Zarządzanie bazami danych
Język SQL

Jarosław Wencel

Zasady współpracy

- Punktualność, wg ustalonych ram
- Przerwy ad-hoc – w zależności od energii grupy
- W razie pytań – najlepiej zadać na bieżąco (chat, ręka w górę)

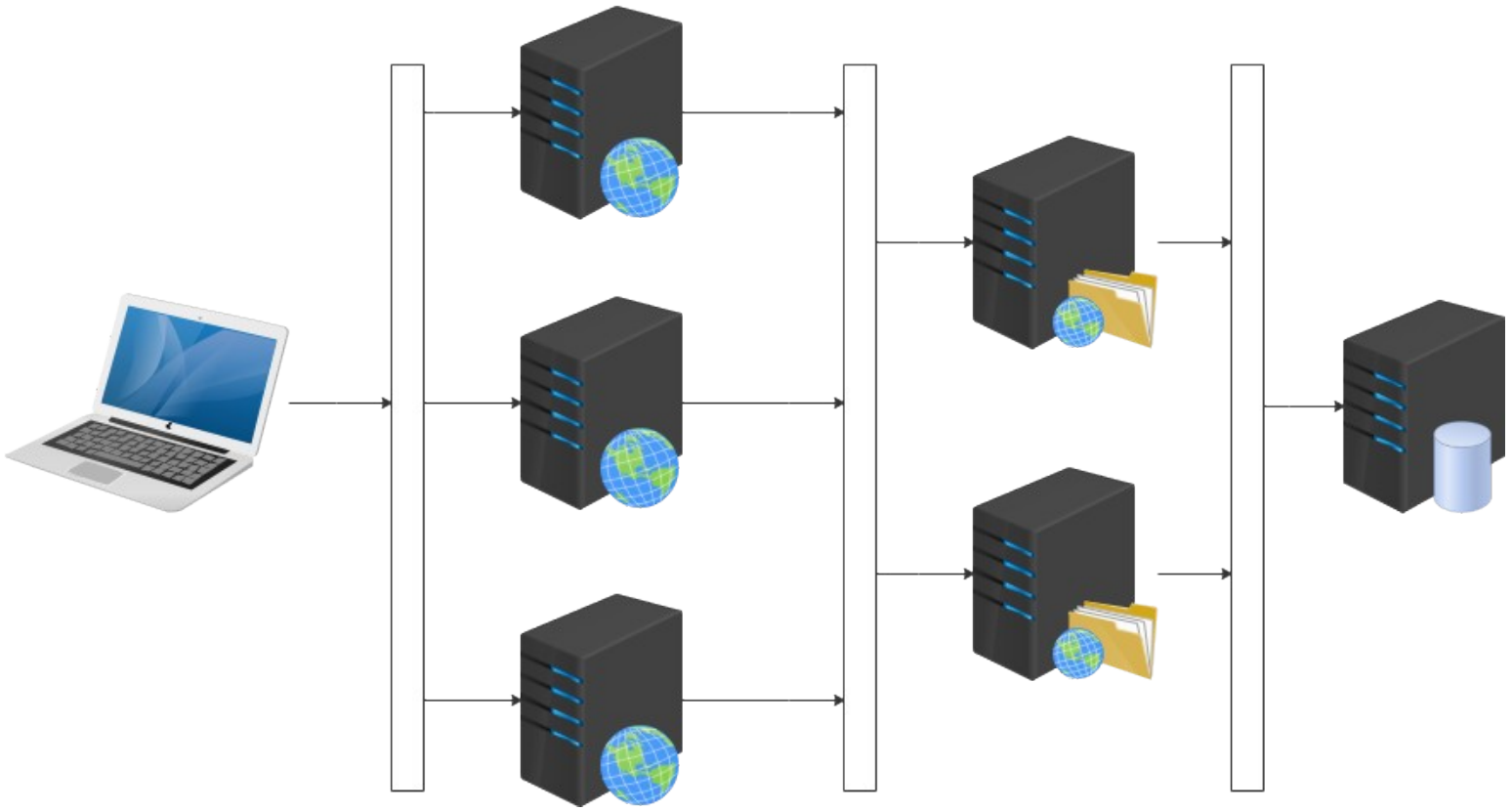
Bibliografia

- C.J. Date *An Introduction to Database Systems* edycja 8
- F.Butzen, D.Forbes *Linux Bazy danych*
- Dokumentacja MariaDB: https://mariadb.com/kb/en/documentation/
- Portal W3schools https://www.w3schools.com/sql/default.asp

Dane

- Dane są znakami, które opisują realnie istniejącą rzecz lub zdarzenie w prawdziwym świecie; przykład: książka – tytuł, autor, ISBN, wydawca, ...
- „to wszystko co jest/może być przetwarzane umysłowo lub komputerowo”
- Wartości znane w rozwiązywaniu problemów fizycznych/matematycznych (np. zadań)
- Informacje lub wiadomości używane do wyciągania wniosków

Stack www



Baza danych

- Baza danych (database) – zbiór danych powiązanych tematycznie, zorganizowany w sposób umożliwiający ich wyszukiwanie według zadanych kryteriów
- Pojęcie „baza danych” rozumiemy potocznie jako system złożony z trzech elementów:
  - właściwej bazy danych – zbioru danych, np. pliki na dysku,
  - systemu (oprogramowania) zarządzającego bazą danych,
  - interfejsu użytkownika, umożliwiającego dostęp do danych

Funkcje baz danych

- Zależne od **użytkownika**
  - tworzenie baz danych i tabel
  - dodawanie i usuwanie danych
  - wyszukiwanie danych („zapytania”, kwerendy)
  - czynności administracyjne
- Zależna od **oprogramowania**
  - zarządzanie fizycznymi zbiorami danych
  - wykonywanie poleceń użytkownika
  - prezentacja wyników operacji

Typy baz danych

- Proste/plikowe bazy danych o pojedynczej tabeli (np. Excel, DBF, własne programy)
- Jednostanowiskowa baza biurowa (np. MS Access)
- Baza typu „klient-serwer” – dostęp z wielu stanowisk (np. Oracle, SQL Server, MariaDB)
- Bazy rozproszone (blockchain, Cassandra)

Struktura bazy danych

- Tabela (table) – dwuwymiarowa struktura przechowująca dane dotyczące określonego tematu
  - kolumny – atrybuty (pola), określony typ danych
  - wiersze – rekordy

Tabela – przykład

Imię	Nazwisko	Adres	Numer
Jan	Kowalski	Kraków, ul. Nowa 10	+48 123 456 789
Marek	Nowak	Warszawa, ul. Krakowska 22	+48 987 654 321

Popularne systemy bazodanowe

- Komercyjne
  - Oracle, Microsoft SQL Server, DB2
- Niekomercyjne, otwarto-źródłowe
  - MariaDB, MySQL, PostgreSQL
- NoSQL
  - MongoDB, Cassandra, Redis

Demo

- Połączenie do bazy przez phpMyAdmin
- Połączenie do bazy przez Workbench
- Model E-R
- Ładowanie pliku

Ćwiczenie

- Załaduj opis bazy danych z pliku lms.mysql.sql do bazy MariaDB
- Wygeneruj model E-R dla utworzonej bazy

Relacyjne bazy danych

- Relacyjna baza danych – zbiór danych zawartych w wielu tabelach połączonych ze sobą relacjami (związkami)
  - jedna tabela dla każdego typu informacji
  - optymalizacja dla dużej ilości danych
  - szybsze wyszukiwanie

Klucze tabeli

- Pozwalają na szybkie wyszukiwanie i kojarzenie informacji przechowywanych w odrębnych tabelach
- Jednoznaczna identyfikacja rekordów – **klucz podstawowy** (primary key)
  - Brak duplikatów
  - Brak wartości pustych (NULL)

Klucze tabeli

- Klucze **jednopolowe** – pojedynczy atrybut identyfikujący rekord, np. numer PESEL
- Klucze **wielopolowe** – gdy żadne z pól samodzielnie nie gwarantuje unikatowej kombinacji, np. autor + tytuł książki
- Klucze typu **autonumeracja** – licznik zwiększany automatycznie przy dodawaniu rekordu

Pola unikatowe i niepuste

- Pola **unikatowe** (unique)
  - nie jest możliwe wprowadzenie dwóch rekordów o identycznej zawartości pola unikatowego
  - klucz podstawowy jest z definicji polem unikatowym
- Pola **niepuste** (not null)
  - pole musi posiadać wartość przy dodawaniu rekordu do bazy
  - klucz podstawowy jest z definicji polem niepustym

Relacje

- Relacje opisują sposób powiązania informacji zawartych w wielu tabelach.
- **Relacja** to związek ustanowiony pomiędzy wspólnymi polami (kolumnami) w dwóch tabelach
- Relacja działa poprzez dopasowanie danych w polach kluczowych — zwykle są to pola o tej samej nazwie w obu tabelach. W większości przypadków dopasowywane pola to **klucz podstawowy** z jednej tabeli, który dostarcza unikatowego identyfikatora dla każdego rekordu, oraz **klucz obcy** w drugiej tabeli.

Klucz obcy

- **Klucz obcy** (foreign key) – jedno lub kilka pól (kolumn) tabeli, które odwołują się do pola lub pól klucza podstawowego w innej tabeli.
- Klucz obcy wskazuje sposób powiązania tabel relacjami
  - typy danych w polach klucza podstawowego i obcego muszą być zgodne
  - nazwy pól klucza podstawowego i obcego nie muszą być identyczne (ale dla wygody często nadaje się tą samą nazwę)

Relacja 1:1

- W relacji jeden-do-jednego każdy rekord w tabeli A może mieć tylko jeden dopasowany rekord z tabeli B, i tak samo każdy rekord w tabeli B może mieć tylko jeden dopasowany rekord z tabeli A.
- Relacji jeden-do-jednego można używać do:
  - podziału tabeli z wieloma polami,
  - odizolowania tabeli ze względów bezpieczeństwa,
  - przechowania informacji odnoszącej się tylko do podzbioru tabeli głównej.

Relacja 1:M

- rekord w tabeli A może mieć wiele dopasowanych do niego rekordów z tabeli B, ale rekord w tabeli B ma tylko jeden dopasowany rekord w tabeli A
- najczęściej występujący typ relacji

Relacja M:N

- Rekord w tabeli A może mieć wiele dopasowanych do niego rekordów z tabeli B i tak samo rekord w tabeli B może mieć wiele dopasowanych do niego rekordów z tabeli A.
- Jest to możliwe tylko przez zdefiniowanie trzeciej tabeli (nazywanej tabelą łącza), której klucz podstawowy składa się z dwóch pól — kluczy obcych z tabel A i B.
- Relacja wiele-do-wielu jest w istocie dwiema relacjami jeden-do-wielu z trzecią tabelą

Niezależność modelu od danych

- pozwala na modyfikowanie struktury danych bez wpływu na istniejące programy,
- do tabel mogą być dodawane kolumny, tabele mogą być dołączane do bazy danych, a nowe relacje mogą być tworzone bez konieczności wprowadzania istotnych zmian do tabel

Etapy projektowania

1. Określenie celu
2. Określenie modelu danych i ich atrybutów (tabel i pól)
3. Określenie jednoznacznych identyfikatorów rekordów
4. Określenie relacji między tabelami
5. Wprowadzenie danych (import, ETL)
6. Opracowanie metod dostępu do danych (formularze, raporty)

Wybór typu pola

- Jakie wartości będą dozwolone dla pola?
- Ile miejsca będą mogły zajmować dane przechowywane w polu?
- Jakie operacje będą wykonywane na wartościach w polu?
- Czy wartości w polu będą sortowane lub indeksowane?
- Czy pole będzie używane do grupowania rekordów w kwerendach lub raportach?

Wybór typu pola #2

- Numeryczne
  https://mariadb.com/kb/en/data-types-numeric-data-types/
- Znakowe
  https://mariadb.com/kb/en/string-data-types/
- Daty i czasu
  https://mariadb.com/kb/en/date-and-time-data-types/
- AUTO_INCREMENT & NULL

Demo

Książka telefoniczna

- Abonenci identyfikowani przez imię i nazwisko
- Dla każdego abonenta mamy możliwość dodania adresów domowego i służbowego, oraz numerów telefonów domowego i komórkowego

Ćwiczenie

- Zaprojektuj bazę danych dla biblioteki
- Rozważ ilość tabel i relacje pomiędzy nimi
- Zdefiniuj nazwy pól oraz ich typy w poszczególnych tabelach
- Zastanów się nad ilością rekordów oraz transakcji w każdej z tabel

Normalizacja struktury bazy

- Organizacja struktury danych wg określonych reguł
- Cele reorganizacji to
  - Zwiększenie elastyczności
  - Eliminacja nadmiarowości i niespójności

Pierwsza postać normalna

W skrócie 1NF

- Zapewnij atomowe atrybuty w tabelach
- Wyeliminuj powtarzające się grupy w poszczególnych tabelach
- Utwórz oddzielną tabelę dla każdego zestawu powiązanych danych
- Identyfikowanie każdego zestawu powiązanych danych za pomocą klucza podstawowego

Druga postać normalna

W skrócie 2NF

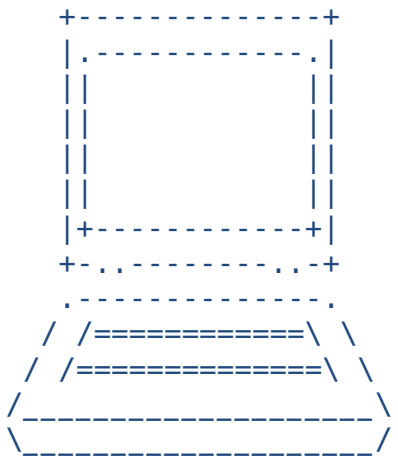
- Tworzenie oddzielnych tabel dla zestawów wartości, które mają zastosowanie do wielu rekordów
- Odnoszą się do tych tabel z kluczem obcym

Trzecia postać normalna

W skrócie 3NF

- Wyeliminuj pola, które nie zależą od klucza, czyli powtórzenia tych samych wartości dla wielu rekordów (np. słowniki)

Demo normalizacji



Ćwiczenie

- Znormalizuj schemat bazy danych z poprzedniego ćwiczenia

Transakcje w bazie danych

- ACID
  - Atomicity,
  - Consistency,
  - Isolation,
  - Durability
- Transakcje zabezpieczają nas przed przypadkowym uszkodzeniem danych podczas zmian wprowadzanych przez wielu użytkowników w tym samym czasie

Język SQL

Co to jest SQL

SQL (ang. Structured Query Language) – strukturalny język zapytań używany do tworzenia, modyfikowania baz danych oraz do umieszczania i pobierania danych z baz danych

Z czego składa się SQL

Język SQL można podzielić na:

- DML (ang. Data Manipulation Language)
- DDL (ang. Data Definition Language)
- DCL (ang. Data Control Language)

Zasady

- W języku SQL nie rozróżnia się wielkich i małych liter (ale wielkie i małe litery stosujemy dla czytelności),
- W poleceniach SQL ignorowane są znaki końca linii (jedno polecenie może zostać zapisane w wielu liniach),
- Każde polecenie SQL powinno być zakończone średnikiem

DML

- DML (Data Manipulation Language) służy do wykonywania operacji na danych – do ich umieszczania w bazie, kasowania, przeglądania, zmiany
- Polecenia z tego zbioru to:
  - SELECT – pobranie danych z bazy
  - INSERT – umieszczenie danych w bazie
  - UPDATE – zmiana danych
  - DELETE – usunięcie danych z bazy
- Dane tekstowe muszą być ujęte w znaki pojedynczego cudzysłowu (')

DDL

- Dzięki DDL (Data Definition Language) można operować na strukturach, w których dane są przechowywane – czyli np. dodawać, zmieniać i kasować tabele lub bazy
- Najważniejsze polecenia tej grupy to:
  - CREATE – utworzenie struktury (bazy, tabeli, indeksu itp.), np. CREATE TABLE, CREATE DATABASE
  - DROP – usunięcie struktury, np. DROP TABLE, DROP DATABASE
  - ALTER – zmiana struktury (dodanie kolumny do tabeli, zmiana typu danych w kolumnie tabeli) np. ALTER TABLE ADD COLUMN

DCL

- DCL (Data Control Language) ma zastosowanie do nadawania uprawnień do obiektów bazodanowych.
- Najważniejsze polecenia w tej grupie to:
  - GRANT: przyznanie wszystkich praw do tabeli EMPLOYEE użytkownikowi PIOTR z opcją pozwalającą mu nadawać prawa do tej tabeli:
    GRANT ALL PRIVILEGES ON EMPLOYEE TO PIOTR
    WITH GRANT OPTION;
  - REVOKE: odebranie użytkownikowi wszystkich praw do tabeli, które zostały przyznane poleceniem GRANT,
  - DENY: zabronienie dostępu do obiektu.

Składnia polecenia SELECT

SELECT <nazwa kolumny>, <nazwa kolumny>, …

FROM <nazwa tabeli>

WHERE <nazwa kolumny + ograniczenia>

GROUP BY <nazwa kolumny>

ORDER BY <nazwa kolumny> ASC lub DESC

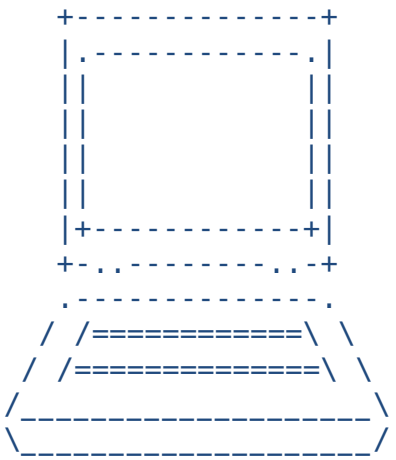
Zapytania SELECT

- Projekcja – wybieramy kolumny do wyświetlenia
- Aliasy – nadajemy nowe nazwy atrybutom po modyfikacji (concat(imie, nazwisko) as imienazwisko)
- Sortowanie – ORDER BY
- Selekcja/filtrowanie – WHERE
- Eliminacja duplikatów – DISTINCT

Operatory logiczne

- = , != , <> , > , >= , < , <=
- BETWEEN ... AND …
- IN (wart1, wart2, wart3)
- LIKE (WHERE imie LIKE ‘A%’)
- IS NULL / IS NOT NULL

Demo



Grupowanie/agregacja

- SELECT count(*) FROM abonenci;
- SELECT imię, count(*) FROM abonenci GROUP BY imię;
- SELECT imię, count(*) FROM abonenci GROUP BY imię HAVING count(*)>2;

Limity

- SELECT * FROM tabela LIMIT 10;
- Różne silniki bazy danych – różne podejścia
  - TOP
  - ROWNUM

Funkcje agregujące

- COUNT
- SUM
- AVG
- MAX
- MIN
- https://mariadb.com/kb/en/aggregate-functions/

Funkcje znakowe

- LEN
- UPPER
- LOWER
- CONCAT
- https://mariadb.com/kb/en/string-functions/

NULL

- IFNULL(a,b) – MariaDB, SQL Server
- NVL(a,b) – Oracle

Jeśli a = NULL zwraca b

HAVING

- Filtr klauzuli HAVING jest stosowany wobec każdej osobnej grupy rekordów otrzymanych w wyniku klauzuli GROUP BY,
- Filtr HAVING pozostawia tylko te rekordy, dla których wartość podanego warunku będzie równa TRUE

Ćwiczenie

- Wyświetl wszystkich pracowników
- Wyświetl wszystkie stanowiska, których nazwa zaczyna się od ‘S’
- Wyświetl wszystkie zespoły na produkcji
- Wyświetl ilość pracowników
- Wyświetl średnie wynagrodzenie
- Wyświetl tylko pięciu pracowników
- Wyświetl pracowników z powtarzającymi się imionami

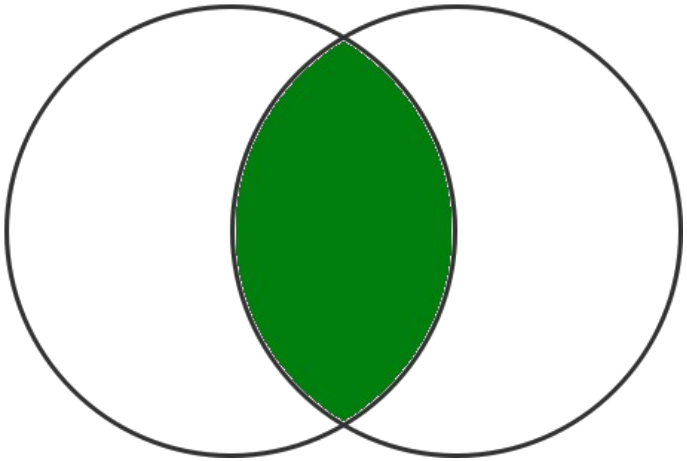
SELECT – złączenia tabel

- INNER JOIN
- LEFT OUTER JOIN
- RIGHT OUTER JOIN

- FULL OUTER JOIN
- CROSS JOIN

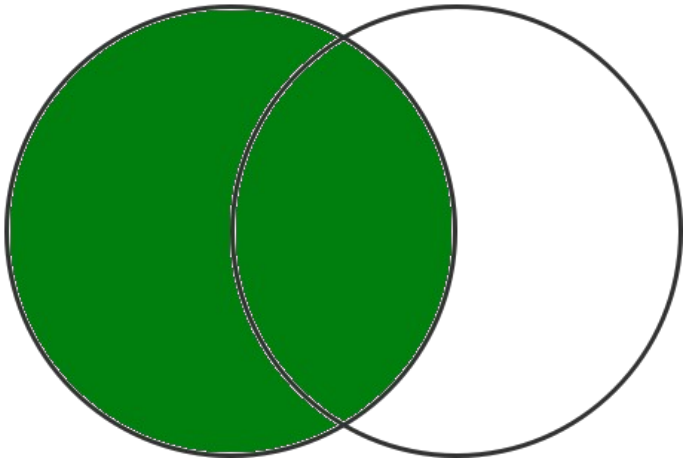
- LEFT OUTER JOIN – WHERE NULL
- RIGHT OUTER JOIN – WHERE NULL
- FULL OUTER JOIN – WHERE NULL

INNER JOIN



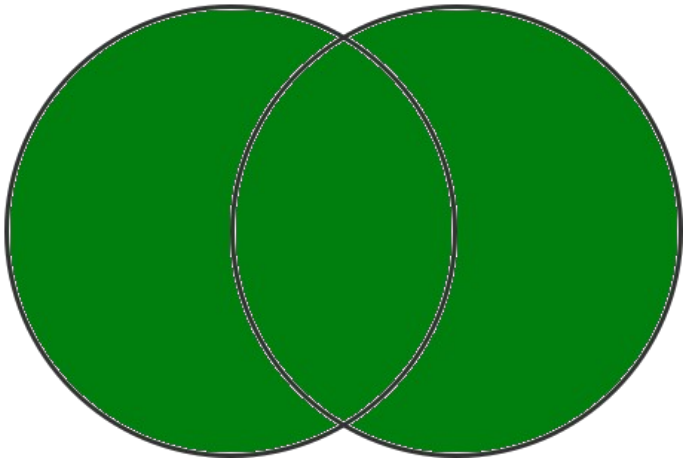
```
SELECT *
FROM t1 INNER JOIN t2
ON t1.a = t2.b
```

LEFT JOIN



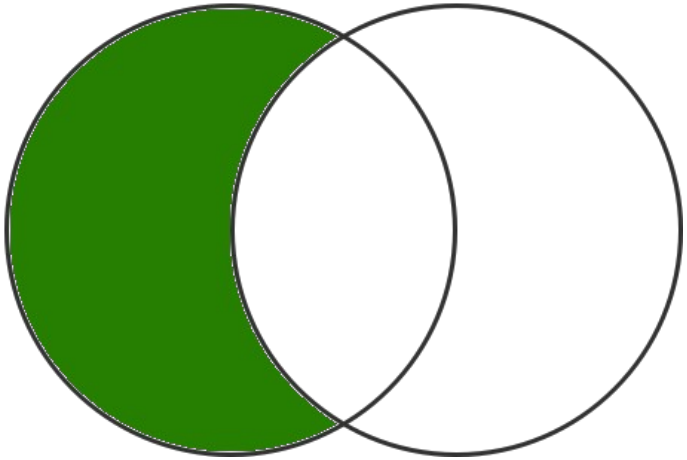
```
SELECT *
FROM t1 LEFT JOIN t2
ON t1.a = t2.b
```

CROSS JOIN



```
SELECT *
FROM t1 CROSS JOIN t2
-- brak klauzuli ON !!
```

OUTER JOIN … WHERE NULL



```
SELECT *
FROM t1 LEFT JOIN t2
ON t1.a = t2.b
WHERE t2.b IS NULL
```

Ćwiczenie

- Wyświetl listę pracowników z przyporządkowaną nazwą stanowiska, zespołem i działem
- Wyświetl nieobsadzone stanowiska
- Wyświetl ilość pracowników w dziale
- Wyświetl średnie wynagrodzenie w dziale
- Wyświetl osobę z maksymalnym wynagrodzeniem w dziale

INSERT

INSERT INTO <nazwa tabeli>
(<nazwa kolumny>, <nazwa kolumny>)
VALUES (<wartość>, <wartość>);

INSERT – przykłady

- Przykład z polami lub bez
- Niepełna ilość pól – wartości defaultowe

SELECT INTO

Standardowy SELECT plus

- INTO OUTFILE '/tmp/outfile.txt';
  żeby zapisać wynik do pliku tekstowego
- INTO DUMPFILE '/tmp/dumpfile.txt';
  żeby zapisać surowy wynik do pliku, np. do wyciągania BLOBów z bazy

UPDATE

UPDATE <nazwa tabeli>
SET <nazwa kolumny> = ‘wartość’
WHERE <nazwa kolumny> = ‘wartość’;

UPDATE – przykłady

- Dlaczego ważny jest WHERE
- Kalkulacje w trakcie
- subquery

Ćwiczenia

- Dodać pracowników do bazy tak, aby obsadzić brakujące stanowiska
- Podwyższyć wynagrodzenie o 10% wszystkim pracownikom, którzy są zatrudnieni ponad 10 lat

DELETE

```
DELETE
FROM <nazwa tabeli>
WHERE <argumenty>;
```

DELETE – przykłady

- O czym warto pomyśleć przed wykonaniem polecenia DELETE?
- Co wydarzy się gdy nie podamy WHERE?

TRUNCATE

TRUNCATE <tbl_name>;

TRUNCATE vs. DELETE

- TRUNCATE usuwa wszystkie wiersze z tabeli, bez wyjątku, przez usunięcie tabeli i stworzenie jej od nowa
- DELETE usuwa wiersz po wierszu, co dla dużych tabel może długo trwać
- TRUNCATE wywołuje COMMIT automatycznie

Ćwiczenie

- Utwórz tabelę KONTRAKTORZY zawierającą pola Imie i Nazwisko, przelozony, wynagrodzenie, data_zatrudn
- Wypełnij tabelę KONTRAKTORZY danymi (10 rekordów), można korzystać z https://random-data-generator.com/generator-losowych-imion-i-nazwisk/

Algebra zbiorów

- UNION – łączenie zbiorów

```
SELECT ...
(INTERSECT [ALL | DISTINCT] | EXCEPT [ALL | DISTINCT] |
UNION [ALL | DISTINCT]) SELECT ...
[(INTERSECT [ALL | DISTINCT] | EXCEPT [ALL | DISTINCT]
| UNION [ALL | DISTINCT]) SELECT ...]
[ORDER BY [column [, column ...]]]
[LIMIT {[offset,] row_count | row_count OFFSET offset}]
```

Algebra zbiorów

- EXCEPT – odejmowanie zbiorów

```
SELECT ...
(INTERSECT [ALL | DISTINCT] | EXCEPT [ALL | DISTINCT] |
UNION [ALL | DISTINCT]) SELECT ...
[(INTERSECT [ALL | DISTINCT] | EXCEPT [ALL | DISTINCT]
| UNION [ALL | DISTINCT]) SELECT ...]
[ORDER BY [column [, column ...]]]
[LIMIT {[offset,] row_count | row_count OFFSET offset}]
```

Algebra zbiorów

- INTERSECT – część wspólna zbiorów

```
SELECT ...
(INTERSECT [ALL | DISTINCT] | EXCEPT [ALL | DISTINCT] |
UNION [ALL | DISTINCT]) SELECT ...
[(INTERSECT [ALL | DISTINCT] | EXCEPT [ALL | DISTINCT]
| UNION [ALL | DISTINCT]) SELECT ...]
[ORDER BY [column [, column ...]]]
[LIMIT {[offset,] row_count | row_count OFFSET offset}]
```

Ćwiczenie

- Wygeneruj listę wszystkich osób pracujących w firmie wraz z ich miesięcznym wynagrodzeniem

Widoki / perspektywy

- Dlaczego stosujemy
  - Upraszczamy skomplikowane zapytania
  - Zarządzanie uprawnieniami/dostępem do danych
  - Gdy w aplikacji nie ma miejsca na skomplikowane zapytania

Ćwiczenie

- Stwórz widok dla zapytania z poprzedniego zadania
- Policz sumę wynagrodzeń w miesiącu w oparciu o nowo stworzony widok

Backup = kopia zapasowa

- Backup logiczny - MYSQLDUMP
  - Zestaw poleceń SQL, które pozwolą nam odtworzyć bazę danych na dany punkt w czasie
- Backup fizyczny - MARIABACKUP
  - Kopia binarna plików bazy danych

Backup - ćwiczenie

- Mysqldump
- https://github.com/SerafinSahary/mybackup

Jakie są Państwa pytania?