1. **ALTER TABLE**

alter table   table\_name  
add  
   (  
   column1\_name column1\_datatype column1\_constraint,    
   column2\_name column2\_datatype column2\_constraint,  
   column3\_name column3\_datatype column3\_constraint  
   );

* **Sa se adauge tabelului *salarii* coloana *prima* de tip int.**

***alter table salarii***

***add prima int;***

* **Sa se stearga coloana de la *prima*.**

***alter table salarii***

***drop column prima;***

* **Sa se adauge tabelului salarii coloana prima de tip int cu constrangerea ca prima sa fie mai mica sau egala decat 100.**

***alter table salarii***

***add prima int check(prima<=100);***

* **Modificarea coloanelor**

ALTER TABLE nume\_tabela

MODIFY(nume\_coloana [tip\_date]

[DEFAULT expresie] [constrangere])

Prin această cerere se pot modifica tipul de date al coloanei, valoarea implicită si

se poate adăuga/modifica o constrângere de tip NULL / NOT NULL pentru acea coloană.

* **Sa se modifice coloana prima astfel incat sa fie de tipul number (7,2) (7 cifre dintre care doua dupa virgula).**

***alter table salarii***

***modify (prima number(7,2) );***

* **Adaugare constrangeri**

ALTER TABLE nume\_tabela

ADD [CONSTRAINT nume] tip(coloana);

Exemplu:

ALTER TABLE ANGAJATI

ADD CONSTRAINT NUME\_3 CHECK(LENGTH(NUME)>3);

ALTER TABLE ANGAJATI

drop constraint Nume\_3

* **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 [CASCADE]

***Reactivare:***

ALTER TABLE nume\_tabel ENABLE PRIMARY KEY;

ALTER TABLE nume\_tabel ENABLE UNIQUE(lista\_coloane);

ALTER TABLE nume\_tabel ENABLE CONSTRAINT nume;

1. **Functii de prelucrare a valorii NULL**

Functiile generale sunt: NVL, NVL2, NULLIF si COALESCE, aceste functii lucreaza cu orice tip de date.

|  |  |
| --- | --- |
| **Functie** | **Descriere** |
| NVL (expr1, expr2) | Daca *expr1* nu este null, returneaza *expr1*. Daca *expr1* este null returneza *expr2* |
| NVL2 (expr1, expr2, expr3) | Daca *expr1* nu este null, NVL2 returneaza *expr2*. Daca *expr1* este null, NVL2 returneza *expr3*. Argumentul *expr1* poate fi orice tip de date. |
| NULLIF (expr1, expr2) | Compara doua expresii si returneaza valoarea null daca sunt egale sau prima expresie daca nu sunt egale. |
| COALESCE (expr1, expr2, ..., expr n) | Returneaza prima expresie non-null din lista de expresii. |

**Sa se afiseze din tabelul *salarii* coloanele *marca, salariu\_baza, prima, total (=salariu\_baza+prima)*.**

***select marca,salariu\_baza,prima,salariu\_baza+NVL(prima,0) as total***

***from salarii;***

***//sa adune cu 0 daca avem NULL***

1. ***Exemple de utilizare a comenzii select***

* Sa se utilizeze pseudocoloana *rownum* pt numerotarea randurilor rezultat al comenzii select.

Rownum este o pseudo-coloana. Aceasta pastreaza numarul inregistrarii, numerotarea inregistrarilor incepe de la 1.

***select rownum as nrCrt, salarii.\* from salarii order by marca desc;***

* Sa se afiseze cele mai mari 10 salarii in ordinea descrescatoare a lor.

***select rownum as nrCrt, salarii.\* from salarii where rownum<=10 order by salariu\_baza desc;***

* Sa se determine totalul cheltuielilor salariale la nivel de departament.

***select cod\_dep, sum(salariu\_baza+NVL(prima,0)) as TOTAL***

***from angajati A inner join salarii B***

***on A.marca=B.marca***

***group by cod\_dep;***

* Sa se afiseze salariatii cu salariu\_baza maxim.

***select \* from angajati A inner join salarii B on A.marca=B.marca***

***where salariu\_baza = ( select max(salariu\_baza) from salarii***

***)***

1. ***Comanda update***

**UPDATE [table-Name](http://docs.oracle.com/javadb/10.6.1.0/ref/rreftablename.html" \l "rreftablename)**

**SET [column-Name](http://docs.oracle.com/javadb/10.6.1.0/ref/rrefcolumnname.html" \l "rrefcolumnname) = Value**

**[ , [column-Name](http://docs.oracle.com/javadb/10.6.1.0/ref/rrefcolumnname.html" \l "rrefcolumnname) = Value,… ]**

**[[WHERE clause](http://docs.oracle.com/javadb/10.6.1.0/ref/rrefsqlj33602.html" \l "rrefsqlj33602)] |**

* **Sa se acorde o prima de 100 lei salariatului cu marca=1;**

***update salarii***

***set prima=100***

***where marca=1;***

* Sa se acorde aceleasi drepturi salariale angajatului cu marca=2 ca si celui cu marca=1.

***update salarii***

***set (salariu\_baza, prima)=(select salariu\_baza, prima***

***from salarii***

***where marca=1***

***)***

***where marca=2;***

* Sa se acorde angajatului cu marca=2 salariul angajatului cu marca=1 cu 200lei mai putin si prima pe jumatate.

***update salarii***

***set (salariu\_baza, prima)=(select salariu\_baza-200, prima/2***

***from salarii***

***where marca=1***

***)***

***where marca=2;***

1. **12. Functia DECODE**

- Aceasta este una dintre cele mai puternice functii Oracle actionand ca o comanda if-then-else sau case dintr-un limbaj procedural.

Forma generală:

DECODE(col | expresie, valoare1, rezultat1 [, valoare2, rezultat2 […]]

[, rezultat\_implicit])

Dacă col | expresie ia valoarea *valoarek,* atunci funcţia DECODE va returna valoarea *rezultatk*.

Sintaxa funcţiei **CASE** este similară comenzii If-Then-Else. Oracle verifică fiecare condiţie începând cu prima condiţie (de la stânga la dreapta). Dacă condiţia respectivă este adevărată (partea WHEN), este returnată valoarea expresiei (partea THEN). Dacă nici o condiţie nu este verificată, se returnează expresia de la ELSE. Partea ELSE este opţională – expresia CASE va returna Null dacă nu este adevărată nici o condiţie.

CASE [expression]

WHEN condition\_1 THEN result\_1

WHEN condition\_2 THEN result\_2

…

WHEN condition\_n THEN result\_n

ELSE result

END

* **Sa se mareasca salariul angajatului cu marca=1 cu 10%, agajatului cu marca=2 cu 15% si a celorlalti cu 5%.**

***update salarii***

***set salariu\_baza=salariu\_baza\* decode(marca, 1, 1.1, 2, 1.15, 1.05)***

***from salarii;***

**//decode este similar cu case din limbajele procedurale**

**//….???**

select sal, case

when sal < 2000 then 'category 1'

when sal < 3000 then 'category 2'

when sal < 4000 then 'category 3'

else 'category 4'

end

from emp;

* Sa se modifice *salariu\_baza* pentru salariatii din departamentul *aprov* conform formulei: *salariu\_baza=salariu\_baza\*1.1* ;

***update salarii S***

***set salariu\_baza=salariu\_baza\*1.1***

***where (select cod\_dep from angajati A inner join salarii B on A.marca=B.marca***

***where A.marca=S.marca)=’aprov’;***

1. **CREAREA VEDERILOR**

O sintaxă simplificată a comenzii create view este următoarea:

CREATE [OR REPLACE] [FORCE | NOFORCE] VIEW nume\_vedere

[(alias [,alias]...)]

AS subinterogare

[WITH READ ONLY]

[WITH CHECK OPTION [CONSTRAINT nume\_constrangere]]

unde

* OR REPLACE recreează vederea dacă ea există deja. Această opţiune poate fi folosită pentru a schimba definiţia unei vederi existente fără a o distruge în prealabil. Avantajul recreării vederii prin opţiunea REPLACE este că în acest caz se păstrează toate privilegiile acordate asupra acestei vederi. De exemplu, să presupunem că după crearea unei vederi, au fost acordate privilegii asupra vederii pentru anumite roluri sau pentru anumiţi utilizatori. Dacă aceea vederea este distrusă şi recreată, atunci toate privilegiile asupra vederii au fost pierdute şi trebuie acordate din nou. Dacă vederea este însă recreată folosind opţiunea OR REPLACE, atunci privilegiile acordate sunt păstrate şi nu mai este necesară acordarea lor încă o dată.
* FORCE este o opţiune care permite crearea vederii indiferent dacă tabelele de bază şi coloanele la care se face referire există sau nu, sau dacă utilizatorul posedă sau nu privilegiile corespunzătoare în legătură cu tabelele respective. Opţiunea opusă, NOFORCE, creează vederea numai dacă tabelele de bază există şi dacă utilizatorul posedă privilegiile corespunzătoare în legătură cu tabelele respective; NOFORCE este opţiunea implicită. Dacă se foloseşte opţiunea FORCE şi un tabel de bază nu există sau una dintre coloane nu este validă, atunci Oracle va crea vederea cu erori de compilare. Dacă mai târziu tabelul în cauză este creat sau coloana este corectată, atunci vederea poate fi folosită, Oracle recompilând-o dinamic înainte de folosire.
* alias specifică numele expresiilor selectate de interogarea vederii. Numărul alias-urilor trebuie să fie acelaşi cu numărul de expresii selectate de către interogarea vederii. Un alias trebuie să fie unic în cadrul unei interogări. Dacă sunt omise alias-urile, Oracle va folosi denumirile coloanelor din interogare. Atunci când interogarea vederii conţine şi expresii, nu doar simple coloane, trebuie folosite alias-uri.
* AS indică interogarea vederii. Aceasta poate fi orice instrucţiune SELECT care nu conţine clauzele ORDER BY şi FOR UPDATE.
* opţiunea WITH READ ONLY asigură că nici o operaţie DML (inserare, ştergere, modificare) nu va fi asigurată asupra vizualizării.
* WITH CHECK OPTION este o constrângere care arată că toate actualizările efectuate prin intermediul vederii vor afecta tabelele de bază numai dacă actualizările respective vor avea ca rezultat numai rânduri care pot fi vizualizate prin intermediul vederii. CONSTRAINT furnizează un nume pentru constrângerea CHECK OPTION. Asupra acestor opţiuni vom reveni puţin mai târziu

**Distrugerea vederilor**

Pentru distrugerea unei vederi se foloseşte comanda drop view:

drop view nume\_vedere;

* Sa se creeze vederea *v\_Angajati* pe baza datelor din tabelul angajati din schema RU si salarii si schema SAL. Vederea se va crea in schema SAL.

***create view v\_Angajati as***

***select A.\*, salariu\_baza, data\_angajarii, vechime\_anterioara***

***from angajati A inner join salarii B on A.marca=B.marca;***

***( am folosit synonym-ul*** *angajati* ***al tabelului*** *ru.angajati* ***)***

***select \* from v\_Angajati;***

* Sa se creeze tabelul *t\_Aprovizionare* pe baza datelor din vederea *v\_Angajati*.

***create table t\_Aprovizionare as***

***select\* from v\_Angajati;***

* Sa se stearga tabelul de la 2.

***drop table t\_Aprovizionare;***

* Sa se creeze tabelul *t\_Ang\_Aprovizionare* pe baza datelor din vederea *v\_Angajati* ce contine salariatii din departamentul Aprov.

***create table t\_Ang\_Aprovizionare as***

***select\* from v\_Angajati where cod\_dep=’aprov’;***

***select \* from t\_Aprovizionare;***

* Sa se afiseze salariatii cu salariu\_baza maxim din fiecare departament.

***select \* from v\_Angajati A***

***where salariu\_baza=(select max(salariu\_baza)***

***from v\_angajati***

***where A.cod\_dep=cod\_dep***

***);***