ORACLE

TIPURI DE DATE ORACLE

NUMBER **DEC, DECIMAL,** **NUMERIC**	Număr real de dimensiune variabilă, cu 38 de cifre semnificative.
NUMBER(n) **DEC(n), DECIMAL(n),** **NUMERIC (n)**	Număr întreg cu maxim n cifre
NUMBER(n, z) **DEC(n, z),** **DECIMAL(n, z),** **NUMERIC (n, z)**	Număr real cu n cifre dintre care z zecimale Subtipuri pentru NUMBER.
DOUBLE PRECISION, **FLOAT**	Subtipuri pentru NUMBER. Numere în virgulă mobilă cu 38 de cifre semnificative.
REAL	Subtip pentru NUMBER. Numere în virgulă mobilă cu 18 cifre semnificative.
INTEGER, INT, **SMALLINT**	Subtipuri pentru NUMBER. Numere întregi cu maxim 38 de cifre BINARY_INTEGER (Întregi între -2^31 şi 2^31).
NATURAL, POSITIVE	Subtipuri pentru BINARY_INTEGER. Numere întregi non-negative, respectiv pozitive.
NATURALN	Subtipuri pentru BINARY_INTEGER
POSITIVEN	Numere întregi nenule non-negative, respectiv pozitive.
SIGNTYPE	Subtip al BINARY_INTEGER. Poate lua doar valorile -1, 0 şi 1.
PLS_INTEGER	Întregi între -2^31 şi 2^31. Similar cu BINARY_INTEGER dar operaţiile cu astfel de numere sunt mai rapide şi în caz de depăşire se ridică o excepţie.
CHAR(n), CHAR	Şir de caractere de lungime fixă, egală cu n. Valoarea maximă pentru n este 2000. Dacă n lipseşte, şir de caractere de lungime 1.
CHARACTER, **CHARACTER(n)**	Identice cu cele anterioare. Introduse pentru compatibilitatea cu alte sisteme.
NCHAR(n)	Analog cu CHAR dar poate stoca şiruri în seturi de caractere naţionale (multioctet)
VARCHAR2(n)	Şir de caractere de lungime variabilă egală cu n. Valoarea maximă pentru n este 4000.
STRING(n), **VARCHAR(n)**	Identice cu VARCHAR2, introduse pentru compatibilitatea cu alte sisteme.
NVARCHAR(n)	Analog cu VARCHAR. Poate stoca şiruri scrise în seturi de caractere naţionale (multioctet)
LONG	Şir de caractere de maxim 2^31 octeţi. Este permisă doar o singură coloană de acest tip pentru o tabelă.
ROWID	Poate stoca un identificator pentru o linie dintr-o tabelă. Pentru conversia la/de la şir de caractere (18 caractere) se pot folosi

	funcţiile SQL ROWIDTOCHAR respectiv CHARTOROWID.
UROWID	Universal ROWID. Poate stoca un identificator logic şi fizic de linie într-o tabelă, indexată sau nu precum şi un identificator de linie extern (nonOracle). Nu este necesară folosirea funcţiilor de conversie la/de la şir de caractere (conversie automată).
RAW(n)	Similar cu VARCHAR2 dar conţine date binare. Valoarea maximă pentru n este 2000.
LONG RAW	Similar cu LONG dar conţine date binare.
DATE	Dată calendaristică (secol, an, lună, zi, oră, minut, secundă).
TIMESTAMP[(n)]	Extensie a tipului DATE. Conţine şi fracţiuni de secundă. Dacă este prezent, n specifică numărul de zecimale pentru acestea. Implicit n = 6
TIMESTAMP [(n)] WITH TIME ZONE	Extinde tipul TIMESTAMP conţinând şi o diferenţă între ora locală şi ora universală (GMT)
TIMESTAMP [(n)] WITH LOCAL TIME ZONE	Similar cu tipul anterior dar la stocarea în baza de date valorile sunt convertite la ora zonei bazei de date iar la regăsire la ora zonei aplicaţiei client.
INTERVAL YEAR [(n)] TO MONTH	Se stochează intervale de ani şi luni. Precizia n specifică numărul de cifre pentru an (între 0 şi 4, implicit 2).
INTERVAL DAY [(z)] TO SECOND [(s)]	Similar cu tipul anterior, dar pentru intervale de zile şi secunde. valorile z şi s sunt preciziile pentru zile, respectiv secunde (0-9, implicit 2 pentru z şi 6 pentru s).

CREAREA TABELELOR

O sintaxă simplificată a comenzii CREATE TABLE este prezentată în continuare:

```
CREATE TABLE [schema.]nume_tabel
( nume_coloana tip_data [DEFAULT expresie]
 [, nume_coloana tip_data [DEFAULT expresie] ... )
[PCTFREE întreg] [PCTUSED întreg]
[TABLESPACE spaţiu_tabel]
[STORAGE parametrii_de_stocare]
```

unde:

- DEFAULT desemnează o valoare implicită pentru coloană, folosită în cazul în care la inserarea unui rând în tabel nu este specificată o valoare explicită pentru coloana în cauză.
- TABLESPACE specifică spaţiul tabel în care va fi stocat tabelul. Dacă acesta nu este menţionat explicit, se va folosi spaţiul tabel implicit (default) al utilizatorului care este proprietarul schemei din care face parte tabelul (vezi exemplul de mai sus).
- Valorile parametrilor PCTFREE şi PCTUSED determină gradul de utilizare a blocurilor din extinderile segmentului tabel.

- Clauza `STORAGE` este folosită pentru setarea parametrilor de stocare (`INITIAL`, `NEXT`, `PCTINCREASE`, `MINEXTENTS`, `MAXEXTENTS`) prin intermediul cărora se specifică mărimea şi modul de alocare a extinderilor segmentului tabel.

CREATE TABLE [schema.]*nume_tabel* [(*descriere_coloana_1, ..., descriere_coloana_n*)]
AS fraza_SELECT;

TIPURI DE CONSTRANGERI

- NOT NULL: valorile nu pot fi nule

Sintaxa:
coloana [CONSTRAINT nume_constr] NOT NULL

- PRIMARY KEY: defineşte cheia primară a unui tabel

Sintaxa la nivel de coloană:
coloana [CONSTRAINT nume_constrangere] PRIMARY KEY

Sintaxa la nivel de tabel:
, [CONSTRAINT nume_constrangere] PRIMARY KEY(lista_coloane)

- ***UNIQUE***: defineşte o altă cheie a tabelului

Sintaxa la nivel de coloană:
coloana [CONSTRAINT nume_constrangere] UNIQUE
Sintaxa la nivel de tabel:
,[CONSTRAINT nume_constrangere] UNIQUE(lista_coloane)

- **FOREIGN KEY**: defineşte o cheie străină (externă)

La nivel de coloană:
coloana [CONSTRAINT nume_constrangere]
REFERENCES tabel(coloana)
[ON DELETE CASCADE | ON DELETE SET NULL]

La nivel de tabel:
, [CONSTRAINT nume_constrangere]
FOREIGN KEY(lista_coloane) REFERENCES tabel(lista_coloane)
[ON DELETE CASCADE | ON DELETE SET NULL]

Daca se doreşte ca la ştergerea liniei conţinând o valoare de cheie să fie şterse suplimentar şi toate liniile care referă această valoare, se specifică ON DELETE CASCADE
Dacă se doreşte ca liniile care referă valoarea cheii respective să nu fie şterse, se specifică ON DELETE SET NULL care are următorul efect: în toate liniile care referă acea valoare ea este înlocuită cu valori nule.

- **CHECK**: introduce o condiţie (expresie logică).

SINTAXA
La nivel de coloană:
coloana [CONSTRAINT nume_constrangere] CHECK (expresie_logica)
La nivel de tabel:
, [CONSTRAINT nume_constrangere] CHECK (expresie_logica)

MODIFICAREA STRUCTURII TABELELOR

- ADAUGARE COLOANA

Sintaxa:
ALTER TABLE nume_tabel
ADD (nume_coloana tip_date [DEFAULT expresie] [constrangere]);

- STERGERE COLOANA

Sintaxa:
ALTER TABLE nume_tabel DROP COLUMN nume_coloana
[CASCADE CONSTRAINTS];

Sau:
ALTER TABLE nume_tabel DROP (lista coloane)
[CASCADE CONSTRAINTS];

-Cererea se poate executa atât în cazul în care tabelul este gol cât şi dacă el conţine deja randuri,
-Dacă tabel are doar o singură coloană, aceasta nu se poate şterge,
-Opţiunea CASCADE CONSTRAINTS şterge suplimentar toate constrângerile de integritate în care sunt implicate coloanele şterse, inclusiv cele de tip FOREIGN KEY care referă valorile coloanei sau coloanelor respective.

- UNUSED

Ştergerea unei coloane este o operaţie costisitoare din punct de vedere al resurselor sistemului.
De aceea, în cazul în care la un moment dat acesta este foarte încărcat ştergerea se poate face în două etape:
Etapa 1: Marcarea ca neutilizate a unei coloane sau a unei liste de coloane prin opţiunea SET UNUSED. Sintaxa este:
ALTER TABLE nume_tabel SET UNUSED(lista_coloane);

sau
ALTER TABLE nume_tabel SET UNUSED COLUMN nume_coloana;
Etapa 2: Ştergerea efectivă a coloanelor marcate ca neutilizate la un moment ulterior de timp, când sistemul nu mai este foarte încărcat. Sintaxa cererii este:
ALTER TABLE nume_tabel DROP UNUSED COLUMNS;

- MODIFICARE COLOANA

Sintaxa:
ALTER TABLE nume_tabel MODIFY(nume_coloana [tip_date] [DEFAULT expresie] [constrangere])
Prin această cerere se poate schimba tipul de date al coloanei, se poate asocia o nouă valoare implicită, se poate adăuga coloanei o constrângere de tip NOT NULL.
Printr-o singură cerere se pot efectua toate operaţiile de mai sus sau doar o parte a lor.

- ADAUGARE CONSTRANGERE

Sintaxa:
ALTER TABLE nume_tabel ADD [CONSTRAINT nume]
tipConstrangere(coloana);
Tipul constrângerii nu poate fi NOT NULL.
În schimb se pot defini constrângeri de CHECK conţinând condiţii de acest tip.
Exemplu: Se doreşte ca in coloana NUME a tabelului SPEC să nu existe valori nule sau identice şi lungimea minimă să fie de 6 caractere:
ALTER TABLE SPEC ADD CONSTRAINT NUME_NENUL CHECK(NUME IS NOT NULL);
ALTER TABLE SPEC ADD CONSTRAINT NUME_UNIC UNIQUE(NUME);
ALTER TABLE SPEC ADD CONSTRAINT NUME_5 CHECK(LENGTH(NUME)>5);

- STERGERE CONSTRANGERE

Sintaxa:
ALTER TABLE nume_tabel DROP PRIMARY KEY [CASCADE];
ALTER TABLE nume_tabel DROP UNIQUE(lista_coloane) [CASCADE];
ALTER TABLE nume_tabel DROP CONSTRAINT nume [CASCADE*];*
Efectul acestor cereri este:
DROP PRIMARY KEY şi DROP UNIQUE specifică ştergerea constrângerii de tip cheie primară/cheie unică pentru tabelul respectiv. Constrângerea poate să nu aibă un nume asociat la definire.
DROP CONSTRAINT specifică ştergerea unei constrângeri având asociat un nume.
Opţiunea CASCADE se aplică în cazul în care există constrângeri dependente şi specifică ştergerea suplimentară a acestora.

- ACTIVARE/DEZACTIVARE CONSTRANGERE

Sintaxa:

Dezactivare:
ALTER TABLE nume_tabel DISABLE PRIMARY KEY [CASCADE]
ALTER TABLE nume_tabel DISABLE UNIQUE(lista_coloane) [CASCADE]
ALTER TABLE nume_tabel DISABLE CONSTRAINT nume [CASCADE]
Reactivare:
ALTER TABLE nume_tabel ENABLE PRIMARY KEY;
ALTER TABLE nume_tabel ENABLE UNIQUE(lista_coloane);
ALTER TABLE nume_tabel ENABLE CONSTRAINT nume;

EFECT
DISABLE specifică dezactivarea constrângerii de integritate respective. Opţiunea CASCADE duce la dezactivarea suplimentară a tuturor constrângerilor dependente.
ENABLE CONSTRAINT specifică reactivarea constrângerii respective
În momentul reactivării sistemul verifică dacă datele la care se referă constrângerea sunt conforme cu aceasta. În cazul în care nu se constată respectarea constrângerii, ea nu este activată şi se generează un mesaj de eroare. În caz de activare cu succes, constrângerile dependente nu sunt reactivate.

GOLIRE TABEL
Sintaxa:
TRUNCATE TABLE nume_tabel [REUSE STORAGE];
Opţiunea REUSE STORAGE este folosită pentru a specifica faptul că spaţiul ocupat de liniile şterse rămâne alocat tabelei şi poate fi folosit la inserările ulterioare în aceasta. În lipsa acestei opţiuni spaţiul respectiv devine disponibil, putând fi utilizat şi pentru alte tabele.

- **STERGERE TABEL**

Sintaxa:
DROP TABLE nume_tabel

- **REDENUMIRE TABEL**

Sintaxa:
RENAME nume_vechi TO nume_nou

1. **Conectare cu contul (DBA)** ***profesor***

Connect profesor/profesor

2. **sa se elimine userii ru si sal**

drop user ru cascade;
drop user sal cascade;

3. **sa se elimine spatiile tabel ts_ru si ts_temp**

drop tablespace *ts_ru* including contents and datafiles;
drop tablespace *ts_temp* including contents and datafiles;

4. **sa se creeze tablespace-ul ts_ru**

create tablespace ts_ru datafile 'd:\date\fd_ru.dbf' size 10M;

5. **sa se creeze tablespace-ul temporar**

create temporary tablespace ts_temp tempfile 'D:\date\f_temp.dbf' size 5M;

6. **sa se creeze user-ul RU cu parola ru**

create user ru identified by ru
default tablespace ts_ru
temporary tablespace ts_temp
quota unlimited on ts_ru ;

4. **sa se creeze userul sal cu parola sal**

create user sal identified by sal
default tablespace ts_ru
temporary tablespace ts_temp
quota unlimited on ts_ru ;

5. **Sa se creeze scriptul rolDezvoltator.sql pentru crearea rolului *dezvoltator* cu privilegiile*: create table, create view, create procedure, create synonym, create sequence*.**

In notepad sau sqldeveloper
save as
d:\date\rolDezvoltator.sql :

create role dezvoltator;
grant create table, create view, create procedure, create synonym, create sequence to dezvoltator;

6. **Sa se execute scriptul rolDezvoltator.sql**

Run SQL command line

SQL> @D:\date\ rolDezvoltator.sql ;

Sau din sqldeveloper:

@"D:\date\ rolDezvoltator.sql ";

7. Sa se acorde acest rol utilizatorilor ru si sal.

grant *dezvoltator* to ru, sal;

7.1 Pentru revocare folosim:

revoke *dezvoltator* from ru,sal;

8. Creare tabele

```
create table departamente
      (  cod_Dep char(10) primary key,
         den_Dep varchar2(30) not null,
         adresa varchar2(50) not null
      );
create table nomenclator_functii
      (    cod_functie char(10) primary key,
         den_functie varchar2(30)
      );

create table angajati
      ( marca int  primary key,
        nume varchar2(30),
        CNP char(13),
        cod_functie char(10),
        cod_Dep char(10) references departamente(cod_Dep),
        foreign key(cod_functie) references nomenclator_functii(cod_functie)
     );
```

9. sa se acorde dreptul „references” utilizatorului(schemei)*sal* pentru tabelul ru.angajati

grant references on *ru.angajati* to *sal*;

Sa se creeze tabelele salarii si pontaj:

```
create table salarii(
        marca int primary key references ru.angajati(marca),
        salariu_baza int,
        data_angajarii date,
        vechime_anterioara int
);

create table pontaj(
        id_pontaj int primary key,
        marca int references ru.angajati(marca),
        luna int,
        anul int,
        ore_lucrate int default 0,
        ore_supl int default 0,
        co int default 0, --concediu de odihna
        cb int default 0  --concediu boala obisnuita
)

select * from ru.angajati
```

10. sa se acorde dreptul „select” pentru tabelul angajati

```
grant select on ru.angajati to sal;

select * from ru.angajati
```

SINONIME

Un sinonim este un nume alternativ pentru un obiect (de tip tabel, vedere, secventa, functie, procedura).

Sinonimele sunt folosite uzual pentru a permite unui utilizator sa foloseasca obiecte din schema altui utilizator, fara a specifica proprietarul acestora. Cand un utilizator foloseste un sinonim, acesta trebuie sa cunoasca doar numele sinonimului, nu si proprietarul sau numele obiectului referit de sinonim.

Un sinonim este de două feluri: public sau privat. Un sinonim public poate fi folosit de către toţi utilizatorii bazei de date. Un sinonim privat este proprietatea numai a unui utilizator, putând fi accesat la fel ca orice obiect din schema acestuia. Pe de altă parte, trebuie reţinut că definirea unui sinonim, fie el public sau privat, nu implică accesul la obiectul referit de acesta de către alţi utilizatori. Pentru a accesa respectivul obiect, utilizatorul trebuie să posede privilegiile adecvate pentru accesarea obiectului.

Pentru a crea un sinonim se foloseşte comanda SQL *create synonym*, având sintaxa:

```
CREATE [PUBLIC] SYNONYM [schema.]nume_sinonim
FOR [schema.]obiect
```

11. **sa se creeze un synonym pentru tabelul ru.angajati (in schema sal)**

create synonym *angajati* for *ru.angajati*

- testam:
select * from angajati

12. **SEQUENCE-generare de auto-numbers**

O secventa este un obiect al bazei de date care serveste la generarea unor numere intregi unice, putand fi folosita simultan de mai multi utilizatori, evitand aparitia conflictelor si blocarii. Ca orice alt obiect al bazei de date, o secventa poate fi creata, modificata sau distrusa. Oracle stocheaza definitiile secventelor in dictionarul de date.

Pentru a crea o secvenţă se foloseşte comanda SQL `CREATE SEQUENCE`, având următoarea sintaxă:

```
CREATE SEQUENCE nume_secvenţă
[INCREMENT BY întreg]
[START WITH întreg]
[MAXVALUE întreg | NOMAXVALUE]
[MINVALUE întreg | NOMINVALUE]
[CYCLE | NOCYCLE]
[CACHE întreg | NOCACHE]
[ORDER | NOORDER]
```

unde

- `INCREMENT BY` specifică intervalul dintre două numere ale secvenţei. Aceasta poate fi orice valoare pozitivă sau negativă, dar nu poate fi 0. Dacă valoarea este pozitivă atunci secvenţa este în ordine descrescătoare, dacă este pozitivă, atunci secvenţa este în ordine crescătoare. Valoarea implicită (default) este 1.
- `START WITH` specifică prima valoare generată de secvenţă. Această opţiune poate fi folosită pentru a începe o secvenţă crescătoare cu o valoare mai mare decât valoarea sa minimă sau pentru a începe o secvenţă descrescătoare cu o valoare mai mică decât valoarea sa maximă. Pentru secvenţe crescătoare valoarea implicită este valoarea minimă a secvenţei, iar pentru secvenţe descrescătoare valoarea implicită este valoarea maximă a secvenţei.

- MINVALUE specifică valoarea minimă a secvenţei. NOMINVALUE specifică o valoare minimă de 1 pentru o secvenţă crescătoare sau -10 pentru o secvenţă descrescătoare. NOMINVALUE este valoarea implicită.
- MAXVALUE specifică valoarea maximă a secvenţei. NOMAXVALUE specifică o valoare maximă de 10 pentru o secvenţă descrescătoare şi 10^{27} pentru o secvenţă crescătoare. NOMAXVALUE este valoarea implicită.
- CYCLE arată că secvenţa continuă să genereze valori şi după ce a atins valoarea maximă (în cazul secvenţelor crescătoare) sau valoarea minimă (în cazul secvenţelor descrescătoare). În acest caz, după ce o secvenţă crescătoare a atins valoarea maximă, ea va genera valoarea sa minimă; după ce o secvenţă descrescătoare a atins valoarea minimă, ea va genera valoarea sa maximă. NOCYCLE arată că secvenţa nu mai poate genera valori după ce a atins valoarea maximă (în cazul secvenţelor crescătoare) sau valoarea minimă (în cazul secvenţelor descrescătoare). Valoarea implicită este NOCYCLE.
- CACHE arată câte valori ale secvenţei sunt prealocate de către Oracle şi ţinute în memorie pentru acces mai rapid. Valoarea minimă a acestui parametru este 2. NOCACHE arată ca nu există valori prealocate ale secvenţei. Dacă se omit amândouă opţiunile, CACHE şi NOCACHE, Oracle prealocă implicit 20 de valori.

13. sa se creeze secventa seq_id_pontaj

create sequence *seq_id_pontaj* increment by 1 start with 1

select seq_id_pontaj.nextval from dual

select seq_id_pontaj.currval from dual //valoarea curenta

insert into pontaj values(*seq_id_pontaj.nextval*, 1, 10,2015,40,0,0,0)

14. sa se acorde privilegiul de modificare a informatiilor din coloanele id_pontaj, marca, luna, anul si ore_lucrate ale tabelului pontaj din schema sal pentru utilizatorul ru.

grant update(id_pontaj, marca, luna, anul, ore_lucrate) on sal.pontaj to ru;