



Wykład 5

SQL

*Structured Query Language*



# 7. SQL

## ★ TEMATYKA:

- Podstawowe informacje o języku SQL
- Struktura języka SQL
- Wyszukiwanie informacji w tabelach
- Zakładanie tabel
- Wpisywanie, aktualizacja, usuwanie danych z tabel

# 7.1. Podstawowe informacje o SQL

- ★ **SQL stanowi najbardziej popularny mechanizm definiowania poleceń i modyfikacji w relacyjnych systemach baz danych (jest ratyfikowany jako standard języka relacyjnych baz danych).**
- ★ **Wszystkie DBMS powinny więc opierać się na tym standardzie.**
- ★ **SQL jest zaimplementowany w takich systemach baz danych (DBMS), jak: DB2, Oracle, InterBase, MySQL, dBase, Sybase, Informix, Paradox, MS SQL. Dzięki temu systemy informacyjne można przenosić na inne platformy.**
- ★ **Podstawowy rdzeń języka SQL jest implementacją algebry relacyjnej**
- ★ **Ujęty w standardzie ANSI/ISO w roku 1986**
  - aktualizacje: SQL-89, SQL 2, SQL 3
- ★ **Język SQL danej bazy, np. MySQL, zawiera:**
  - polecenia SQL ujęte w standardzie
  - rozszerzenia standardu – polecenia specyficzne dla konkretnego systemu baz danych

# Podstawowe informacje o SQL

- ★ **SQL - Structured Query Language, strukturalny język zapytań**
- ★ **SQL jest językiem czwartej generacji, który został w ciągu wielu lat opracowany przez grupę badawczą IBM w późnych latach siedemdziesiątych. Język ten został stworzony dla relacyjnego systemu zarządzania bazą danych o nazwie DB2 (produktu firmy IBM).**
- ★ **Stał się międzynarodowym standardem dla języków baz danych i występuje obecnie w produktach większości liczących się firm, zajmujących się sprzedażą oprogramowania dla baz danych.**
- ★ **SQL jest zaimplementowany w takich systemach baz danych (DBMS), jak: DB2, Oracle, InterBase, MySQL, dBase, Sybase, Informix, Paradox, MS SQL. Dzięki temu systemy informacyjne można przenosić na inne platformy.**
- ★ **Polecenia SQL mają postać zbliżoną do zdań w języku angielskim i są stosowane w celu uzyskania dostępu do danych i sterowania operacjami w bazie danych.**
- ★ **Podstawowy rdzeń języka SQL jest implementacją algebry relacyjnej**
- ★ **Ujęty w standardzie ANSI/ISO w roku 1986**
  - aktualizacje: SQL-89, SQL 2, SQL 3
- ★ **Język SQL danej bazy, np. MySQL, zawiera:**
  - polecenia SQL ujęte w standardzie
  - rozszerzenia standardu – polecenia specyficzne dla konkretnego systemu baz danych

# Typy składni języka SQL

## ★ Składnię języka SQL dzieli się na trzy typy:

- **DML (Data Manipulation Language)** - stosowany przez wszystkich użytkowników bazy danych. Służy do wybierania i manipulowania danymi znajdującymi się w bazie. Za jego pomocą, można dodawać, usuwać, wybierać czy uaktualniać dane.
  - Przykłady komend: SELECT- wydobywa dane z tabeli, UPDATE- uaktualnia dane w tabeli, DELETE - kasuje dane z tabeli, INSERT INTO - wprowadza dane do tabeli
- **DCL (Data Control Language)** - stosowany przez administratorów bazy danych. Służy on do zapewnienia bezpieczeństwa dostępu do danych znajdujących się w bazie. Za jego pomocą można przykładowo nadawać lub odbierać uprawnienia poszczególnym użytkownikom, czy całym grupom.
- **DDL (Data Definition Language)** - czyli język definiowania struktur danych jest wykorzystywany do utrzymywania struktury bazy danych. Dotyczy więc obiektów i poleceń jakie można na nich wykonywać.
  - Przykłady komend: CREATE TABLE - tworzy nową tabelę, ALTER TABLE - zmienia istniejącą tabelę, DROP TABLE - kasuje istniejącą tabelę, CREATE INDEX - tworzy indeks, DROP INDEX - usuwa indeks

## ★ Integralność danych włączona jest do części definicji danych

## ★ Język SQL nie zawiera zwykłych instrukcji sterowania IF ... THEN ..., DO ...WHILE i in.

# Rodzaje poleceń SQL

## ★ Polecenia SQL dotyczą:

- tworzenia i usuwania baz danych, tabel, kluczy
- wprowadzania, uaktualniania i usuwania danych
- wyszukiwania danych
- ustawiania praw dostępu do danych
- administracji bazą danych
- zarządzania transakcjami

## ★ Sposób wprowadzania do bazy poleceń SQL:

- w programie działającym z linii poleceń
- w programie z graficznym interfejsem użytkownika
- w skryptach i programach komunikujących się z bazą danych
- pośrednio, przy użyciu graficznego interfejsu użytkownika

# SQL - struktura języka, klauzule

## ★ Język definiowania struktur danych (DDL):

- **CREATE** - tworzenie tablic, perspektyw, indeksów;
- **DROP** - usuwanie obiektów;
- **ALTER** - modyfikacja struktury.

## ★ Język wyszukiwania i manipulowania danymi (DML):

- **SELECT** - wyszukiwanie danych w tabelach.
- **INSERT** - wstawianie danych do tabeli;
- **DELETE** - usuwanie danych;
- **UPDATE** - aktualizacja danych.

## ★ Język Nadzoru (DCL):

- **GRANT** - nadzorowanie uprawnień;
- **REVOKE** - odbieranie uprawnień;
- **BEGIN TRANSACTION ... END TRANSACTION** - programowanie transakcji;
- **ROLLBACK [WORK]** - przywrócenie tabeli do stanu przed transakcją.

## ★ Inne klauzule:

- **FROM** - nazwa tabeli;
- **WHERE** - warunek;
- **DISTINCT** - udostępnienie nie powtarzających się kolumn;
- **NOT, AND, OR** - spójniki wewnątrz warunku klauzuli.

# SQL - struktura języka, typy danych

## ★ Typy danych:

### ■ Typy napisowe

- **CHAR(N)** - napis znakowy o stałej długości, max długości 255 znaków  
**CHARACTER(N)**;
- **VARCHAR (N)** – napis znakowy o zmiennej długości, (do 255 znaków)  
**CHARACTER VARYING(N)**
- **BIT(N)** – ciąg bitów o stałej długości N
- **BIT VARYING(N)** – ciąg bitów o zmiennej długości, max N

### ■ Typy liczbowe

- **NUMERIC(M,N)**, - liczba stałoprzecinkowa o długości M z N miejscami po przecinku  
**DECIMAL(M,N)**
- **INTEGER** - liczba całkowita (długość zależy od implementacji)  
**INT**
- **SMALLINT** - liczba całkowita ze zmniejszoną liczbą cyfr (długość zależy od implementacji)
- **FLOAT(M,N)** - liczba rzeczywista zmiennoprzecinkowa  
**REAL**

### ■ Typy daty i godziny

- **DATE**
- **TIME**
- **TIMESTAMP** – czas z uwzględnieniem ułamków sekund
- **INTERVAL** – przedział pomiędzy datami

### ■ Typy logiczne

- **LOGICAL** - wartości logiczne: .T. .F. (.T. .N.).

## ★ Poszczególne implementacje języka różnią się w zakresie typów danych



# SQL - struktura języka

## ★ Funkcje agregujące:

- **COUNT(nazwa pola)** – zlicza ilość rekordów;
- **AVG(nazwa pola)** – wyznacza średnią arytmetyczną wartości w kolumnie;
- **SUM(nazwa pola)** – liczy sumę wartości w kolumnie;
- **MIN(nazwa pola)** – zwraca minimalną wartość w kolumnie;
- **MAX(nazwa pola)** – zwraca wartość maksymalną kolumny.

## ★ Obiekty:

- **DATABASE** - baza danych;
- **TABLE** - jedna tabela w bazie danych;
- **INDEX** - tablica indeksów do tabeli;
- **VIEW** - widok, perspektywa z nazwą (część tabeli);
- **SYNONYM** - alternatywna nazwa tabeli lub widoku (synonim).

# Zasady ogólne

- ★ Język SQL nie rozróżnia małych i wielkich liter w słowach kluczowych i nazwach (baz danych, tabel, indeksów i kolumn).
- ★ Język SQL rozróżnia litery w nazwach danych, trzeba pamiętać, że muszą być pisane dokładnie tak jak są umieszczone w tabeli.
- ★ Legalne są nazwy zbudowane ze znaków alfanumerycznych, nie zaczynające się od cyfry. Nie są dozwolone nazwy składające się wyłącznie z cyfr.
- ★ Nie należy w nazwach stosować znaków przestankowych i "@";
- ★ Każda komenda SQL kończy się średnikiem (;) i może składać się z wielu linii tekstu;
- ★ Wartości napisowe podaje się tak: "napis", lub tak: 'napis';
- ★ Wartości liczbowe zapisuje się w "zwykły" sposób, ewentualnie z kropką dziesiętną lub w notacji wykładniczej (np. -32032.6809e+10), gdy chodzi o wartości zmiennoprzecinkowe.

# Zasady ogólne (2)

## ★ **Polecenie języka SQL składa się z dwóch rodzajów słów:**

- Zarezerwowanych – które są integralną częścią języka i nie mogą być zmieniane ani dzielone pomiędzy wierszami.
- Zdefiniowanych przez użytkownika - reprezentują nazwy różnych obiektów bazy danych takich jak: indeksy, widoki, tabele, kolumny, relacje.

## ★ **Oznaczenia wykorzystane w składni poleceń**

- Słowa pisane dużymi literami - słowa zarezerwowane np. SELECT, FROM, REVOKE, GROUP BY itd.
- Słowa pisane małymi literami - słowa nie zarezerwowane, zdefiniowane przez użytkownika. Są to więc np. nazwy przestrzeni tabel, nazwy kolumn, tabel itp.

# Polecenie SELECT

- ★ Polecenie SELECT jest najpopularniejszym poleceniem języka SQL. Służy ono do wyszukiwania danych wg określonych w zapytaniu warunków.
- ★ Ogólna struktura polecenia SELECT jest następująca:

**SELECT** [nazwy kolumn, wyrażenia, funkcje]

**FROM** [nazwy tabel lub widoków]

**WHERE** [warunek wyboru wierszy]

**GROUP BY** [nazwy kolumn wg]

**HAVING** [warunek grupowania wybranych wierszy]

**ORDER BY** [nazwy lub pozycje kolumn]

# Składnia polecenia SELECT

★ Składnia polecenia (w notacji Bekusa):

```
SELECT [ALL | DISTINCT] { * | nazwa_kolumny [AS nowa_nazwa] [, ... ] }  
FROM nazwa_tabeli [alias] [, ... ]  
[WHERE warunek]  
[GROUP BY nazwa_kolumny]  
[HAVING warunek] [{ UNION [ALL] | INTERSECT | EXCEPT} SELECT ...]  
[ORDER BY nazwa_kolumny] [ASC | DESC]
```

[ xxx ]      – nie obowiązkowa obecność elementu w instrukcji  
{ xxx }      – obowiązkowa obecność elementu w instrukcji  
|            – należy wybrać jeden z elementów rozdzielonych „|”  
...          – możliwość powtórzenia konstrukcji.

ALL          – zbiór ostateczny zawiera wszystkie rekordy spełniające warunki zapytania (również powtarzające się)  
DISTINCT    – zbiór ostateczny zawiera wszystkie unikalne rekordy spełniające warunki zapytania (bez powtórzeń).  
\*            – oznacza tu, że wszystkie kolumny wszystkich tabel będą włączone do zbioru ostatecznego.  
ASC          – określa sortowanie w porządku rosnącym (takie jest domyślne)  
DESC        – oznacza sortowanie w porządku malejącym

# Realizacja instrukcji SELECT

★ Procedura wykonania instrukcji SELECT polega na realizacji klauzul FROM, WHERE, GROUP BY, HAVING i ORDER BY w następnej kolejności:

1. **FROM** – określić nazwę (nazwy) tablicy (tablic), która jest potrzebna(i) dla formowania zbioru ostatecznego.
2. **WHERE** – włączyć filtrację rekordów tabel. Uwzględniane będą tylko rekordy spełniające zapisany warunek.
3. **GROUP BY** – sformować grupy rekordów, mających identyczne wartości w kolumnach tabeli, nazwa której jest podana w wyrażeniu tej instrukcji.
4. **HAVING** – wykorzystać filtrację grupy rekordów. Warunek (warunki) tej filtracji jest wyznaczony w wyrażeniu instrukcji.
5. **ORDER BY** – sformować rozkaz wyników ostatecznego zbioru. Rozkaz jest zadany w wyrażeniu tej instrukcji.

# Wyszukiwanie – wybór kolumn (rzut)

- ★ Wyszukiwanie danych – wyświetlenie wybranych kolumn

```
SELECT rok, tytuł, wykonawca  
FROM albumy;
```

- ★ W ten sposób można uzyskać powtarzające się wyniki:

```
SELECT wykonawca  
FROM albumy;
```

- ★ Eliminacja powtórzeń wyników:

```
SELECT DISTINCT wykonawca  
FROM albumy;
```

albumy(tytuł, wykonawca, album, rok, gatunek)

tytuł	wykonawca	album	rok	gatunek

# Wyszukiwanie – wybór wierszy. Operatory

★ Wyszukiwanie rekordów spełniających zadany warunek – instrukcja WHERE

```
SELECT *  
FROM albumy  
WHERE wykonawca = 'Pink Floyd';
```

*Restrykcja*

## Operatory używane w instrukcji SELECT ... WHERE:

- porównania: = <> < > <= >= <=>
- logiczne: NOT ! AND && OR || XOR
- IS NULL, IS NOT NULL
- *expr* BETWEEN *min* AND *max* (NOT BETWEEN)
- *expr* IN (*lista*) (NOT IN)

```
SELECT wykonawca, rok  
FROM albumy  
WHERE tytuł = 'The Best Of' AND rok < 1970;
```

*Rzut + restrykcja*

```
SELECT *  
FROM albumy  
WHERE wykonawca IN ('Pink Floyd', 'Dire Straits')  
AND (rok < 1975 OR rok BETWEEN 1979 AND 1983);
```



# Sortowanie wyników

## ★ Sortowanie wyników wg zadanej kolumny:

- `ORDER BY pole` – w porządku rosnącym
- `ORDER BY pole ASC` – jw.
- `ORDER BY pole DESC` – w porządku malejącym

```
SELECT *  
FROM albumy  
ORDER BY rok DESC;
```

```
SELECT *  
FROM albumy  
WHERE rok = 1989  
ORDER BY wykonawca DESC;
```

# Sortowanie wyników c.d.

- ✳ Jeśli jest potrzeba sortowania wg dwóch (lub więcej kolumn), jest to możliwe poprzez umieszczenie większej liczby kolumn w klauzuli ORDER BY, jak poniżej:

```
SELECT *  
FROM albumy  
ORDER BY rok DESC, wykonawca;
```

- ✳ W klauzuli ORDER BY możemy się odwołać do kolumny poprzez wpisanie jej pozycji na liście SELECT.

```
SELECT wykonawca, album, rok  
FROM albumy  
ORDER BY 3, wykonawca;
```

- ✳ W zapytaniu SELECT można użyć tylko jednej klauzuli ORDER BY, którą umieszczamy zawsze na końcu zapytania.

# Grupowanie wyników

## ★ Tworzenie zestawień przez grupowanie wyników:

- użycie funkcji, np. **COUNT**, **SUM**, **MAX**, **MIN**, **AVG**
- nazwanie kolumny z wynikami (opcjonalnie) – **AS**
- zgrupowanie wyników – **GROUP BY**

Przykład – obliczenie ilości albumów wszystkich wykonawców:

```
SELECT wykonawca, COUNT(*)  
FROM albumy  
GROUP BY wykonawca;
```

```
SELECT wykonawca, COUNT(*) AS ilosc  
FROM albumy  
GROUP BY wykonawca  
ORDER BY ilosc DESC;
```

# Grupowanie wyników c.d.

- ★ Ograniczenie rekordów uzyskanych w wyniku grupowania - operator **HAVING**
- ★ Nie należy mylić instrukcji **WHERE** i **HAVING**!

Obliczenie ilości albumów wszystkich wykonawców.  
Wyświetlenie tylko tych, którzy mają więcej niż 5 albumów:

```
SELECT wykonawca, COUNT(*) AS ilosc  
FROM albumy  
GROUP BY wykonawca  
HAVING ilosc > 5;
```

# Ograniczenie liczby wyników

## ★ Ograniczenie liczby zwracanych wyników – **LIMIT**

- **LIMIT** *n* – *n* pierwszych wyników
- **LIMIT** *m, n* – *n* wyników, pomijając *m* pierwszych

10 wykonawców o największej liczbie albumów:

```
SELECT wykonawca, COUNT(*) AS ilosc  
FROM albumy  
GROUP BY wykonawca LIMIT 10;
```

20 następnych wyników (11-30):

```
SELECT wykonawca, COUNT(*) AS ilosc  
FROM albumy  
GROUP BY wykonawca LIMIT 10,20;
```

# Symbole wieloznaczne

★ Symbole wieloznaczne używane w instrukcji WHERE:

% - zastępuje dowolny ciąg znaków

\_ - zastępuje jeden znak

★ Operator symboli wieloznacznych: LIKE, NOT LIKE

```
SELECT *  
FROM albumy  
WHERE album NOT LIKE 'The Best in 197_';
```

# Wyszukiwanie przy użyciu polecenia SELECT

- ★ Aby wybrać wszystkie pola (kolumny) stosujemy „\*” a brak warunku WHERE pozwoli wyszukać wszystkie kolumny z tabeli:

```
SELECT *  
FROM Moduły;
```

- ★ Aby wybrać niektóre kolumny z tabeli, wpisujemy:

*Rzut*

```
SELECT NazwaModułu, Poziom  
FROM Moduły;
```

- ★ Aby wybrać z tabeli tylko wiersze spełniające zadany warunek wpisujemy:

*Restrykcja*

```
SELECT *  
FROM Moduły  
WHERE Poziom = „1”
```

- ★ Aby wybrać niektóre kolumny z tabeli i tylko rekordy spełniające określony warunek wpisujemy

*Rzut+Restrykcja*

```
SELECT NazwaModułu, Poziom  
FROM Moduły  
WHERE Poziom = „1”
```

# Wyszukiwanie w wielu tabelach

★ Pobieranie danych z więcej niż jednej tabeli

★ Przykład bazy danych – dwie tabele:

■ albumy

IDA	Wykonawca	Album	Rok	Gatunek

■ utwory

IDU	Utwór	Czas	IDA

★ Wybranie wszystkich możliwych kombinacji rekordów z obu tabel (iloczyn kartezjański):

```
SELECT *  
FROM albumy, utwory;
```



# Wyszukiwanie w wielu tabelach c.d.

- ★ Uwzględnienie relacji między tabelami:

```
SELECT *  
FROM albumy, utwory  
WHERE albumy.IDA = utwory.IDA;
```

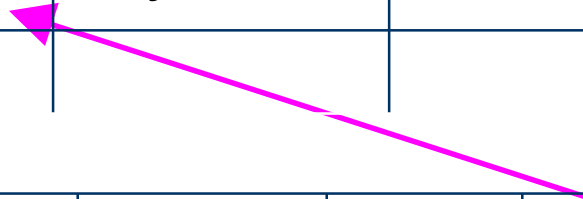
- ★ Łączy ze sobą rekordy obu tabel mające takie same dane w polach, które są połączone relacją:

■ albumy

IDA	wykonawca	album	rok	gatunek

■ utwory

IDU	utwór	czas	IDA



# Wyszukiwanie w wielu tabelach

## ★ Wybór kolumn:

```
SELECT albumy.wykonawca, albumy.album, utwory.utwor, utwory.czas  
FROM albumy, utwory  
WHERE albumy.IDA = utwory.IDA;
```

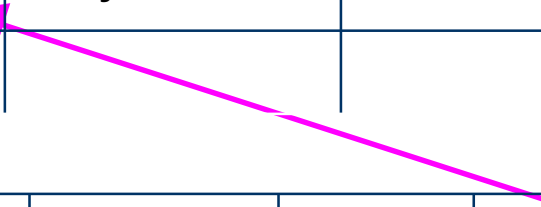
## ★ Krótsza wersja – aliasy nazw tabel:

```
SELECT a.wykonawca, a.album, u.utwor, u.czas  
FROM albumy a, utwory u  
WHERE a.IDA = u.IDA;
```

IDA	wykonawca	album	rok	gatunek

IDU	utwór	czas	IDA



# Tworzenie tabel

## ★ Utworzenie tabeli wymaga podania:

- nazwy tabeli,
- nazw pól (kolumn) w tabeli,
- typów danych dla każdej kolumny,
- maksymalny rozmiar każdej kolumny.

```
CREATE TABLE nazwa_tabeli
(nazwa_pola1 typ_pola1 [UNIQUE] [NOT NULL],
 nazwa_pola2 typ_pola2 [UNIQUE] [NOT NULL]
 ....
 nazwa_polaN typ_polaN [UNIQUE] [NOT NULL]);
```

```
CREATE TABLE albumy
(id INT,
 wykonawca VARCHAR(30),
 tytuł VARCHAR(30),
 rok YEAR,
 liczba-utworow SMALLINT,
 opis TEXT);
```

# Tworzenie tabel c.d.

- ★ Kolumny tabeli mogą być:
  - `NOT NULL` – nie mogą mieć wartości nul,
  - `UNIQUE` – jednoznaczne (nie powtarzające się),
- ★ Do definicji kolumny możemy dodać klauzulę określającą jej wartość domyślną:  
`nazwa_pola typ DEFAULT wartość`

```
CREATE TABLE albumy
(
    id                INT NOT NULL UNIQUE,
    wykonawca         VARCHAR(30) NOT NULL,
    tytuł             VARCHAR(30) NOT NULL,
    rok               YEAR,
    liczba-utworow    SMALLINT DEFAULT 10,
    opis              TEXT);
```

# Tworzenie tabel – ustalanie kluczy

## ★ Podczas definiowania tabeli można określić klucze

- główny – dopisując odpowiednią klauzulę przy właściwej kolumnie

`nazwa_pola typ PRIMARY KEY`

lub dodając klauzulę na końcu definicji tabeli

`PRIMARY KEY (nazwa_pola)`

- zewnętrzny

`FOREIGN KEY (nazwa_pola IDENTIFIES tabela_wskazywana)`

```
CREATE TABLE Moduły
  (NazwaModułu CHAR(15) PRIMARY KEY,
  Poziom SMALLINT,
  KodKursy CHAR(3),
  NrPrac NUMBER(5)
  FOREIGN KEY (NrPrac IDENTIFIES Wykładowcy);
```

```
CREATE TABLE Wykładowcy
  (NrPrac NUMBER(5),
  NazwiskoPrac VARCHAR(20),
  Status VARCHAR(10),
  PRIMARY KEY (NrPrac));
```

# Tworzenie tabel – zasady integralności

- ★ Podczas definicji klucza zewnętrznego (obcego) można określić warunki propagacji w przypadku usunięcia rekordu zawierającego klucz (lub jego modyfikacji):

**NO ACTION** – nie można usunąć klucza do którego odnosi się klucz obcy,

**CASCADE** – powoduje automatyczne usunięcie powiązanych wierszy w tabeli klucza obcego po usunięciu wiersza w tabeli z kluczem głównym,

**SET DEFAULT** – powoduje po usunięciu wiersza w tabeli z kluczem głównym, ustawienie wartości klucza obcego na wartość domyślną,

**SET NULL** – powoduje po usunięciu wiersza w tabeli z kluczem głównym, ustawienie wartości klucza obcego na wartość NULL.

```
CREATE TABLE Moduły
  (NazwaModułu    CHAR(15) PRIMARY KEY,
   Poziom         SMALLINT,
   KodKursu       CHAR(3),
   NrPrac         NUMBER(5)
  FOREIGN KEY (NrPrac IDENTIFIES Wykładowcy
  ON DELETE SET NULL
  ON UPDATE CASCADE) ;
```

```
CREATE TABLE Wykładowcy
  (NrPrac         NUMBER(5),
   NazwiskoPrac   VARCHAR(20),
   Status         VARCHAR(10),
  PRIMARY KEY (NrPrac)) ;
```

# Tworzenie tabel – dziedziny

- ★ Dziedziny możemy częściowo określić podając odpowiedni typ kolumn
- ★ Można ograniczyć zakres kolumn klauzulom **CHECK**

```
CREATE TABLE Moduły
(NazwaModułu CHAR(15) PRIMARY KEY,
Poziom SMALLINT,
KodKursy CHAR(3),
NrPrac NUMBER(5)
FOREIGN KEY (NrPrac IDENTIFIES Wykładowcy
ON DELETE SET NULL
ON UPDATE CASCADE
CHECK (Poziom IN 1,2,3)
CHECK (NrPrac BETWEEN 100 AND 10999);
```

- ★ Można zdefiniować własną dziedzinę

```
CREATE DOMAIN NazwaDziedziny AS Typ
[DEFAULT wartość]
[CHECK Zakres]
```

```
CREATE DOMAIN Poziomy AS INTEGER(1)
DEFAULT 1
CHECK (VALUE BETWEEN 1 AND 3);
```