Systemy Baz Danych

Wykład III

Język SQL – polecenia DQL – c.d.,
polecenia DDL i DML

Powtórzenie wiadomości – cz. 2

Materiał wykładu

Wykład zawiera kontynuację przeglądu podstawowych wiadomości o języku Structured Query Language (SQL). Przedstawione są polecenia należące do Data Definition Language (DDL) i Data Manipulation Language (DML). Stanowi powtórzenie materiału wykładanego w ramach przedmiotu Relacyjne Bazy Danych dla studentów studiów inżynierskich Wydziału Informatyki PJWSTK.

Wykład jest przeznaczony dla studentów przedmiotu **Systemy Baz Danych** prowadzonego dla studiów inżynierskich Wydziału Informatyki PJWSTK.

DDL

Data Definition Language – operacje na obiektach bazy danych

Utworzenie nowej tabeli

```
CREATE TABLE nazwa_tabeli(
Nazwa_kolumny typ_danych [więzy_spójności],
...);
```

Więzy spójności danych (ang. Constraints) to zespół reguł, które gwarantują logiczną spójność danych wprowadzanych i przechowywanych w bazie.

System bazy danych musi zagwarantować, że więzy spójności pozostaną prawdziwe przy wszystkich operacjach wykonywanych na bazie danych w postaci transakcji, akcji wyzwalaczy, ładowania danych do bazy danych czy ich importu.

Realizację zdefiniowanych więzów spójności zapewnia system bazy danych (SZBD).

Więzy spójności encji

- Więzy klucza głównego **PRIMARY KEY** – wartości w określonych kolumnach jednoznacznie identyfikują wiersz – każda wartość musi być unikalna i nie jest dopuszczalna pseudowartość **NULL**. Na kolumnie (kolumnach) PK automatycznie jest zakładany indeks. Dla jednej tabeli może istnieć tylko jeden klucz główny.
- Więzy klucza jednoznacznego **UNIQUE** – wartości w kolumnach jednoznacznie identyfikują wiersz, ale jest dopuszczalna pseudowartość **NULL**. Na kolumnach tworzących klucz jednoznaczny jest automatycznie zakładany indeks . W jednej tabeli można utworzyć więcej niż jeden klucz jednoznaczny.
- Więzy **NOT NULL** – w kolumnie nie jest dozwolona wartość **NULL**.
- Więzy **CHECK** – definiują warunek, który nie może być fałszywy dla każdego wiersza w tabeli, ale może przyjmować wartość logiczną NULL.

Więzy spójności referencyjnej

Więzy spójności referencyjnej pozwalają zdefiniować zależności pomiędzy wartościami zapisanami w kolumnach wiążących tabele w układzie klucz_główny – klucz_obcy:

- zbiór wartości w kolumnach klucza obcego jest zawsze podzbiorem zbioru wartości odpowiadającego mu klucza głównego,
- wartością klucza obcego może być **NULL** – wówczas klucz obcy nie wskazuje na żaden wiersz tabeli nadrzędnej,
- system gwarantuje istnienie wiersza wskazywanego przez wartość klucza obcego, przy wszystkich możliwych operacjach na tabelach, w których biorą udział klucze główne i obce.

Definiowanie więzów spójności

- **NOT NULL** – pseudo-wartość *Null* nie jest dozwolona w danej kolumnie,
- **PRIMARY KEY** – kolumna stanowi klucz główny,
- **FOREIGN KEY REFERENCES** *nazwa_tabeli* – kolumna stanowi klucz obcy odwołujący się do klucza głównego podanej tabeli,
- **UNIQUE** – kolumna stanowi klucz jednoznaczny (wartości w kolumnie nie powtarzają się),
- **CHECK** (*warunek logiczny*) – warunek jaki ma być spełniony dla wartości w wierszu,
- **DEFAULT** *wartość domyślna* – wartość domyślna dla kolumny,
- opcjonalne słowo kluczowe **CONSTRAINT** wprowadza explicite nazwę dla więzów spójności.

Jeżeli nazwy więzów nie są zadeklarowane explicite, nada je automatycznie SZBD.

Przykład utworzenia tabel Dept i Salgrade

```
CREATE TABLE   Dept(
               Deptno    INT              PRIMARY KEY,
               Dname     VARCHAR(14)      NOT NULL,
               Loc       VARCHAR(13)
);

CREATE TABLE   Salgrade(
               Grade     INT              PRIMARY KEY,
               Losal     NUMERIC(7,2)     NOT NULL,
               Hisal     NUMERIC(7,2)     NOT NULL,
CONSTRAINT     chk_salgrade_1 CHECK (Losal < Hisal)
);
```

Przykład utworzenia tabeli Emp

```
CREATE TABLE   Emp(
               Empno       INT              PRIMARY KEY,
               Ename       VARCHAR(10)      NOT NULL,
               Job         VARCHAR(9),
               Mgr         INT              REFERENCES Emp,
               Hiredate    DATETIME,
               Sal         NUMERIC(7,2),
               Comm        NUMERIC(7,2),
               Deptno      INT NOT NULL     REFERENCES Dept,
CONSTRAINT     chk_Emp_1   CHECK (Comm < Sal)
);
```

Składnia więzów „w linii” i „poza linią”

W linii :

```
CREATE TABLE   Osoba(
               IdOsoby     INT            PRIMARY KEY,
               Imie        VARCHAR(20),
               Nazwisko    VARCHAR(50)
);
```

Poza linią:

```
CREATE TABLE   Osoba(
               IdOsoby     INT,
               Imie        VARCHAR(20),
               Nazwisko    VARCHAR(50)
PRIMARY KEY    (IdOsoby),
NOT NULL       (Nazwisko)
);
```

Składnia więzów „w linii” i „poza linią”

Poza linią, z określeniem nazwy więzów spójności:

```
CREATE TABLE  Osoba (
              IdOsoby       INT,
              Imie          VARCHAR(20),
              Nazwisko      VARCHAR(50),
CONSTRAINT    PK_Osoba      PRIMARY KEY (IdOsoba),
CONSTRAINT    UQ_Nazwisko   UNIQUE(Nazwisko)
);
```

Wszystkie powyższe przykłady zostały zaprezentowane w dialekcie **MS SQL Server**. Ich składnia w **ORACLE** nie różni się niczym, poza nazwami typów danych.

Generowanie jednoznacznych numerów MS SQL Server

W **MS SQL Server** używamy opcji **IDENTITY** dla kolumny z typem danych **INT**

Przykład:

```
CREATE TABLE    Osoba(
                Id_osoby    INT IDENTITY    PRIMARY KEY,
                Imie        VARCHAR(10),
                Nazwisko    VARCHAR(30)
                );
```

Wprowadzając nowy rekord do tabeli nie podajemy wówczas wartości kolumny klucza głównego, jest ona generowana automatycznie. Odczytać ostatnio wprowadzoną wartość dla danej sesji można odwołując się do zmiennej systemowej T-SQL **@@IDENTITY** albo lepiej do funkcji **Scope_Identity()**

Właściwość **IDENTITY** może zostać przypisana do co najwyżej jednej kolumny w tabeli.

Generowanie jednoznacznych numerów MS SQL Server

Innym rozwiązaniem jest użycie specjalnego obiektu klasy **Sequence** (sekwencja). To rozwiązanie dostępne jest w **MS SQL Server** od wersji 2012. Obiekty **Sequence** nie są powiązane z tabelami. Wygenerowanie nowej, unikalnej wartości, nie musi być związane z dopisywaniem nowego rekordu do tabeli. Wygenerowana wartość nie musi zostać użyta w tabeli.

Utworzenie sekwencji:

```
CREATE SEQUENCE Nazwa_sekwencji
AS              Typ_danych
[START WITH X]
[INCREMENT BY Y];
```

Gdzie:

Typ danych - typ liczb całkowitych
X – pierwsza wygenerowana wartość; 1 jeśli pominięte
Y – wartość inkrementacji (może być ujemna); 1 jeśli pominięte

Generowanie jednoznacznych numerów MS SQL Server

Usunięcie sekwencji:

```
DROP SEQUENCE Nazwa_sekwencji;
```

Restart sekwencji:

```
ALTER SEQUENCE Nazwa_sekwencji
RESTART WITH Z;
```

Przykład użycia sekwencji do wygenerowania wartości klucza głównego:

```
CREATE SEQUENCE Test
AS Int
START WITH 50
INCREMENT BY 10;

INSERT INTO Dzial (IdDzial, Nazwa, Lokalizacja)
VALUES       (NEXT VALUE FOR Test, 'LOGISTIC', 'DETROIT');
```

Generowanie jednoznacznych numerów ORACLE

Utworzenie sekwencji:

```
CREATE SEQUENCE Nazwa_sekwencji
[START WITH X]
[INCREMENT BY Y];
```

Gdzie:

X – pierwsza wygenerowana wartość; 1 jeśli pominięte
Y – wartość inkrementacji (może być ujemna); 1 jeśli pominięte

Usunięcie sekwencji:

```
DROP SEQUENCE Nazwa_sekwencji;
```

Zmiana wartości inkrementacji sekwencji:

```
ALTER SEQUENCE Nazwa_sekwencji
INCREMENT BY Z;
```

W ORACLE nie można zmieniać wartości startowej sekwencji.

Generowanie jednoznacznych numerów ORACLE

Przykład użycia sekwencji:

```
CREATE SEQUENCE Test
AS Int
START WITH 50
INCREMENT BY 10;

INSERT INTO Dzial (IdDzial, Nazwa, Lokalizacja)
VALUES       (Test.Nextval, 'LOGISTIC', 'DETROIT');

Dbms_output.Put_line ('Dopisano nowy departament
                       z deptno = ' || Test.Currval);
```

Jak widać, w **ORACLE** wygenerowanie nowej wartości sekwencji następuje przez odwołanie do jej właściwości **Nextval**

…a odczytanie ostatnio wygenerowanej wartości, poprzez odwołanie do właściwości **Currval**.

Wartość wygenerowana poza tabelą musi zostać podstawiona na zmienną Pl/SQL.

Zmiana schematu tabeli - kolumny

Odpowiednio dodanie nowej kolumny do istniejącej tabeli, zmiana typu danych kolumny i usunięcie kolumny.

MS SQL Server i Standard

```
ALTER TABLE Nazwa_tabeli ADD nazwa_kolumny Typ_danych;
ALTER TABLE Nazwa_tabeli ALTER COLUMN nazwa_kolumny Typ_danych;
ALTER TABLE Nazwa_tabeli DROP COLUMN nazwa_kolumny;
```

ORACLE:

```
ALTER TABLE Nazwa_tabeli ADD (nazwa_kolumny Typ_danych);
ALTER TABLE Nazwa_tabeli MODIFY (nazwa_kolumny Typ_danych);
ALTER TABLE Nazwa_tabeli DROP COLUMN nazwa_kolumny;
```

Nawiasy w składni **ORACLE** są opcjonalne.

Zmiana schematu tabeli – dodanie więzów CHECK

MS SQL Server i Standard :

```
ALTER TABLE      Emp
ADD CONSTRAINT   Emp_Sal_comm CHECK (Sal + Comm <=10000);
```

ORACLE:

```
ALTER TABLE      Emp
ADD (CONSTRAINT  Emp_Sal_comm CHECK (Sal + Comm <=10000));
```

Warunek zdefiniowany jako więzy **CHECK** sprawdzany jest podczas operacji wstawiania nowego wiersza oraz aktualizacji danych w kolumnie. Akceptowane są dane, dla których wartością logiczną warunku jest **TRUE** lub **NULL (!!!)**. SZBD nie dopuszcza do pojawienia się danych, dla których warunek przyjmie wartość **FALSE**. W takim przypadku podnoszony jest wyjątek (błąd).

Zmiana schematu tabeli – więzy NOT NULL

Więzy **NOT NULL** traktowane są jako właściwości kolumny, zatem ich deklaracja jest możliwa poprzez zmianę schematu kolumny:

Przykład deklaracji więzów **NOT NULL**:

MS SQL Server

```
ALTER TABLE  Kontener
ALTER COLUMN Pojemnosc Decimal(4,2) NOT NULL;
```

ORACLE:

```
ALTER TABLE  Kontener
MODIFY       (Pojemnosc Numeric(4,2) NOT NULL);
```

Zmiana schematu tabeli – więzy DEFAULT

Więzy **NOT NULL** oraz **DEFAULT** traktowane są w ORACLE jako właściwości kolumny, w MS SQL Server jako więzy, których deklaracja jest możliwa poprzez zmianę schematu kolumny:

MS SQL Server

```
ALTER TABLE     Kontener
ADD CONSTRAINT Def_Pojemnosc
DEFAULT         250
FOR             Pojemnosc;
```

ORACLE:

```
ALTER TABLE  Kontener
MODIFY       (Pojemnosc Numeric (6,2) DEFAULT  250);
```

W specyfikacji wartości domyślnej mogą występować stałe, funkcje SQL, **Sysdate** (**ORACLE**) i **Getdate** (**MS SQL Server**). Nie mogą występować nazwy kolumn i funkcje Pl/SQL i T-SQL. W **ORACLE** nie mogą występować sekwencje.

Zmiana schematu tabeli – więzy referencyjne

Przykład deklaracji cyklicznych odwołań pomiędzy tabelami Miasto i Panstwo

MS SQL Server

```
ALTER TABLE       Miasto
ADD CONSTRAINT    FK_Panstwo_miasto
FOREIGN KEY       (IdPanstwo) REFERENCES Panstwo;

ALTER TABLE       Panstwo
ADD CONSTRAINT    FK_Miasto_panstwo
FOREIGN KEY       (IdMiasto) REFERENCES Miasto;
```

ORACLE:

```
ALTER TABLE    Miasto
ADD            (Panstwo_Id Int
               ,CONSTRAINT FK_panstwo_miasto
               FOREIGN KEY (Panstwo_Id) References Panstwo);
```

… i analogicznie dla tabeli Panstwo.

Akcje referencyjne przy DELETE i UPDATE

ON DELETE NO ACTION – przy próbie usunięcia wiersza, do którego są odwołania przez wartości kluczy obcych podnoszony jest błąd, a polecenie **DELETE** jest wycofywane. Jest to domyślne ustawienie w ORACLE i MS SQL Server i jego deklaracja może być pominięta,

ON DELETE SET DEFAULT - przy usuwaniu wiersza, do kolumn klucza obcego wierszy odwołujących się do wiersza usuwanego następuje wstawienie wartości domyślnej tych kolumn (oczywiście, jeżeli są zdefiniowane; jeżeli brak jest definicji wartości domyślnej, a kolumna dopuszcza NULL, zostanie użyta ta pseudowartość),

ON DELETE SET NULL – jak w poprzednim przypadku wartość domyślna, tak tutaj wstawiana jest pseudowartość NULL,

ON DELETE CASCADE – wraz z usuwanym wierszem tabeli nadrzędnej usuwane są odwołujące się do niego wiersze z tabel podrzędnych.

W **ORACLE** nie jest implementowane ustawienie **ON DELETE SET DEFAULT**

Włączanie i wyłączanie więzów spójności

ORACLE

W **ORACLE** możliwe jest włączanie i wyłączanie działania więzów integralności w instrukcji ALTER TABLE:

Wyłączanie

```
ALTER TABLE          nazwa_tabeli
DISABLE CONSTRAINT   nazwa_więzów;
```

Ponowne włączanie

```
ALTER TABLE          nazwa_tabeli
ENABLE  CONSTRAINT   nazwa_więzów;
```

Włączanie i wyłączanie więzów spójności

MS SQL Server

W **MS SQL Server** operacje włączania / wyłączania dotyczy wyłącznie więzów referencyjnych **FOREIGN KEY** oraz więzów **CHECK**.

Wyłączanie:

```
ALTER TABLE          nazwa_tabeli
NOCHECK CONSTRAINT   nazwa_więzów;
```

Włączanie:

```
ALTER TABLE          nazwa_tabeli
CHECK CONSTRAINT     nazwa_więzów;
```

Wyłączanie wszystkich więzów w tabeli:

```
ALTER TABLE          nazwa_tabeli
NOCHECK CONSTRAINT   ALL;
```

UWAGA: Wyłączenie więzów referencyjnych nie oznacza możliwości usunięcia z tabeli nadrzędnej rekordów, do których istnieją odwołania z tabel podrzędnych.

Usuwanie tabeli (i innych obiektów)

```
DROP TABLE nazwa_tabeli;
```

Gdy inne tabele mają klucze obce odwołujące się do usuwanej tabeli i nie zostały zadeklarowane inne akcje referencyjne niż **ON DELETE NO ACTION** operacja nie powiedzie się.

Analogicznie usuwane są obiekty innych klas:

```
DROP typ_obiektu nazwa_obiektu;
```

Np.

```
DROP VIEW Alamakota;
```

DML, perspektywy

Data Manipulation Language – operacje na danych

Wstawianie nowych rekordów do tabeli

Składnia podstawowa (pełna):

```
INSERT INTO nazwa_tabeli     (lista nazw kolumn)
VALUES                       (lista wartości);
```

Składnia uproszczona (sugeruję nie używać):

```
INSERT INTO nazwa_tabeli
VALUES          (lista wartości);
```

Listy są rozdzielane przecinkami

Źródłem danych dla instrukcji **INSERT** może być instrukcja **SELECT** odczytująca wartości z innych tabel. Instrukcja **SELECT** może także zawierać wyrażenia.

```
INSERT INTO   nazwa_tabeli_1 (lista_nazw_kolumn)
SELECT        lista_wyrażeń
FROM          nazwa_tabeli_2;
```

Wstawianie nowych rekordów do tabeli

Istnieje możliwość wstawienia danych odczytanych instrukcją **SELECT** do tworzonej „w locie” tabeli, o strukturze zgodnej ze strukturą danych zwracanych przez **SELECT**.

ORACLE

```
CREATE TABLE nazwa_nowej_tabeli AS (SELECT … FROM…);
```

MS SQL Server

```
SELECT w1, w2, … INTO nazwa_nowej_tabeli
FROM   źródła_rekordów;
```

UWAGA: W **ORACLE** zbliżoną składnią operuje instrukcja podstawienia odczytanych danych na zmienne! Nie należy tego mylić!

Modyfikacja rekordów istniejących w tabeli

```
UPDATE   nazwa_tabeli
SET      nazwa_kolumny = wyrażenie1, ...
[WHERE   warunek];
```

Pominięcie warunku **WHERE** spowoduje modyfikację danych we wszystkich wierszach tabeli (!!!).

Usuwanie rekordów z tabeli

```
DELETE FROM  nazwa_tabeli
[WHERE       warunek];
```

Pominięcie warunku **WHERE** spowoduje usunięcie danych ze wszystkich wierszy tabeli.

Pewną alternatywę dla instrukcji **DELETE** stanowi instrukcja **TRUNCATE**:

```
TRUNCATE TABLE nazwa_tabeli;
```

Utworzenie perspektywy

```
CREATE VIEW Nazwa_perspektywy [(nazwa_kolumny, ...)]
AS          instrukcja_select;
```

Problem sortowania wierszy w definicji perspektywy w **MS SQL Server**:

Jeżeli w definicji perspektywy ma zostać użyta klauzula **ORDER BY**, w instrukcji **SELECT** musi znaleźć się klauzula **TOP**.

Np.

```
CREATE VIEW     Urzednicy (Numer, Nazwisko, Placa)
AS
SELECT          TOP 99.99 PERCENT Empno, Ename, Sal
FROM            Emp
WHERE           Job = 'CLERK‘
ORDER BY        Empno;
```

Dla wartości 100 **PERCENT** widok zwróci wiersze nieposortowane!

Problem ten nie występuje w **ORACLE**.

Perspektywa z opcją sprawdzania

Poprzez perspektywy mogą być wykonywane operacje DML, jeśli perspektywa odwołuje się do jednej tabeli, na liście **SELECT** są tylko nazwy kolumn, nie zawiera **DISTINCT**, podzapytań i klauzul **GROUP BY** i **HAVING**.

CREATE VIEW *nazwa_perspektywy [(nazwa_kolumny,...)]*
AS *instrukcja_SELECT*
WITH CHECK OPTION;

Opcja **WITH CHECK OPTION** powoduje, że przy wykonywaniu instrukcji **INSERT** i **UPDATE** następuje sprawdzenie, czy wstawiany bądź modyfikowany wiersz spełnia warunek określony w klauzuli **WHERE**:

- jeśli spełnia, operacja jest wykonywana,
- jeśli nie spełnia, operacja nie zostanie wykonana.