy długość w oktetach).	`bit_length('jose') → 32`
`char_length ( text ) → integer`	Zwraca liczbę znaków w ciągu.	`char_length('josé') → 4`
`lower ( text ) → text`	Konwertuje ciąg na małe litery, zgodnie z regułami lokalizacji bazy danych.	`lower('TOM') → tom`
`lpad ( string text, length integer [, fill text ] ) → text`	Rozszerza ciąg do długości `length`, dodając `fill` (domyślnie spacje) z lewej strony. Jeśli ciąg jest dłuższy, zostaje przycięty.	`lpad('hi', 5, 'xy') → xyxhi`
`ltrim ( string text [, characters text ] ) → text`	Usuwa najdłuższy ciąg zawierający tylko znaki z `characters` (domyślnie spacje) z początku ciągu.	`ltrim('zzzytest', 'xyz') → test`
`normalize ( text [, form ] ) → text`	Konwertuje ciąg do określonej formy normalizacji Unicode.	`normalize(U&'\0061\0308bc', NFC) → U&'\00E4bc'`
`octet_length ( text ) → integer`	Zwraca liczbę bajtów w ciągu.	`octet_length('josé') → 5 (jeśli kodowanie serwera to UTF8)`

Funkcja/Operator	Opis	Przykład użycia
`octet_length ( character ) → integer`	Zwraca liczbę bajtów w ciągu, nie usuwając trailing spaces.	`octet_length('abc '::character(4)) → 4`
`overlay ( string text PLACING newsubstring text FROM start integer [ FOR count integer ] ) → text`	Zastępuje część ciągu, zaczynając od `start`-tej pozycji i długości `count`, nowym ciągiem.	`overlay('Txxxxas' placing 'hom' from 2 for 4) → Thomas`
`position ( substring text IN string text ) → integer`	Zwraca pierwszy indeks wystąpienia `substring` w `string`, lub 0, jeśli nie jest obecny.	`position('om' in 'Thomas') → 3`
`rpad ( string text, length integer [, fill text ] ) → text`	Rozszerza ciąg do długości `length`, dodając `fill` (domyślnie spacje) z prawej strony. Jeśli ciąg jest dłuższy, zostaje przycięty.	`rpad('hi', 5, 'xy') → hixyx`
`rtrim ( string text [, characters text ] ) → text`	Usuwa najdłuższy ciąg zawierający tylko znaki z `characters` (domyślnie spacje) z końca ciągu.	`rtrim('testxxzx', 'xyz') → test`
`substring ( string text [ FROM start integer ] [ FOR count integer ] ) → text`	Wyciąga podciąg z `string`, zaczynając od `start`-tej pozycji i kończąc po `count` znakach.	`substring('Thomas' from 2 for 3) → hom`
`substring ( string text FROM pattern text ) → text`	Wyciąga pierwszy podciąg pasujący do wzorca POSIX regular expression.	`substring('Thomas' from '...$') → mas`
`substring ( string text SIMILAR pattern text ESCAPE escape text ) → text`	Wyciąga pierwszy podciąg pasujący do wzorca SQL regular expression.	`substring('Thomas' similar '%#"o_a#"_' escape '#') → oma`
`trim ( [ LEADING | TRAILING | BOTH ] [ characters text ] FROM string text ) → text`	Usuwa najdłuższy ciąg znaków z początku, końca lub obu stron `string` zawierający tylko znaki z `characters` (domyślnie spacje).	`trim(both 'xyz' from 'yxTomxx') → Tom`
`unicode_assigned ( text ) → boolean`	Zwraca `true`, jeśli wszystkie znaki w ciągu mają przypisane kody Unicode.	`unicode_assigned('abc') → true`

Funkcja/Operator	Opis	Przykład użycia
`upper ( text ) → text`	Konwertuje ciąg na wielkie litery, zgodnie z regułami lokalizacji bazy danych.	`upper('tom') → TOM`

PL/pgSQL Function

PL/pgSQL to proceduralny język programowania umożliwiający rozszerzenie funkcjonalności PostgreSQL poprzez tworzenie funkcji,procedur,wyzwalaczy.

PL/pgSQL umożliwia:

- tworzenie funkcji i procedur z wykorzystaniem logiki proceduralnej
- definiowanie zmiennych lokalnych
- obsługa wyjątków
- użycie pętli, instrukcji warunkowych

Instalacja PL/pgSQL

```
CREATE EXTEMSION IF NOT EXISTS plpgsql;
```

Elementy funkcji

określa typ zwracanej funkcji blok kodu z logiką instrukcja która określa watość zwracaną przez funckje język w ktorym jest napisana funckja

Przykładowe funkcje

```
CREATE OR REPLACE FUNCTION example_function() RETURNS TEXT AS $$
BEGIN
    RETURN 'HELLO';
END;
$$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE FUNCTION add_number(a INT, b INT) RETURNS INT AS $$
BEGIN
    RETURN a+b;
END;
$$ LANGUAGE plpgsql;

SELECT add_number(10,30);
-- 40
```

```
CREATE OR REPLACE FUNCTION get_all_users() RETURNS TABLE(user_id INT,
user_name TEXT) AS $$
BEGIN
    RETURN QUERY SELECT id,name FROM users;
```

```
END;
$$ LANGUAGE plpgsql;

SELECT * FROM get_all_users();
```

Funkcje z wyjątkami

```
CREATE OR REPLACE FUNCTION calcuate_tax(price NUMERIC, tax_rate NUMERIC
DEFAULT 23)
RETURNS NUMERIC AS $$
BEGIN
    IF b=0 THEN
        RETURN NULL;
    ELSE
        RETURN a/b;
    END IF;
END;
$$ LANGUAGE plpgsql;
```

Best practies

- dokładne określenie typów danych parametrów i wartości zwracanych
- obaługa wyjątków EXCEPTION
- unikanie ciężkich operacji- powinny być zoptymalizowane i unikać długich pętli
- testowanie funkcji z różnymi zestawami danych
- uważać na rekursje
- uważać na operacje zmieniające dane
- uważać na nieefektywne zapytania w pętlach

JĘZYKI PLSQL

- SQL proste operacje,łatwe
- PL/pgSQL logika,pętle warunki
- C wymaga kompilacji, wysoka wydajność
- PL/Python ogromne korzyści, numpy,pandas język plpython3u
- PL/Perl elastyczny do manipulacji tekstu i operacji na ciągach znaków
- PL/Tcl do wykonywania operacji proceduralnych rzadko używany
- PL/V8 PL/JavaSript S
- PL/R do analiz statystycznych i wizualizacji danych

Jakie języki `SELECT lanname AS language, lanpltrusted AS trusted FROM pg_language;`

Jak zainstalowac

```
CREATE EXTENSION plpgsql;
CREATE EXTENSION plpython3u;
```

ROLE

Tworzenie i usuwanie roli

Jako komenda sql

```
CREATE ROLE name;
DROP ROLE name;
```

W terminalu

```
createuser name
dropuser name
```

Tworzenie roli z logowaniem

```
CREATE ROLE name  LOGIN;
CREATE USER name;
```

`CREATE USER` jest równoważne z `CREATE ROLE`, z wyjątkiem tego że `CREATE USER` zawiera `LOGIN` domyślnie, podczas gdy `CREATE ROLE` nie.

Role

- login privilege

```
CREATE ROLE name LOGIN;
```

- superuser status

```
CREATE ROLE name SUPERUSER;
```

Superużytkownik bazy danych omija wszystkie kontrole uprawnień, z wyjątkiem prawa do logowania. Jest to niebezpieczny przywilej i nie należy go używać nieostrożnie; najlepiej jest wykonywać większość swojej pracy jako rola, która nie jest superużytkownikiem.

- database creation

```
CREATE ROLE name CREATEDB;
```

Rola musi mieć jawnie nadane uprawnienia do tworzenia baz danych (oprócz superużytkowników, ponieważ omijają oni wszystkie kontrole uprawnień).

- role creation

```
CREATE ROLE name CREATEROLE;
```

Rola musi mieć jawnie nadane uprawnienia do tworzenia kolejnych ról (oprócz superużytkowników, ponieważ omijają one wszystkie kontrole uprawnień). Aby utworzyć taką rolę, użyj . Rola z uprawnieniami może zmieniać i usuwać role, które zostały przyznane użytkownikowi za pomocą opcji.

- initiating replication

```
CREATE ROLE name REPLICATION LOGIN;
```

Rola musi mieć jawne uprawnienia do inicjowania replikacji strumieniowej (oprócz superużytkowników, ponieważ omijają oni wszystkie kontrole uprawnień).

- password

```
CREATE ROLE name PASSWORD 'string';
ALTER ROLE name WITH PASSWORD 'nowe_hasło';
```

Hasło jest istotne tylko wtedy, gdy metoda uwierzytelniania klienta wymaga od użytkownika podania hasła podczas łączenia się z bazą danych. Metody uwierzytelniania passwordi md5 wykorzystują hasła. Hasła bazy danych są oddzielne od haseł systemu operacyjnego.

- inheritance of privileges

```
CREATE ROLE name NOINHERIT;
```

Rola dziedziczy uprawnienia ról, których jest członkiem, domyślnie. Jednak aby utworzyć rolę, która nie dziedziczy uprawnień domyślnie,

- bypassing row-level security

```
CREATE ROLE name BYPASSRLS;
```

Rola musi mieć jawne pozwolenie na ominięcie każdej polityki zabezpieczeń na poziomie wiersza (RLS)

- connection limit

```
CREATE ROLE name CONNECTION LIMIT 'integer';
```

Limit połączeń może określać, ile równoczesnych połączeń może utworzyć rola. -1 (domyślnie) oznacza brak limitu.

Role predefiniowane

Rola	Dozwolony dostęp
pg_read_all_data	Odczyt wszystkich danych (tabele, widoki, sekwencje), jakby użytkownik miał prawa SELECT do tych obiektów oraz prawa USAGE do wszystkich schematów, nawet jeśli nie ma ich explicite. Ta rola nie ma atrybutu BYPASSRLS ustawionego. Jeśli RLS jest używany, administrator może ustawić BYPASSRLS na rolach, którym ta rola jest przydzielona.
pg_write_all_data	Zapis wszystkich danych (tabele, widoki, sekwencje), jakby użytkownik miał prawa INSERT, UPDATE i DELETE do tych obiektów oraz prawa USAGE do wszystkich schematów, nawet jeśli nie ma ich explicite. Ta rola nie ma atrybutu BYPASSRLS ustawionego. Jeśli RLS jest używany, administrator może ustawić BYPASSRLS na rolach, którym ta rola jest przydzielona.
pg_read_all_settings	Odczyt wszystkich zmiennych konfiguracyjnych, nawet tych, które normalnie są widoczne tylko dla superużytkowników.
pg_read_all_stats	Odczyt wszystkich widoków pg_stat_* i używanie różnych rozszerzeń związanych ze statystykami, nawet tych, które normalnie są widoczne tylko dla superużytkowników.
pg_stat_scan_tables	Wykonywanie funkcji monitorujących, które mogą blokować tabele na poziomie ACCESS SHARE, potencjalnie na długi czas.
pg_monitor	Odczyt/wykonywanie różnych widoków i funkcji monitorujących. Ta rola jest członkiem pg_read_all_settings, pg_read_all_stats i pg_stat_scan_tables.
pg_database_owner	Brak dostępu. Członkostwo składa się z właściciela bieżącej bazy danych.
pg_signal_backend	Wysyłanie sygnału do innego backendu w celu anulowania zapytania lub zakończenia jego sesji.
pg_read_server_files	Zezwala na odczyt plików z dowolnej lokalizacji, do której baza danych ma dostęp na serwerze, za pomocą funkcji COPY i innych funkcji dostępu do plików.

Rola	Dozwolony dostęp
pg_write_server_files	Zezwala na zapis do plików w dowolnej lokalizacji, do której baza danych ma dostęp na serwerze, za pomocą funkcji COPY i innych funkcji dostępu do plików.
pg_execute_server_program	Zezwala na uruchamianie programów na serwerze bazy danych jako użytkownik, pod którym działa baza danych, za pomocą funkcji COPY i innych funkcji umożliwiających uruchamianie programów po stronie serwera.
pg_checkpoint	Zezwala na wykonanie polecenia CHECKPOINT.
pg_maintain	Zezwala na wykonanie VACUUM, ANALYZE, CLUSTER, REFRESH MATERIALIZED VIEW, REINDEX oraz LOCK TABLE na wszystkich relacjach, jakby użytkownik miał prawa MAINTAIN do tych obiektów, nawet jeśli nie ma ich explicite.
pg_use_reserved_connections	Zezwala na używanie zarezerwowanych slotów połączeń poprzez reserved_connections.
pg_create_subscription	Zezwala użytkownikom z uprawnieniami CREATE na bazie danych na wykonanie polecenia CREATE SUBSCRIPTION.

Nadawanie roli predefiniowanej

```
CREATE ROLE nowy_uzytkownik WITH LOGIN PASSWORD 'haslo';
GRANT pg_monitor TO nowy_uzytkownik;
```

Uprawnienia nadawanie, usuwanie

stworzenie nowej roli ``sql CREATE ROLE nowy_uzytkownik WITH LOGIN PASSWORD 'haslo';

```
nadanie uprawnień roli predefiniowanych do roli
```sql
GRANT pg_monitor TO nowy_uzytkownik;
GRANT pg_monitor, pg_read_all_data TO nowy_uzytkownik;
```

Rodzaje uprawnień do roli

- na poziomie bazy danych
  - CONNECT – pozwala na połączenie z bazą danych.
  - CREATE – pozwala na tworzenie nowych obiektów (tabel, widoków itp.) w bazie danych.
  - TEMPORARY – pozwala na tworzenie tymczasowych obiektów w czasie trwania sesji.
```

```
GRANT CONNECT, CREATE ON DATABASE nazwa_bazy TO nowy_uzytkownik;
```

- na poziomie tabeli
  - SELECT – pozwala na odczyt danych z tabeli.
  - INSERT – pozwala na wstawianie nowych wierszy do tabeli.
  - UPDATE – pozwala na aktualizowanie istniejących wierszy w tabeli.
  - DELETE – pozwala na usuwanie wierszy z tabeli.
  - TRUNCATE – pozwala na usuwanie wszystkich wierszy z tabeli.
  - REFERENCES – pozwala na tworzenie kluczy obcych, które odnoszą się do tej tabeli.
  - TRIGGER – pozwala na tworzenie i zarządzanie triggerami dla tabeli.

```
GRANT SELECT, INSERT ON TABLE employees TO nowy_uzytkownik;
GRANT UPDATE ON TABLE employees TO nowy_uzytkownik;
```

- na poziomie widoku
  - SELECT – pozwala na odczyt danych z widoku.

```
GRANT SELECT ON VIEW employee_view TO nowy_uzytkownik;
```

- na poziomie funkcji
  - EXECUTE – pozwala na wykonywanie funkcji

```
GRANT EXECUTE ON FUNCTION my_function TO nowy_uzytkownik;
```

- na poziomie schematu
  - USAGE – pozwala na używanie obiektów w schemacie (np. wywoływanie funkcji, odwoływanie się do tabel).
  - CREATE – pozwala na tworzenie nowych obiektów w schemacie

```
GRANT USAGE ON SCHEMA public TO nowy_uzytkownik;
GRANT CREATE ON SCHEMA public TO nowy_uzytkownik;
```

Aby usunąć uprawnienia zamiast `GRANT` trzeba napisać `REVOKE`

```

```

```
REVOKE USAGE, CREATE ON SCHEMA public FROM nowy_uzytkownik;
```

Template

Template 0

`Template0` to specjalny szablon, który zawiera minimalną konfigurację bazy danych, bez dodatkowych rozszerzeń i zmian, które mogłyby być dodane do `template1`.

- Cel: template0 zapewnia czystą bazę danych, bez żadnych rozszerzeń ani dodatkowych obiektów.
- Zakaz modyfikacji: Nie można zmieniać zawartości template0. Jest to czysta baza danych, której zawartość jest stała.
- Przeznaczenie: Używa się jej, gdy chcesz, aby nowa baza danych była absolutnie czysta, bez żadnych rozszerzeń, które mogą być obecne w template1.

Przykład utworzenia bazy danych na podstawie template0

```
CREATE DATABASE nowa_baza TEMPLATE template0;
```

template0:

- Zawiera minimalną konfigurację.
- Nie zawiera rozszerzeń ani dodatkowych obiektów.
- Jest w pełni czysta i niezmieniana po zainstalowaniu PostgreSQL.

Template 1

`Template1` to standardowy szablon, który zawiera wszystkie rozszerzenia i modyfikacje, które zostały dodane do systemu przez administratorów lub przez samego użytkownika.

- Cel: `template1` jest domyślnym szablonem do tworzenia nowych baz danych. Zawiera ona domyślną konfigurację, rozszerzenia i inne zmiany, które zostały wprowadzone po instalacji PostgreSQL.
- Modyfikacja: Można wprowadzać zmiany w `template1`. Na przykład, instalowanie rozszerzeń w `template1` spowoduje, że te rozszerzenia będą również obecne w nowych bazach danych tworzonych na jej podstawie.
- Przeznaczenie: Jeśli chcesz, aby nowa baza zawierała te same rozszerzenia i konfiguracje co `template1`, to jest to odpowiedni wybór.

Przykład utworzenia bazy danych na podstawie `template1`

```
CREATE DATABASE nowa_baza;
```

`Template1` jest używana domyślnie, jeśli nie podasz jawnie szablonu podczas tworzenia nowej bazy danych.

template1:

- Zawiera rozszerzenia, takie jak pg_stat_statements i inne, które mogą być dodane przez administratorów.
- Jest modyfikowana przez użytkowników, którzy chcą, aby wszystkie nowe bazy danych dziedziczyły te zmiany.
- Jest używana jako domyślny szablon dla nowych baz danych.

Kiedy wprowadzisz zmiany do bazy `template1`, na przykład zainstalujesz nowe rozszerzenie, nowe bazy danych utworzone na podstawie `template1` będą miały te zmiany.

Statystyka

1. Wyświetlanie statystyk tabeli

Aby wyświetlić statystyki dla tabeli, można użyć polecenia:

Skopiuj kod `\d+ nazwa_tabeli` To polecenie wyświetli dodatkowe informacje o tabeli, w tym statystyki związane z jej kolumnami, takie jak liczba wierszy i indeksów.

2. Wyświetlanie statystyk kolumny

Możesz sprawdzić statystyki dla poszczególnych kolumn, np.:

```
SELECT * FROM pg_stats WHERE tablename = 'nazwa_tabeli';
```

W wynikach zobaczysz dane dotyczące kolumny, takie jak:

`n_distinct`: szacowana liczba unikalnych wartości w kolumnie. `most_common_values (MCV)`: najczęściej występujące wartości w kolumnie. `null_frac`: udział wartości NULL. `avg_width`: średnia szerokość kolumny.

3. Wyświetlanie rozszerzonych statystyk

Aby zobaczyć rozszerzone statystyki, które mogą zawierać informacje o zależnościach między kolumnami, użyj polecenia:

`\dX` Rozszerzone statystyki mogą obejmować:

`Schema`: Schemat, w którym znajdują się statystyki. `Name`: Nazwa obiektu rozszerzonych statystyk. `Definition`: Definicja wyrażenia statystyki. `Ndistinct`: Liczba unikalnych wartości. `Dependencies`: Zależności od innych obiektów.

4. Zbieranie statystyk

Aby zebrać (lub zaktualizować) statystyki w bazie danych, użyj polecenia:

```
ANALYZE
```

To polecenie zbiera statystyki dla wszystkich tabel w bazie danych lub dla pojedynczej tabeli:

```
ANALYZE nazwa_tabeli;
```

Proces zbierania statystyk jest używany przez planistę zapytań do optymalizacji wykonania zapytań.

5. Podstawowe informacje o statystykach w bazie

Aby wyświetlić ogólne informacje o tabelach, ich statystykach, a także inne szczegóły, możesz skorzystać z zapytania do systemowych tabel, np.:

```
SELECT * FROM pg_stat_user_tables;
```

To zapytanie zwróci informacje o statystykach użytkownika, takie jak liczba odczytów i zapisów w tabelach.

6. Sprawdzanie wydajności zapytań

Aby zbierać informacje o wydajności zapytań (np. czas wykonania), włącz statystyki zapytań:

```
SET track_activities = true;
```

Następnie możesz sprawdzić aktywność zapytań:

```
SELECT * FROM pg_stat_activity;
```

7. Wyświetlanie statystyk indeksów

Aby uzyskać informacje o indeksach w bazie danych, użyj polecenia:

`\di` Możesz także zapytać o szczegóły dotyczące indeksów w systemowej tabeli:

```
SELECT * FROM pg_indexes WHERE tablename = 'nazwa_tabeli';
```

Te polecenia pozwolą Ci monitorować, analizować i optymalizować wydajność zapytań w PostgreSQL za pomocą statystyk.

Konfiguracja

Ustawnianie ilości połączeń do bazy dla roli

Tworzenie roli z ograniczeniem liczby jednoczesnych połączeń dla tej roli.

```
CREATE ROLE name CONNECTION LIMIT 10;
```

Ustawianie globalnego limitu połączeń

```
SHOW config_file; -- pokazuje ścieżkę do pliku
-- w pliku postresql.conf zmień
max_connections = 100
```

Ustawianie limitu połączeń dla bazy danych

```
ALTER DATABASE nazwa_bazy CONNECTION LIMIT 50; -- ustawienie limitu dla
bazy
SELECT datname, datconnlimit FROM pg_database; -- sprawdzenie bieżącego
limitu
```

Tabele katalogowe *pg_catalog*

Zwykłe tabele na których możemy używać `SELECT`,`UPDATE`,`DELETE`. Nie zaleca się modyfikacji ręcznej, chyba że w wyjątkowych sytuacjach.

```
SELECT datconnlimit FROM pg_database WHERE datname="nazwa_bazy";        //
wyświetlenie limitu połączeń z bazą danych
ALTER DATABASE nazwa_bazy CONNECTION LIMIT 1
//zmiana limitu połączeń na 1. -1 dla wszystkich
UPDATE pg_database SET dataconnlimit=-1 WHERE datname='nazwa_bazy';
//nie zalecania zmiana limitu połączeń, ponieważ  przy nieprawidłowej
wartości nie wyrzuca błędu np. "-2"
```

Plik postresql.conf

- można zmienić port - domyślny 5432
- maksywalną liczbę połączeń - domyślnie 100
- adresy IP, na których serwer nasłuchuje połączeń (listen_addresses)

- logowanie i monitorowanie (zbieranie logów, katalog z logami, nazwa plików logów, rejestrowanie zapytań sql)
- zarządzanie transakcjami (domyślny poziom izolacji transakcji)
- bezpieczeństwo (metoda szyfrowanai haseł, włączenie połączeń szyfrowanych)

Plik pg_hba.conf

- rodzaje połączenia -local - połączenie lokalne -host - połączenie przez TCP/IP -hostssl - połączenie przez TCP/IP wymagające SSL
- adres
    - adres IP klienta lub zakres adresów
- nazwa-bazy
    - all wszystkie bazy
- metoda uwierzetylnienia
    - trust - bez uwierzetylnienia
    - md5 - hasło szyfrowane md5
    - password -uwierzytelnienie za pomocą hasła w tekście
    - peer - uwierzytenienie przez porównanie nazwy usera systemowego
    - scram-sha-256 - hasło w formacie sha256
    - reject - odrzucenie połączenia
    - cert - certyfikat klienta SSL

Plik pg_catalog

`pg_catalog` to systemowy schemat w PostgreSQL, który zawiera wbudowane tabele i widoki systemowe. Jest to integralna część bazy danych PostgreSQL i przechowuje metadane dotyczące struktury bazy danych, takie jak informacje o tabelach, kolumnach, indeksach, funkcjach, a także o uprawnieniach i konfiguracjach.

Ustawnienia PostgreSQL

- Replikacja (Replication) - Konfiguracja replikacji danych pomiędzy serwerami
- Dzienniki Write-Ahead Logs (WAL) - Zarządzanie logami
- Zużycie zasobów (Resource consuption) - Kontrola nad zasobami sprzętowymi tj. RAM, CPU przeznaczonych na działanie serwera
- Planowanie zapytań (Query planning) - optymalizacja planów zapytań, aby uzyskać najlepszą wydajność
- Logowanie (Logging) - zapis logów na serwerze

- Uwierzetylnienie
- Zbieranie staystyk
- Zarządznie blokadami
- Obsługa błędów
- Opcje debugowania

Inne

Wyświetlenie wartości id

```
SELECT * FROM klienci_id_seq;

SELECT last_value FROM klienci_id_seq;
```

Wartość następnego id

```
SELECT nextval('klienci_id_seq');
```

Uprawnienia do baz

- `-C` pozwla tworzyc nowe schematy
- `-c` sprawdza uprawnienia roli przy próbie połączenie do bazy
- `-T` tworzenie tabel tymczasowych niszczone po zakończeniu sesji usera

jeśli nie ma wpisu w *Access privileges* to rola przypisna jest PUBLIC

Encoding pozwla na przechowywanie 1 lub wielobajtowych zestawów znaków tj *SQL_ACII* lub *UTF-8*.

Inne atrybuty baz danych

- Maintenance - *datafrozenxid* - do określenia czt baza wymaga procesu vacuum
- Zarządzanie przestrzenią dyskową- *dattablespace* -określa przestrzeń tabel dla baz danych
- Równoczesność- *datconnlimit* - liczba pozwolonych połączeń. Wartość -1 oznacza brak ograniczeń
- Ochrona- *datallowconn* - wyłącz możliwość połączenia z bazą danych, głównie w celu ochrony template0 przed modyfikacjami

---

Tablespace w bazach danych

Przestrzeń dyskowa którą wskazujemy gdzie nasze dane będą przechowywane. Każda baza/tabela może być na innym dysku, katalogu.

Utrzymaniowość i zarządzanie miejscem (Maintenance)

- Jeśli na dysku brakuje miejsca, można utworzyć tablespace na innej partycji. Dzięki temu możemy przenieść dane do nowej lokalizacji, unikając problemów związanych z brakiem miejsca

Optymalizacyjne

- Przenoszenie danych na szybsze dyski

Tworzenie tablespace

```
CREATE TABLESPACE fast_storage LOCATION '/mnt/ssd_partition/';
```

Używanie tablespace

```
CREATE TABLE moja_tabela(
    id SERIAL PRIMARY KEY;
    nazwa VARCHAR(255) NOT NULL;
)TABLESPACE fast_storage;
```

Ustawienia kontekstu w jaki sposób będziemy zmieniać wartości

- Internal(wewnętrzy) Nie można zmieniać bezpośrednio
- Postmaster Zmiana parametrów wiąże się z restartem postgres
- Sighup Trzeba wysłać sygnał po zmianie do procesu
- Backend Nie wymaga restartu i od razu powinny działać
- Superuser Przez komendę set, będą działać cały czas
- User W trakcie działanie bieżącej sesji

`SET` i `SHOW` służą do zmiany i sprawdzania wartości parametrów ustawień

Zmiana w `postgresql.conf` ma efekt globalny

Przeładowanie konfiguracji `SELECT pg_reload_conf();`

Pełne nazwy są czasochłonne, dlatego preferuje się korzystanie z nazw obiektów, które zawierają tylko nazwę obiektu bez schematu.

PostgreSQL umożliwia ustawienie `search_path` które ma ścieżkę wyszukiwania.

Modelowanie danych

Reprezentowanie pojedynczych encji

Diagramy ER

Obiekty owalne - proste

Obiekty prostikątne mają atrybuty złożone

Znaczenie modeli danych odgrywają rolę w utrzymaniu spójności danych w systemach współpracujących. Źle zdefiniowane encje mogą prowadzić do zamieszania i niespójności w całej organizacji. np. (klient vs kontrahent), moze prowadzić do nieporozumień.

Trzy modele danych:

- konceptualny
- logiczny - struktura danych okreslnonych technologii
- fizyczny - implementacja bazy danych

Model encji i relacji (ER)

Jest typem modelu danych konceptualnego zaprojektowanych do uchwycenia i reprezentowania encji oraz ich relacji.

Używany przez developerów i biznes

Praktyki w modelowaniu

- nadmiarowość danych - zbyt duze moga powodowac problemy tj. niespójnosc i degradacja wydajnosci
- saturacjia wartosci null - problem aby w JSON dany klucz nie przyjmuje wartosci NULL
- ścisłe powiązanie - ścisłe powiązanie miedzy enacjami moze prowadzic do sztywnych struktur, co utrudnia przyszle zmiany

pytania kontrolne check list

co jest kluczem głównym?

jaka domyślna wartość kolumny

jaki typ kolumny

ograniczenia

uprawnienia

klucze obce

cykl życia danych

jakie operacje są dozwolone

KLUCZE naturalne a zastepcze

klucze naturalne - oczywiste i rozpoznawalne - mogą się zmieniać. a użycie klucza zastępczego zapewnia ze referencja do innych wierszy nie zostanie utracony, poniewać klucz zastepczy sie nie zmienia.

błędne założenia o kluczach naturalnych

klucze zastepcze moga wspierac projekty bazy danych tymczasowej

klucze zastepcze czesto uzywaja kompaktowych typow danych (liczby calkowite)

klucz zastepcze sa generowane automatyczne co moze prowadzic do roznych wynikow w roznych bazach testowych

klucz zastepczy nie jest opisowy