SQL - podstawy

Definiowanie podstawowych
elementów tablic i proste zapytania

Tworzenie tabel

Nazwa

CREATE TABLE Nazwa(...);

Tworzenie tabel

Nazwa

kolumna1	kolumna2	kolumna3

CREATE TABLE Nazwa(k1,k2,...);

- nazwy kolumn

Tworzenie tabel

Nazwa

kolumna1	kolumna2	kolumna3
typ danych 1	typ danych 2	typ danych 3

CREATE TABLE Nazwa(k1 t1,k2 t2,...);

- nazwy kolumn
- typy danych w kolumnach: CHAR, VARCHAR, TEXT, INT, REAL, DATE, TIMESTAMP ;

Tworzenie tabel

Nazwa

kolumna1	kolumna2	kolumna3
typ danych 1	typ danych 2	typ danych 3
klucz w relacji: PRIMARY KEY	klucz obcy: REFERENCES...	inne: NOT NULL, DEFAULT, CHECK

CREATE TABLE Nazwa(k1 t1 f11 f12,k2 t2 f2,f3...);

- nazwy kolumn
- typy danych w kolumnach: CHAR, VARCHAR, TEXT, INT, REAL, DATE, TIMESTAMP ;
- więzy kolumn - klucze, klucze obce, NOT NULL, UNIQUE, DEFAULT, [CONSTRAINT nazwa] CHECK

Tworzenie tabel - przykład

CREATE TABLE ();

Tworzenie tabel - przykład

nick	grupa	data_rej

CREATE TABLE **Os** (
nick ,
grupa ,
data_rej);

Tworzenie tabel - przykład

nick	grupa	data_rej
nazwa - słowo	*nazwa - słowo*	*dzień rejestracji*

CREATE TABLE **Os** (
nick ,
grupa ,
data_rej);

Tworzenie tabel - przykład

nick	grupa	data_rej
varchar(20)	*varchar(20)*	*date*

CREATE TABLE **Os** (
nick *varchar(20)* ,
grupa *varchar(20)* ,
data_rej *date*);

Tworzenie tabel - typy kolumn

- **Typy teksu:** CHAR, CHAR(5), VARCHAR, VARCHAR(15), TEXT,
- **Typy liczbowe:** INT, REAL, FLOAT,...
- **Typy daty i czasu:** DATE, TIME, TIMESTAMP,...
- **Typy logiczne:** typ BOOLEAN przyjmuje wartości TRUE, FALSE i UNKNOWN
- **Typy binarne** (zdjęcia, muzyka,...): BLOB (BYTEA),...
- **SERIAL** - licznik automatycznie zwiększający wartość przy każdym odwołaniu;

Typy liczbowe

Typ	#bajt	Opis	Zakres
smallint	2	całkowita	-32768 .. 32767
integer	4	całkowita	-2147483648 .. 2147483647
bigint	8	całkowita	-9223372036854775808 .. 9223372036854775808
decimal		dokładna	(131072 cyfry).(16383 cyfry)
numeric		dokładna	(131072 cyfry).(16383 cyfry)
real	4	zmiennoprz.	6 cyfr dziesiętnych
double precision	8	zmiennoprz.	15 cyfr dziesiętnych
small-serial	2	autoincrement	1 .. 32767
serial	4	autoincrement	1 .. 2147483647
bigserial	8	autoincrement	1 .. 9223372036854775808

Typy liczbowe

Typ	#bajt	Opis	Zakres
smallint	2	całkowita	-32768 .. 32767
integer	4	całkowita	-2147483648 .. 2147483647
bigint	8	całkowita	-9223372036854775808 .. 9223372036854775808
decimal		dokładna	(131072 cyfry).(16383 cyfry)
numeric		dokładna	(131072 cyfry).(16383 cyfry)
real	4	zmiennoprz.	6 cyfr dziesiętnych
double precision	8	zmiennoprz.	15 cyfr dziesiętnych
small-serial	2	autoincrement	1 .. 32767
serial	4	autoincrement	1 .. 2147483647
bigserial	8	autoincrement	1 .. 9223372036854775808

Typy znakowe (tekstowe)

Typ	Opis
character varying(n), **varchar(n)**	zmienna długość do podanej maksymalnej n
character(n) **char(n)** **char = char(1)**	stała długość n, krótsze teksty uzupełnione spacjami
text, **varchar**	dowolna, zmienna długość tekstu

Typy znakowe (tekstowe)

Typ	Opis
character varying(n), **varchar(n)**	zmienna długość do podanej maksymalnej n
character(n) **char(n)** **char = char(1)**	stała długość n, krótsze teksty uzupełnione spacjami
text, **varachar**	dowolna, zmienna długość tekstu

Typy daty i czasu

Typ	Opis	Bajty
timestamp [without time zone]	data i czas z dokładnością do 1 mikrosek.	8
date	data (bez godziny)	4
time [without time zone]	godz:min:sek:mikrosek	12
interval	przedział czasu: 3 4:05:06.25 3 days 4 hours 5 minutes 6.25 seconds	12

Tworzenie tabel - typy kolumn

- **Typy tekstu:** CHAR, CHAR(5), VARCHAR, VARCHAR(15), TEXT,
- **Typy liczbowe:** INT, REAL, FLOAT,...
- **Typy daty i czasu:** DATE, TIME, TIMESTAMP,...
- **Typy logiczne:** typ BOOLEAN przyjmuje wartości TRUE, FALSE i UNKNOWN
- **Typy binarne** (zdjęcia, muzyka,...): BLOB (BYTEA),...
- **SERIAL** - licznik automatycznie zwiększający wartość przy każdym odwołaniu;
- **Typ wyliczeniowy** (wyk,cw,pr,rep)
- **Struktury:** rekord, tablica, obiekt geometryczny…
- **Inne:** adresy internetowe, XML, JSON

Tworzenie tabel - przykład

nick	**grupa**	**data_rej**
varchar(20)	*varchar(20)*	*date*
klucz główny	klucz obcy do tabeli Gr	wymagana data rejestracji

CREATE TABLE **Os** (
nick *varchar(20)* ,
grupa *varchar(20)* ,
data_rej *date*);

Tworzenie tabel - przykład

nick	**grupa**	**data_rej**
varchar(20)	*varchar(20)*	*date*
primary key	klucz obcy do tabeli Gr	wymagana data rejestracji

CREATE TABLE **Osoba** (
nick *varchar(20)* primary key,
grupa *varchar(20)* ,
data_rej *date*);

Tworzenie tabel - przykład

nick	**grupa**	**data_rej**
varchar(20)	*varchar(20)*	*date*
primary key	references Gr(id)	wymagana data rejestracji

CREATE TABLE **Osoba** (
nick *varchar(20)* primary key,
grupa *varchar(20)* references Gr(id),
data_rej *date*);

Tworzenie tabel - przykład

nick	**grupa**	**data_rej**
varchar(20)	*varchar(20)*	*date*
primary key	references Gr(id) on delete set null	wymagana data rejestracji

CREATE TABLE **Osoba** (
nick *varchar(20)* primary key,
grupa *varchar(20)* references Gr(id) on delete set null,
data_rej *date*);

Tworzenie tabel - przykład

nick	**grupa**	**data_rej**
varchar(20)	*varchar(20)*	*date*
primary key	references Gr(id) on delete set null	not null

CREATE TABLE **Osoba** (
nick *varchar(20)* primary key,
grupa *varchar(20)* references Gr(id) on delete set null,
data_rej *date* not null);

Tworzenie tabel - przykład

nick	**grupa**	**data_rej**
varchar(20)	*varchar(20)*	*date*
primary key	references Gr(id) on delete set null	not null default current_date

CREATE TABLE **Osoba** (
nick *varchar(20)* primary key,
grupa *varchar(20)* references Gr(id) on delete set null,
date_rej *date* not null default current_date);

Tworzenie tabel - więzy

Tworzenie tabel - więzy

Klucz obcy:
FOREIGN KEY grupa REFERENCES Gr(id)

Tworzenie tabel - więzy

Klucz obcy:

FOREIGN KEY grupa REFERENCES Gr(id)
ON DELETE [set null | **restrict** |set default| cascade]
ON UPDATE [set null | restrict |set default| **cascade**]

Tworzenie tabel - więzy

Klucz obcy:
FOREIGN KEY grupa REFERENCES Gr(id)
ON DELETE [set null | **restrict** |set default| cascade]
ON UPDATE [set null | restrict |set default| **cascade**]

Wartość domyślna:
DEFAULT 0, DEFAULT now()

Tworzenie tabel - więzy

Klucz obcy:
FOREIGN KEY grupa REFERENCES Gr(id)
ON DELETE [set null | **restrict** |set default| cascade]
ON UPDATE [set null | restrict |set default| **cascade**]

Wartość domyślna:
DEFAULT 0, DEFAULT now()

Pola niepuste:
NOT NULL;

Tworzenie tabel - więzy

Klucz obcy:
FOREIGN KEY grupa REFERENCES Gr(id)
ON DELETE [set null | **restrict** |set default| cascade]
ON UPDATE [set null | restrict |set default| **cascade**]

Wartość domyślna:
DEFAULT 0, DEFAULT now()

Pola niepuste:
NOT NULL;

Inne warunki:
CHECK (data_rej<=current_date);
CHECK (pewnosc BETWEEN 0 AND 100);

Tworzenie tabel - więzy nazwane

Tworzenie tabel - więzy nazwane

Nadawanie nazwy więzom:

```
CONSTRAINT pwnsc_lim CHECK (pewnosc BETWEEN 0
AND 100);
CONSTRAINT ref_grupa FOREIGN KEY (grupa)
REFERENCES Gr(id);
```

Tworzenie tabel - więzy nazwane

Nadawanie nazwy więzom:
CONSTRAINT pwnsc_lim CHECK (pewnosc BETWEEN 0 AND 100);
CONSTRAINT ref_grupa FOREIGN KEY (grupa) REFERENCES Gr(id);

Więz nazwany można usunąć:

- ALTER TABLE Os DROP CONSTRAINT ref_grupa

Tworzenie tabel - więzy nazwane

Nadawanie nazwy więzom:
CONSTRAINT pwnsc_lim CHECK (pewnosc BETWEEN 0 AND 100);
CONSTRAINT ref_grupa FOREIGN KEY (grupa) REFERENCES Gr(id);

Więz nazwany można usunąć:

- ALTER TABLE Os DROP CONSTRAINT ref_grupa

Więz nazwany można zawiesić do końca transakcji:

- SET CONSTRAINT pwnsc_lim DEFERRED
- SET CONSTRAINT pwnsc_lim IMMEDIATE

Tworzenie tabel - więzy nazwane

Nadawanie nazwy więzom:

CONSTRAINT pwnsc_lim CHECK (pewnosc BETWEEN 0 AND 100) **DEFERRABLE INITIALLY DEFERRED**;
CONSTRAINT ref_grupa FOREIGN KEY (grupa) REFERENCES Gr(id);

Więz nazwany można usunąć:

- ALTER TABLE Os DROP CONSTRAINT ref_grupa

Więz nazwany można zawiesić do końca transakcji:

- SET CONSTRAINT pwnsc_lim DEFERRED
- SET CONSTRAINT pwnsc_lim IMMEDIATE

Tworzenie tabel - więzy nazwane

Nadawanie nazwy więzom:

CONSTRAINT pwnsc_lim CHECK (pewnosc BETWEEN 0 AND 100) **DEFERRABLE INITIALLY IMMEDIATE**;
CONSTRAINT ref_grupa FOREIGN KEY (grupa) REFERENCES Gr(id);

Więz nazwany można usunąć:

- ALTER TABLE Os DROP CONSTRAINT ref_grupa

Więz nazwany można zawiesić do końca transakcji:

- SET CONSTRAINT pwnsc_lim DEFERRED
- SET CONSTRAINT pwnsc_lim IMMEDIATE

Klucze złożone i niuanse CHECK

Klucze złożone i niuanse CHECK

nick: *varchar(20)*	**moment**: *timestamp default now()*	**wpis**: *text*

```
CREATE TABLE Wpisy(
nick varchar(20) references Os(nick),
moment timestamp not null default now()
            CHECK (moment=now()),
wpis text CHECK (NOT substr('*',wpis)));
```

Klucze złożone i niuanse CHECK

nick: *varchar(20)*	**moment**: *timestamp default now()*	**wpis**: *text*
jeden fragment klucza	drugi fragment klucza	

```
CREATE TABLE Wpisy(
nick varchar(20) references Os(nick),
moment timestamp not null default now()
            CHECK (moment=now()),
wpis text CHECK (NOT substr('*',wpis)));
```

Klucze złożone i niuanse CHECK

nick: *varchar(20)*	**moment**: *timestamp default now()*	**wpis**: *text*
jeden fragment klucza	drugi fragment klucza	

```
CREATE TABLE Wpisy(
nick varchar(20) references Os(nick),
moment timestamp not null default now()
            CHECK (moment=now()),
wpis text CHECK (NOT substr('*',wpis)),
CONSTRAINT klucz_wpis_os PRIMARY KEY (nick,moment));
```

Klucze złożone i niuanse CHECK

nick: *varchar(20)*	**moment**: *timestamp default now()*	**wpis**: *text*	id: ***SERIAL***

```
CREATE TABLE Wpisy(
nick varchar(20) references Os(nick),
moment timestamp not null default now()
            CHECK (moment=now()),
wpis text CHECK (NOT substr('*',wpis)),
id SERIAL PRIMARY KEY);
```

Klucze złożone i niuanse CHECK

nick: *varchar(20)*	**moment**: *timestamp default now()*	**wpis**: *text*	id: ***SERIAL***

```
CREATE TABLE Wpisy(
nick varchar(20) references Os(nick),
moment timestamp not null default now()
        CHECK (moment=now()),
wpis text, CHECK (strpos(wpis, '*')=0)
id SERIAL PRIMARY KEY);
```

Klucze złożone i niuanse CHECK

nick: *varchar(20)*	**moment**: *timestamp default now()*	**wpis**: *text*	id: ***SERIAL***

```
CREATE TABLE Wpisy(
nick varchar(20) references Os(nick),
moment timestamp not null default now()
        CHECK (moment=now()),
wpis text, CHECK (strpos(wpis, '*')=0)      # wpis IS NULL ?
id SERIAL PRIMARY KEY);
```

Zmiana schematu tablicy

Zmiana schematu tablicy

- ALTER TABLE Os ADD [COLUMN] adres text;
- ALTER TABLE Os DROP [COLUMN] grupa;

Zmiana schematu tablicy

- ALTER TABLE Os ADD [COLUMN] adres text;
- ALTER TABLE Os DROP [COLUMN] grupa;

- ALTER TABLE Os ALTER adres SET DEFAULT 'Ełk';
- ALTER TABLE Os ALTER grupa DROP NOT NULL;

Zmiana schematu tablicy

- ALTER TABLE Os ADD [COLUMN] adres text;
- ALTER TABLE Os DROP [COLUMN] grupa;

- ALTER TABLE Os ALTER adres SET DEFAULT 'Ełk';
- ALTER TABLE Os ALTER grupa DROP NOT NULL;

- DROP TABLE Gr [CASCADE]; - konsekwencje dla tablic, które odwołują się do Gr przez klucz obcy;

Zmiana schematu tablicy

- ALTER TABLE Os ADD [COLUMN] adres text;
- ALTER TABLE Os DROP [COLUMN] grupa;

- ALTER TABLE Os ALTER adres SET DEFAULT 'Ełk';
- ALTER TABLE Os ALTER grupa DROP NOT NULL;

- DROP TABLE Gr [CASCADE]; - konsekwencje dla tablic, które odwołują się do Gr przez klucz obcy;

- ALTER TABLE Os ADD CONSTRAINT...
- ALTER TABLE Os DROP CONSTRAINT...
- SET CONSTRAINT ... DEFERRABLE

Wypełnianie tablicy

Wypełnienie tablicy

- INSERT INTO Os VALUES('Ala',NULL,now());
- INSERT INTO Os(nick) VALUES('Ola');
- INSERT INTO Os SELECT...

Wypełnienie tablicy

- INSERT INTO Os VALUES('Ala',NULL,now());
- INSERT INTO Os(nick) VALUES('Ola');
- INSERT INTO Os SELECT...

Wypełnienie tablicy

- INSERT INTO Os VALUES('Ala',NULL,now());
- INSERT INTO Os(nick) VALUES('Ola');
- INSERT INTO Os SELECT...

- DELETE FROM Os;
- DELETE FROM Os WHERE grupa IS NULL;

Wypełnienie tablicy

- INSERT INTO Os VALUES('Ala',NULL,now());
- INSERT INTO Os(nick) VALUES('Ola');
- INSERT INTO Os SELECT...

- DELETE FROM Os;
- DELETE FROM Os WHERE grupa IS NULL;

Wypełnienie tablicy

- INSERT INTO Os VALUES('Ala',NULL,now());
- INSERT INTO Os(nick) VALUES('Ola');
- INSERT INTO Os SELECT...

- DELETE FROM Os;
- DELETE FROM Os WHERE grupa IS NULL;

- UPDATE Os SET nick='Ala95' WHERE nick='Ala';
- UPDATE wpis SET moment=moment+100;

Wypełnienie tablicy

- INSERT INTO Os VALUES('Ala',NULL,now());
- INSERT INTO Os(nick) VALUES('Ola');
- INSERT INTO Os SELECT...

- DELETE FROM Os;
- DELETE FROM Os WHERE grupa IS NULL;

- UPDATE Os SET nick='Ala95' WHERE nick='Ala';
- UPDATE wpis SET moment=moment+100;

Zapytania SELECT

Zapytania SELECT

```
SELECT [DISTINCT] A,C,B
FROM R, S
WHERE F
ORDER BY B,C;
```

Zapytania SELECT

```
SELECT [DISTINCT] A,C,B
FROM R, S                          -- (1) złączenie
WHERE F
ORDER BY B,C;
```

1. Obliczanie (rozumienie) zapytania rozpoczynamy od złączenia relacji wymienionych po FROM

Zapytania SELECT

```
SELECT [DISTINCT] A,C,B
FROM R, S                    -- (1) złączenie
WHERE F                      -- (2) selekcja
ORDER BY B,C;
```

1. Obliczanie (rozumienie) zapytania rozpoczynamy od złączenia relacji wymienionych po FROM
2. Następnie wybieramy krotki złączenia spełniające warunek selekcji F podany w klauzuli WHERE

Zapytania SELECT

```
SELECT [DISTINCT] A,C,B        -- (3) rzutowanie,
FROM R, S                      -- (1) złączenie,
WHERE F                        -- (2) selekcja
ORDER BY B,C;
```

1. Obliczanie (rozumienie) zapytania rozpoczynamy od złączenia relacji wymienionych po FROM
2. Następnie wybieramy krotki złączenia spełniające warunek selekcji F podany w klauzuli WHERE
3. Wynik rzutujemy na kolumny wskazane w klauzuli SELECT ewentualnie usuwając duplikaty (DISTINCT)

Zapytania SELECT

```
SELECT  DISTINCT  A,C,B        -- (3) rzutowanie,
FROM R, S                     -- (1) złączenie,
WHERE F                       -- (2) selekcja
ORDER BY B,C;
```

1. Obliczanie (rozumienie) zapytania rozpoczynamy od złączenia relacji wymienionych po FROM
2. Następnie wybieramy krotki złączenia spełniające warunek selekcji F podany w klauzuli WHERE
3. Wynik rzutujemy na kolumny wskazane w klauzuli SELECT ewentualnie usuwając duplikaty (DISTINCT)

Zapytania SELECT

```
SELECT [DISTINCT] A,C,B        -- (3) rzutowanie,
FROM R, S                      -- (1) złączenie,
WHERE F                        -- (2) selekcja
ORDER BY B,C;                  -- (4) sortowanie
```

1. Obliczanie (rozumienie) zapytania rozpoczynamy od złączenia relacji wymienionych po FROM
2. Następnie wybieramy krotki złączenia spełniające warunek selekcji F podany w klauzuli WHERE
3. Wynik rzutujemy na kolumny wskazane w klauzuli SELECT ewentualnie usuwając duplikaty (DISTINCT)
4. Wynik porządkujemy wg klauzuli ORDER BY

SELECT - klauzula WHERE

SELECT - klauzula WHERE

Zarejestrowani w ciągu ostatnich 10 dni,

SELECT - klauzula WHERE

Zarejestrowani w ciągu ostatnich 10 dni,

```
SELECT nick FROM Os
WHERE data_rej>=current_date-10;
```

SELECT - klauzula WHERE

Zarejestrowani w ciągu ostatnich 10 dni,

```
SELECT nick FROM Os
WHERE data_rej>=current_date-10;
```

SELECT - klauzula WHERE

Zarejestrowani w ciągu ostatnich 10 dni,

```
SELECT nick FROM Os
WHERE data_rej>=current_date-10;
```

SELECT - klauzula WHERE

Zarejestrowani w ciągu ostatnich 10 dni,

```
SELECT nick FROM Os
WHERE data_rej>=current_date-10;
```

Można wykonywać operacje arytmetyczne na datach i liczbach: data-data to liczba dni, data +- liczba to data; w przypadku dat ważna jest także konwersja do odpowiedniego formatu (timestamp::date lub cast);

SELECT - klauzula WHERE

Zarejestrowani w ciągu ostatnich 10 dni,
których nick zaczyna się na "ab",

```
SELECT nick FROM Os
WHERE data_rej>=current_date-10;
```

SELECT - klauzula WHERE

Zarejestrowani w ciągu ostatnich 10 dni,
których nick zaczyna się na "ab",

```
SELECT nick FROM Os
WHERE data_rej>=current_date-10 AND
      nick LIKE 'ab%';
```

SELECT - klauzula WHERE

Zarejestrowani w ciągu ostatnich 10 dni,
których nick zaczyna się na "ab",

```
SELECT nick FROM Os
WHERE data_rej>=current_date-10 AND
      nick LIKE 'ab%';
```

Dla tekstów mamy operacje porównywania przybliżonego LIKE, znaki zastępcze (% zastępuje dowolny ciąg znaków a _ dowolny znak), konkatenację || i wiele innych funkcji...

SELECT - klauzula WHERE

Zarejestrowani w ciągu ostatnich 10 dni,
których nick zaczyna się na "ab",
którzy nie mają przypisanej grupy

```
SELECT nick FROM Os
WHERE data_rej>=current_date-10 AND
      nick LIKE 'ab%';
```

SELECT - klauzula WHERE

Zarejestrowani w ciągu ostatnich 10 dni,
których nick zaczyna się na "ab",
którzy nie mają przypisanej grupy

```
SELECT nick FROM Os
WHERE data_rej>=current_date-10 AND
      nick LIKE 'ab%' AND grupa IS NULL;
```

SELECT - klauzula WHERE

Zarejestrowani w ciągu ostatnich 10 dni,
których nick zaczyna się na "ab",
którzy nie mają przypisanej grupy

```
SELECT nick FROM Os
WHERE data_rej>=current_date-10 AND
      nick LIKE 'ab%' AND grupa IS NULL;
```

Sprawdzenie, czy pole jest puste, wykonujemy za pomocą operatora IS NULL (niepuste - IS NOT NULL).

SELECT - klauzula SELECT

SELECT - klauzula SELECT

Nicki osób z grup LIKE '%temp%'

SELECT - klauzula SELECT

Nicki osób z grup LIKE '%temp%'

```
SELECT nick

FROM Os WHERE grupa LIKE '%temp%';
```

SELECT - klauzula SELECT

Nicki osób z grup LIKE '%temp%'
z adresem,

```
SELECT nick

FROM Os WHERE grupa LIKE '%temp';
```

SELECT - klauzula SELECT

Nicki osób z grup LIKE '%temp%'
z adresem,

```
SELECT nick, adres

FROM Os WHERE grupa LIKE '%temp%';
```

SELECT - klauzula SELECT

Nicki osób z grup LIKE '%temp%'
z adresem,

```
SELECT nick||' z '||adres

FROM Os WHERE grupa LIKE '%temp%';
```

SELECT - klauzula SELECT

Nicki osób z grup LIKE '%temp%'
z adresem,

```
SELECT nick||' z '||adres

FROM Os WHERE grupa LIKE '%temp%';
```

Z wybranych kolumn (i stałych) możemy wyliczyć nowe kolumny pokazywane w relacji wynikowej.

SELECT - klauzula SELECT

Nicki osób z grup LIKE '%temp%'
z adresem,
oraz proponowaną datą wyrejestrowania

```
SELECT nick||' z '||adres

FROM Os WHERE grupa LIKE '%temp%';
```

Z wybranych kolumn (i stałych) możemy wyliczyć nowe kolumny pokazywane w relacji wynikowej.

SELECT - klauzula SELECT

Nicki osób z grup LIKE '%temp%'
z adresem,
oraz proponowaną datą wyrejestrowania

```
SELECT nick||' z '||adres,
       data_rej+30 as "Data wypisu"
FROM Os WHERE grupa LIKE '%temp%';
```

Z wybranych kolumn (i stałych) możemy wyliczyć nowe kolumny pokazywane w relacji wynikowej.

SELECT - klauzula SELECT

Nicki osób z grup LIKE '%temp%'
z adresem,
oraz proponowaną datą wyrejestrowania

```
SELECT nick||' z '||adres,
       data_rej+30 as "Data wypisu"
FROM Os WHERE grupa LIKE '%temp%';
```

Z wybranych kolumn (i stałych) możemy wyliczyć nowe kolumny pokazywane w relacji wynikowej.
Dodajemy jeszcze jedną kolumnę wyliczaną i nadajemy jej nazwę.

SELECT - klauzula SELECT

Nicki osób z grup LIKE '%temp%'
z adresem,
oraz proponowaną datą wyrejestrowania

```
SELECT nick||' z '||adres,
       data_rej+30 as "Data wypisu"
FROM Os WHERE grupa LIKE '%temp%';
```

Pełne rekordy grup LIKE ‘%temp%’

```
SELECT id, nr, status, nazwa, założyciel
  FROM Gr WHERE id LIKE '%temp%';
```

SELECT - klauzula SELECT

Nicki osób z grup LIKE '%temp%'
z adresem,
oraz proponowaną datą wyrejestrowania

```
SELECT nick||' z '||adres,
       data_rej+30 as "Data wypisu"
FROM Os WHERE grupa LIKE '%temp%';
```

Pełne rekordy grup LIKE ‘%temp%’

```
SELECT *
  FROM Gr WHERE id LIKE '%temp%';
```

Klauzula FROM

Klauzula FROM

Wypiszmy nicki i adresy wszystkich osób

```
SELECT nick, adres
FROM Os;
```

Klauzula FROM

Wypiszmy nicki i adresy wszystkich osób
wraz z datami zamieszczenia wpisów

```
SELECT nick, adres
FROM Os;
```

1. Potrzebujemy danych z tablicy Wpisy

Klauzula FROM

Wypiszmy nicki i adresy wszystkich osób
wraz z datami zamieszczenia wpisów

```
SELECT nick, adres
FROM Os, Wpisy;
```

1. Potrzebujemy danych z tablicy Wpisy
2. Dopisanie na liście FROM tworzy iloczyn kartezjański

Klauzula FROM

Wypiszmy nicki i adresy wszystkich osób
wraz z datami zamieszczenia wpisów

```
SELECT nick, adres
FROM Os, Wpisy
WHERE Os.nick = Wpisy.nick;
```

1. Potrzebujemy danych z tablicy Wpisy
2. Dopisanie na liście FROM tworzy iloczyn kartezjański
3. Warunek selekcji powoduje, że złączenie ma sens

Klauzula FROM

Wypiszmy nicki i adresy wszystkich osób
wraz z datami zamieszczenia wpisów

```
SELECT nick, adres, (moment)::date
FROM Os, Wpisy
WHERE Os.nick = Wpisy.nick;
```

1. Potrzebujemy danych z tablicy Wpisy
2. Dopisanie na liście FROM tworzy iloczyn kartezjański
3. Warunek selekcji powoduje, że złączenie ma sens
4. Teraz na listę wynikową możemy dodać datę wpisu

Klauzula FROM

Wypiszmy nicki i adresy wszystkich osób
wraz z datami zamieszczenia wpisów

```
SELECT DISTINCT nick, adres, (moment)::date
FROM Os, Wpisy
WHERE Os.nick = Wpisy.nick;
```

1. Potrzebujemy danych z tablicy Wpisy
2. Dopisanie na liście FROM tworzy iloczyn kartezjański
3. Warunek selekcji powoduje, że złączenie ma sens
4. Teraz na listę wynikową możemy dodać datę wpisu
5. Dodajemy DISTINCT, by usunąć powtórzenia z wyniku

Klauzula FROM

Wypiszmy nicki i adresy wszystkich osób
wraz z datami zamieszczenia wpisów

```
SELECT DISTINCT nick, adres, (moment)::date
FROM Os, Wpisy
WHERE Os.nick = Wpisy.nick;
```

1. Potrzebujemy danych z tablicy Wpisy
2. Dopisanie na liście FROM tworzy iloczyn kartezjański
3. Warunek selekcji powoduje, że złączenie ma sens
4. Teraz na listę wynikową możemy dodać datę wpisu
5. Dodajemy DISTINCT, by usunąć powtórzenia z wyniku
6. Nazwa kolumny nick nie jest unikalna w zapytaniu!!!

Klauzula FROM

Wypiszmy nicki i adresy wszystkich osób
wraz z datami zamieszczenia wpisów

```
SELECT DISTINCT OS.nick, adres, (moment)::date
FROM Os, Wpisy
WHERE Os.nick = Wpisy.nick;
```

1. Potrzebujemy danych z tablicy Wpisy
2. Dopisanie na liście FROM tworzy iloczyn kartezjański
3. Warunek selekcji powoduje, że złączenie ma sens
4. Teraz na listę wynikową możemy dodać datę wpisu
5. Dodajemy DISTINCT, by usunąć powtórzenia z wyniku
6. Nazwa kolumny nick nie jest unikalna w zapytaniu!!!

Klauzula FROM - aliasy relacji

Wypiszmy pary osób (ich nicki),
które zarejestrowały się tego samego dnia

Klauzula FROM - aliasy relacji

Wypiszmy pary osób (ich nicki),
które zarejestrowały się tego samego dnia

```
SELECT Os.nick, Os.nick
FROM Os, Os
WHERE Os.data_rej=Os.data_rej;
```

1. Potrzebujemy dwóch kopii relacji Os: Os i Os

Klauzula FROM - aliasy relacji

Wypiszmy pary osób (ich nicki),
które zarejestrowały się tego samego dnia

```
SELECT Os.nick, Os.nick
FROM Os, Os
WHERE Os.data_rej=Os.data_rej;
```

1. Potrzebujemy dwóch kopii relacji Os: Os i Os
2. Aby je rozróżnić nadajemy im nazwy (aliasy) o1 i o2

Klauzula FROM - aliasy relacji

Wypiszmy pary osób (ich nicki),
które zarejestrowały się tego samego dnia

```
SELECT Os.nick, Os.nick
FROM Os o1, Os o2
WHERE Os.data_rej=Os.data_rej;
```

1. Potrzebujemy dwóch kopii relacji Os: Os i Os
2. Aby je rozróżnić nadajemy im nazwy (aliasy) o1 i o2

Klauzula FROM - aliasy relacji

Wypiszmy pary osób (ich nicki),
które zarejestrowały się tego samego dnia

```
SELECT o1.nick, o2.nick
FROM Os o1, Os o2
WHERE o1.data_rej=o2.data_rej;
```

1. Potrzebujemy dwóch kopii relacji Os: Os i Os
2. Aby je rozróżnić nadajemy im nazwy (aliasy) o1 i o2

Klauzula FROM - aliasy relacji

Wypiszmy pary osób (ich nicki),
które zarejestrowały się tego samego dnia

```
SELECT o1.nick, o2.nick
FROM Os o1, Os o2
WHERE o1.data_rej=o2.data_rej;
```

1. Potrzebujemy dwóch kopii relacji Os: Os i Os
2. Aby je rozróżnić nadajemy im nazwy (aliasy) o1 i o2

Klauzula FROM - aliasy relacji

Wypiszmy pary osób (ich nicki),
które zarejestrowały się tego samego dnia

```
SELECT o1.nick, o2.nick
FROM Os o1, Os o2
WHERE o1.data_rej=o2.data_rej;
```

1. Potrzebujemy dwóch kopii relacji Os: Os i Os
2. Aby je rozróżnić nadajemy im nazwy (aliasy) o1 i o2
3. Poprawiamy jeszcze zapytanie, by uniknąć powtórzeń i par symetrycznych;

Klauzula FROM - aliasy relacji

Wypiszmy pary osób (ich nicki),
które zarejestrowały się tego samego dnia

```
SELECT o1.nick, o2.nick
FROM Os o1, Os o2
WHERE o1.data_rej=o2.data_rej AND
        o1.nick < o2.nick;
```

1. Potrzebujemy dwóch kopii relacji Os: Os i Os
2. Aby je rozróżnić nadajemy im nazwy (aliasy) o1 i o2
3. Poprawiamy jeszcze zapytanie, by uniknąć powtórzeń i par symetrycznych;

Złączenia w klauzuli FROM

Złączenia w klauzuli FROM

Os JOIN Wpis USING(nick)

złącz po jednakowych wartościach nick

Złączenia w klauzuli FROM

Os JOIN Wpis USING(nick)

złącz po jednakowych wartościach nick

Os JOIN Gr ON grupa=id

złącz według warunku grupa=id

Złączenia w klauzuli FROM

Os JOIN Wpis USING(nick)
złącz po jednakowych wartościach nick

Os JOIN Gr ON grupa=id
złącz według warunku grupa=id

Os NATURAL JOIN Wpis
złącz według wszystkich wspólnych kolumn

Złączenia w klauzuli FROM

Os JOIN Wpis USING(nick)
złącz po jednakowych wartościach nick

Os JOIN Gr ON grupa=id
złącz według warunku grupa=id

Os NATURAL JOIN Wpis
złącz według wszystkich wspólnych kolumn =
= kolumn o takich samych nazwach

Złączenia w klauzuli FROM

Os JOIN Wpis USING(nick)
złącz po jednakowych wartościach nick

Os JOIN Gr ON grupa=id
złącz według warunku grupa=id

Os NATURAL JOIN Wpis
złącz według wszystkich wspólnych kolumn =
= kolumn o takich samych nazwach

Os LEFT JOIN Wpis USING(nick)
złącz dodając do wyniku osoby bez wpisów z NULL w kolumnach wpisów;

Złączenia w klauzuli FROM

Os JOIN Wpis USING(nick)
złącz po jednakowych wartościach nick

Os JOIN Gr ON grupa=id
złącz według warunku grupa=id

Os NATURAL JOIN Wpis
złącz według wszystkich wspólnych kolumn =
= kolumn o takich samych nazwach

Os LEFT JOIN Wpis USING(nick)
złącz dodając do wyniku osoby bez wpisów z NULL
w kolumnach wpisów (RIGHT i FULL OUTER JOIN);

Złączenia w klauzuli FROM

Złączenie jest łączne lewostronnie i złączenia zewnętrzne nie są wzajemnie przemienne.

Złączenia w klauzuli FROM

Złączenie jest łączne lewostronnie i złączenia zewnętrzne nie są wzajemnie przemienne.

Os LEFT JOIN Wpis USING(nick) JOIN Zal USING(id)

Złączenia w klauzuli FROM

Złączenie jest łączne lewostronnie i złączenia zewnętrzne nie są wzajemnie przemienne.

Os LEFT JOIN Wpis USING(nick) JOIN Zal USING(id)
(Os LEFT JOIN Wpis USING(nick)) JOIN Zal USING(id)

Złączenia w klauzuli FROM

Złączenie jest łączne lewostronnie i złączenia zewnętrzne nie są wzajemnie przemienne.

Os LEFT JOIN Wpis USING(nick) JOIN Zal USING(id)
(Os LEFT JOIN Wpis USING(nick)) JOIN Zal USING(id)

Złączenia w klauzuli FROM

Złączenie jest łączne lewostronnie i złączenia zewnętrzne nie są wzajemnie przemienne.

Os LEFT JOIN Wpis USING(nick) JOIN Zal USING(id)
(Os LEFT JOIN Wpis USING(nick)) JOIN Zal USING(id)
Tylko osoby, które mają wpis z załącznikiem

Os LEFT JOIN (Wpis USING(nick) JOIN Zal USING(id))

Osoby, które mają wpis z załącznikiem, są z tym wpisem pozostałe są z pustymi polami wpisu

Złączenia w klauzuli FROM

Złączenie jest łączne lewostronnie i złączenia zewnętrzne nie są wzajemnie przemienne.

Os LEFT JOIN Wpis USING(nick) JOIN Zal USING(id)
(Os LEFT JOIN Wpis USING(nick)) JOIN Zal USING(id)
Tylko osoby, które mają wpis z załącznikiem

Os LEFT JOIN (Wpis USING(nick) JOIN Zal USING(id))

Osoby, które mają wpis z załącznikiem, są z tym wpisem; pozostałe są z pustymi polami wpisu

Operacje na zbiorach

Operacje na zbiorach

(SELECT nick,data_rej as "data" FROM Os WHERE data_rej BETWEEN '2009.01.01' AND '2009.12.31')

Operacje na zbiorach

(SELECT nick,data_rej as "data" FROM Os WHERE data_rej BETWEEN '2009.01.01' AND '2009.12.31')

(SELECT nick,data1 as "data" FROM ArchOs WHERE data1 BETWEEN '2009.01.01' AND '2009.12.31' AND data2>'2009.12.31');

Operacje na zbiorach

```
(SELECT nick,data_rej as "data" FROM Os WHERE
data_rej BETWEEN '2009.01.01' AND '2009.12.31')
UNION ALL
(SELECT nick,data1 as "data" FROM ArchOs WHERE
data1 BETWEEN '2009.01.01' AND '2009.12.31' AND
data2>'2009.12.31');
```

Operacje na zbiorach

```
(SELECT nick,data_rej as "data" FROM Os WHERE
data_rej BETWEEN '2009.01.01' AND '2009.12.31')
UNION ALL
(SELECT nick,data1 as "data" FROM ArchOs WHERE
data1 BETWEEN '2009.01.01' AND '2009.12.31' AND
data2>'2009.12.31');
```

- Wykonując sumę musimy zwrócić uwagę na zgodność typów dodawanych relacji;

Operacje na zbiorach

```
(SELECT nick,data_rej as "data" FROM Os WHERE
data_rej BETWEEN '2009.01.01' AND '2009.12.31')
UNION ALL
(SELECT nick,data1 as "data" FROM ArchOs WHERE
data1 BETWEEN '2009.01.01' AND '2009.12.31' AND
data2>'2009.12.31');
```

- Wykonując sumę musimy zwrócić uwagę na zgodność typów dodawanych relacji;
- Suma (UNION) automatycznie usuwa duplikaty; jeśli chcemy je pozostawić stosujemy UNION ALL;

Operacje na zbiorach

```
(SELECT nick,data_rej as "data" FROM Os WHERE
data_rej BETWEEN '2009.01.01' AND '2009.12.31')
UNION
(SELECT nick,data1 as "data" FROM ArchOs WHERE
data1 BETWEEN '2009.01.01' AND '2009.12.31' AND
data2>'2009.12.31');
```

- Wykonując sumę musimy zwrócić uwagę na zgodność typów dodawanych relacji;
- Suma (UNION) automatycznie usuwa duplikaty; jeśli chcemy je pozostawić stosujemy UNION ALL;

Operacje na zbiorach

```
(SELECT nick,data_rej as "data" FROM Os WHERE
data_rej BETWEEN '2009.01.01' AND '2009.12.31')
UNION
(SELECT nick,data1 as "data" FROM ArchOs WHERE
data1 BETWEEN '2009.01.01' AND '2009.12.31' AND
data2>'2009.12.31');
```

- Wykonując sumę musimy zwrócić uwagę na zgodność typów dodawanych relacji;
- Suma (UNION) automatycznie usuwa duplikaty; jeśli chcemy je pozostawić stosujemy UNION ALL;
- Pozostałe operacje to EXCEPT (MINUS) i INTERSECT

Operacje na zbiorach

```
(SELECT nick,data_rej as "data" FROM Os WHERE
data_rej BETWEEN '2009.01.01' AND '2009.12.31')
UNION
(SELECT nick,data1 as "data" FROM ArchOs WHERE
data1 BETWEEN '2009.01.01' AND '2009.12.31' AND
data2>'2009.12.31');
```

- Wykonując sumę musimy zwrócić uwagę na zgodność typów dodawanych relacji;
- Suma (UNION) automatycznie usuwa duplikaty; jeśli chcemy je pozostawić stosujemy UNION ALL;
- Pozostałe operacje to EXCEPT (MINUS) i INTERSECT
- EXCEPT i INTERSECT też domyślnie usuwają duplikaty

Porządkowanie wyniku

Porządkowanie wyniku

- Usuwanie duplikatów DISTINCT
- Nazywanie kolumn AS "data_wyp"
- Sortowanie: ORDER BY nazwa lub liczba i DESC/ASC
- Ograniczenie liczby krotek: LIMIT, OFFSET

Porządkowanie wyniku

- Usuwanie duplikatów DISTINCT
- Nazywanie kolumn AS "data_wyp"
- Sortowanie: ORDER BY nazwa lub liczba i DESC/ASC
- Ograniczenie liczby krotek: LIMIT, OFFSET

```
SELECT DISTINCT nick,
        EXTRACT(hour FROM moment) AS "data_wyp"
FROM Os JOIN Wpis USING(nick)
WHERE (moment::date)=current_date-1
ORDER BY "data_wp" DESC, nick
LIMIT 10;
```

Porządkowanie wyniku

- Usuwanie duplikatów DISTINCT
- Nazywanie kolumn AS "data_wyp"
- Sortowanie: ORDER BY nazwa lub liczba i DESC/ASC
- Ograniczenie liczby krotek: LIMIT, OFFSET

```
SELECT DISTINCT nick,
            EXTRACT(hour FROM moment) AS "data_wyp"
FROM Os JOIN Wpis USING(nick)
WHERE (moment::date)=current_date-1
ORDER BY "data_wp" DESC, nick
LIMIT 10;
```

Porządkowanie wyniku

- Usuwanie duplikatów DISTINCT
- Nazywanie kolumn AS "data_wyp"
- Sortowanie: ORDER BY nazwa lub liczba i DESC/ASC
- Ograniczenie liczby krotek: LIMIT, OFFSET

```
SELECT DISTINCT nick,
        EXTRACT(hour FROM moment) AS "data_wyp"
FROM Os JOIN Wpis USING(nick)
WHERE (moment::date)=current_date-1
ORDER BY "data_wp" DESC, nick
LIMIT 10;
```

Porządkowanie wyniku

- Usuwanie duplikatów DISTINCT
- Nazywanie kolumn AS "data_wyp"
- Sortowanie: ORDER BY nazwa lub liczba i DESC/ASC
- Ograniczenie liczby krotek: LIMIT, OFFSET

```
SELECT DISTINCT nick,
        EXTRACT(hour FROM moment) AS "data_wyp"
FROM Os JOIN Wpis USING(nick)
WHERE (moment::date)=current_date-1
ORDER BY "data_wp" DESC, nick
LIMIT 10;
```

Porządkowanie wyniku

- Usuwanie duplikatów DISTINCT
- Nazywanie kolumn AS "data_wyp"
- Sortowanie: ORDER BY nazwa lub liczba i DESC/ASC
- Ograniczenie liczby krotek: LIMIT, OFFSET

```
SELECT DISTINCT nick,
        EXTRACT(hour FROM moment) AS "data_wyp"
FROM Os JOIN Wpis USING(nick)
WHERE (moment::date)=current_date-1
ORDER BY "data_wp" DESC, nick
LIMIT 1 OFFSET 100;
```

Porządkowanie wyniku

- Usuwanie duplikatów DISTINCT
- Nazywanie kolumn AS "data_wyp"
- Sortowanie: ORDER BY nazwa lub liczba i DESC/ASC
- Ograniczenie liczby krotek: LIMIT, OFFSET

```
SELECT DISTINCT nick,
        EXTRACT(hour FROM moment) AS "data_wyp"
FROM Os JOIN Wpis USING(nick)
WHERE (moment::date)=current_date-1
ORDER BY "data_wp" DESC, nick
LIMIT 10;
```

Porządkowanie wyniku

- Usuwanie duplikatów DISTINCT
- Nazywanie kolumn AS "data_wyp"
- Sortowanie: ORDER BY nazwa lub liczba i DESC/ASC
- Ograniczenie liczby krotek: LIMIT, OFFSET

```
SELECT DISTINCT nick,
        EXTRACT(hour FROM moment) AS "data_wyp"
FROM Os JOIN Wpis USING(nick)
WHERE (moment::date)=current_date-1
ORDER BY "data_wp" DESC, nick
LIMIT 10;
```

Porządkowanie wyniku

- Usuwanie duplikatów DISTINCT
- Nazywanie kolumn AS "data_wyp"
- Sortowanie: ORDER BY nazwa lub liczba i DESC/ASC
- Ograniczenie liczby krotek: LIMIT, OFFSET

```
SELECT DISTINCT nick,
        EXTRACT(hour FROM moment) AS "data_wyp"
FROM Os JOIN Wpis USING(nick)
WHERE (moment::date)=current_date-1
ORDER BY 2 DESC, 1
LIMIT 10;
```

SQL - podstawy

Podsumowanie

DDL (definiowanie tabel)

```
CREATE Os (
  id SERIAL PRIMARY KEY,
  nazwisko text NOT NULL,
  adres text,
  data_rej date DEFAULT current_date);

CREATE Grupa (
  id SERIAL PRIMARY KEY,
  nazwa text UNIQUE NOT NULL,
  założyciel integer REFERENCES Os ON DELETE SET NULL,
  typ char CHECK typ IN (‘z’,’s’,’p’));

DROP TABLE Os [CASCADE];
```

DDL (definiowanie tabel)

```
CREATE Os (
  id SERIAL PRIMARY KEY,
  nazwisko text NOT NULL,
  adres text,
  data_rej date DEFAULT current_date);

CREATE Grupa (
  id SERIAL PRIMARY KEY,
  nazwa text UNIQUE NOT NULL,
  założyciel integer REFERENCES Os ON DELETE SET NULL,
  typ char CHECK typ IN (‘z’,’s’,’p’));

DROP TABLE Os [CASCADE];
```

SERIAL to nie tyle typ danych, co dodatkowy obiekt w bazie - sekwencja generująca na żądanie nową wartość: nextval(“sek”), setval(“sek”,wartość);

DDL (definiowa

```
CREATE Os (
  id SERIAL PRIMARY KEY,
  nazwisko text NOT NULL,
  adres text,
  data_rej date DEFAULT current_da

CREATE Grupa (
  id SERIAL PRIMARY KEY,
  nazwa text UNIQUE NOT NULL,
  założyciel integer REFERENCES Os ON DELETE SET NULL,
  typ char CHECK typ IN (‘z’,’s’,’p’));

DROP TABLE Os [CASCADE];
```

PRIMARY KEY jest unikalny, niepusty i powoduje utworzenie indeksu (np. B-drzewa)
UNIQUE sprawdza unikalność wartości niepustych; też powoduje utworzenie indeksu.
UNIQUE i NOT NULL to klucz alternatywny.

Query Language

SELECT [DISTINCT]

- *, tabela.*
- atrybuty
- stałe
- wyrażenia (kolumny) wyliczane z atrybutów i stałych
- przemianowane kolumny

FROM

- lista relacji R, S, … (R X S X…)
- aliasy relacji R r, S s
- złączenia relacji:
  - R JOIN S ON(warunek)
  - R JOIN S USING (wspólne kolumny)
  - R NATURAL JOIN S
  - R [LEFT|RIGHT|FULL] [OUTER] JOIN S [ON|USING]

WHERE

- warunek selekcji: R.id=S.id OR S.nazwa LIKE ‘%temp%’ OR typ IS NULL

DDL (definic

```
CREATE Os (
  id SERIAL PRIMARY KEY,
  nazwisko text NOT NULL,
  adres text,
  data_rej date DEFAULT curre

CREATE Grupa (
  id SERIAL PRIMARY KEY,
  nazwa text UNIQUE NOT NULL,
  założyciel integer REFERENCES Os ON DELETE SET NULL,
  typ char CHECK typ IN (‘z’,’s’,’p’));

DROP TABLE Os [CASCADE];
```

FOREIGN KEY… REFERENCES (klucz obcy) wymaga, by w tabeli nadrzędnej istniała krotka wskazywana przez klucz. Usunięcie krotki nadrzędnej może zostać:

- zablokowane (RESTRICT|NO ACTION)
- kaskadowo usunąć krotki podrzędne
- wykasować lub zmienić na wartość domyślną klucze obce w krotkach podrzędnych.

Analogiczne efekty wywołuje modyfikacja krotki nadrzędnej.

DDL (definic

```
CREATE Os (
  id SERIAL PRIMARY KEY,
  nazwisko text NOT NULL,
  adres text,
  data_rej date DEFAULT curre

CREATE Grupa (
  id SERIAL PRIMARY KEY,
  nazwa text UNIQUE NOT NULL,
  założyciel integer REFERENCES Os ON DELETE SET NULL,
  typ char CHECK typ IN (‘z’,’s’,’p’));

DROP TABLE Os [CASCADE];
```

FOREIGN KEY… REFERENCES (klucz obcy) wymaga, by w tabeli nadrzędnej istniała krotka wskazywana przez klucz.
Usunięcie krotki nadrzędnej może zostać:

- zablokowane (RESTRICT|NO ACTION)
- kaskadowo usunąć krotki podrzędne
- wykasować lub zmienić na wartość domyślną klucze obce w krotkach podrzędnych.

Analogiczne efekty wywołuje modyfikacja krotki nadrzędnej.

Akcja referencyjna:
Usunięcie tabeli, na którą inne wskazują przez FOREIGN KEY, powoduje ostrzeżenie.
Usunięcie z opcją CASCADE powoduje usunięcie tabeli i więzu klucza obcego.

DDL (definiowanie tabel)

```
CREATE Os (
  id SERIAL PRIMARY KEY,
  nazwisko text NOT NULL,
  adres text,
  data_rej date DEFAULT current_date);

CREATE Grupa (
  id SERIAL PRIMARY KEY,
  nazwa text UNIQUE NOT NULL,
  założyciel integer REFERENCES Os ON DELETE SET NULL,
  typ char CHECK typ IN ('z','s','p'));

DROP TABLE Os [CASCADE];
```

Więz **CHECK** dotyczy jednej krotki. Jest sprawdzany w momencie jej wstawienia lub modyfikacji.

DDL (zmiany schematu)

ALTER TABLE tabela [ADD|DROP] COLUMN opis_kolumny;
ALTER TABLE tabela [ADD|DROP] CONSTRAINT nazwa_więzu

- CHECK (...)
- FOREIGN KEY … REFERENCES …
- PRIMARY KEY ...

ALTER TABLE tabela

- ALTER COLUMN kolumna
  - [SET|DROP] DEFAULT
  - [SET|DROP] NOT NULL
  - TYPE nowy_typ
- RENAME TO nowa_nazwa;

ALTER tabela RENAME TO nowa_nazwa;