Wykład I

*Terminy, określenia:*

- Relacja – tabela
- Krotka – wiersz ( rekord )
- Atrybut – kolumna (pole )
- Stopień relacji – liczba atrybutów
- Klucz główny **relacji** – jednoznaczny identyfikator krotki
- Dziedzina – zbiór dopuszczalnych wartości danego atrybutu
- Schemat relacji **R( $A_1$, $A_2$, ... ,$A_n$)**
- **Baza danych – zbiór relacji**

**SQL** – Structured Query Language

- stworzony na początku lat 70 ubiegłego wieku w IBM przez Donalda Messerly'ego, Donalda Chamberlina oraz Raymonda Boyce'a pod nazwą SEQUEL
- pierwszy SZBD – System R utworzony przez IBM w 1973 r wykorzystujący SEQUEL
- pierwszy komercyjny SZBD stworzony przez Relational Software ( obecnie Oracle ) w 1979 r dla amerykańskiej marynarki wojennej, CIA i kilku agencji rządowych
- od 1986 r SQL jest oficjalnym standardem ( ANSI ), od 1987 ( ISO )
- aktualny standard SQL: 2011

**Cechy SQL:**

- język deklaratywny
- język interpretowany
- pod język danych
- dostęp do danych z poziomu relacji
- zbliżony konstrukcją do języka naturalnego
- wykorzystuje logikę trójwartościową
- wykorzystuje algebrę relacyjną
- nie jest w pełni zgodny z modelem relacyjnym

**Możliwości SQL**

- wyszukiwanie danych
- dopisywanie, usuwanie i modyfikacja danych
- tworzenie, usuwanie i modyfikowanie obiektów bazo_danowych
- zarządzanie transakcjami
- zarządzanie bazą danych

**Formy języka**

- SQL interakcyjny ( autonomiczny )
- statyczny kod SQL ( *Static SQL* )
  - osadzony SQL ( *Embedded SQL* )
  - język modułów
- dynamiczny kod SQL ( Dynamic SQL )

## Składowe SQL

- **DML** ( *Data Manipulation Language* ) do manipulacji danymi, zawierający polecenia **SELECT, UPDATE, INSERT, DELETE** itp.

- **DDL** ( *Data Definition Language* ) do definiowania obiektów baz danych; polecenia **CREATE, DROP, ALTER** itp.

- **DCL** ( *Data Control Language* ) do zarządzania bazą danych polecenia **COMMIT, ROLLBACK, GRANT, REVOKE** itp.

## Wady SQL

- standard
  - nie jest przestrzegany przez różnych producentów narzędzi do baz danych
  - nie określa zachowań bazy danych w wielu sytuacjach ( np.: indeksowanie, składowanie )
- zbliżenie SQL'a do języka mówionego spowodowało większą złożoność składni
- czasem brak konsekwencji w składni wynikająca z pozostawienia starych form
  - zbyt łatwe uzyskiwanie wyniku iloczynu kartezjańskiego

## Składniki polecenia ( instrukcji ) **SQL**

- klauzule, niektóre opcjonalne
- wyrażenia
- predykaty określające warunki
- średnik kończący polecenie
- wielkość liter w składni nie ma znaczenia

## Podstawowe typy danych **SQL** wg **ANSI**

- **CHARACTER**( *n* ), **CHAR**( *n* ) - ciąg znaków o stałej długości określone przez rozmiar *n*
- **CHARACTER VARYING**( *n* ), **CHAR VARYING**( *n* ) - ciąg znaków o zmiennej długości, nie przekraczającej podanego rozmiaru *n*
- **NUMERIC**( *p* , *s* ), **DECIMAL**( *p* , *s* ) - liczby o ustalonej precyzji *p* i skali *s*
- **INTEGER**, **INT**, **SMALLINT** – liczby całkowite
- **FLOAT**( b ), **DOUBLE PRECISION**, **REAL** – liczby zmiennoprzecinkowe
- **DATE** – data w formacie: YYYY-MM-DD
- **TIME** – czas w formacie: HH:MI:SS
- **TIMESTAMP** – data wraz z czasem – znacznik czasu w formacie : YYYY-MM-DD HH:MI:SS
- **INTERVAL** – przedział czasu
- **BLOB** – dane binarne
- **CLOB** – dane znakowe
- **XML** – dane w formacie XML

Wykład I

**Przedziały czasu i arytmetyka czasu**
Wyróżnia się dwa typy przedziałów czasowych:

- lata do miesięcy
- dni do sekundy

***Przykłady :***

**INTERVAL** '3-5' **YEAR TO MONTH** – przedział 3 lat i 5 miesięcy
**INTERVAL** '5' **YEAR** – przedział 5 lat
**INTERVAL** '5 10:12' **DAY TO MINUTE** - przedział 5 dni 10 godzin i 12 minut

- na przedziałach czasu, datach, liczbach można wykonywać operacje: dodawania, odejmowania a nawet mnożenia

***Składnia zapytania***

**SELECT [ DISTICT ]** < *lista_wynikowa* > **FROM** < *źródło_danych* >
**[ WHERE** < *warunek_logiczny_dla_wierszy* > **]**
**[ GROUP BY** < *kryterium_grupowania* >
**[ HAVING** < *warunek_logiczny_dla_grup* > **] ]**
**[ ORDER BY** < *kryterium_porządkowania* > **] ;**

*Klauzule **SELECT** oraz **FROM** są obowiązkowe; kolejność poszczególnych klauzul jest ważna*

Klauzula **SELECT** określa schemat relacji wynikowej;

- słowo kluczowe **DISTINCT** ( bez duplikatów) – stosujemy aby w wyniku otrzymać różne krotki ( wiersze )
- praktykuje się użycie operatora * - lista wynikowa zawiera wszystkie atrybuty z relacji źródłowych
- lista wynikowa to elementy oddzielane przecinkiem
- element z listy może być atrybutem z relacji źródłowych, wyrażeniem typu liczbowego, tekstowego, datowego w szczególności funkcją
- istnieje możliwość stosowania aliasów ( przezwisk ) dla wyrażeń

Klauzula **FROM** określa źródło danych;

- może to być jedna ( *nazwana* ) relacja, lub połączone relacje
- w szczególności ( *nienazwana* ) relacja wynik podzapytania

Klauzula **WHERE** określa warunek logiczny jaki ma być spełniony przez krotki
( wiersze tabeli ) relacji wynikowej
Warunek logiczny budujemy używając operatory:

- porównania ( =, <>, <, <=, >, LIKE, NOT LIKE )
- zawierania w przedziale ( BETWEEN . . . AND )
- zawierania w zbiorze ( IN )
- operatorów logicznych (ALL, AND, ANY, EXISTS, NOT, OR )
- operatory do obsługi wartości NULL ( IS NULL, IS NOT NULL)

oraz atrybuty z relacji źródłowych, literały, a także funkcje

Klauzula **GROUP BY** umożliwia podzielenie relacji wynikowej na podzbiory – agregację
- pojedynczy podzbiór/grupę stanowią krotki, dla których kryterium grupowania ma identyczną wartość.
- na liście wynikowej klauzuli SELECT mogą wystąpić jedynie elementy związane z kryterium grupowania, funkcje agregujące (COUNT, SUM, AVG, MIN, MAX i inne)

Klauzula **HAVING** określa warunki dla podzbiorów/grup, które powstały w wyniku agregacji
- jest możliwa tylko z klauzulą GROUP BY

Klauzula **ORDER BY** umożliwia posortowanie danych wynikowych według zadanego kryterium
- kryterium może to być lista atrybutów z relacji źródłowych, wyrażenie, alias lub liczba wskazująca pozycje na liście wynikowej klauzuli SELECT
- możliwy jest porządek sortowania; rosnący, malejący, losowy
- opcja (klauzula) ORDER BY występuje w składni tylko raz

***Przykłady:***

```
SELECT Nr_albumu, Rok, Gr_dziekan FROM Studenci
WHERE rodzaj_studiow='MGR_ST_UZUP' AND Specjalnosc LIKE 'Z%'
ORDER BY 2, 3 ;
```

```
SELECT Rok, Semestr, Specjalnosc, Gr_dziekan, Count(*) FROM Studenci
WHERE rodzaj_studiow='MGR_ST_UZUP' GROUP BY Rok, Semestr,Specjalnosc,Gr_dziekan
ORDER BY Rok, Specjalnosc ;
```

## Operatory zbiorowe UNION, INTERSECT, EXCEPT/MINUS

używane są do przeprowadzenia operacji ( z teorii zbiorów ) sumy, przecięcia, różnicy na dwóch lub więcej zgodnych relacjach będących wynikami zapytań

***Przykłady:***

```
SELECT Imiona, Nazwisko FROM Studenci WHERE rok=1
UNION
SELECT Imie, Nazwisko FROM Aktorzy
ORDER BY 2;
```

```
SELECT Imiona FROM Studenci WHERE rok=1
INTERSECT
SELECT upper(Imie) FROM Aktorzy WHERE kraj='PL' ;
```

## Złączenia tabel w języku SQL

Notacja w standardzie ANSI SQL

```
FROM < tabela1> [ AS <alias> ] { CROSS JOIN |
 [ NATURAL ] [ <rodzaj_połączenia> ] JOIN < tabela2 > [ AS <alias > ]
{ ON <warunek1> [ { AND | OR } <warunek2> ] [ . . . ] ] | USING <atrybut1 > [ , . . . ] ) } }
```

Wykład I

Typy połączeń

**Iloczyn kartezjański** – daje w wyniku relację składającą się ze wszystkich możliwych kombinacji krotek obu łączonych relacji

***Przykład:***

```
SELECT  k.imiona,k.rok,m.imiona,m.rok  FROM  Studenci k CROSS  JOIN Studenci m
        WHERE     k.imiona IN ('AGNIESZKA','KATARZYNA')
                  AND  to_char(k.data_urodzenia,'YYYY')='1991'
                  AND  m.imiona IN ('PATRYK','ŁUKASZ','KAROL')
                  AND to_char(m.data_urodzenia,'YYYY')='1989' ;
```

**Złączenia  naturalne** – daje w wyniku iloczyn kartezjański łączonych relacji ograniczony do tych krotek, dla których atrybuty o tych samych nazwach i typach w obu relacjach miały równe wartości

***Przykład:***

```
SELECT nazwisko, imie, rola, tytul  FROM  Filmy  NATURAL JOIN  Obsady
        WHERE rezyser  LIKE 'Wajda%' ;
```

**Złączenie warunkowe** ( tetha złączenie ) daje w wyniku iloczyn kartezjański łączonych relacji ograniczony do tych krotek, dla których został spełniony warunek tetha

***Przykłady:***

```
SELECT Nazwa, Nazwisko FROM Pracownicy [ INNER ] JOIN Zespoly  USING ( Id_zesp);

SELECT Nazwisko, Placa_min, Placa, Placa_max FROM Pracownicy [ INNER ] JOIN Etaty
        ON  Nazwa=Etat;
```

*stary zapis*

```
SELECT  *  FROM Pracownicy p, Zespoly z
    WHERE  p.Id_zesp=z.Id_zesp;
```

**Złączenie  zewnętrzne** – rozszerza rezultat złączenia równościowego o te krotki jednej lub obu relacji, dla których nie odnaleziono odpowiedników w drugiej relacji. Krotki stanowiące rozszerzenie są wypełnione wartością NULL.
Złączenia zewnętrzne dzielą się na lewostronne, prawostronne, pełne

***Przykład:***

```
SELECT Nazwa,Count(Numer) Liczba, sum(placa+nvl(Dodatek,0)) Budzed  FROM
    Pracownicy [ OUTER ] FULL  JOIN  Zespoly USING(Id_zesp)
GROUP  BY Nazwa ;
```

**Samozłączenie –** złączenie relacji samej z sobą

***Przykład:***

```
SELECT  p.Nazwisko AS Pracownik, s.Nazwisko AS Kierownik  FROM
        Pracownicy p JOIN  Pracownicy s     ON (p.Szef = s.Numer );
```

**Podzapytania –** mogą być zagnieżdżone w klauzulach SELECT, FROM, WHERE oraz HAVING

***Przykłady:***

```
SELECT Nazwisko,Placa  FROM  Pracownicy
        WHERE  Placa = ( SELECT min( Placa )  FROM  Pracownicy );


SELECT * FROM
        ( SELECT Nazwisko, Placa  FROM  Pracownicy
                ORDER  BY 2 DESC )
WHERE rownum<=3 ;


SELECT  Imiona,count(*)  FROM  Studenci
        WHERE rodzaj_studiow='MGR_ST_UZUP'
                GROUP  BY Imiona  HAVING count(*) =
( SELECT  max(count(*))  FROM  Studenci  WHERE  rodzaj_studiow='MGR_ST_UZUP'
                        GROUP  BY Imiona) ;


SELECT Nr_albumu, Rok  FROM  Studenci s
        WHERE Data_urodzenia =
                ( SELECT max(Data_urodzenia) FROM Studenci WHERE Rok = s.Rok ) ;
```

*Wykorzystano*
*Wykłady  dr inż. Olga Siedlecka-Lamch - Systemy baz danych z roku 2012*