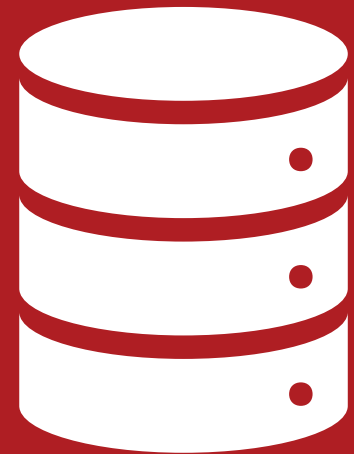


| SQL, relacyjne bazy danych





Kto mówi

Michał Michalczuk

Full-Stack software developer @ Goyello.
Co-owner pub-bistro @ Rzecz Jasna

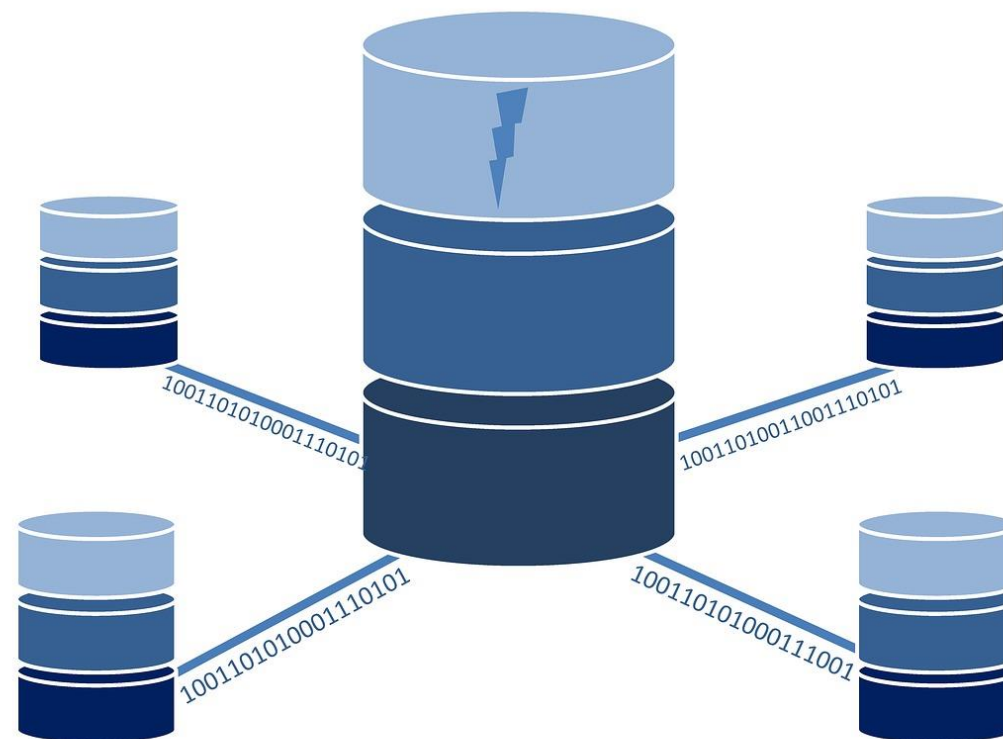
A solid red vertical bar on the left side of the slide.

Plan na dzisiaj

- Czym są Bazy Danych? Trochę nazewnictwa
- Czym jest SQL?
- Zapytania o dane
- Wprowadzanie, aktualizacja i usuwanie danych
- Design bazy danych

Czym jest Baza Danych?

- Kontener do przechowywania danych
- Dane w ustrukturyzowanej formie
- Ujednolicony dostęp do danych
- Możliwość operacji CRUD



A serwer bazodanowy?

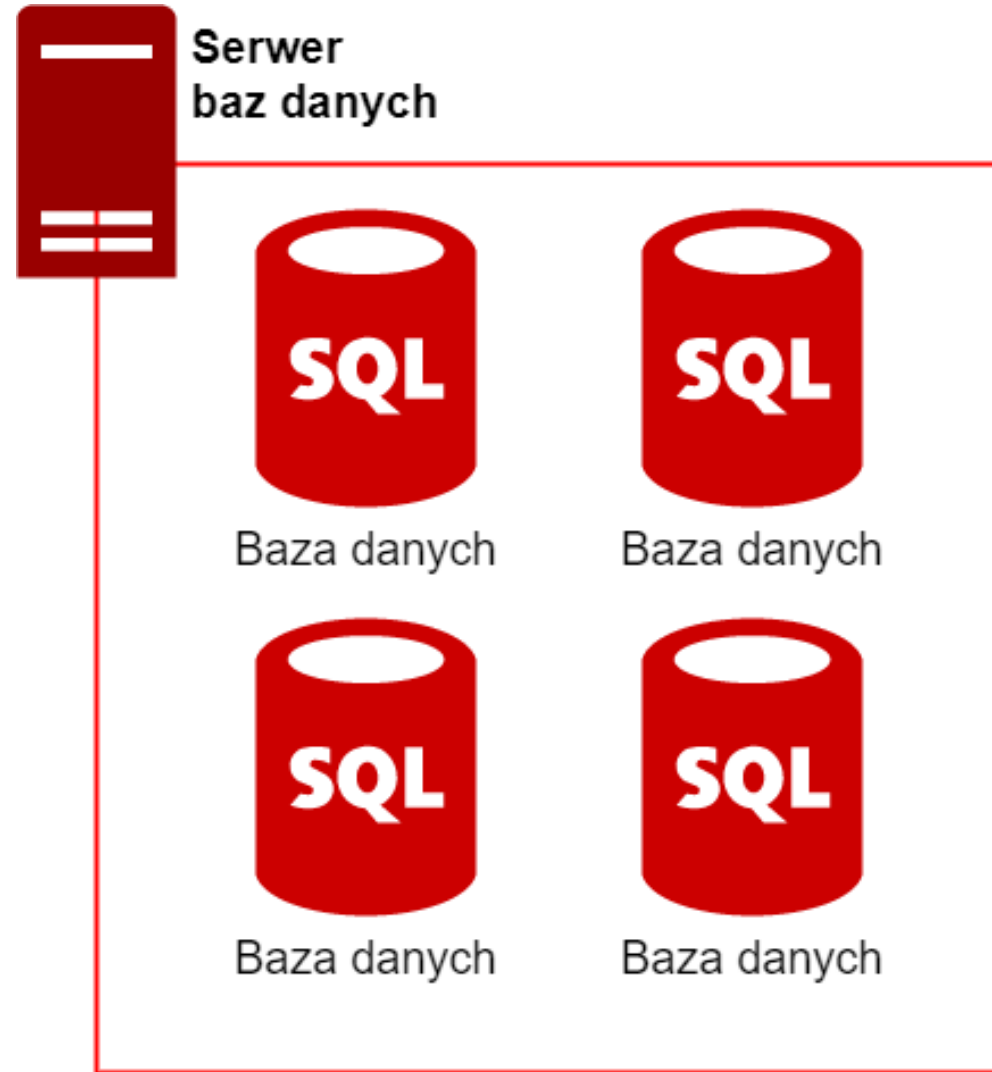
- Kontener w który jest wiele kontenerów
- Oprogramowanie które służy do zarządzania bazami danych
- Wiele baz danych
- Np twój komputer



Jak to wygląda?

Serwer

Bazy danych



Tabele, kolumny itd

- Tabela - zbiór danych o jednym znaczeniu wspólnej strukturze
- Kolumna – składowa tabeli o tym samym typie i znaczeniu
- Rekord – wpis w tabeli



Jak to wygląda?

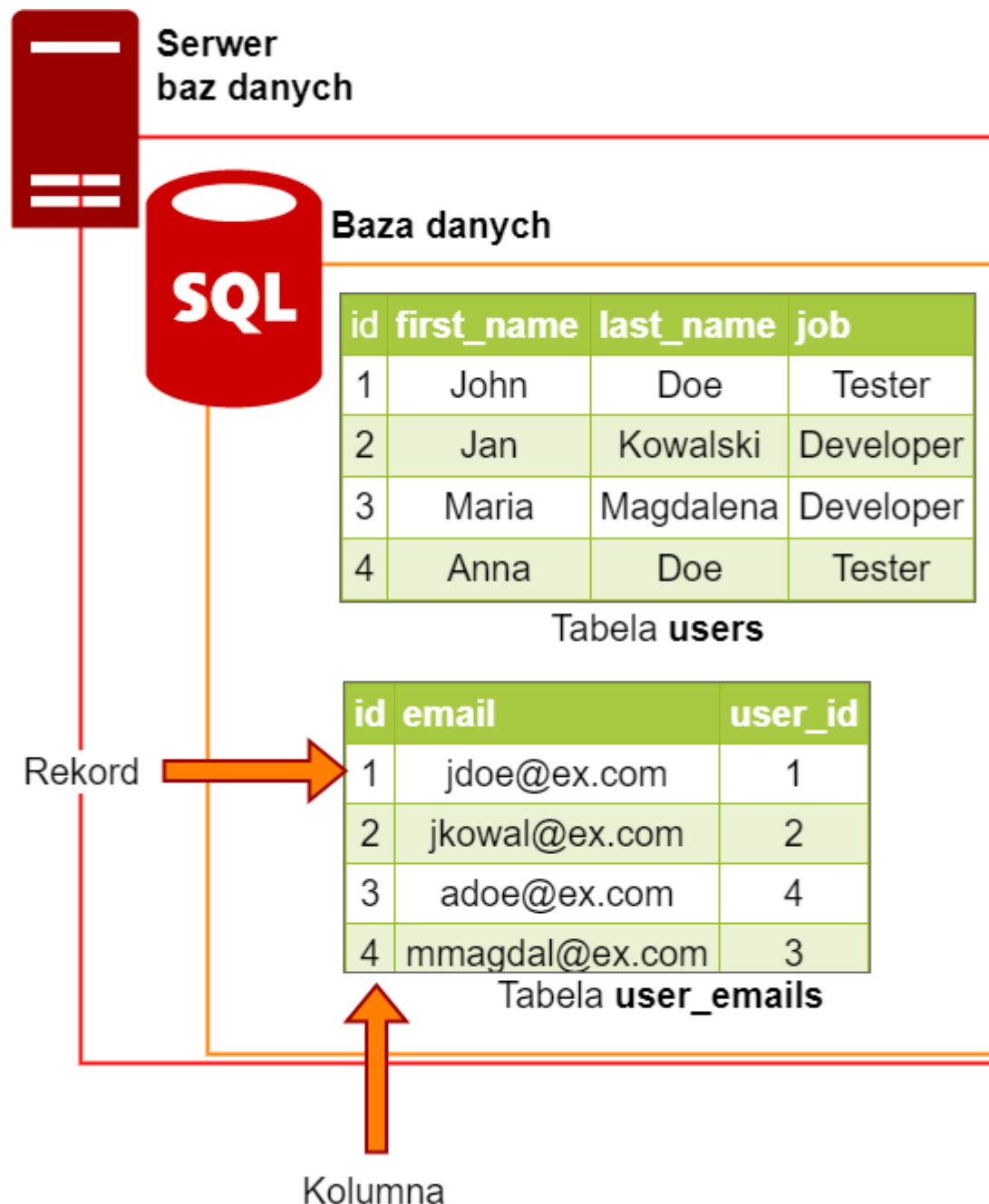
Serwer

Bazy danych

Tabele

Kolumny

Rekordy



Silniki baz danych

“Framework”

Często własny
dialekt SQL

Różne możliwości

Różne
zastosowanie



PostgreSQL



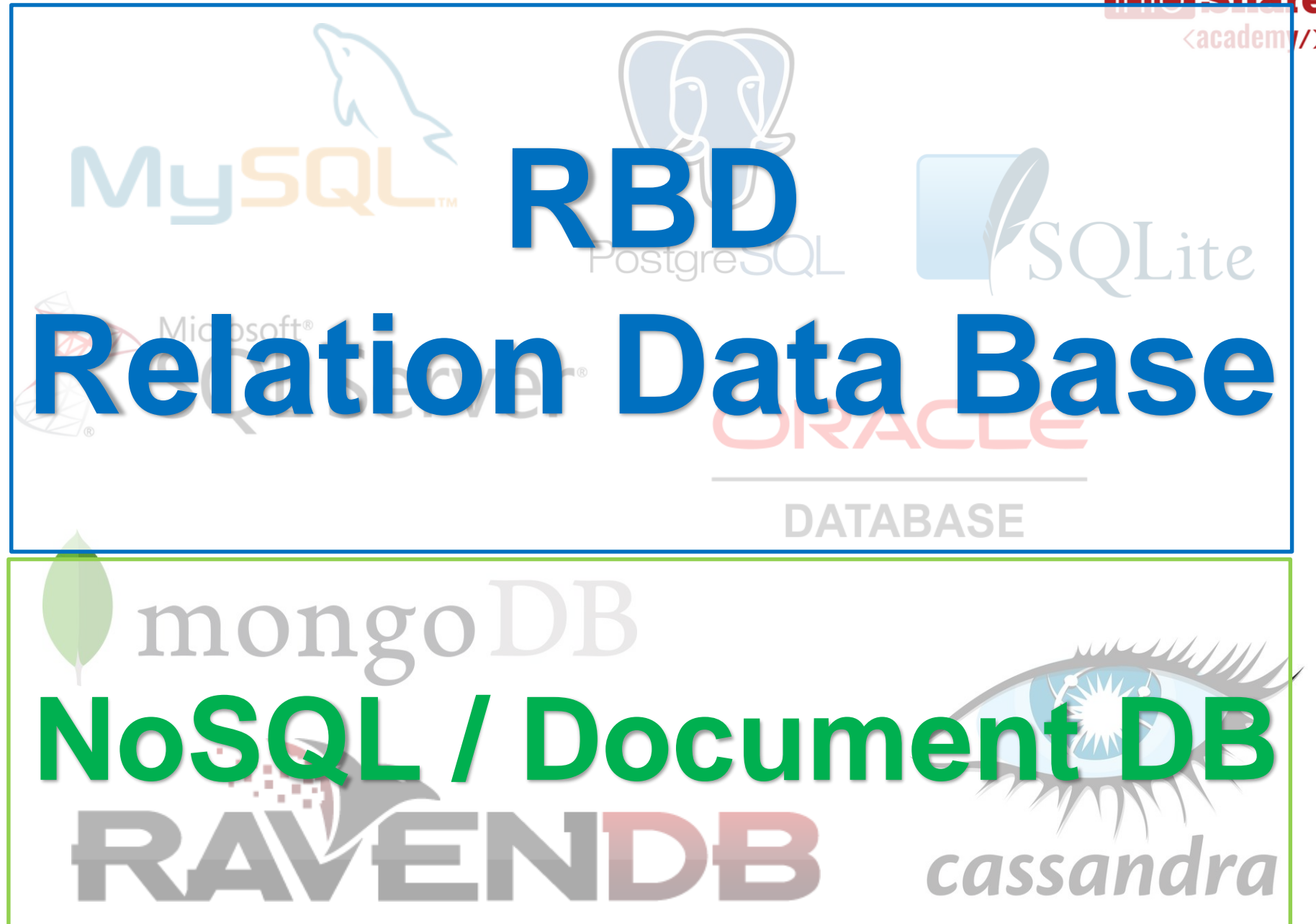
Silniki baz danych

“Framework”

Często własny
dialekt SQL

Różne możliwości

Różne
zastosowanie



“ *A relational database is
a collection of data items
organized as a set of formally
described tables*

Simple as it is

╰_(ツ)_╯

SQL VS. NOSQL OVERSIMPLIFIED

SQL

VS

NoSQL

```
SELECT * FROM Customers_tbl WHERE
Last_Name='Smith';
```

Cust_No	Last_Name	First_Name
560779	Smith	Juan
207228	Smith	George
173996	Smith	Ben
477610	Smith	Conrad

```
Get customer.firstname,customer.lastname,customer.productID.* where Last_Name='Whitelock'
```

Key	Value
746133	Firstname: George Lastname: Whitelock productID: 2012: 5
135225	Firstname: Luke Lastname: Whitelock productID: 1285: 1 1077: 5
884256	Firstname: Sam Lastname: Whitelock productID: 1442: 2

MySQL

www.mysql.com

SQL

RDMS

Popularny

Open Source

Darmowy

Prosty



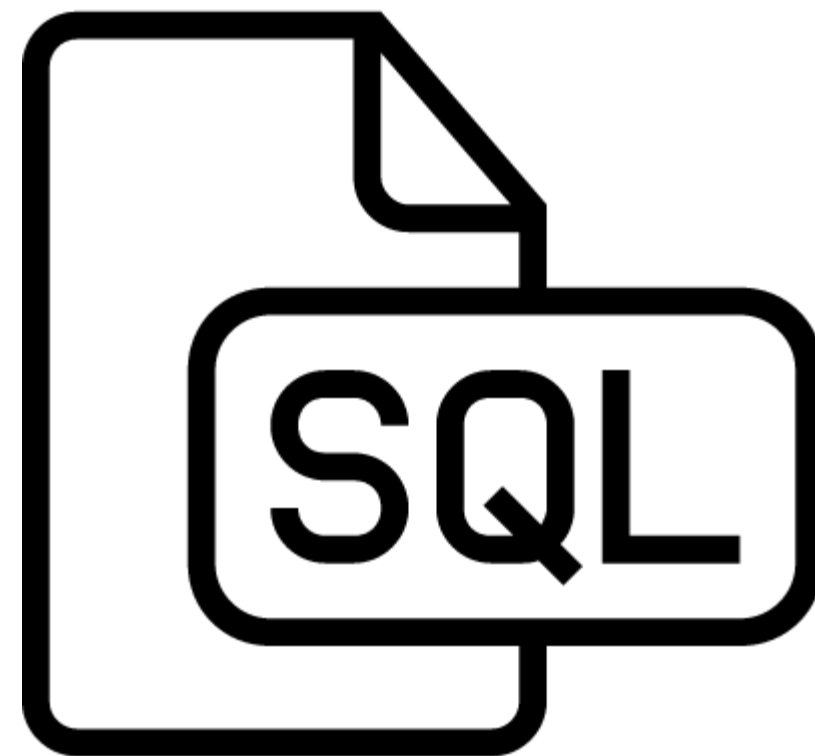
Polecam trzymać się głównie standardowych non-vendor-specific typów, komend itd.
Dzisiaj skupimy się wyłącznie na standardach.

SQL

Sam SQL to język

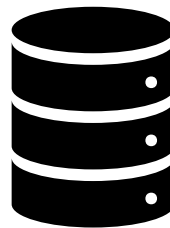
Structured
Query
Language

- Standard
- Manipulacja bazą relacyjną
- Język deklaratywny
- Definiowanie danych + manipulacja
- Różne silniki = różne „dialekty” SQL



Tworzenie bazy oraz tabel

Jak stworzyć bazę oraz tabele w niej



Serwer bazy danych MySQL jako kontener

- Wygodnie
- Szybko
- Brak śmiecia na maszynie
- Będziecie używać bazy jako kontenera w projekcie

```
docker run -it -e MYSQL_ROOT_PASSWORD=root -p 3306 -d mysql
```

```
docker ps -a
```


| Tworzenie bazy danych

```
CREATE DATABASE db_name;
```

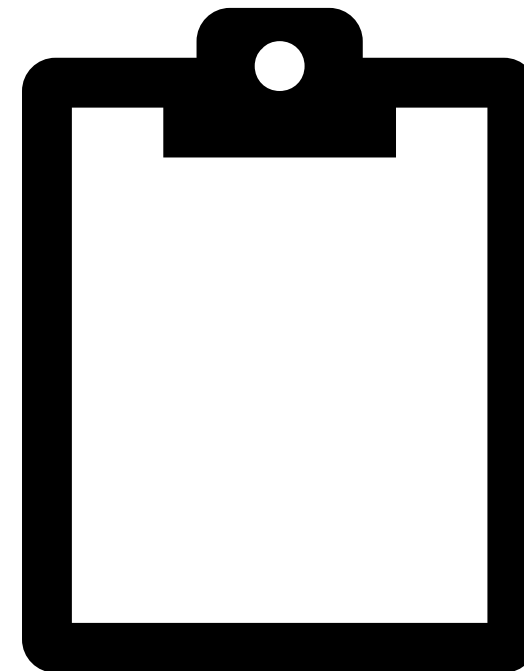
- Stwórz bazę o nazwie *db_name*

```
use db_name;
```

- Teraz wszystkie komendy idą do *db_name*

```
> mysql -h host -u user db_name
```

- Połączenie do *db_name* przez CLI



Usuwanie bazy danych

```
DROP DATABASE db_name;
```

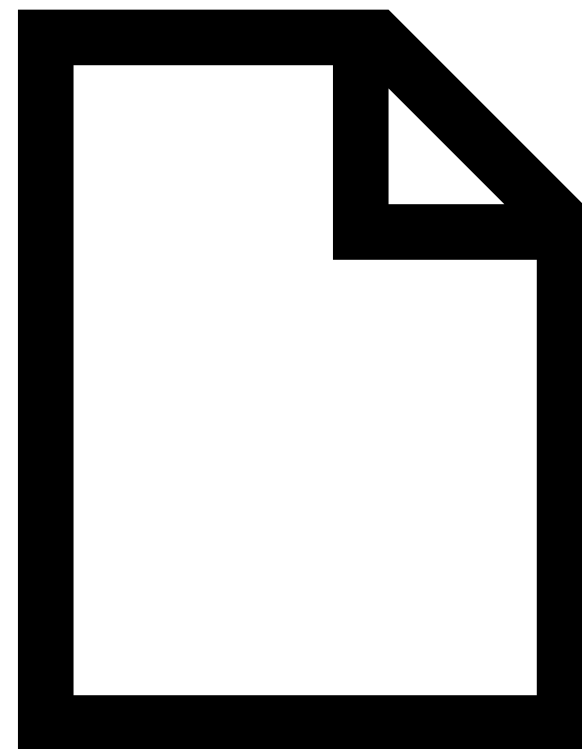
- Usuń bazę danych *db_name*



Tworzenie tabeli

```
CREATE TABLE table_name (  
    column1 datatype,  
    column2 datatype,  
    column3 datatype,  
    . . .  
);
```

Stwórz tabelę *table_name* z kolumnami *column1*, *column2* ... każda o wskazanym typie *datatype*.



Typy danych w SQL

- Numeryczne
 - Stało przecinkowe
 - Zmiennie przecinkowe
 - Logiczne (bool)
- Daty oraz czas
- Znakowe
- BLOB (Binary Large Object)



dev.mysql.com/doc/refman/5.7/en/data-types.html

Typy danych Numeryczne

Type	Storage (Bytes)	Minimum Value (Signed/Unsigned)	Maximum Value (Signed/Unsigned)
TINYINT	1	-128	127
UNSIGNED (not standard)		0	255
SMALLINT	2	-32768	32767
UNSIGNED (not standard)		0	65535
MEDIUMINT	3	-8388608	8388607
UNSIGNED (not standard)		0	16777215
INT	4	-2147483648	2147483647
UNSIGNED (not standard)		0	4294967295
BIGINT	8	-9223372036854775808	9223372036854775807
UNSIGNED (not standard)		0	18446744073709551615

dev.mysql.com/doc/refman/5.7/en/integer-types.html

Typy danych

Stało przecinkowe

- `DECIMAL (n, p)`
 - `n` – łączna ilość cyfr
 - `p` – ilość cyfr po przecinku

Np: `DECIMAL (5, 2)`

min wartość: -999.99

max wartość: 999.99

Typy danych

Zmiennie przecinkowe

- FLOAT
- DOUBLE

To są wartości zbliżone, nie są przechowywane jako “konkretne wartości”.

- Tricky przy porównywaniu
- Błędy zaokrągleń
- Zależne od wielu zmiennych, np CPU

dev.mysql.com/doc/refman/5.7/en/problems-with-float.html

Typy danych Daty

Format dat w MySQL :

`yyyy-mm-dd`

np.: 2017-11-26

Format czasu w MySQL

`hh:mm:ss`

np.: 15:16:05

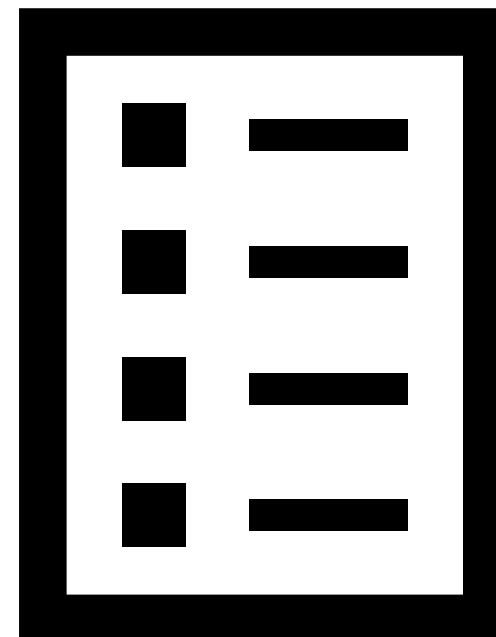
Opisz tabelę

```
DESCRIBE table_name;
```

Pokasz deklarację tabeli *table_name*.

```
SHOW TABLES;
```

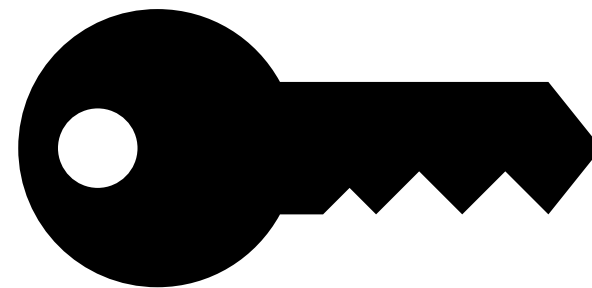
Pokaż tabele w aktualnej bazie danych.



| Klucz główny Primary Key (PK)

```
CREATE TABLE table_name (  
    id INT NOT NULL,  
    ...  
    PRIMARY KEY(id)  
);
```

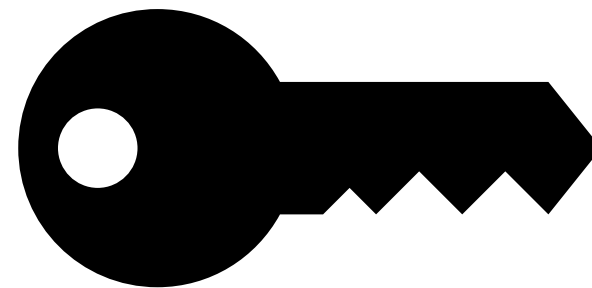
- Unikat
- Najczęściej INT/Unsigned INT lub UUID
- Niezmienny
- Baza pilnuje integralności



| Klucz główny Primary Key (PK)

```
CREATE TABLE table_name (  
    id INT NOT NULL AUTO_INCREMENT,  
    ...  
    PRIMARY KEY(id)  
);
```

Dla INT, zostawmy bazie jego
ustawianie: ***AUTO_INCREMENT***



Modyfikacja tabeli

Jak zmienić nazwę tabeli, pola lub parametry pól



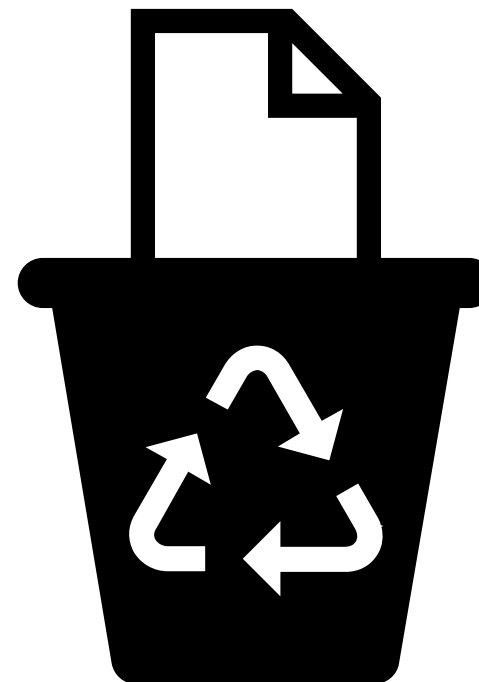
Usuwanie tabeli

```
DROP TABLE table_name;
```

Usuń tabelę *table_name* – strukturę oraz wszystkie dane.

```
TRUNCATE TABLE table_name;
```

Usuń tylko dane z tabeli *table_name*.



Modyfikacja struktury tabeli



Dodaj kolumnę

```
ALTER TABLE table_name  
ADD column_name datatype;
```

Zmień nazwę kolumny

```
ALTER TABLE table_name  
CHANGE column_name new_name datatype;
```

Zmodyfikuj kolumnę

```
ALTER TABLE table_name  
MODIFY column_name datatype;
```

Usuń kolumnę

```
ALTER TABLE table_name  
DROP column_name;
```



Modyfikacja struktury tabeli

Uwaga

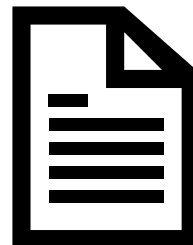


Operacje mogą się nie powieść jeśli mamy **klucze obce** lub **constraint'y**.

Ale o tym za chwilę.

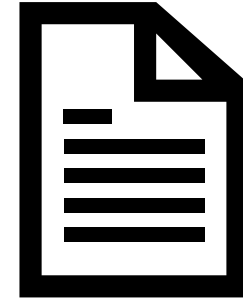
Wprowadzanie danych

Dodajemy rekordy do tabel



Wprowadzanie danych

Insert



Dodaj do *table_name*

```
INSERT INTO table_name(column1, column2, ...)  
VALUES (value1, value2, ...);
```

Wymień kolumny które chcesz uzupełnić, a potem w tej samej kolejności wartości.

Dodanie do *table_name* wszystkich pól

```
INSERT INTO table_name  
VALUES (value1, value2, ...);
```

Dodanie wielu rekordów naraz

```
INSERT INTO table_name(..columns..)  
VALUES (values..), (values..),  
      (values..);
```

Zapytania o dane

Data queries



Zapytania o dane: SELECT

Pobierz z tabeli



Pobierz konkretne pola (wymień kolumny)

```
SELECT column1, column2, ...  
FROM table_name;
```



column1	column2
row 1 value1	row 1 value2
row 2 value1	row 2 value1
row 3 value1	row 3 value1

...



Pobierz wszystko z tabeli

```
SELECT *  
FROM table_name;
```

Zapytania o dane: SELECT

Pobierz z tabeli z aliasem



Nadaj specjalną nazwę w wyniku

```
SELECT column1 AS „alias1”, column2, ...  
FROM table_name;
```

Np.:

```
SELECT first_name AS „Imie”, last_name  
FROM table_name;
```



Imie	last_name
Andrzej	Nowak
Anna	Maria
Jan	Kowalski

Zapytania o dane: SELECT

Transformuj dane wyjściowe



Połącz imię i nazwisko w jedno pole

```
SELECT CONCAT(first_name, last_name) AS „Kto”  
FROM table_name;
```



Kto
Andrzej Nowak
Anna Maria
Jan Kowalski

Policz rok urodzenia

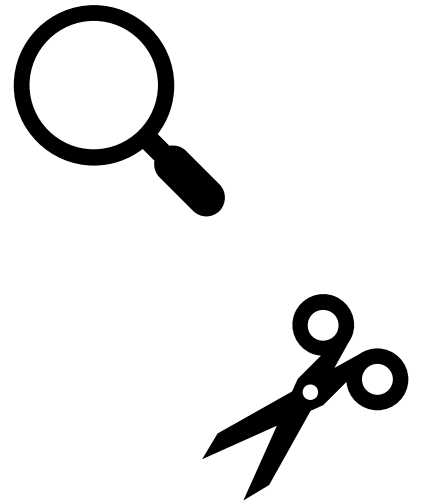
```
SELECT id, 2017-age AS „Rok urodzenia”  
FROM table_name;
```



id	Rok urodzenia
1	1956
2	1988
3	1970

Zapytania o dane: SELECT

Pobierz z tabeli - odfiltrowane



Odfiltruj – pobierz tylko te spełniające warunek

```
SELECT column1, column2, ...  
FROM table_name  
WHERE condition;
```

Np.:

```
SELECT * FROM users  
WHERE id = 5;
```

```
SELECT * FROM users  
WHERE age >= 10;
```

```
SELECT * FROM users  
WHERE city = ,Gdańsk';
```

PS też możesz pobrać całość lub wybrane kolumny

Zapytania o dane: SELECT

Złożone warunki

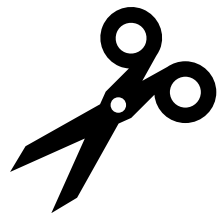
Składamy warunki logiczne jak w Javie.

AND, OR

Np.:

```
SELECT * FROM users  
WHERE age >= 10 AND city = ,Gdańsk';
```

```
SELECT * FROM users  
WHERE age >= 10 OR city = ,Gdańsk';
```



Operatory :

=, >, <, !=, <>, <=, >=

„Nie równa się”

Zapytania o dane: SELECT

Złożone warunki – IN/NOT IN



Gdy chcemy sprawdzić czy wartość pola zawiera się w liście

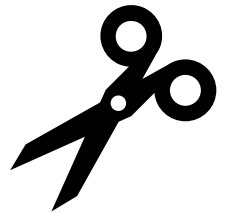
```
SELECT * FROM users  
WHERE age IN (10, 12, 25);
```

Lub gdy wartość nie zawiera się na liście

```
SELECT * FROM users  
WHERE age NOT IN (11, 15, 30);
```


Zapytania o dane: SELECT

Złożone warunki – BETWEEN



Jeśli chcemy sprawdzić czy wartość pola zawiera się w zakresie

```
SELECT * FROM table_name  
WHERE column BETWEEN min AND max;
```

Lub gdy wartość nie zawiera się na liście

```
SELECT * FROM users  
WHERE age BETWEEN 10 AND 30;
```



```
SELECT * FROM users  
WHERE age >= 10 AND age <= 30;
```

Zapytania o dane: SELECT

Złożone warunki – LIKE, mini regex



```
SELECT * FROM users
```

```
WHERE name LIKE 'ANDR%';
```

Jeśli szukamy zbliżonych wartości. Np. „**kot**”, „**koty**”, „**kotowate**”.

Możemy wyszukać wartości kot + dowolny ciąg `kot%`.

Możemy też określić który znak jest „dowolny”. Np.: „**koc**”, „**kac**”, „**kxc**”.

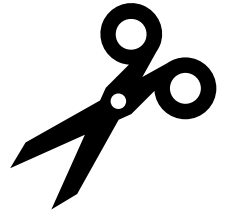
Dowolny znak na 2giej pozycji: `k_c`

Kombinacja dla „**koc**”, „**kac**”, „**koci**”, „**kacap**”, „**kicia**”

Dowolny znak na 2giej pozycji oraz dowolny ciąg na końcu: `k_c%`

Zapytania o dane: SELECT

Złożone warunki – LIKE, mini regex



LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that starts with "a"
WHERE CustomerName LIKE '%a'	Finds any values that ends with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%_%'	Finds any values that starts with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that starts with "a" and ends with "o"

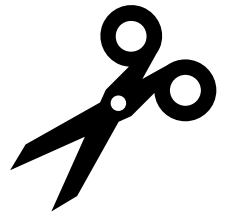
https://www.w3schools.com/sql/sql_wildcards.asp

Zapytania o dane: SELECT

Złożone warunki – kombo

Możemy dowolnie łączyć warunki

```
SELECT * FROM users
WHERE age BETWEEN 10 AND 30
      AND
      (first_name LIKE „%are%”
      OR first_name = ,Andrzej’)
AND id > 100;
```



Sortowanie wyników

Sortujemy rosnąco lub malejąco w kolejności kolumn

```
SELECT column1, column2, ...  
FROM table_name  
ORDER BY column1, column2, ... ASC|DESC;
```

Np.:

```
SELECT * FROM users  
ORDER BY age DESC;
```

```
SELECT * FROM users  
ORDER BY age, name;
```

```
SELECT * FROM users  
ORDER BY age DESC, name ASC;
```



ASC – rosnąco, DESC - malejąco

Domyślnie: ASC

Podzbiory wyników



Możemy pobrać tylko część wyników. Np. pierwsze *n*.

```
SELECT column1, column2, ...  
FROM table_name  
LIMIT n;
```

Aby zrobić małą „paginację”, możemy ominąć część wyników, *off* – liczba do ominięcia

```
SELECT column1, column2, ...  
FROM table_name  
LIMIT n  
OFFSET off;
```

Kolejność operacji



Zapytania o dane

Agregaty



Agregowanie wyników



Wyświetlenie największe, najmniejszej, średniej lub po prostu ilości

```
SELECT MIN(col1), MAX(col2), AVG(col3), COUNT(*)  
FROM table_name;
```

Np.:

```
SELECT MIN(age), MAX(age), AVG(age), COUNT(*)  
FROM users;
```

MIN – minimum, **MAX** – maximum,
AVG – średnia, **COUNT** - ilość

Grupowanie wyników

Podziel wynik na grupy

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s);
```

Np.:

```
SELECT city FROM users
GROUP BY city;
```



Po grupowaniu mamy wynik tylko do danych po których grupujemy.

Grupowanie wyników z filtrowaniem grup

Podziel wynik na grupy, ale odfiltruj jeszcze wyniki grupowania

```
SELECT column_name(s) FROM table_name  
WHERE condition  
GROUP BY column_name(s)  
HAVING group_condition;
```

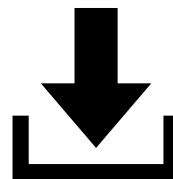
Np.:

```
SELECT city, AVG(age) AS avg FROM users  
GROUP BY city  
HAVING avg > 30;
```

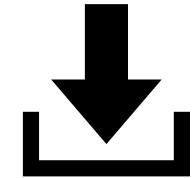


Aktualizacja danych

Update



Update



Zaktualizuj wartości, dla wybranych kolumn w odfiltrowanych rekordach

```
UPDATE table_name  
SET column1 = value1, column2 = value2, ...  
WHERE condition;
```

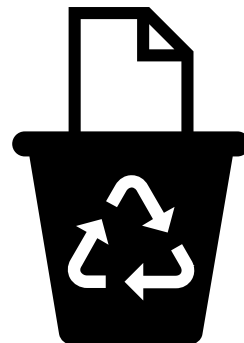
Np.:

```
UPDATE users  
SET city = „Gdynia”, age=„33”  
WHERE id = 5;
```

```
UPDATE employees  
SET salary = salary*1.1  
WHERE year_result > 120;
```

Usuwanie danych

Delete



Usuwanie danych

Usuwamy rekordy w tabeli. Po odfiltrowaniu.

```
DELETE table_name  
WHERE condition;
```

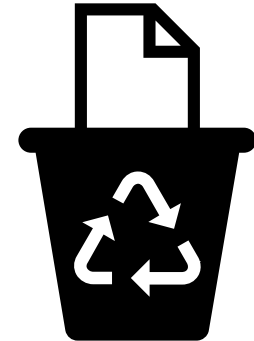
Np.:

```
DELETE users  
WHERE id = 5;
```

```
DELETE employees  
WHERE year_result < 10;
```

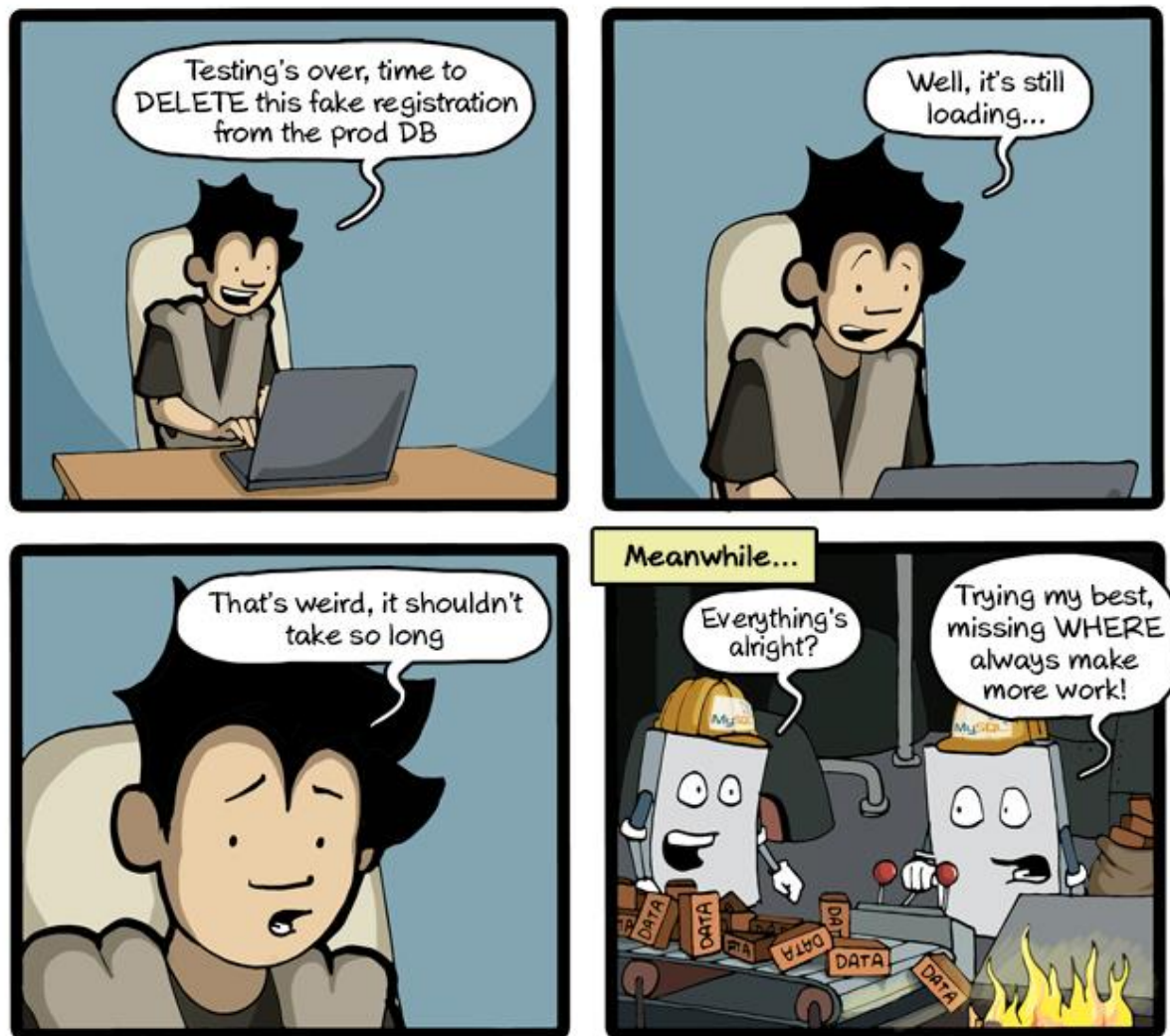
Usunąć wszystkie rekordy w tabeli

```
TRUNCATE TABLE table_name;
```



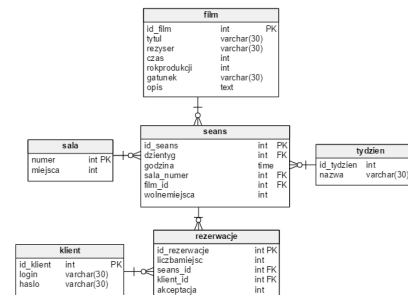
Usuwanie danych

Nie zapomnij o warunku

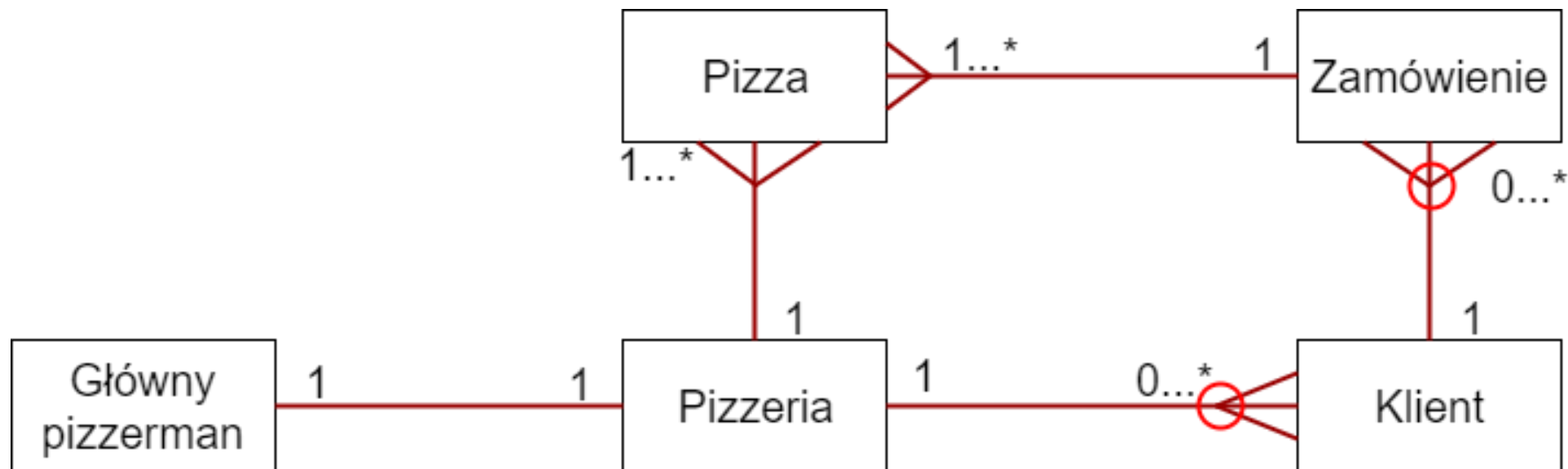


Relacje w bazie

Relacje pomiędzy tabelami, klucze obce



Zamodelujmy pizzerię



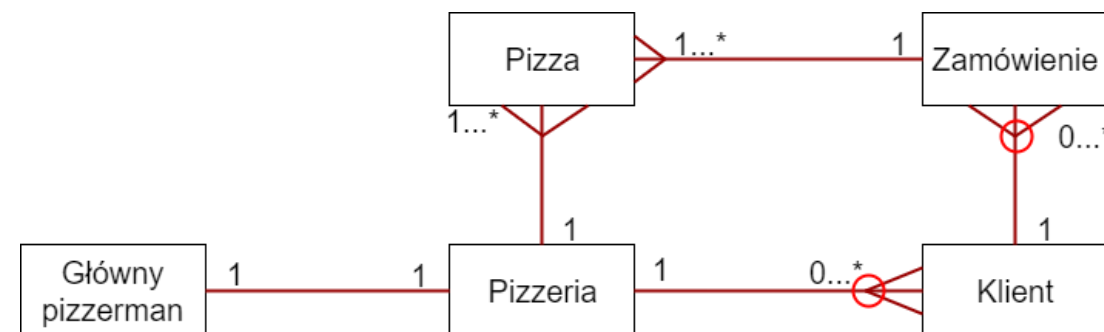
Entity Relationship Diagram ERD

- Zależności
- Kolumny
- Klucze
- Tabela = Encja

Tabela – rzeczownik

Relacja – czasownik

Np.: Zamówienie ma wiele pizz



RELACJE



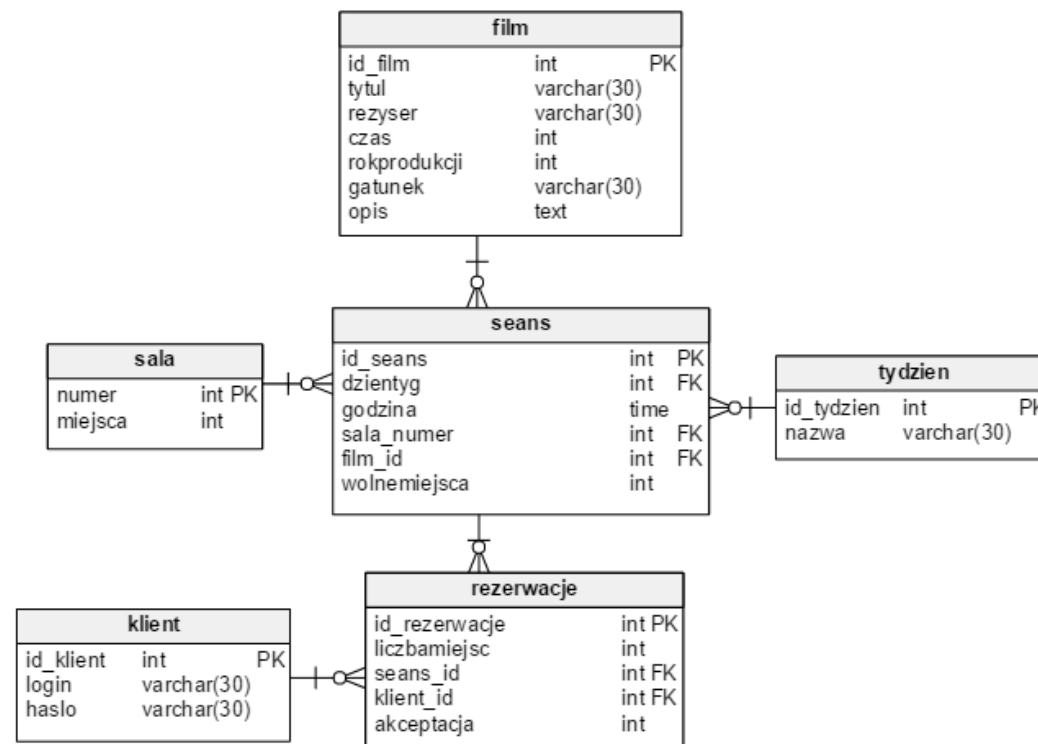
- one-to-zero or many



- many-to-zero or many



- one-to-zero or one

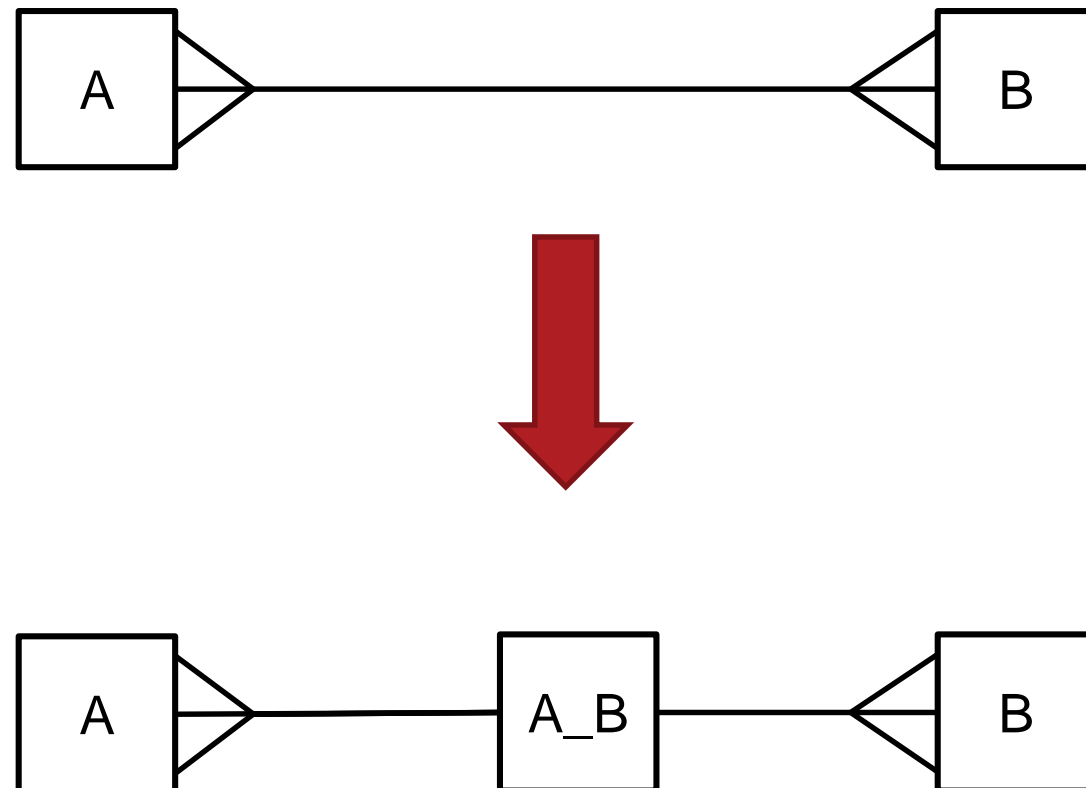


RELACJE

Many to Many

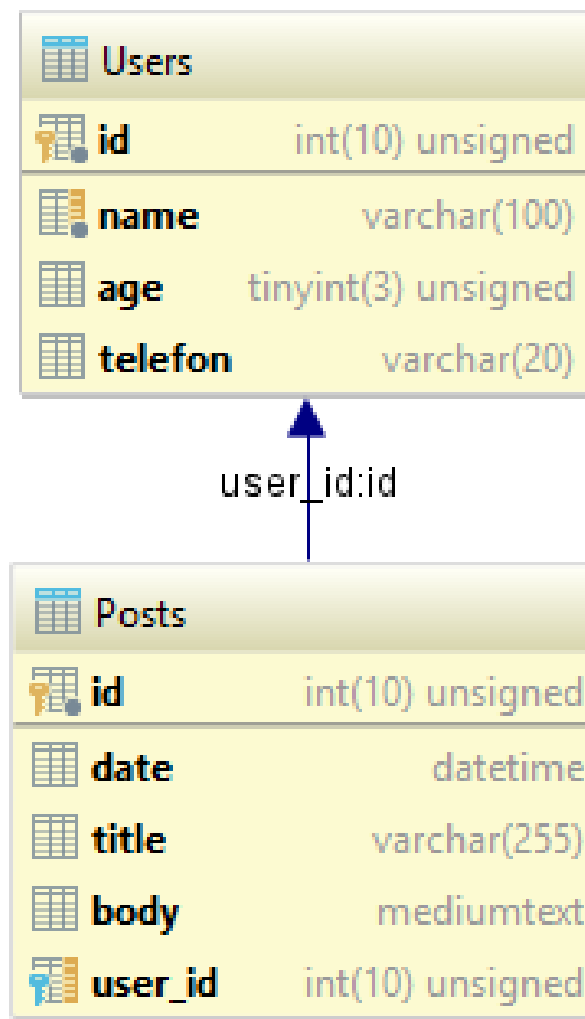
Tą relację musimy zamodelować
z małą pomocą

tabeli pośredniczącej A_B



Klucz obcy Foreign Key (FK)

- Link pomiędzy tabelami
- Pole (lub pola) które wskazują na PR w innej tabeli
- Baza dba o intergalność



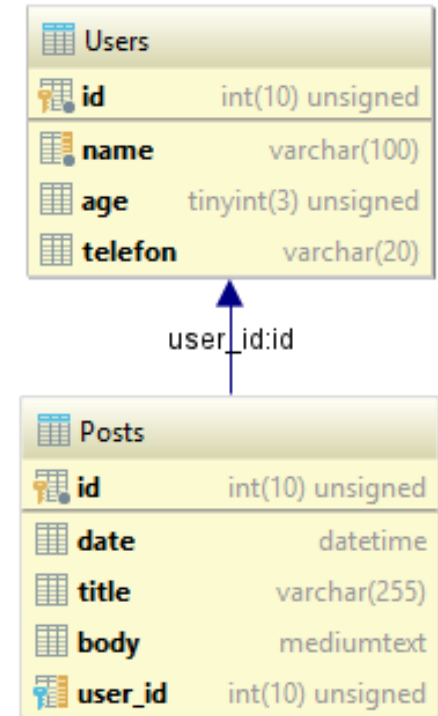
Klucz obcy

Tworzenie tabeli

```
CREATE TABLE Posts (
    id INT UNSIGNED AUTO_INCREMENT,
    body MEDIUMTEXT,
    user_id INT UNSIGNED,
    PRIMARY KEY (id),
    FOREIGN KEY (user_id) REFERENCES Users (id)
);
```

Nazwa pola które będzie kluczem

Tabela do której tworzymy relacje
oraz pole na które wskazuje klucz.



Klucz obcy

Alter tabeli

Jeśli nie dodaliśmy klucza od razu .. Nie szkodzi. Możemy zmodyfikować schemę.

```
ALTER TABLE Posts
```

```
ADD FOREIGN KEY (user_id) REFERENCES Users (id) ;
```



Nazwa pola które będzie kluczem




Tabela do której tworzymy relacje
oraz pole na które wskazuje klucz.

Klucz obcy

Usuwanie rekordów



Klucz obcy jest warunkiem który musimy spełniać, element o danym ID na który wskazuje istnieć.

To warunek integralności danych.

Nie możemy usunąć rekordu na który wskazują (mają klucze obce wskazujące na niego) rekordy w innej/innych tabelach.

Musimy najpierw usunąć “dzieci” lub usunąć constraint – klucz obcy.

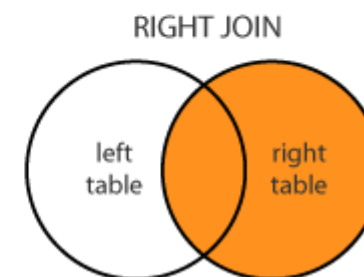
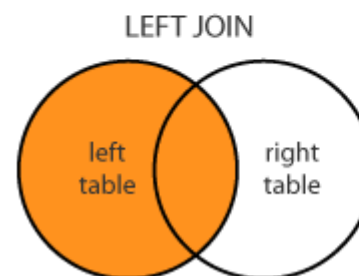
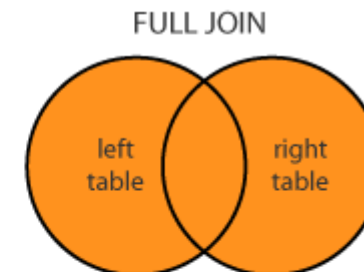
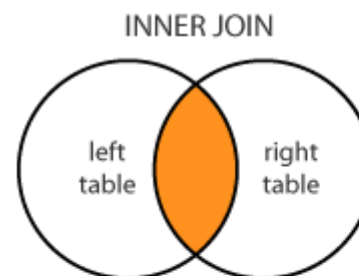
JOIN

Łączymy table w zapytaniach



JOIN

- Wyniki z wielu tabel
- Musi być pole po którym łączymy
- Najczęściej po kluczach
- Ale nie koniecznie



JOIN

INNER JOIN – część wspólna

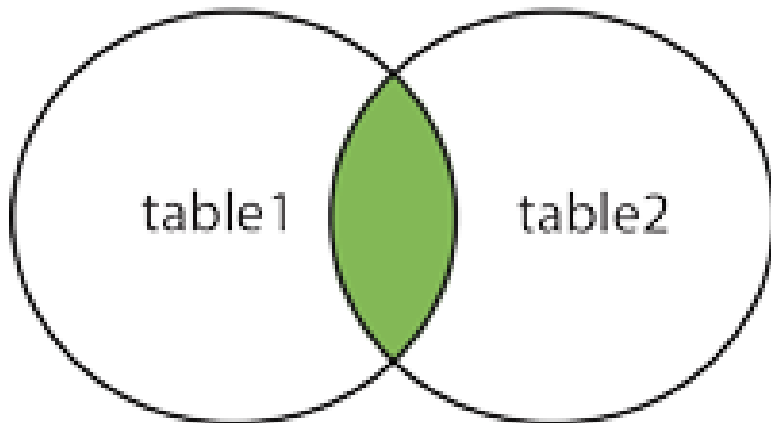
```
SELECT column_name(s)
```

Domyślny JOIN jest INNER

```
FROM table1
```

```
INNER JOIN table2 ON table1.column_name = table2.column_name;
```

INNER JOIN



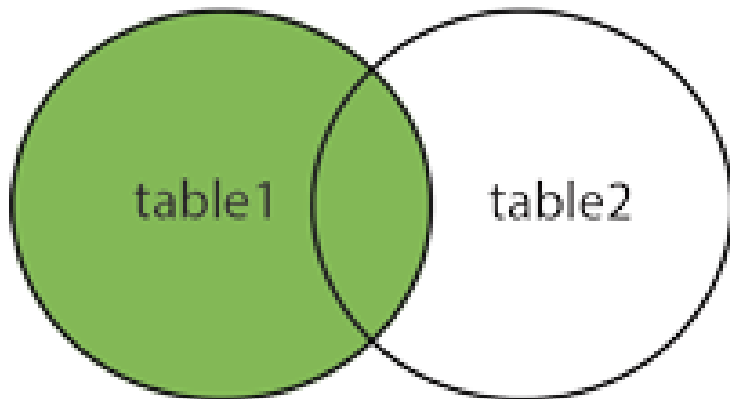
Pola muszą mieć ten sam typ

JOIN

LEFT JOIN – część wspólna + cała lewa tabela

```
SELECT column_name(s)  
FROM table1  
LEFT JOIN table2 ON table1.column_name = table2.column_name;
```

LEFT JOIN



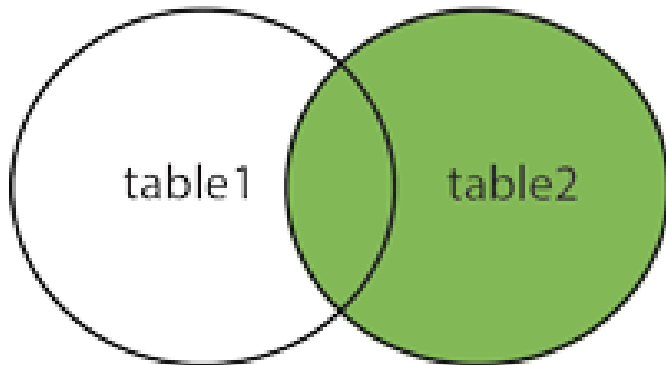
Pola muszą mieć ten sam typ

JOIN

LEFT JOIN – część wspólna + cała prawa tabela

```
SELECT column_name(s)  
FROM table1  
RIGHT JOIN table2 ON table1.column_name = table2.column_name;
```

RIGHT JOIN



Pola muszą mieć ten sam typ

JOIN

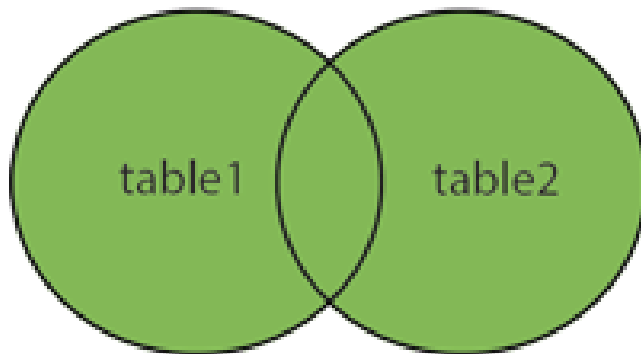
LEFT JOIN – część wspólna + cała prawa tabela + cała lewa

```
SELECT column_name(s)
```

```
FROM table1
```

```
FULL OUTER JOIN table2 ON table1.column_name = table2.column_name;
```

FULL OUTER JOIN



Pola muszą mieć ten sam typ

Przydatne funkcje, operatory

IS NULL, NOW(), CONCAT(...) ...



IS NULL

Wygodny do sprawdzania czy pole ma wartość **NULL**.

```
SELECT * FROM Posts  
WHERE user_id IS NULL;
```

Albo odwrotnie – gdzie jest uzupełnione

```
SELECT * FROM Posts  
WHERE user_id IS NOT NULL;
```

Round

Zaokrąglenie wyników:

```
ROUND(value, precision)
```

```
mysql> SELECT ROUND(4.43123, 2);
```

```
+-----+  
| ROUND(4.43125) |  
+-----+  
|                |  
|                |  
+-----+
```

Concat

Łączenie stringów:

```
CONCAT(value1, value2, value3, ...)
```

```
mysql> SELECT CONCAT('Jan', 'Kowalski');
```

```
+-----+
| CONCAT('Jan', 'Kowalski') |
+-----+
|           Jan Kowalski    |
+-----+
```

NOW() CURRENT_DATE()

```
mysql> SELECT NOW();
```

```
+-----+  
| NOW() |  
+-----+  
|      2017-11-26 17:30:11 |  
+-----+
```

```
mysql> SELECT CURRENT_DATE();
```

```
+-----+  
| CURRENT_DATE() |  
+-----+  
|      2017-11-26 |  
+-----+
```

Porównywanie dat

```
SELECT * FROM users  
WHERE birth_date BETWEEN '1970-07-05' AND '2011-11-10';
```

```
SELECT * FROM users  
WHERE birth_date > '2011-11-10';
```

```
SELECT * FROM users  
WHERE birth_date > DATE('2011-11-10');
```

```
SELECT * FROM users  
WHERE birth_date > STR_TO_DATE('2011-11-10', '%d/%m/%Y');
```

Parę przydatnych linków

- Dokumentacja MySQL: <https://dev.mysql.com/doc/refman/5.7/en/>
- Diagram ER (ERD): <https://www.lucidchart.com/pages/er-diagrams>
- Interaktywny kurs SQL: <https://sqlbolt.com>
- Ściągawka z najważniejszych funkcji i komend SQL: <https://www.w3schools.com/sql/default.asp>
- Absolutne minimum + syntax: https://www.w3schools.com/sql/sql_quickref.asp
- Obraz MySQL w Docker store: <https://store.docker.com/images/mysql>
- IntelliJ i Database tools: <https://www.jetbrains.com/help/idea/writing-and-executing-sql-statements.html>
- Konwencje MySQL: <http://www.sqlstyle.guide/>
- MySQL dump / restore dla serwera bazodanowego w kontenerze Docker:
<https://gist.github.com/michalczukm/e81ba6ccba4132243221f276d08be511>



Dziękuję za uwagę



Kontakt

michalczukm



michalczukm@gmail.com