Wykład II

polecenie **UPDATE**
służy do aktualizacji zawartości wierszy tabel lub perspektyw

składnia:

UPDATE { <*nazwa_tabeli*> | <*nazwa_perspektywy*>}
SET { {<*nazwa_atrybutu1*> = <*wyrażenie*> | **DEFAULT** | **NULL**},
{<*nazwa_atrybutu2*> = <*wyrażenie*> | **DEFAULT** | **NULL**}, . . .}
[**WHERE** <*warunek_logiczny*>];

Przykłady:

UPDATE Studenci **SET** Rok = Rok + 1 **WHERE** Rodzaj_studiow =' INŻ_ST';

UPDATE Studenci **SET** Rok = Rok – 1 **WHERE** Nr_albumu **IN** ('111345','100678');

UPDATE Pracownicy **SET** Placa = Placa * 1.2
WHERE Numer **IN** (**SELECT** Szef **FROM** Pracownicy);

polecenie **INSERT**
pozwala wstawiać do tabeli lub perspektywy nowe wiersze.

składnia:

INSERT INTO { <*nazwa_tabeli*> | <*nazwa_perspektywy*>} [(<*nazwa_atrybutu1*> [, . . .])]
{ **DEFAULT VALUES** | **VALUES** (<*wartość1* [, . . .]) | **SELECT . . .** } ;

Przykłady:

INSERT INTO Zespoly **VALUES** (85,' Inteligentne Systemy','Armii Krajowej 36');

INSERT INTO Pracownicy (Numer, Nazwisko) **VALUES** (700,'Nowak');

INSERT INTO Absolwenci (Nazwisko, Imiona, Rocznik)
SELECT Nazwisko, Imiona, to_char(sysdate,'YYYY') **FROM** Studenci
WHERE Rodzaj_studiow='INŻ_ST' **AND** Semestr='VII' ;

Polecenie **DELETE**
Usuwa wiersze z danej tabeli lub perspektywy

Składnia:
DELETE FROM { <*nazwa_tabeli*> | <*nazwa_perspektywy*> }
[**WHERE** <*warunek_logiczny*>];

Przykłady:
DELETE Studenci **WHERE** Rok = 5;

DELETE FROM Etaty ;

Wykład II

DDL (*ang. Data Definition Language*) zawiera niezbędne polecenia do tworzenia **CREATE**, usuwania **DROP** i modyfikacji struktur danych **ALTER**

Tabela (*ang. table*) jest podstawową jednostką służącą do przechowywania danych w bazach W Oracle spotykamy różne typy tabel:

- **relacyjna**
- **tymczasowa** – tworzy się za pomocą polecenia CREATE GLOBAL TEMPORARY TABLE . . . , w tabelach tymczasowych przechowywane są dwa rodzaje danych tymczasowych; dane tymczasowe istniejące przez czas trwania transakcji i dane tymczasowe istniejące przez czas trwania sesji. Czas istnienia danych tymczasowych kontroluje klauzula ON COMMIT. Klauzula ON COMMIT DELETE ROWS usuwa wszystkie wiersze z tabeli tymczasowej w momencie, gdy wykonywane jest polecenie COMMIT lub ROLLBACK. Natomiast klauzula ON COMMIT PRESERVE ROWS zachowuje wszystkie wiersze w tabeli, nawet po zakończeniu transakcji. Po zakończeniu sesji wszystkie wiersze wstawione do tabeli tymczasowej zostaną usunięte
- **indeksowa** – przechowuje wiersze tabeli w indeksie drzewa binarnego, gdzie każdy z węzłów drzewa binarnego zawiera kolumnę będącą kluczem (indeksowaną) oraz co najmniej jedną kolumnę nieindeksowaną
- **obiektowa** – posiadają wiersze, do których można odwoływać się za pośrednictwem identyfikatorów obiektów (OID)
- **zewnętrzna** – umożliwiają użytkownikowi sięganie do zewnętrznego źródła danych, na przykład pliku tekstowego , tak jakby był on przechowywany w bazie danych. Metadane tabeli przechowywane są w słowniku systemu, niemniej sama zawartość tabeli przechowywana jest na zewnątrz
- **klastrowana** – pozwala zredukować przestrzeń potrzebną do przechowywania kolumn, które powtarzają się w łączonych tabelach. Takie wspólne kolumny nazywane są wartościami klucza klastra
- **partycjonowana** – staje się łatwiejsza do zarządzania; firma Oracle zaleca dzielić na partycje wszelkie tabele , których rozmiar przekracza 2GB. Stosuje się partycje zakresowe, listowe, haszowane.

CREATE [GLOBAL TEMPORARY] TABLE [<*schemat*>.]<*nazwa_tabeli*>
(<*wlasności_relacyjne* >**) [ON COMMIT { DELETE | PRESERVE } ROWS]**
[< *własności_fizyczne* >] ;

Definicja kolumn

CREATE TABLE < *nazwa_tabeli* >
(<*nazwa_kolumny1*> <**typ>** [(*rozmiar*)] [**SORT**]
[**DEFAULT** < *wyrażenie* >]
[**ENCRYPT** < *spec_szyfrowania* >]
[< *więzy_kolumny1*>] [, <*nazwa_kolumny2* > . . .]
) ;

Własności fizyczne

CREATE TABLE < nazwa_tabeli >
(<*własności_relacyjne* >)
[{ < *atrybuty_segmentu* > [< *kompresja* >]
| **ORGANIZATION** { < t*yp_organizacji_segmentu* > }
| **CLUSTER** < *nazwa* > (< *nazwa_kolumny1* [, < *kolumna2*] . . .) }] ;

Przykłady:

```
CREATE  CLUSTER Zamowienie_klaster
( Zamowienie_numer NUMBER(6) )
SIZE 50
HASH  IS  Zamowienie_numer  HASHKEYS 100000;

CREATE  TABLE Klient_zamowienie
( Zamowienie_numer  NUMBER(6)  PRIMARY  KEY,
Zamowienie_data  DATE,
Klient_numer  NUMBER )
CLUSTER Zamowienie_klaster( Zamowienie_numer) ;

CREATE  TABLE Demo
( Id NUMBER( 10 )  )
TABLESPACE przykład
STORAGE ( INITIAL 6144 );

CREATE  TABLE  Dzialy
( Id  NUMBER( 2 ) PRIMARY  KEY ,
  Nazwa  VARCHAR2( 50 ) ,
  Lokalizacja   VARCHAR2( 20 )
)
ORGANIZATION  INDEX
PCTHRESHOLD  30
OVERFLOW  TABLESPACE Nadmiar ;


CREATE  TABLE  Klienci
( Id  NUMBER( 6 ) ,
  Nazwisko  VARCHAR2( 50 ) ,
  Imiona  VARCHAR2( 30 ) ,
  Kraj  VARCHAR2( 30 ) ,
  Email  VARCHAR2( 25 )
)
PARTITION  BY LIST ( Kraj )
(
        PARTITION  Azja  VALUES  ( 'CHINY' , ' TAJLANDIA') ,
        PARTITION  Europa  VALUES  ( 'NIEMCY' , 'WŁOCHY' ) ,
        PARTITION  Ameryka VALUES  ( 'USA' , 'KANADA' ),
        PARTITION  Reszta  VALUES  ( DEFAULT )
);

CREATE  TABLE  Pracownicy_archiwum  AS
        SELECT  *  FROM  Pracownicy  WHERE  Id_zesp = 20 ;
```

Wykład II

Definiowanie ograniczeń (więzów spójności)
sposoby :

- dla kolumny – definiowane „inline” lub „out_of_line” (za wyjątkiem NOT NULL)
- dla tabeli – definiowane „out_of_line”
- tworzone w poleceniu modyfikacji struktury tabeli – wyłącznie „out_of_line”
- każde ograniczenie jest **nazwane** przez użytkownika lub przez system

Definicja ograniczeń „inline”

CREATE TABLE < *nazwa_tabeli* >
(< *nazwa_kolumny1* > < **typ>** [(*rozmiar*)]
[**DEFAULT** < *wyrażenie* >] [**SORT**]
[[**CONSTRAINT** < *nazwa_ogr_kolumny* >] {
[NOT] NULL
| UNIQUE
| PRIMARY KEY
| REFERENCES < *nazwa_tabeli* > (< *nazwa_kolumny1*>[,< *nazwa_kolumny2* >] . . .)
[ON DELETE { CASCADE | SET NULL }]
| CHECK (<*warunek_logiczny* >) } **[DISABLE | ENABLE]], . .) ;**

Przykłady:

CREATE TABLE Etaty
(Nazwa **VARCHAR2**(15) constraint pk_etaty **PRIMARY KEY,**
Placa_min **NUMBER**(7,1) constraint war1 **CHECK** (Placa_min > 1500),
Placa_max **NUMBER**(7,1) **NOT NULL**,
constraint war **CHECK** (Placa_max >= Placa_min)) ;

CREATE TABLE Kursy
(Kod **NUMBER**(3) constraint p_kod **PRIMARY KEY**,
Nazwa **VARCHAR2**(30) **NOT NULL**,
Koszt **NUMBER**(7,1) constraint c_koszt **CHECK** (Koszt > 0));

CREATE TABLE Pracownicy_kursy
(Numer **NUMBER**(4) constraint fk_numer **REFERENCES** Pracownicy,
Kod **NUMBER**(3) constraint fk_kod **REFERENCES** Kursy,
constraint pk_pk **PRIMARY KEY** (Numer, Kod));

DROP TABLE [< *schemat*>.]<*nazwa_tabeli* >
[**CASCADE CONSTRAINTS**] [**PURGE**] ;

Polecenie powyższe

- usuwa definicję tabeli i dane z tabeli oraz wyzwalacze, indeksy związane z tabelą
- zwalnia fizyczny segment i jego rozszerzenia
- ustawia atrybut niepoprawności dla powiązanych perspektyw, synonimów, procedur itp.

ALTER TABLE . . .

Dodanie kolumn ; **ALTER TABLE** < *nazwa_tabeli* > **ADD**
(< *nazwa_kol1* >< **typ>** [**DEFAULT** < *wyrażenie* >] [< *więzy_kol1*>]
[, < *nazwa_kol2* > <**typ>** [**DEFAULT** < *wyrażenie* >] [< *więzy_kol2*>] . . .) ;

Modyfikacja kolumn; **ALTER TABLE** < nazwa_tabeli > **MODIFY**
(< *nazwa_kol1* > <**typ>** [**DEFAULT** < *wyrażenie* >] [< *więzy_kol1*>]
[, < *nazwa_kol2* > <**typ**>[**DEFAULT** < *wyrażenie* >] [< *więzy_kol2*>] . . .) ;

Usunięcie kolumn; **ALTER TABLE** < *nazwa_tabeli* > **DROP**
COLUMN < *nazwa_kol1* **>** | (**<** *nazwa_kol1* **>** [, **<** *nazwa_kol2* >] . . .) ;

Zmiana nazwy kolumny; **ALTER TABLE** < *nazwa_tabeli* > **RENAME**
COLUMN < *stara_nazwa_kol* **> TO** < *nowa_nazwa_kol* > ;

Modyfikacja ograniczeń:

ALTER TABLE < *nazwa_tabeli* >
{ ADD CONSTRAINT < *nazwa_ograniczenia* >
| **PRIMARY KEY**
| **UNIQUE** (< *nazwa_kol1*>) [, UNIQUE (<*nazwa_kol2* >]
| **MODIFY CONSTRAINT** < *nazwa_ograniczenia* >
| **PRIMARY KEY**
| **UNIQUE** (< *nazwa_kol1*>) **[, UNIQUE** (<*nazwa_kol2* > **]**
| **RENAME CONSTRAINT** < *stara_nazwa* > TO < *nowa_nazwa* >
| **DROP { PRIMARY KEY |** **UNIQUE (** < *nazwa_kol1* > [, < *nazwa_kol2*] . . . **)**
[CASCADE] [KEEP | DROP INDEX] | CONSTRAINT < *nazwa_ograniczenia* >
[CASCADE] }
| **ENABLE | DISABLE { PRIMARY KEY | CONSTRAINT** < *nazwa_ograniczenia* > **} } ;**

Indeks (*ang. index*) - umożliwia szybszy dostęp do wierszy tabeli w sytuacji, gdy pobierany z niej mały podzbiór wierszy. Można go tworzyć dla pojedynczej kolumny lub wielu kolumn. Poszczególne pozycje indeksu zapisywane są w strukturze o postaci B-drzewa, , dzięki czemu wyszukiwanie klucza właściwego dla danego wiersza wymaga wykonania bardzo niewielu operacji wejścia-wyjścia.
System Oracle oferuje kila różnych rodzajów indeksów:

- unikatowe
- z odwróconym kluczem
- funkcyjne
- bitmapowe
- połączeniowe

CREATE [OR REPLACE] VIEW < *nazwa_perspektywy* >
[< *nazwa_kol1* [, < *nazwa_kol2*]. . .] **AS**
SELECT . . . [**WITH CHECK OPTION | WITH READ ONLY**] ;

Perspektywy:

- widoki, wirtualne tabele – prezentują wybrane zawartości tabel lub innych perspektywy
- wyrażenia , z klauzuli SELECT podzapytania, muszą być zaopatrzone w aliasy lub zaproponować schemat dla wyniku podzapytania
- istnieje możliwość modyfikacji danych za pośrednictwem perspektyw „ prostych „

Przykłady:

```
CREATE OR REPLACE VIEW Studenci_stacjonarni AS
SELECT Nr_albumu, Nazwisko, Imiona, Rok, Rodzaj_studiow FROM Studenci
    WHERE Rodzaj_studiow IN ( 'INŻ_ST','MGR_ST_UZUP');


CREATE OR REPLACE VIEW Koszty_szkolen ( Numer, Ile, Kwota) AS
    SELECT Numer, Count(Numer), Sum(koszt)
        FROM Pracownicy_kursy JOIN Kursy USING(Kod)
            GROUP BY Numer ;
```

Słownik danych (METADANE)
Zbiór tabel i perspektyw systemowych przechowujących informacje na temat struktury bazy danych, zawartych w niej obiektów, kontach użytkowników, uprawnieniach, itp.

Nazwy tych perspektyw zbudowane są z dwóch członów:
przedrostków **USER** – obiekty z konta danego użytkownika
ALL – obiekty , do których użytkownik ma dostęp
DBA – obiekty całej bazy danych
nazw obiektów połączone znakiem **_**
tabel USER_TABLES, ALL_TABLES
perspektyw USER_VIEWS
ograniczeń USER_CONSTRAINTS, ALL_CONS_COLUMNS
indeksów USER_INDEXES
sekwencji USER_SEQUENCES
obiektów USER_OBJECTS

Przykłady:

```
SELECT Column_name,Constraint_type FROM
    All_constraints c JOIN All_cons_columns USING(Constraint_name)
        WHERE c.Table_name='STUDENCI' ;

SELECT column_name,constraint_name,constraint_type
FROM User_constraints c JOIN User_cons_columns cc USING( constraint_name)
    WHERE c.table_name='OCENY' ;
```