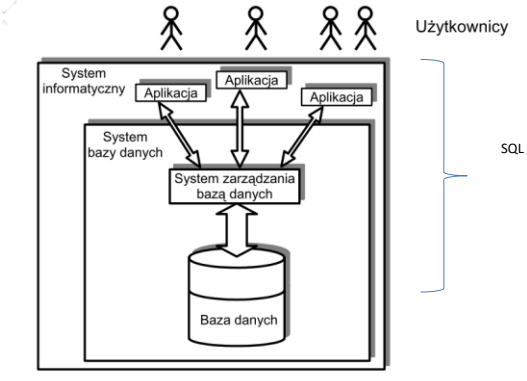
Podstawy języka SQL

- Strukturalny język zapytań SQL
- SQL – charakterystyka składni
- Klauzule SQL
- Funkcje w bazach danych
- Rozkazy języka SQL – tworzenie tabel i operacje na tabelach, wyszukiwanie informacji i ich zamiana
- Podzapytania
- Złączenia i widoki
- Transakcje

Strukturalny język zapytań SQL

- Historia języka SQL
- Podgrupy języka SQL
- Selekcja, projekcja, złączenie, suma



Czym jest SQL?

SQL to Strukturalny Język Zapytań (z ang. Stuctured Query Language) służący do wykonywania wszelkich operacji **w systemach relacyjnych baz danych** RDBMS (z ang. Relational Database Management System).
Jest to język uniwersalny(dialekty), stosowany zarówno w systemach komercyjnych, jak: DB2, MS SQL czy Oracle, jak i rozwijanych na zasadach OpenSource, jak PostgreSQL czy darmowe wersje MySQL.
Początki SQL sięgają wczesnych lat 70. ubiegłego wieku, kiedy to w firmie IBM został rozpoczęty projekt bazodanowy o nazwie System/R, na którego potrzeby powstał język SEQUEL (z ang. Structured English Query Language). Druga wersja SEQUEL-a, SEQUEL/2, w roku 1997 została przemianowana na SQL.
SQL pozwala na wykonywanie następujących operacji:
Uzyskiwanie (pobieranie), modyfikowanie, wstawianie, usuwanie, sterowanie danych

Podgrupy SQLa

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 (np. CREATE TABLE, CREATE DATABASE, ...) – utworzenie struktury (bazy, tabeli, indeksu itp.),
DROP (np. DROP TABLE, DROP DATABASE, ...) – usunięcie struktury,
ALTER (np. ALTER TABLE ADD COLUMN ...) – zmiana struktury (dodanie kolumny do tabeli, zmiana typu danych w kolumnie tabeli).

DML (**Data Manipulation Language**) służy do wykonywania operacji na danych – do ich umieszczania w bazie, kasowania, przeglądania oraz dokonywania zmian. Najważniejsze polecenia z tego zbioru to:
INSERT – umieszczenie danych w bazie,
UPDATE – zmiana danych,
DELETE – usunięcie danych z bazy.
Dane tekstowe muszą być zawsze ujęte w znaki pojedynczego cudzysłowu (').

Podgrupy SQLa

DCL (**Data Control Language**) ma zastosowanie do nadawania uprawnień do obiektów bazodanowych. Najważniejsze polecenia w tej grupie to:

GRANT - służące do nadawania uprawnień do pojedynczych obiektów lub globalnie konkretnemu użytkownikowi
REVOKE – służące do odbierania wskazanych uprawnień konkretnemu
DENY - służy głównie do odbierania uprawnień dostępu do obiektów bazodanowych

DQL (**Data Query Language**) to język formułowania zapytań do bazy danych. W zakres tego języka wchodzi jedno polecenie - **SELECT**.

Często SELECT traktuje się jako część języka DML, ale to podejście nie wydaje się właściwe, ponieważ DML z definicji służy do manipulowania danymi - ich tworzenia, usuwania i uaktualniania.

TCL Transaction Control Language – język sterowania przepływem danych (kontrola transakcji) . Najważniejsze polecenia w tej grupie to:

COMMIT – zatwierdzenie transakcji
ROLLBACK – wycofanie transakcji
SAVEPOINT – punkt przywracania transakcji

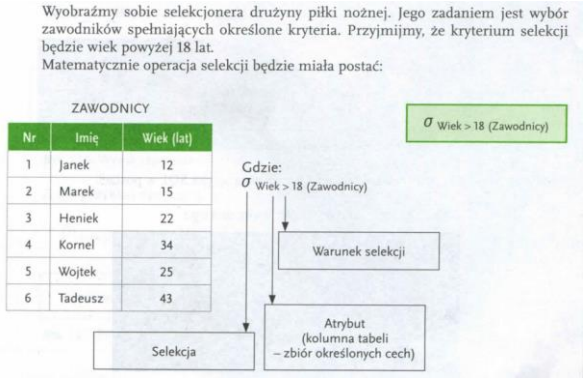
Podczas pracy z bazą danych zachodzi konieczność wydobycia określonych informacji, zwłaszcza takich, które spełniają pożądany przez nas warunek. Ponieważ zmienna relacyjna (tabela) ma dane pogrupowane dzięki atrybutom (kolumnom), możemy na takich zbiorach danych wykonywać operacje.
Na początek trochę teorii. Operacje, o których mówimy, przedstawia się za pomocą operatorów algebry relacyjnej. Operator taki na wejściu pobiera argumenty będące relacjami, natomiast zwraca relację wynikową. Operatory algebry relacyjnej przedstawia tabela:

∪ б ⋈ ∏

Operator	Symbol
Selekcja	б
Projekcja	∏
Złączenie	⋈
Suma	∪

SELEKCJA

Wyobraźmy sobie selekcjonera drużyny piłki nożnej. Jego zadaniem jest wybór zawodników spełniających określone kryteria. Przyjmijmy, że kryterium selekcji będzie wiek powyżej 18 lat.
Matematycznie operacja selekcji będzie miała postać:



W strukturalnym języku zapytań taka selekcja będzie miała postać:

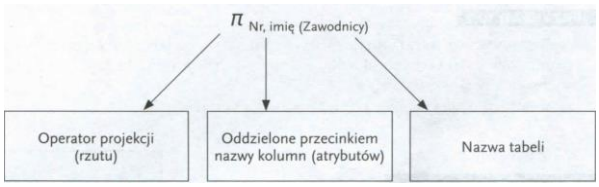
SELECT * FROM zawodnicy WHERE wiek > 18;

W efekcie otrzymamy wynik:

```
mojabaza=# SELECT * FROM zawodnicy WHERE wiek > 18;
 nr |  imie   | wiek
----+---------+------
  3 | Heniek  |   22
  4 | Kornel  |   34
  5 | Wojtek  |   25
  6 | Tadeusz |   43
(4 wiersze)
```

Należy pamiętać, że zapytanie kończymy, wpisując znak średnika ;

Projekcja nazywana bywa również **rzutem** i możemy ją zdefiniować **jako wybór kolumn**. Operator projekcji na wejściu przyjmuje nazwy kolumn, a na wyjściu zwraca ich zawartość. Możemy prześledzić to na wcześniejszym przykładzie z selekcjonerem piłkarskim. Wyobraźmy sobie sytuację, że selekcjoner będzie potrzebował listy wszystkich zawodników wraz z kandydatami (zawodnikami poniżej 18 roku życia). Lista będzie musiała składać się z imion i numerów zawodników. Aby otrzymać taki zbiór za pomocą algebry relacyjnej, należy posłużyć się wyrażeniem:



Wyrażenie to możemy również zapisać za pomocą języka SQL w postaci:

SELECT nr, imie FROM zawodnicy;

```
mojabaza=# SELECT nr, imie FROM zawodnicy;
 nr |  imie
----+---------
  1 | Janek
  2 | Marek
  3 | Heniek
  4 | Kornel
  5 | Wojtek
  6 | Tadeusz
(6 wierszy)
```

Złączenie (*JOIN*) służy do pobierania danych z dwóch lub większej liczby tabel w celu porównania lub zestawienia. W tabelach biorących udział w złączeniu muszą występować kolumny, które są zgodne i spełniają warunki pozwalające na dokonanie złączenia. **Zaleca się, aby kolumny te łączyły dwie relacje (tabele) na zasadzie: klucz podstawowy, klucz obcy**, chociaż warunek ten nie jest niezbędny do wykonania złączenia.
Dla przykładu wykorzystamy dwie tabele: zawodnicy i pokoje.

nr [PK] integer	imie character varying(50)	wiek integer
1	Janek	12
2	Marek	15
3	Heniek	22
4	Kornel	34
5	Wojtek	25
6	Tadeusz	43

id_pokoju [PK] integer	cena_za_dobe integer	id_goscia integer
1	50	1
2	60	2
3	70	3
4	20	4
5	10	5
6	23	6

1.Mateusz Kowalski
2.Mateusz Kowalski

Gdy założymy, że numer gościa (kolumna **id_goscia**) odpowiada numerowi zawodnika (kolumnie **nr**), wówczas, chcąc otrzymać zestawienie **imienia** zawodnika oraz **ceny**, którą ma zapłacić za pokój, wykonujemy złączenie:
SELECT imię,cena_za_dobe FROM zawodnicy JOIN pokoje ON nr=id_goscia;

```
mojabaza=# SELECT imie, cena_za_dobe FROM zawodnicy JOIN pokoje ON nr=id_goscia;
  imie   | cena_za_dobe
---------+--------------
 Marek   |           60
 Janek   |           50
 Heniek  |           70
 Kornel  |           20
 Wojtek  |           10
 Tadeusz |           23
(6 wierszy)
```

A oto przykład operacji równozłączenia zapisany językiem algebry relacji:

$$(\Pi_{\text{imie(ZAWODNICY)}}) \bowtie_{\text{zawodnicy.id = pokoje.id\_pokoju}} (\Pi_{\text{cena\_za\_dobe(POKOJE)}})$$

Operator złączenia

Znak równości determinujący typ złączenia – równozłączenie

Suma (Union)
Teoriomnogościowa suma dotyczy dwóch relacji o tym samym schemacie - językiem algebry relacji może być zapisana jako A∪B.
Dla przykładu wykorzystaliśmy dwie tabele: **prac_p_pomoc** i **pracownicy_bhp** Jak zostało to przedstawione na poniższej ilustracji, tabele mają tę samą strukturę.

id_pracownika [PK] integer	imie character varying(50)	nazwisko character varying(50)	wiek integer
1	Jan	Nowak	24
2	Wojciech	Kowalski	30
3	Michal	Niemczak	23
4	Renata	Zawadzka	25
5	Irena	Szalowska	55
6	Wiktoria	Kowalska	23
7	Bozena	Przykoscinska	40

id_pracownik [PK] integer	imie character varying(50)	nazwisko character varying(50)	wiek integer
1	Jan	Nowak	24
2	Wojciech	Kowalski	30
3	Marian	Nowak	33
4	Marcin	Tracz	26
5	Ewa	Werner	23

$$\Pi_{\text{id\_pracownika,imie,nazwisko,wiek}}(\text{prac\_p\_pomoc})$$

$$\cup\ \Pi_{\text{id\_pracownika,imie,nazwisko,wiek}}(\text{pracownicy\_bhp})$$

```
mojabaza=# SELECT * FROM prac_p_pomoc
mojabaza-# UNION
mojabaza-# SELECT * FROM pracownicy_bhp;
 id_pracownika |   imie   |    nazwisko    | wiek
---------------+----------+----------------+------
             3 | Marian   | Nowak          |   33
             5 | Ewa      | Werner         |   23
             3 | Michal   | Niemczak       |   23
             7 | Bozena   | Przykoscinska  |   40
             2 | Wojciech | Kowalski       |   30
             4 | Marcin   | Tracz          |   26
             5 | Irena    | Szalowska      |   55
             4 | Renata   | Zawadzka       |   25
             6 | Wiktoria | Kowalska       |   23
             1 | Jan      | Nowak          |   24
(10 wierszy)
```