Vilniaus Gedimino technikos universitetas
Informacinių technologijų katedra

Duomenų bazių valdymas 1

Lektorė Janina Galkauskaitė

Mokomoji knyga

2007

Duomenų bazių valdymas
Lektorė Janina Galkauskaitė

DUOMENŲ BAZIŲ VALDYMO SISTEMA 1

Medžiaga skirta studentams, pradedantiems studijuoti ORACLE Duomenų bazių valdymo sistemą.
Čia pateikiama informacija apie bendriausius informacinių sistemų kūrimo principus ir ORACLE SQL kalbos realizaciją.

ĮŽANGA Į ORACLE: SQL IR PL/SQL

SQL (struktūrinė užklausų kalba) - apima duomenų bazės struktūros kūrimą , duomenų saugojimą, peržiūrą bei manipuliavimą duomenimis reliacinėse duomenų bazėse.
PL/SQL kursas apima medžiagą, kuri moko kaip reikia kurti PL/SQL blokus aplikacijoms.

INFORMACINIŲ SISTEMŲ KŪRIMAS

Bet kokios informacinės sistemos kūrimo ir gyvavimo ciklas susideda iš:

- Strategijos ir analizės
- Sistemos Projektavimo
- Programavimo ir dokumentavimo
- Sistemos Testavimo
- Sistemos Diegimo

DUOMENŲ SAUGOJIMAS SKIRTINGOMIS PRIEMONĖMIS

Kiekviena organizacija turi poreikį tam tikrai informacijai saugoti. Bibliotekoje tai gali būti sąrašas knygų, skaitytojų ir kt. Tai gali būti informacija apie darbuotojus, departamentus, atlyginimus. Tie informacijos gabalai yra vadinami duomenimis. Organizacija gali saugoti duomenis skirtingais būdais ir skirtingomis priemonėmis, pvz. popieriniai dokumentai, elektroninės lentelės arba duomenų bazės.

Duomenų bazė (DB) yra organizuota informacijos kolekcija. Valdyti DB reikalinga valdymo sistema yra vadinama duomenų bazių valdymo sistema (DBVS). DBVS yra programos, kurios atvaizduoja arba modifikuoja duomenis DB pagal vartotojo poreikius. Yra pagrindiniai keturi DB tipai: hierarchinės DB, tinklinės DB, reliacinės DB ir naujausias tipas- objektinė reliacinė DB.

Viena iš DBVS yra ORACLE. ORACLE skirta reliacinei DB aptarnauti.

Reliacinės DB koncepcija buvo sukurta 1970 m . Dr. E.F.Codd pasiūlė reliacinį duomenų modelį. Tai yra reliacinės DB valdymo sistemos pagrindas. Reliaciniam modelis būdinga:

- objektų ir ryšių kolekcijos
- operatoriai, veikiantys per ryšius nustatant kitus ryšius
- duomenų vientisumas būtinas dėl jų tikslumo ir nuoseklumo

RELIACINĖS DB (RDB) APIBRĖŽIMAI

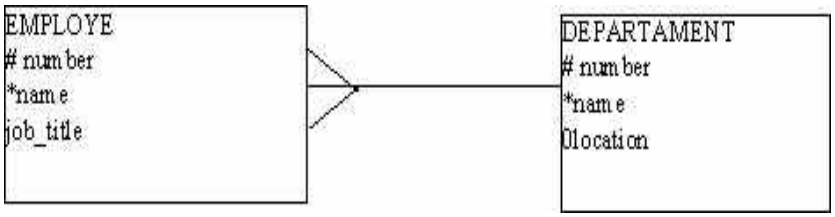
RDB naudoja dvimates lenteles informacijai talpinti ir ryšius tarp jų.
Pvz. gali reikėti saugoti informaciją apie darbuotojus. RDB jūs kuriate keletą lentelių, skirtų saugoti skirtingoms informacijos dalims apie darbuotojus, t.y. darbuotojų lentelė, departamentų lentelė ir atlyginimų lentelė. Taip informacija yra skaidoma, kad nesikartotų ta pati informacija keletą kartų. Pvz. Tame pačiame departamente gali dirbti keletas darbuotojų, tai kad nereikėtų kiekvienam iš jų kartoti to paties departamento pavadinimo, jį galima iškelti į kitą lentelę, ir tik reikalui esant kreiptis į departamentų lentelę. Taip būtų taupoma vieta.

DUOMENŲ MODELIS

Sistemos projektuotojas kuria modelį kad pateikti idėją vartotojui ir ją vystyto taip, kad jis taptų tinkamas DB projektavimui. Pirmiausia veiklos modelis gimsta galvoje- po to kuriamas esybių ryšių modelis, tada projektuojamos lentelės ir galiausiai jos kuriamos.

ESYBIŲ RYŠIŲ MODELIS

Tai efektyvi sistema, kai duomenys yra dalinami į diskretiškas porcijas (kategorijas), kitaip dar vadinamas esybėmis. Šios lentelės vėliau yra surišamos, nustatant ryšius tarp jų, t.y. sukuriamas esybių ryšių modelis. Esybių ryšių modelis yra iliustracija įvairių veiklos esybių ir ryšių tarp jų. ER modelis atsiranda iš veiklos aprašymo ir sukuriamas per analizės fazę sistemos gyvavimo ciklo kūrimo metu. ER modelis atskiria informaciją pagal veiklos darbus ir reikalavimus . Veikla gali kisti , bet informacijos tipas lieka tas pats. Be to, duomenų struktūra taip pat turėtų likti ta pati. Pavyzdys žemiau atspindi esybę EMPLOE, skirtą saugoti duomenis apie darbuotojus, o esybė DEPARTAMENT saugo informaciją apie departamentus.



Esybė atspindi informaciją apie objektą kurią reikia saugoti. Tai departamentai, darbuotojai, jų algos. Esybė sudaryta iš atributų. Atributai aprašo esybę. Pvz. darbuotojo numeris, vardas, pareigos ir t.t. Bet kuris iš atributų gali būti privalomas arba nebūtinas. Tai įtakoja vėliau DB aptarnavimą užpildant ją duomenimis.

RYŠIAI

Ryšiai nustato priklausomybę tarp esybių. Jie parodo priklausomybės būtinumą ir laipsnį. Pavyzdžiu gali būti surišti darbuotojai ir departamentai, t.y. kiekvienam darbuotojui yra priskirtas departamentas, arba kitaip galima sakyti, kad departamente yra vienas ar keli darbuotojai.

Esybės žymimos stačiakampiu, nurodant jos vardą didžiosiomis raidėmis.
Atributai yra žymimi mažosiomis raidėmis nurodant jų vardus, ir prieš vardą pažymint sutartinius ženklus:
#- nurodo unikalų numerį- t. y. pirminį raktą (Primary Key)
*- būtinai turi būti užpildyta atributo reikšmė
0 – atributas nebūtinai turi būti užpildytas

Ryšiams nustatyti gali būti naudojamos skirtingos linijos, jungiančios esybes (esybių ryšiai):
Punktyrinė linija- reiškia nebūtinus elementus t.y. ryšys "gali būti"
Ištisinė linija – privalomas elementas reiškia "turi būti'
"Varnos koja" ryšio gale - nustato laipsnį, indikuojantį "vienas ar daugiau"
Paprasta linija ryšio gale – tai elementas, reiškiantis laipsnį "vienas ir tik vienas"
Ryšiai bet kuria kryptimi turi Vardą, priklausomumo privalumą ir ryšio laipsnį.

RELIACINĖS DUOMENŲ BAZĖS TERMINOLOGIJA

Reliacinė DB gali būti sudaryta iš vienos ar kelių lentelių. Lentelė yra pagrindinė duomenų struktūros saugykla RDBVS. Lentelė saugo visus būtinus duomenis iš realaus pasaulio apie tam tikrus objektus. Pvz. Darbuotojus, pirkėjus, važtaraščius.

❷ EMPNO	ENAME	❸ JOB	MGR	HIREDATE	SAL	❹ COMM	DEPT
7839	KING	PREZID ❻		01-NOV-81	8000		10
7698	BLAKE	MANA	7839	01-MAY-82	5000	❺ 20	⊇
7782	CLARK	SALES	7698	01-NOV-81	2300		10

❶- eilutė atspindi visus duomenis apie vieną darbuotoją. Kiekviena eilutė lentelėje turi būti identifikuojama pirminiu raktu, kuris taip pat apsaugo nuo pasikartojančių eilučių. Eilučių tvarka yra nesvarbi, ją galima nustatyti kai duomenys yra pateikiami vartotojui.
❷- Stulpelis šiuo atveju yra skirta darbuotojų numeriams, kuris taip pat suprantamas kaip pirminis raktas. Darbuotojo numeris yra unikalus lentelėje EMP. Pirminis raktas privalo būti užpildytas reikšmėmis. Bendru atveju stulpelis saugo informaciją pateiktą atribute.
❸ -Stulpelis, kuri nėra raktinė. Stulpelis atspindi tam tikrą duomenų rūšį lentelėje, pvz. Darbuotojo pareigas visiems darbuotojams. Stulpelių tvarka yra nesvarbi talpinant duomenis .

❹Stulpelis, kurioje nurodomas departamento numeris ir kuris yra antrinis raktas. Antrinis raktas nustato, kaip lentelės surištos viena su kita. Antrinis raktas remiasi pirminiu arba unikaliu raktu kitoje lentelėje. Pvz. DEPTNO unikaliai identifikuoja departamentą DEPT lentelėje.
❺Laukas- yra stulpeliss ir eilutės susikirtimo taške. Jame gali būti nurodyta tik viena reikšmė.
❻Laukas gali neturėti jokios reikšmės. Tai vadinama NULL reikšme. Lentelėje EMP tik darbuotojai, kurie yra pardavėjai, gali turėti lauko COMM ne nulinę reikšmę.

KELIOS SURIŠTOS LENTELĖS

Kiekviena lentelė talpina duomenis, tiktai iš vienos esybės. Pvz. EMP lentelė atitinka informaciją iš esybės EMPLOYES. Kadangi duomenys iš skirtingų esybių yra skirtingose lentelėse, gali prireikti kombinuoti dviejų ar daugiau lentelių informaciją, atsakant į pateiktus klausimus. Pvz.. reikia žinoti ne tik darbuotojo informaciją, bet ir jo departamento lokacijos vietą. Tam reikalinga informacija iš dviejų lentelių: EMP ir DEPT. RDBVS leidžia surišti duomenis iš vienos lentelės su duomenimis iš kitos lentelės per pirminius ir antrinius raktus. Antrinis raktas yra stulpelis, kuri surišta su pirminiu raktu toje pat ar kitoje lentelėje. Galimybė surišti duomenis tarp dviejų lentelių leidžia organizuoti duomenis atskirai valdomose lentelėse.

Reikalavimai pirminiam ir antriniam raktams:
- Pirminiame rakte negali kartotis tos pačios reikšmės
- Pirminis raktas negali būti keičiamas
- Antrinis raktas remiasi duomenų reikšmėmis pirminiame rakte.
- Antrinio rakto reikšmė turi būti poroje su esamu pirminiu raktu ar gali būti unikali ar netgi gali būti NULL.
- Lentelės per pirminį ir antrinį raktus yra surištos logiškai bet ne fiziškai

RELIACINĖ DUOMENŲ BAZIŲ VALDYMO SISTEMA (RDBVS)

Reliacinės DB valdymo sistemos ypatybės leidžia talpinti duomenis ir tvarkyti juos su visais reliacinių DB privalumais ir joms sukurtais PL/SQL programiniais vienetais.

KLIJENTAS/SERVERIS ARCHITEKTŪRA

Klientas/Serveris yra palyginti naujas DB modelis, kuris daugiavartotojiškose duomenų bazėse atlieka darbo paskirstymą tarp keleto kompiuterių. Kliento /serverio pagrindiniai komponentai:
- Serveris,
- Klijentas ir Kliento priedai,
- Programinė įranga, skirta palaikyti ryšį tarp kliento ir serverio.

Serveris valdo visą DB aptarnavimo mechanizmą. Jo funkcijos yra :
1. aptarnauti bent vieną duomenų bazę, su kuria vienu metu dirba keletas vartotojų;
2. reguliuoti priėjimą prie duomenų bazės,

3. organizuoti duomenų apsaugą, taikant įvairias apsaugos formas:
- SQL*Plus Product_User_Profle table;
- Roles ir privilegijas, o taip pat duomenų archyvavimo ir kopijų atstatymo metodus.

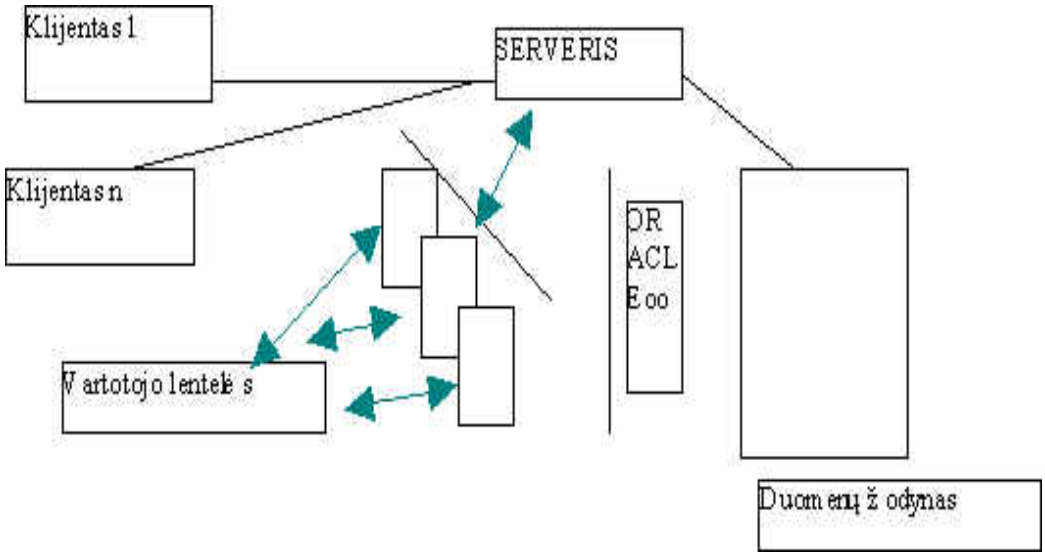
ORACLE - yra vienas iš daugelio klijentas/serveris modelių, efektyviai valdančių savo resursus, informaciją esančią DB ir aptarnaujantis daugybę vartotojų, besikreipiančių į DB su užklausomis, arba atnaujinančių informaciją per tinklą. ORACLE serveris susideda iš komponentų, skirtų darbui su pagrindinėmis reliacinio modelio sritimis:

Duomenų struktūros aprašymas - tai operacinės sistemos failų rinkinys, kuriuose saugoma informacija apie:

-Lentelės sritis. Lentelės sritis yra loginis duomenų bazės dalijimas.
-Lentelės apribojimų sritis
-Priėjimo prie duomenų taisyklės
-Transakcijos

Duomenų tikslingumas-projektavimo metu yra nustatomas tam tikros taisyklės, kurios leidžia įrašyti į lentelę ar į bazę duomenis, atitinkančius tas taisykles. Tikslingumas skirtas tam, kad mes galėtume vienareikšmiškai prieiti prie tų duomenų, kurie mums reikalingi.

Duomenų bazių lentelės -yra standartiniai loginiai saugojimo vienetai. Lentelė-tai masyvas tarpusavyje susijusios informacijos, t.y.duomenų įrašai su vienoda struktūra. Lentelės atributai-stulpeliss, o įrašai lentelėje - yra lentelės eilutės. Kartu su lentelėmis yra aprašoma ir lentelės sritis. Sukurdamas naują lentelę, vartotojas serveriui praneša, kur reikia saugoti vartotojų duomenis, nurodydamas lentelės srities specifikaciją. Administratorius užduoda kiekvienai kuriamai duomenų bazei pradinius failų vardus ir pradinius failų dydžius ir ši informacija yra **SYSTEM-**lentelės pagrindiniai duomenys. Šioje srityje **SYSTEM** saugojamas duomenų žodynas **(DATA DICTIONARY)**. Kitos lentelės sritys gali būti sukuriamos administratoriaus vėliau ir naudojamos įvairių klientų.



Galutinis Oracle sprendimas Pav. 1

Klijentas- sistemos dalis, kuri vykdo kliento priedus ir leidžia prisijungti prie DB, esančios serveryje. Kliento priedai gali būti vykdomi tame pačiame kompiuteryje kur yra serveris, bet taip pat kliento priedai gali būti vykdomi kliento kompiuteryje.

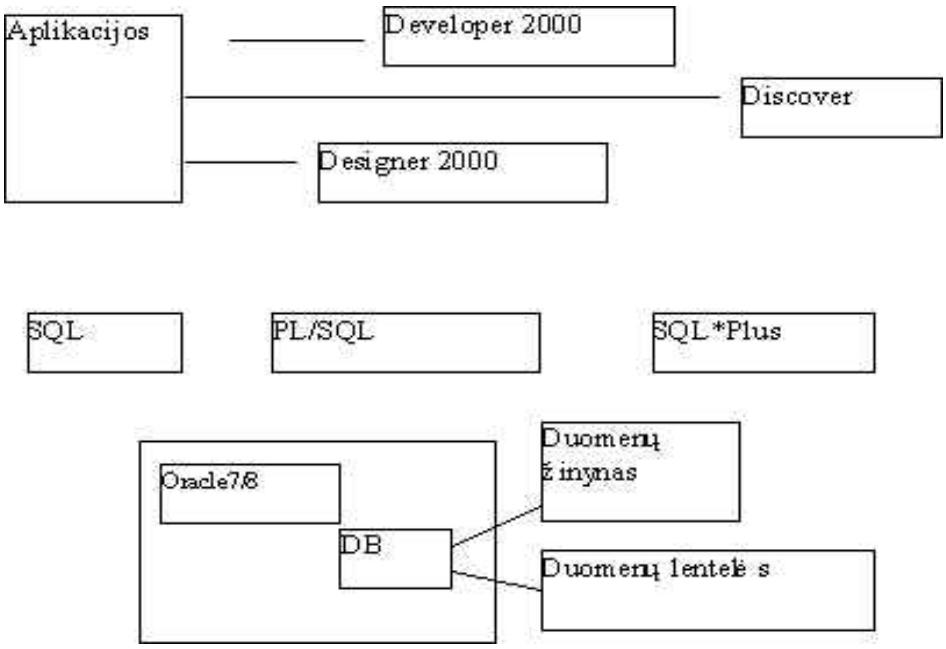
Kliento priedai-išorinė sąsaja- tai yra dalis sistemos, kurią vartotojas naudoja duomenų apdorojimui. Juose taip pat numatyta visa duomenų apdorojimo logika bei duomenų teisingumo analizė. Klijentas siunčia užklausas serveriui ir gauna duomenis iš serverio bei juos analizuoja. Sistemoje klijentas/serveris klijentas dirba ne su visu failu, o tik su atskirais jo elementais, kaip pavyzdžiui lentelės eilutėmis (įrašais).
Pagaliau yra programinė įranga, esanti ir kliente ir serveryje, kuri leidžia palaikyti ryšį tarp kliento ir serverio.

ORACLE DUOMENŲ BAZIŲ PROGRAMINĖ ĮRANGA

1. Vienam vartotojui (personal)
2. Darbo grupei (workgroup)
3. Įmonei ar organizacijai (enterprise).

ORACLE vartotojai skirstomi pagal jų atliekamas funkcijas:

- ORACLE DB administratoriai:
  1. Kuria ir palaiko RDB,
  2. Organizuoja duomenų išsaugojimą ir atstatymą,
  3. Organizuoja naujų vartotojų registravimą ir prijungimą,
  4. Tvarko fizinę duomenų saugojimo erdvę,
  5. Konfigūruoja nacionalinės kalbos aplinką.
- ORACLE aplikacijų kūrėjai ir aptarnautojai,-programuotojai, informacinių sistemų projektuotojai, analitikai - programuotojai.
- Sistemų kūrėjai ir projektuotojai- sistemų analitikai, sistemų administratoriai, aplikacijų projektuotojai.



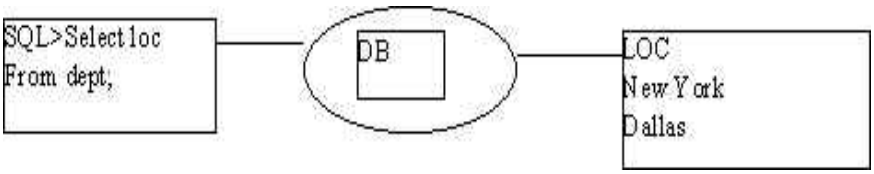
RDB SAVYBĖS:

- Gali būti kuriama ar net modifikuojama naudojant SQL kalbą
- Sudaryta iš lentelių rinkinio, kurios fiziškai tarpusavyje nesusijusios
- Naudoja priskyrimo operatorius

Ryšys su RDB organizuotas naudojant SQL operatorius:

- Įvedamas SQL operatorius
- SQL operatorius siunčiamas DB
- Duomenys iš DB yra parodomi Klijentui

Grafiškai tai atrodytų per pateiktą paveikslą



SQL APIBRĖŽIMAI

Oracle SQL operatoriai atitinka standartus ir pateikti lentelėje 1

Lentelė 1

SELECT	Duomenų pateikimas iš lentelių
INSERT UPDATE DELETE	Manipuliavimas duomenimis (DML)
CREATE ALTER DROP RENAME TRUNCATE	Duomenų apibrėžimo kalba (DDL)
COMMIT ROLBACK SAVEPOINT	Transakcijų kontrolė
GRANT REVOKE	Duomenų kontroliavimo komandos (DCL)

Mokantis dirbsime su trimis lentelėmis: EMP, DEPT, SALGRADE
EMP – lentelė, kurioje saugoma informacija apie darbuotojus
DEPT lentelė kurioje saugoma informacija apie departamentus
SALGRADE lentelė, kurioje saugoma informacija apie atlyginimų paskirstymą į grupes.

EMP - Informacija apie darbuotojus
EMPNO NUMBER(4) PRIMARY KEY,
ENAME VARCHAR2(10),
JOB VARCHAR2(10),
MGR NUMBER(4),
HIREDATE DATE,
SAL NUMBER(7,2),
COMM NUMBER(7,2),
DEPTNO NUMBER(2) NOT NULL, FOREIGN KEY DEPT(DEPTNO);
DEPT- Departamentų pavadinimų lentelė
DEPTNO NUMBER(4) PRMARY KEY,

DNAME VARCHAR2(14),
LOC VARCHAR2(14));

SALGRADE – atlyginimų gradavimo lentelė

GRADE NUMBER,
LOSAL NUMBER(7,2),
HISAL NUMBER(7,2));

STRUKTŪRINĖ UŽKLAUSŲ KALBA(SQL) –(STRUCTURED QUERY LANGUAGE)

SQL kalbos operatoriai

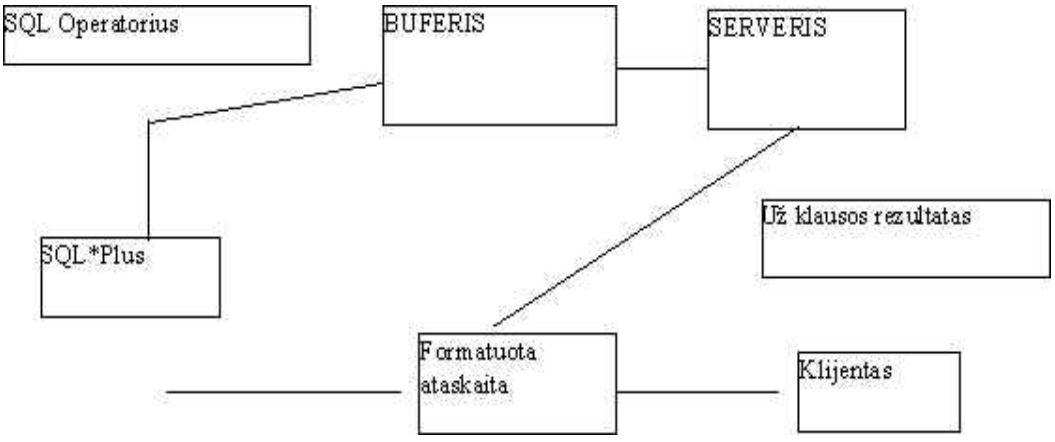
Jie rašomi laikantis taisyklių:

- Operatorių pavadinimai rašomi bet kokiom raidėm.
- Operatoriai sudaryti iš raktinių žodžių ir paragrafų
- Operatoriai gali būti pernešami į kitą eilutę
- Raktiniai žodžiai negali būti padalinti per kelias eilutes, jie turi būti užrašomi nepertraukiami.
- Paragrafai aiškumo dėlei paprastai rašomi skirtingose eilutėse.

SQL operatorių vykdymas:

Operatoriai pateikiami vykdymui SQL*Plus aplinkoje. Visos operatoriaus eilutės yra numeruojamos. Kai sutinkamas kabliataškis, operatorius yra vykdomas.
Jie talpinami buferyje. Tik vienas operatorius vienu metu gali būti buferyje.
Norint vykdyti operatorių iš buferio, jo pabaigoje vietoj kabliataškio pateikiamas simbolis "/"

Operatorių formuojame atsakydami į SQL*Plus paklausimą.



SQL užklausos operatorius SELECT sugeba pateikti vartotojui duomenis iš DB įvairiais metodais:

SELECTION metodas pateikia vartotojui duomenis eilutėmis iš lentelių . Galima naudoti įvairius paieškos kriterijus duomenims išrinkti iš lentelės.
PROJECTION metodas atvaizduoja pasirinktus stulpelius. Taip pat galima pasirinkti įvairius atrankos kriterijus
JOIN metodas pateikia vartotojui duomenis bendrai iš kelių lentelių, pagal sukurtus ryšius tarp tų lentelių.
Ryšiams nustatyti naudojamos tie patys stulpeliai abiejuose lentelėse.
SELECT operatorius atvaizduoja informaciją iš lentelių per pateiktas UŽKLAUSAS (QUERY)
UŽKLAUSA- bendras terminas naudojamas aptarnaujant RDB (reliacines duomenų bazes) per SQL operatorius, ir ji formuojama per prašymų pateikimą serveriui.

Bendra operatoriaus SELECT sintaksė:

SELECT [DISTINCT] {sąrašas norimos pateikti informacijos…}
FROM lentelė(,s)
[WHERE duomenų išrinkimo sąlyga]
[ORDER BY - sąrašas elementų]
[GROUP BY - sąrašas elementų];

SELECT operatoriuje pateikiamas sąrašas norimos atvaizduoti informacijos :
* -visos lentelės stulpeliai . Stulpelių vardai imami iš lentelių struktūros arba išskaičiuojamos išraiškos. Sąraše stulpelių vardai gali eiti kartu su alternatyviai formuojamu stulpelio vardu. Stulpelių vardai sąraše pateikiami per kablelį .
FROM – pateikiamas sąrašas lentelių , iš kur pateiksime informaciją vartotojui.
[**WHERE** duomenų išrinkimo sąlyga]
[**ORDER BY** - sąrašas elementų, pagal ką rūšiuojame duomenis, juos pateikiant užklausos rezultate]
[**GROUP BY** - sąrašas elementų, pagal ką grupuojame duomenis grupinių funkcijų atveju];
Standartiškai operatorius SELECT atvaizduoja visus duomenis iš lentelės, tame tarpe ir besikartojančius, tokia tvarka, kaip jie buvo įvesti į lentelę. Atvaizduojant duomenis iš lentelių, virš stulpelių spausdinami stulpelių pavadinimai iš lentelių struktūros. Yra galimybė pakeisti stulpelių antraštes ir jas suformuoti aiškesnes, skirtingas nuo stulpelių vardų (alternatyvūs). Formuojant naujas antraštes yra taikomos tam tikros taisyklės:

- Naujas pavadinimas rašomas per tarpą už stulpelio vardo.
- Pagal nutylėjimą jie formuojami didžiosiomis raidėmis.
- Jei naujoje antraštėje reikia tarpų ar kokių specialių simbolių, kaip pvz . #, tai visa naujoji antraštė pateikiama tarp dvigubų kabučių .

DISTINCT naudojame kai nenorime užklausoje rodyti lentelėje pasikartojančias reikšmes.

Pavyzdžiai

SQL>SELECT Deptno FROM Emp;

Išveda departamentų numerius iš lentelės EMP. Yra darbuotojų po kelis kiekviename departamente, tad rezultate matysime visus įrašus (eilutes), esančius lentelėje, bet pateikus užklausą kitu būdu jau matysime tik po vieną tokį departamento numerį.

- SQL>SELECT DISTINCT Deptno FROM Emp;

- SELECT * FROM Dept; - Išveda visas eilutes su visais stulpeliais iš lentelės Dept. "*" nurodo, jog reikia parodyti visus stulpelius. Nėra užduotos jokios sąlygos, vadinasi reikia parodyti visas eilutes.

- SELECT Loc,Deptno FROM Dept;

Išveda tik dviejų stulpelių reikšmes iš lentelės Dept taip pat be jokių atrankos sąlygų.

Išvedimo metu yra taikomos tam tikros išdėstymo taisyklės:

Viršuje spausdinami stulpelių vardai, o po jų eina stulpelių reikšmės. Tekstas ir data stulpelyje išlyginami pagal kairį kraštą, o skaičiai pagal dešinį kaštą.

SELECT operatoriaus pagalba galima atvaizduoti duomenis vartotojui reikiama tvarka, atliekant visus Reliacinės Algebros veiksmus, jei jie yra reikalingi.
Gali būti atvaizduotos išskaičiuotos reikšmės. Tam naudojami aritmetiniai operatoriai
+ sudėtis,
- atimtis,
/ dalyba,
* daugyba.

Jei naudojame keletą aritmetinių veiksmų, tai pirmiausia atliekami daugybos ir dalybos veiksmai. Jei yra vieno lygio operatoriai, tai veiksmai atliekami iš kairės į dešinę eilės tvarka. Veiksmų sekai pakeisti gali būti naudojami skliaustai. Tada veiksmai pirmiausia atliekami skliaustuose.

- SELECT Ename,Sal,Sal+300 FROM Emp;

Išveda informaciją iš lentelės Emp dviejų nurodytų stulpelių, ir formuoja trečią stulpelį, atatinkamai išskaičiuodama jo reikšmes. Stulpelio pavadinimas sutaps su išskaičiuojama išraiška.
Kalbant apie atvaizduojamus duomenis dar kartą grįšime prie NULL reikšmės lauke. NULL yra reikšmė, kuri yra netinkama, neatvaizduojama, nežinoma ar nepritaikoma. Tai ne tas pats kas nulis arba tarpas.

- SELECT Ename,Job,Comm FROM Emp;

Jei eilutėje trūksta duomenų kuriame nors stulpelyje, jo reikšmė lauke vadinama NULL. Kai kurių tipų stulpeliai negali būti NULL Tai Pirminiai raktai. Jų tipas lentelių kūrimo metu nustatomas pagal nutylėjimą kaip NOT NULL.
Comm- Šis stulpelis gali būti NULL lentelėje Emp.

Jei stulpelio reikšmė aritmetinėje išraiškoje yra NULL, tai rezultatas yra formuojamas NULL, nepriklausomai nuo kitų išraiškos elementų. Pvz. Dalyba iš nulio duos klaidą. Tačiau jei dalinsite iš stulpelio, kurio reikšmė yra NULL, rezultatas bus nenuspėjamas.

- SELECT Ename NAME,12*Sal+Comm FROM EMP WHERE Ename='KING';

Rezultate gausime tik vardą, nes antras stulpelis yra išskaičiuojama reikšmė, tačiau ten yra stulpelis, kuris turi NULL reikšmę (konkrečiai šiam darbuotojui) ir visas reiškinys tampa NULL. NAME šiuo atveju yra alternatyvus stulpelio vardas.

Duomenų išrinkimui pagal užduotą sąlygą formuojama užklausa, naudojant paragrafą **WHERE**. Sąlygoje (arba gali būti užduotos kelios sąlygos) gali būti nurodyti stulpelių vardai, kuriems yra taikoma sąlyga ir ši sąlyga yra tikrinama kiekvienai lentelės eilutei, formuojant užklausos rezultatą.
Sąlyga gali būti pateikiama kaip aritmetinė išraiška ar kaip simbolių eilutė, surišta per lyginimo operatorius =,>,<,>=,<=,<>.

***WHERE** stulpelio vardas; sąlyga; kito stulpelio vardas arba konstanta ar reikšmių sąrašas taip formuojama sąlyga duomenų atrinkimui.*

- SELECT Ename,Job,Deptno FROM EMP **WHERE Job='CLERK';**

pateiks rezultate tik darbuotojus su nurodyta specialybe. Sąlygoje svarbu pateikti lyginimui skirtus duomenis reikiamam formate (dešinėj pusėj raidžių dydis, datos formatas ir kt.)

- SELECT Ename, Job,Deptno
  FROM EmpP
  WHERE Hiredate='01-JAN-95';

- SELECT * FROM Emp **WHERE Sal>=1500**;

- SELECT Ename,Sal,Comm FROM Emp
  WHERE Sal<=Comm; sąlygoje dalyvauja du stulpeliai, bet nenurodyta konkreti reikšmė.

Simbolinės eilutės arba data sąlygoje turi būti pateikta tarp viengubų kabučių.

Paragrafe WHERE gali būti formuojama sąlyga, naudojant sąlygos pateikimo būdus:

- **BETWEEN ... AND** -kurie nurodo žemiausios ir aukščiausios reikšmių intervalą, išrenkant duomenis iš lentelės, t.y. rezultate pateikia duomenis tam tikram reikšmių intervalui.
- **IN** (sąrašas reikšmių, kurios turi būti pateiktos užklausos rezultate, pateikiamos per kablelį)
- **LIKE** (šablonas duomenų išrinkimui iš lentelės)

- **IS NULL** - (nurodo, jog vartotoją domina tik eilutės lentelėje su neapibrėžta lauko reikšme)

❖ SELECT Ename,Sal
FROM EMP
WHERE Sal BETWEEN 1000 AND 1500;

Pateikia darbuotojus, kurių alga yra intervale tarp 1000 ir 1500.
BETWEEN ..AND operatorius apima ir kraštutines reikšmes. Mažesnioji reikšmė turi būti nurodyta pirma, didesnioji - antra.

❖ SELECT Empno,Ename,Sal,Mgr
FROM EMP
WHERE Mgr IN(7902,7566); pateiks sąraše tik du darbuotojus. Šią sąlygą galima būtų pateikti ir kitaip.

❖ SELECT *
FROM EMP
WHERE Deptno IN(10,20); Išves darbuotojus, kurie dirba 10 ir 20 departamentuose.

LIKE galima naudoti šabloną pagal kurį atrenkami duomenys iš lentelės. Sąlygoje gali būti pateikiami tik simboliniai ar skaitmeniniai duomenys. Mažiausiai turi būti du simboliai formuojant šabloną.
% - reiškia nulį arba keletą simbolių
_ reiškia vieną simbolį

❖ SELECT Ename
FROM EMP
WHERE Ename LIKE 'S%'; parodo visus darbuotojų vardus, kurių pavardė prasideda "S" raide, bet neparodys, kurių vardai prasidės "s" raide(mažąja). LIKE operatorius gali būti kaip trumpinys.

❖ SELECT Ename,Hiredate
FROM EMP
WHERE Hiredate LIKE '% 81'; - Visi kurie pradėjo dirbti 81 metais.

Galima įvairiai konstruoti šabloną

❖ SELECT Ename
FROM EMP
WHERE Ename LIKE '_A%'; Parodys visus darbuotojus, kurių varde bus raidė "A"

NULL paragrafas gali būti naudojamas norint pateikti duomenis su NULL reikšme.

- SELECT Ename,Mgr
  FROM EMP
  WHERE Mgr IS NULL; pateiks duomenis, kur Mgr neužpildytas. Tai būtų darbuotojas "KING".

IS NULL tikrina nurodyto stulpelio reikšmę.

- SELECT Ename,Job,Comm
  FROM EMP
  WHERE Comm IS NULL; parodo visus darbuotojus, kurie negauna komisinio atlyginimo.

Sudėtinga sąlyga gali būti pateikta naudojant **AND arba OR**. Tokiu būdu galima pateikti kelias sąlygas duomenų išrinkimui.

- SELECT Empno,Ename,Job,Sal
  FROM EMP
  WHERE Sal>=1100 AND Job='CLERK';

Rezultate pasirodys visi darbuotojai, kurie užima nurodytas pareigas ir jų alga viršija 1100.

- SELECT Empno,Ename,Job,Sal
  FROM EMP
  WHERE Sal>1100 OR Job='CLERK';

Rezultate pasirodys visi darbuotojai, kurie užima nurodytas pareigas arba jų alga viršija 1100.

Su kiekvienu sąlygos paragrafu gali būti naudojamas ir paragrafas **NOT**, kuris reiškia sąlygos neigimą.

- SELECT *
  FROM EMP
  WHERE Comm IS NOT NULL; Rezultate parodo visus darbuotojus, kurie gauna komisinį atlyginimą.

… **WHERE NOT Job IN ("CLERK','MANAGER')**
… WHERE Sal NOT BETWEEN 1000 AND 1500

Pirmenybės, atliekant lyginimo operaciją, pateiktos žemiau lentelėje 2.

Lentelė 2

Eilės Tvarka	Operatorius
1	Visi lyginimo operatoriai
2	NOT

3	AND
4	OR

- SELECT Ename,Job,Sal
  FROM EMP
  WHERE job='SALESMAN' OR job='PRESIDENT' AND sal>1500;

Dvi sąlygos nurodant pareigas su atlyginimu, kai pareigos pateikiamos taip pat per sąlygą OR (arba) t.y. kai pareigos gali turėti skirtingas reikšmes, bet abiem atvejais atlyginimui keliami tie patys reikalavimai.

- SELECT Ename,Job,Sal
  FROM EMP
  WHERE (job='SALESMAN' OR job='PRESIDENT') AND sal>1500;

Irgi dvi sąlygos, bet tikrinama pirma pareigybes ir jei yra viena iš nurodytų, toliau tikrinamas atlyginimas.

Normaliai išvedant duomenis iš lentelės į ekraną, duomenų tvarka yra tokia, kaip jie buvo suvesti į lentelę. Norint pakeisti duomenų atvaizdavimo tvarką, operatoriuje SELECT galima naudoti paragrafą **ORDER BY**, kuriame nustatome, kaip turi būti pateikti duomenys užklausos rezultate. Standartiniu būdu, naudojant ORDER BY, duomenų vaizdavimas vyksta pagal alfabetą nuo A iki Z arba skaičių didėjimo tvarka. Norint pakeisti duomenų atvaizdavimo tvarką atvirkščiai, prie paragrafo ORDER BY naudojame papildomą žodį **DESC**. Rūšiuoti duomenis galima pagal kelis stulpelius ir pagal kiekvieną iš jų skirtinga tvarka.
Parametrą rūšiavimui galima taip pat dirbtinai sukurti, tačiau jį nebūtina atvaizduoti užklausos rezultate.
Jei naudojame paragrafą ORDER BY, jis turi būti nurodomas operatoriaus SELECT gale.

- SELECT *
  FROM EMP
  ORDER BY Hiredate; rūsiuoja pagal priėmimo į darbą datą

- SELECT *
  FROM EMP
  ORDER BY Hiredate DESC; rūšiouja tąip pat pagal datą, bet atvirkštine tvarka,

T.y. vėliuasia priimti į darbą pasirodys užklausos rezultate.

FUNKCIJOS VIENAI EILUTEI

Funkcijos yra galingas įrankis ir gali būti naudojamos:
- -atlikti skaičiavimus su duomenimis lentelėse;
- -modifikuoti duomenis;
- -sugrupuoti duomenis išvedimo metu;
- -koreguoti duomenų formatą jų išvedimo metu;
- -konvertuoti stulpelių duomenų tipą.

Yra keletas skirtingų funkcijų tipų. Vienas iš jų yra funkcijos vienai eilutei. Jos veikia kiekvieną eilutę ir grąžina rezultatą atskirai kiekvienai eilutei. Kita funkcijų grupė veikia visą įrašų grupę ir pateikia vieną rezultatą kiekvienai grupei.

Funkcijos vienai eilutei yra:
- Simbolinės
- Skaitmeninės
- Datos
- Duomenų konvertavimo.

Funkcijos vienai eilutei dirba su duomenimis, priimdamos jom skirtus argumentus ir grąžindamos reikšmę. Veikia kiekvieną eilutę, gali keisti duomenų tipą ir gali būti nutrauktos vykdymo metu. Bendra sintaksė taikant funkcijas pateikiamoms užklausoms yra:

Funkcijos_vardas {stulpelis|išraiška,[arg1,arg2]}

Vienos eilutės funkcijos naudojamos duomenų laukų pertvarkymui, pateikiant duomenis užklausos rezultate. Jos naudoja vieną arba keletą argumentų ir užklausoje grąžina vieną reikšmę kiekvienai eilutei.Gali būti naudojami keli argumentai. Funkcijų argumentais gali būti - konstanta, kintamojo reikšmė, stulpelio vardas, bet kokia išraiška. Funkcijos gali būti naudojamos su operatoriumi SELECT ir eina kartu su paragrafais WHERE arba ORDER BY.

Simbolinės funkcijos priima simbolinius duomenis ir gali grąžinti arba simbolinius arba skaitmeninius duomenis.
Skaitmeninės funkcijos priima skaitmeninius duomenis ir taip pat grąžina skaitmeninis duomenis.
Datos tipo funkcijos veikia datos tipo reikšmes. Visos datos tipo funkcijos grąžina duomenų reikšmes kaip datos tipo, išskyrus MONTHS_BETWEEN funkciją, kuri grąžina skaitmeninę reikšmę.
Konvertavimo funkcija priima bet kokio tipo duomenis ir juos konvertuoja pagal funkcijos veikimo kryptį.

Bendros funkcijos tai yra: NVL ir DECODE

Simbolinės Funkcijos (CHARACTER).

Tokioms funkcijoms reikia vieno ar kelių argumentų ir rezultatas yra formuojamas kaip įprasta kiekvienai eilutei. Panagrinėsime kai kurias simbolines funkcijas, kurios dažnai naudojamos pateikiant duomenis užklausų rezultate arba juos atrenkant iš lentelių.

LOWER(stulpelis|reikšmė) - Bet kokiam formate parašytus simbolius rezultate pateikia mažosiomis raidėmis. Labai naudinga nurodant sąlygą užklausoje kai reikia suvienodinti informacijos formatą duomenų atrinkimo metu. Be to ji naudinga aplikacijose pateikiant informaciją vienodam pavidale, jei ji yra suvesta skirtingam formate.
UPPER(stulpelisėlė|reikšmė)- Bet kokiam formate parašytus simbolius rezultate pateikia didžiosiomis raidėmis. Labai naudinga nurodant sąlygą užklausoje kai reikia suvienodinti informacijos formatą duomenų atrinkimo metu. Be to ji naudinga aplikacijose pateikiant informaciją vienodam pavidale, jei ji yra suvesta skirtingam formate.

INITCAP(stulpelis|reikšmė)- Bet kokiam formate parašytus simbolius rezultate išveda pirmą simbolį didžiąja raide, o kitus mažosiomis raidėmis . Labai naudinga nurodant sąlygą užklausoje kai reikia suvienodinti informacijos formatą duomenų atrinkimo metu. Be to ji naudinga aplikacijose pateikiant informaciją vienodam pavidale, jei ji yra suvesta skirtingam formate.

Kita grupė yra duomenų manipuliavimo funkcijos:

CONCAT(stulpelis1|reikšmė1,stulpelis2|reikšmė2)- -Apjungia pirmą simbolinę reikšmę su antra simboline reikšme. Ekvivalenti operatoriui ||. Pvz.

- SELECT CONCAT(ename,job) JOB FROM Emp WHERE Empno=7900;

LPAD(stulpelis|reikšmė,n,'eilutė') -Prideda tiek trūkstamų (laisvai pasirinktų) simbolių iš kairės, kad sulygintų rezultatą iki nustatyto pločio ir išlygintų pagal dešinį kraštą.

- stulpelis |reikšmė - objektas, kurį reikia sutvarkyti
- n - reikiamas simbolių kiekis užklausos rezultate (plotis)
- 'eilutė'- užpildymo simboliai. Jei ji praleista, užpildoma tarpais.

RPAD(stulpelis|reikšmė,n,'eilutė') -Prideda tiek trūkstamų simbolių iš dešinės, kad sulygintų iki nustatyto pločio. Galima naudoti dėl teksto išlyginimo iš kairės.

- stulpelis|reikšmė - objektas, kurį reikia sutvarkyti
- n - reikiamas simbolių kiekis užklausos rezultate
- 'eilutė'- užpildymo simboliai. Jei tai praleista, užpildo tarpais.

Funkciją galima taip pat naudoti, norint susiaurinti stulpelį išvedimo metu.

SUBSTR(stulpelis|'simbolių_eilutė',pozicija,n)- rezultate pateikia n simbolių eilutę, sudarytą iš simbolių, kuriuos paima nuo nurodytos pozicijos iš stulpelio arba pateiktos simbolių eilutės. Jei **n** nenurodyta, tai rezultate pateikiami simboliai iki galo, pradedant nurodyta pozicija.

INSTR(stulpelis|reikšmė,'eilutė')- nustato pasirinktos simbolių eilutės pirmojo simbolio poziciją stulpelyje arba eilutėje.

LTRIM(stulpelis|reikšmė,'eilutė/s') - nuima iš kairės pusės nurodytus vedančius simbolius. Jei eilutė nenurodyta, tai pašalina vedančius tarpus.

RTRIM(stulpelis|reikšmė,'eilutė/s') - nuima iš dešinės pusės nurodytus vedančius simbolius. Jei 'eilutė/s' nenurodyta, tai pašalina vedančius tarpus.

SOUNDEX(stulpelis|reikšmė) - suranda panašiai skambančias eilutės dalis.

LENGTH(stulpelis|reikšmė) - nustato simbolių skaičių stulpelyje arba reikšmėje.

TRANSLATE(stulpelis|reikšmė,iš,į) - nurodo kad galima pakeisti simbolius iš vieno į kitą išvedimo metu.

REPLACE(stulpelis|reikšmė,eilutė,keičiama_eilutė)

- SELECT Ename,CONCAT(Ename,job),LENGTH(Ename),INSTR(Ename,'A')
  FROM Emp
  WHERE SUBSTR(Job,1,5)='SALES'

Pateiks rezultate darbuotojo vardą, taip pat vardą sujungtą su pareigom kaip vieną žodį, vardo simbolių skaičių bei simbolio A pozicijos numerį varde tik tiems darbuotojams kurių darbo stulpelio penki pirmieji simboliai yra nurodyti tarp kabučių paragrafe WHERE.

Skaitmeninės Funkcijos (NUMBER)

Šios funkcijos skirtos darbui su skaičiais. Jos orientuotos į skaičius ir rezultatą grąžina taip pat skaičių pavidale. Tai yra:

ROUND(stulpelis|išraiška,n)- Užklausos rezultate suapvalina skaičių iki nurodytų n pozicijų. Jei n - neigiamas skaičius, tai rezultatas yra suapvalinamas iki sveiko skaičiaus.

TRUNC(stulpelis|išraiška,n) -Užklausos rezultate atmeta nereikalingą informaciją iki reikiamų nurodytų n pozicijų. Jei n yra neigiamas skaičius, tai iš kairės pusės esantys skaičiai paverčiami nuliu.

SIGN(stulpelis|išraiška)- tikrina ženklą. Jei neigiamas skaičius, grąžina -1, jei nulis -0, jei teigiamas skaičius - grąžina +1.

CEIL(stulpelis|išraiška) - suranda didžiausią sveiką skaičių didesnį arba lygų nurodytai reikšmei.

FLOOR(stulpelis|reikšmė) - suranda mažiausią sveiką skaičių mažesnį arba lygų nurodytai reikšmei.

POWER(stulpelis|išraiška,n) - kėlimas n-tuoju laipsniu.

SQRT(stulpelis|reikšmė) - kvadratinės šaknies traukimas. Jei reikšmė yra NULL ar neigiama ta grąžinama NULL reikšmė.

ABS(stulpelis|reikšmė)- grąžina absoliutinę reikšmę.

MOD(pirma_reikšmė,antra_reikšmė)- grąžina liekaną, kuri gaunama padalinus pirmą_reikšmę iš antros_reikšmės.

Matematinės Funkcijos

LOG(m,n)- grąžina logaritmo reikšmę

SIN(n) -grąžina sinuso reikšmę
SINH(n)

TAN(n) - grąžina tangento reikmę
TANH(n)

COS(n)- grąžina kosinuso reikšmę
COSH(n)

EXP(n)

Datos Tipo Funkcijos (DATE)

Funkcijos skirtos darbui su datomis. Sistemoje data yra saugoma kaip skaičius, todėl jų visų darbo rezultatas taip pat yra skaičius. Pagal nutylėjimą data suvedama ir atvaizduojama DD-MON-YY formate.

SYSDATE - pseudostulpelis, iš kurios galima paimti ir atvaizduoti užklausos rezultate einamąją datą. Ši data yra lentelėje SYS.DUAL ir yra pasiekiama bet kuriam vartotojui, nors jos savininkas yra vartotojas SYS, todėl reikia pilnai nurodyti paieškos kelią t.y. užklausoje reikia rašyti - SELECT SYSDATE FROM SYS.DUAL. Kadangi data yra saugoma kaip skaičiai, todėl su DATE tipo duomenimis galima atlikti aritmetinius veiksmus:

DATE+number - rezultatas yra DATE -(number suprantamas kaip dienų skaičius)
DATE - number - rezultatas yra DATE -(number suprantamas kaip dienų skaičius)
DATE - DATE -- atima vieną datą iš kitos rezultatas yra dienų skaičius tarp nurodytų datų
DATE+number/24 - prideda valandų skaičių prie datos rezultatas yra DATE .

- SELECT HIREDATE, HIREDATE+7, HIREDATE-7, SYSDATE-HIREDATE
  FROM EMP;

Datos tipo funkcijas galima priskirti prie duomenų konvertavimo funkcijų.
Datos tipo funkcijos:

MONTHS_BETWEEN(DATE1,DATE2)- nustato mėnesių skaičių tarp dviejų nurodytų datų. Rezultatas yra teigiamas, jei DATE1 yra vėlesnė nei DATE2. Rezultatas yra neigiamas, jei DATE1 yra ankstesnė nei DATE2. Trupmeninė rezultato dalis yra mėnesio dalis.

ADD_MONTHS (DATE1,n)- prie datos prideda n kalendorinių mėnesių. n turi būti sveikas skaičius, bet ženklas nesvarbu.

NEXT-DAY (DATE1,'simb') -nustato artimiausios nurodytos savaitės dienos datą. "simb' gali būti skaičius nustatantis dieną arba simbolių eilutė.

LAST-DAY(DATE1)- nustato paskutinę mėnesio dieną nurodytai DATA1.

ROUND(DATA1,'FMT')- Grąžina datą suapvalintą iki nustatytos pagal 'FMT' modelį. Jei 'FMT' praleistas, grąžinama suapvalinta data iki artimiausios datos.

ROUND(DATA1 'MONTH")- nustato DATA1 iki mėnesio tikslumo.

TRUNCATE(DATA1,'char')

MONTHS_BETWEEN('01-SEP-95','11-JAN-94')	19.6774194
ADD_MONTHS('11-JAN-94',6 '	11-JUL-94
NEXT_DAY('01-SEP-95''FRIDAY')	08-SEP-95
LAST_DAY('01-SEP-95')	30-SEP-95
ROUND('25-JUL-95','MONTH')	01-AUG-95
ROUND('25-JUL-95','YEAR')	01-JAN-96
TRUNC('25-JUL-95','MONTH')	01-JUL-95
TRUNC('25-JUL-95','YEAR')	01-JAN-95

- SELECT
  Empno,Hiredate,ROUND(Hiredate,'MONTH',TRUNC(Hiredate,'MONTH')
  FROM Emp
  WHERE Hiredate like '%87';

DUOMENŲ KONVERTAVIMO FUNKCIJOS

Jos leidžia konvertuoti duomenis iš vieno formato į kitą. SQL siūlo tris duomenų konvertavimo funkcijas:

- TO_CHAR -perveda duomenis iš NUMBER tipo arba DATE tipo duomenų į simbolių eilutę.
- TO_NUMBER - kai simbolių eilutėje pateiktus skaičius perveda į skaičių formatą. Simbolių eilutė sudaryta iš skaičių, bet tik jie aprašyti kaip simboliai (tarp kabučių).
- TO_DATE - datą, aprašytą kaip simbolių eilutė, perveda į datą pagal užduotą ***FMT*** formatą. Jei formatas nenurodytas, tai bus formatuojama standartiškai DD-MON-YY.

TO_CHAR(DATA, 'datos šablonas')- nurodo, kad užduota data bus konvertuojama į formatą, tinkamą išvedimui ir pateikiama pagal nurodytą datos šablono formatą.

❖ SELECT TO_CHAR(SYSDATE,'DAY, DDTH MONTH YYYY') FROM SYS.DUAL;

Rezultate matysime TEUSDAY , 07TH MAY 1995

Konvertavimo metu yra taikomos konvertavimo taisyklės:

- Datos šablonas užduodamas tarp viengubų kabučių.
- Šablone gali būti naudojami visi datos formato elementai.
- Stulpelio (datos) vardas nuo šablono atskirtas kableliu.
- Mėnesio dienoms užklausos rezultate pagal nutylėjimą išskiriamos 9 pozicijos. Iki trūkstamo formato yra užpildoma tarpais . Norint pašalinti nereikalingus susidariusius tarpus reikia taikyti išvedimo formatą. Formatą taip pat naudojame keičiant simbolių dydį.

FMT- formatas, naudojamas išvedimo formatui pakeisti. Tą pačią funkciją galima taikyti ir norint išvesti laiką iš datos.

❖ SELECT TO_CHAR(SYSDATE, 'HH:MM:SS')
FROM SYS.DUAL;

SYS.DUAL sisteminiai kintamieji:
DAY - |
DDTH -|
MONTH | pseudostulpelio, naudojamos šablonui užduoti.
YYYY - |
FMT – formatas taip pat naudojamas, kai norime atmesti tarpus ir nereikšminius nulius.

❖ SELECT Ename, TO_CHAR(Hiredate, 'FMDD MONTH YYYY') HIREDATE
FROM EMP;

- SELECT TO_CHAR(SYSDATE, ‘FMDay,ddth Month YYYY’)
  FROM SYS.DUAL;

Rezultate matysime savaitės dienos pavadinimą be nereikšminių tarpų, už kablelio bus parodyta mėnesio diena, mėnesio vardas ir metai iš pseudostulpelio SYSDATE, kuriame yra saugoma einamoji data. Ta pati funkcija naudojama taip pat ir dėl skaičių pervedimo į simbolinį formatą.

TO_CHAR(Number,’FMT’)

- SELECT TO_CHAR(SAL,’$9,999’)
  FROM EMP;

Galiojančius datos formato elementus galima rasti lentelėje naudojant Help sistemą. Skaičių formatai pateikti lentelėje 3 apačioje.

Lentelė 3

9	Atvaizduoja skaičių
0	Reiškia, kad nulis turi būti rodomas
$	Parodo rezultate dolerio ženklą, kurio vieta priklauso nuo rezultato ilgio
L	Naudojamas vietos valiutai irgi kad jos vieta taptų lanksčiai kintama
.	Spausdina dešimtainį tašką
,	Spausdina skirtuką tarp tūkstančių

TO_NUMBER (stulpelis/reikšmė) - perveda skaičius, pateiktus simboliniame formate, į skaičių formatą. Tai yra labai svarbu lyginimo operacijose.

- SELECT EMPNO,ENAME,JOB,SAL
  FROM EMP
  WHERE Sal>TO_NUMBER(‘1500’);

Skaičiaus formato elementus matome pateiktoje lentelėje 3.

TO_DATE(‘SIMBOLINĖ DATA,’FMT’’) - iš simbolinės eilutės datą perveda į datos formatą. Tai svarbu, kai norime atlikti veiksmus su datom, atliekant paiešką, ypač jei data užduota nestandartiniu būdu.

- SELECT ENAME,HIREDATE
  FROM Emp
  WHERE HIREDATE=TO_DATE(‘JUNE 4, 1984’, ‘MONTH DD,YYYY’);

Bendrosios Funkcijos

Yra funkcijų, kurios gali būti taikomos bet kokio tipo duomenims.

- **NVL funkcija**

Jei išskaičiuojamoj išraiškoj yra bent vienas neapibrėžtos reikšmės elementas, tai rezultatas formuojamas taip pat NULL (neapibrėžtas). Kad taip neatsitiktų, galime išskaičiuojant rezultatą naudoti funkciją NVL, skirtą pervesti neapibrėžtą reikšmę į nurodytą (dažniausiai nulį).

NVL(stulpelis/išraiška,reikšmė)
NVL - perverčia NULL reikšmę, sutinkamam stulpelyje ar išraiškoj, į nurodytą reikšmę. Duomenų tipai turi sutapti.

❖ SELECT SAL*2+**NVL(COMM,0)**
FROM Emp;

❖ SELECT **NVL(HIREDATE,'01-JUN-96')**
FROM Emp;

❖ SELECT NVL(JOB,'NEAIŠKIOS-PAREIGOS')
FROM Emp;

- **GREATEST Funkcija**

GREATEST(stulpelis|išraiška1,stulpelis|išraiška2) - nustato kuri iš pateiktų išraiškų yra didesnė. Prieš lyginimą, duomenų formatai suvienodinami.

- **LEAST Funkcija**

LEAST(stulpelis|išraiška1,stulpelis|išraiška2) - nustato kuri iš pateiktų išraiškų yra mažesnė. Prieš lyginimą, duomenų formatai suvienodinami.

- **DECODE** funkcija yra labai galinga priemonė, formuojant rezultatą užklausos metu.

DECODE (stulpelis/išraiška, paiešk1,rezult1,[paiešk2,rezult2,.....][default];
stulpelis/išraiška- gali būti bet kokio tipo duomenys
paiešk1, paiešk2 ...- turi būti tokio paties tipo, kaip ir stulpelis/išraiška
rezult1,rezult2 ..- turi būti tokio tipo, kaip ir paiešk1,paiešk2.
default - nebūtinas parametras, bet jis nurodo, kad rezultatas bus formuojamas pagal nutylėjimą, jei paieška buvo nesėkminga.
DECODE funkcija atlieka užklausą pagal nuorodą **paiešk1** ir, jei ši paieška sėkminga, tai rezultate formuoja naują reikšmę, užduotą **rezult1**. Jei tokia paieška nesėkminga, rezultate formuoja pagal nutylėjimą reikšmę iš lentelės, o jei tai nenurodyta, grąžina reikšmę NULL.

- SELECT Job, Sal,
  DECODE(Job, ‘ANALYST’, Sal*1.1,‘CLERK’, Sal*1.15,‘MANAGER’, Sal*1.20,
  Sal) DECODED_JOB FROM Emp;

Rezultate, jei tik lentelės Emp stulpelyje JOB pasitaikys reikšmė ‘ANALYST’ tai skaičiuos vienaip algą, dar kitaip dėl ‘CLERK’ ir dar kitaip dėl ‘MANAGER’. Visiems kitiems paliks tokią algą, kuri buvo lentelėje.

Įdėtinės Funkcijos

Funkcijos vienai eilutei gali būti įdėtos per kelis lygius. Jos yra vykdomos nuo giliausio lygio iki paskutinio lygio.

F3(F2(F1(col,arg1),arg2),arg3)

Pirmas žingsnis yra vykdant F1 funkciją su jos argumentais ir rezultatas1 yra naudojamas kaip vienas iš argumentų funkcijai F2.
F2 funkcija formuoja reultatą2, kuris yra kaip vienas iš argumentų funkcijai F3.
Galutinis rezultatas yra rezultatas3

- SELECT Ename,NVL(TO_CHAR(Mgr),’No Manager’)
  FROM Emp
  WHERE Mgr IS NULL;

Pirmiausia skaitmeninį stulpelį paverčia simboline eilute. Po to gautą neapibrėžtą reikšmę pakeičia tekstu.

DARBAS SU KELIOM LENTELĖM. DUOMENŲ IŠRINKIMAS IŠ KELIŲ LENTELIŲ.

Kartais reikia parodyti duomenis vienoje užklausoje iš kelių lentelių. Eilutės iš vienos lentelės su eilutėm iš kitos lentelės yra sujungiamos per bendrus duomenis, esančius PRIMARY KEY ir FOREIGN KEY stulpeliuose. Tam yra taip vadinamas lentelių duomenų jungimas. Atliekama užklausa kaip paprastai per SELECT operatorių. Bet kada galima nurodyti apjungimo sąlygą per paragrafą WHERE, jei tik jūs neplanuojate pateikti visų eilučių sujungimo vienoje lentelėje su visomis eilutėmis kitoje lentelėje.

SELECT lentele1.stulpelis,lentele2.stulpelis
FROM lentelė1,lentelė2
WHERE lentelė1.stulpelis=lentelė2.stulpelis.
Jei yra tokie patys stulpelių vardai abiejuose lentelėse, prieš stulpelio vardą reikia nurodyti lentelės vardą.
Eilutės iš vienos lentelės sujungiamos su eilutėmis iš kitos lentelės. Kai apjungimo sąlyga yra neteisinga ar nėra galutinai apibrėžta, rezultatas yra PROODUCT tipo apjungimas ir visos lentelių apjungimo kombinacijos yra pateikiamos užklausoje. Visos eilutės pirmoje lentelėje yra sujungiamos su visomis eilutėmis kitoje lentelėje. Rezultate gauname

didžiulį kiekį eilučių ir dažnai tai yra nesąmoningas rezultatas. Kad išvengti tokių nesąmoningų rezultatų yra rašoma apjungimo sąlyga. Apjungimo tipai skirstomi:

- per lygybę
- per nelygybę

Gali būti ir kiti apjungimo metodai:

- išorinis
- automatinis
- priskyrimo.

Apjungiant dvi lenteles nurodoma viena sąlyga, apjungiant tris lenteles reikia konstruoti dvi apjungimo sąlygas, o apjungiant keturias lenteles, būtina konstruoti tris sąlygas. Ši apjungimo taisyklė gali būti pavaizduota per formulę:
#lentelė-1=minimalus lentelių apjungimo sąlygų skaičius.
#- apjungiamų lentelių kiekis. Ši sąlyga nereikšminga, jei lentelės sujungtos per PRIMARY KEY ir FOREIGN KEY.
Norint nustatyti darbuotojo departamento pavadinimą, jūs lyginate lentelės Emp stulpelio DEPTNO reikšmes su reikšme stulpelyje DEPTNO lentelėje Dept. Ryšys tarp Emp ir Dept yra lygybės-sąlyga per stulpelį DEPTNO, kai nustatytas pilnas atitikimas abiejuose lentelėse, tada eilutė pasirodo užklausos rezultate. Tai yra jungimas per lygybę.

❖ SELECT ENAME,JOB,DNAME,LOC
FROM EMP,DEPT
WHERE EMP.DEPTNO=DEPT.DEPTNO;

Norint patikslinti stulpelį, jei ji turi tokį patį pavadinimą skirtingose lentelėse, reikia prieš ją parašyti lentelės vardą, nuo stulpelio vardo atskiriant jį tašku.

❖ SELECT ENAME,JOB,DNAME,LOC
FROM Emp,Dept
WHERE EMP.DEPTNO=DEPT.DEPTNO
ORDER BY DEPT.DEPTNO;

Nelygybės sąlyga naudojama, kai nėra tiesioginio ryšio tarp lentelių. Darbuotojo alga yra tarp mažiausios ir aukščiausios ribos ir nėra tiesioginio ryšio tarp lentelės Emp ir Salgrade. Kai reikia išrinkti duomenis tarp užduotų ribų, naudojamas paragrafas BETWEEN…. AND.

❖ SELECT E.ENAME,E.JOB,E.SAL.S.GRADE
FROM EMP E,SALGRADE S
WHERE E.SAL BETWEEN S.LOSAL AND S.HISAL;

Nerekomenduojama naudoti ” =“. Kiti ženklai, tokie kaip ”< =“ ,“> =“ yra galimi.

Išorinis jungimas.

Jei eilutė nepatenkina sąlygos, tada eilutė nepasirodys užklausos rezultate. Pvz. lygybės apjungimo sąlygos iš EMP ir DEPT atveju kai departamento numeris 40, rezultate nebus nieko, nes nėra darbuotojų, dirbančių šiame departamente.
Tačiau nesamas eilutes galime išvesti, jei panaudosime išorinio apjungimo sąlygas. Tuo atveju ženklas "+" dedamas toje lygybės pusėje, kurioje trūksta informacijos. Tuo atveju yra sukuriama eilutė su neapibrėžtu NULL lauku. Negali būti sąlygoje kartu su paragrafu OR(arba), taip pat ir paragrafu IN.
"+" gali būti tik vienoje lygybės pusėje. Ta pati lentelė , esant išoriniam jungimui, gali būti naudojama poroje tik su viena lentele.

Bendra sintaksė tuo išorinio jungimo atveju
SELECT lentelė.stulpelis,lentelė.stulpelis
FROM lentelė1,lentelė2
WHERE lentelė1.stulpelis(+)=lentelė2.stulpelis

Arba
SELECT lentelė.stulpelis,lentelė.stulpelis
FROM lentelė1,lentelė2
WHERE lentelė1.stulpelis=lentelė2.stulpelis(+)

- SELECT ENAME,JOB,DNAME,LOC
  FROM EMP,DEPT
  WHERE EMP.DEPTNO(+)=DEPT.DEPTNO;
  ORDER BY DEPT.DEPTNO;

- SELECT E.ENAME,D.DEPTNO,D.DNAME
  FROM EMP E,DEPT D
  WHERE E.DEPTNO(+)=D.DEPTNO AND DEPTNO IN (30,40);

Lentelių Sinonimai

Aprašant stulpelių vardus kartu su lentelių vardais gali būti prarandama daug laiko, ypač jei vardai yra ilgi. Vietoj jų galima vartoti sinonimus. Kaip ir stulpelių sinonimai, tai yra metodas, leidžiantis suteikti lentelei naują vardą. Galima naudoti lentelės sinonimą, aprašant kiekvieno stulpelio nuorodą.

- SELECT D.DEPTNO,E.ENAME,E.JOB,D.DNAME
  FROM Emp E,Dept D
  WHERE E.DEPTNO=D.DEPTNO
  ORDER BY DEPT.DEPTNO;

E yra Emp lentelės sinonimas, D yra Dept lentelės sinonimas. Sinonimai galioja tik tam konkrečiam operatoriui SELECT.

Lentelės Prijungimas Prie Savęs Pačios.

Tai galima atlikti tik naudojant sinonimus, lyg tai būtų skirtingos lentelės. Tai naudojama, kai tos pačios lentelės eilutės sujungiamos su tos pačios lentelės kitos eilutės informacija.
Pvz. reikia išrinkti darbuotojus, kurių alga mažesnė negu jo šefo.
Tuomet jūs tą pačią lentelę peržiūrite du kartus. Tam galima formuoti du lentelės sinonimus.

- SELECT E.ENAME EMP_NAME,E.SAL EMP_SAL,M.ENAME MGR_NAME,M.SAL MGR_SAL FROM Emp E,Emp M WHERE E.MGR=M.EMPNO AND E.SAL<M.SAL;

Sąlyga reiškia, kad darbuotojo MGR numeris yra toks pat kaip ir menedžerio EMPNO. Paragrafe FROM yra kreipinys į lentelę Emp du kartus ir abiem atvejais yra formuojamas sinonimas. E- logiškai reiškia darbuotojus, o M reiškia menedžerius. Tai yra taip vadinamų menamų ryšių pavyzdys. Tuo atveju būtina formuoti stulpelių antraštę, kitaip tai bus neaiški informacija.

Priskyrimo Operatoriai

UNION, INTERSECT, MINUS yra naudingi, kai formuojamos užklausos, kurios kreipiasi į skirtingas lenteles. Kombinacija formuoja rezultatą iš vienos ar daugiau užklausų į vieną rezultatą. Sudėtinė užklausa gali būti sudaryta iš vieno ar kelių SQL SELECT operatorių, sujungtų vieno ar kelių priskyrimo operatorių. Priskyrimo operatorius yra dažnai vadinamas vertikaliu apjungimu, kadangi apjungimas nėra apjungimas tarp eilučių, bet tai yra apjungimas tarp stulpelių.

- SELECT JOB
  FROM EMP
  WHERE DEPTNO=10
  UNION SELECT
  FROM EMP
  WHERE DEPTNO=30;

Apjungia dvi SELECT užklausas pagal DEPTNO. Rezultate matome eilutes, kurios patenkina bent vieną iš dviejų sąlygų

UNION ALL taip pat apjungia vieną ar daugiau užklausų, tik skirtumas yra tas, kad rezultate matome netgi pasikartojančias reikšmes.

- SELECT JOB
  FROM EMP
  WHERE DEPTNO=10
  UNION ALL SELECT
  FROM EMP
  WHERE DEPTNO=30;

INTERSECT operatorius apjungimo metu į užklausą grąžina tik abiem atvejais sutampančias reikšmes

- SELECT JOB
  FROM EMP
  WHERE DEPTNO=10
  INTERSECT SELECT
  FROM EMP
  WHERE DEPTNO=30;

MINUS operatorius apjungimo metu grąžina tik skirtingas reikšmes.

- SELECT JOB
  FROM EMP
  WHERE DEPTNO=10
  MINUS SELECT
  FROM EMP
  WHERE DEPTNO=30;

PRISKYRIMO OPERATORIŲ NAUDOJIMO TAISYKLĖS

- Turi būti tas pats stulpelių kiekis ir tas pats duomenų tipas. Pasikartojančios reikšmės ignoruojamos (išskyrus UNION ALL) ir negali būti naudojamas paragrafas DISTINCT.
- Paragrafas ORDER BY dėl rūšiavimo gali būti tik užklausos gale ir tik vieną kartą.
- Užklausų kiekis neribotas ir vykdoma nuosekliai nuo pirmos iki paskutinės užklausos.

FUNKCIJOS VISAI GRUPEI (GRUPINĖS)

Šios funkcijos veikia per visą lentelę arba per lentelės sritį ir tik tada pateikia bendrą rezultatą kiekvienai grupei atskirai arba visai lentelei. Pagal nutylėjimą visos lentelės eilutės skaitomos kaip viena grupė, jei nenurodyta kitaip. Paragrafas GROUP BY komandoje SELECT gali būti naudojamas sudalinti lentelę į grupes. Grupinėse funkcijose NULL reikšmė ignoruojama. Grupinėms funkcijoms galima priskirti funkcijas:

AVG([DISTINCT|ALL],n) - nustato vidurkį grupei
COUNT([DISTINCT|ALL],Išr|*) - nustato eilučių, kurių išraiška yra apibrėžta, kiekį lentelėje
MAX([DISTINCT/ALL],Išr) -nustato didžiausią reikšmę
MIN([DISTINCT/ALL],Išr) - nustato mažiausią reikšmę
STDDEV([DISTINCT|ALL],n) -nustato nukrypimus nuo **n**
SUM([DISTINCT/ALL],n) - nustato bendrą sumą, ignoruojant neapibrėžtas reikšmes
VARIANCE([DISTINCT|ALL],n) –

Bet kuriuo atveju funkcijos taikomos visai grupei, bet jei nurodyta DISTINCT, tai funkcija taikoma tik skirtingoms reikšmėms, t.y. pasikartojančios reikšmės yra ignoruojamos. Visais atvejais, jei yra NULL reikšmė, tai įrašas ignoruojamas, išskyrus funkciją COUNT, kai naudojama **'*'**.

- SELECT MAX(SAL),MIN(SAL),AVG(SAL) FROM Emp;
- SELECT COUNT(*) from EMP where DEPTNO=20;

Norint sugrupuoti įrašus lentelėje, naudojam paragrafą **GROUP BY**

❖ SELECT AVG(SAL)
FROM Emp;

❖ SELECT AVG(SAL)
FROM Emp
GROUP BY JOB;

Kiekvienai pareigybei išvedamas darbo užmokesčio vidurkis, tačiau toks atvaizdavimo būdas nėra akivaizdus, todėl rekomenduojama šalia spausdinti ir pareigybę:

❖ SELECT JOB,AVG(SAL)
FROM Emp GROUP BY JOB;

❖ SELECT JOB, AVG(SAL)
FROM Emp
WHERE JOB='MANAGER'
GROUP BY JOB ;

Duomenys lentelėje sugrupuojami pagal darbą, ir dar po to galima išrinkti konkrečiai nurodytą darbą ir suskaičiuoti atlyginimo vidurkį.

Viduje vienos grupės galima dar grupuoti smulkiau. Tarkime skirtinguose departamentuose, norime grupuoti informaciją pagal pareigybes :

❖ SELECT JOB, AVG(SAL)
FROM Emp
WHERE JOB='MANAGER'
GROUP BY DEPTN,JOB ;

❖ SELECT Deptno,Job,Sum(sal)
FROM EMP
GROUP BY Deptno,Job;

Pateikiant rezultatą pirmiausia duomenys grupuojami pagal departamento numerį, o po to jo viduje dar grupuojama pagal darbo pobūdį. Taip sutvarkius, sumuojamas bendras atlyginimas suformuotose grupėse.

Grupinėse funkcijose yra svarbu išvedant duomenis, užklausą formuoti taip, kad nebūtų individualių laukų, kurie nėra įtraukti į paragrafą GROUP BY.

- SELECT DEPTN,MIN(SAL)
  FROM EMP; - klaidinga užklausa, nes funkcija MIN yra grupinė, o laukas DEPTNO yra kiekvienam įraše ir tai nesuderinama. Teisingai atrodytų

- SELECT DEPTN,MIN(SAL)
  FROM EMP
  GROUP BY DEPTNO;

Dabar DEPTNO jau yra grupės pavadinimas ir nebėra individualus laukas.
Gali būti grupuojami duomenys, pateikiant juos kaip grupę grupėje. Naudojant grupines funkcijas yra galimybė išjungti arba įjungti iš rezultato atskiras duomenų grupes. Tam galime naudoti paragrafą HAVING, kuris panaudojamas arba iki GROUP BY arba po jo. Turi tokią pačią prasmę kaip ir operatorius WHERE, tačiau kuris yra negalimas grupinėse funkcijose.

Norint parodyti vidutinį atlyginimą darbuotojams, pagal sąlygą, jog mus domina tik departamentai, kuriuose dirba daugiau nei trys žmonės, formuosime užklausą:

- SELECT DEPTNO,AVG(SAL)
  FROM EMP
  GROUP BY DEPTNO HAVING COUNT(*)>3;

Norint parodyti tik darbus, kurių didžiausias atlyginimas yra >=$3000, formuosim užklausą:

- SELECT JOB,MAX(SAL)
  FROM EMP
  HAVING MAX(SAL)>=3000
  GROUP BY JOB;

Nesant grupavimo, geriau atrankos sąlygai užduoti naudoti paragrafą WHERE.
Žemiau pateikiamas pavyzdys yra neteisingas, nes grupinėi funkcija naudojama poroje su paragrafu WHERE

- SELECT DEPTNO,AVG(SAL)
  FROM EMP
  WHERE AVG(SAL)>2000
  GROUP BY DEPTNO;

Reiktų rašyti

- SELECT DEPTNO,AVG(SAL) FROM EMP HAVING AVG(SAL)>2000
GROUP BY DEPTNO;

Bendru atveju sintaksė būtų tokia

SELECT *stulpelis, grupinė_funkcija*
FROM *lentelė*
[WHERE Sąlyga]|[HAVING sąlyga]
[GROUP BY sąrašas_grupavimui]
[HAVING grupės sąlyga]
[ORDER BY rūšiavimo sąlyga]

- SELECT Job,Sum(Sal) Benbra_Suma FROM EMP
  WHERE Job NOT LIKE 'SALES%'
  GROUP BY Job
  HAVING Sum(Sal)>5000
  ORDER BY Sum(Sal)

Rūšiuoja pagal gautą sumą grupėse.
Benbra_Suma formuoja naują stulpelio antraštę
Skaičiuoja visiems darbuotojams, išskyrus "SALES"

Įdėtinės grupinės funkcijos

- SELECT Max(Avg(Sal))
  FROM EMP
  GROUP BY Deptno;

Pirma suskaičiuoja atlyginimo vidurkį departamente, o po to išrenka didžiausią vidurkį tarp departamentų.

SQL*PLUS APLINKA. SQL-PLUS komandos ir jų naudojimas

SQL*Plus yra produktas, kurio aplinkoje galima vykdyti SQL ir PL/SQL komandas. Tai yra draugiškas interfeisas, sukurtas vartotojui. Jis turi savo nuosavą kalbą, kurią naudoja užklausų rezultatams formatuoti.
Taip pat yra pagalbos sistema, kurioje aprašoma kiekviena ORACLE sistemos komanda.
Gali būti įvedamos trijų tipų komandos:

SQL*Plus komandos - SQL*Plus komandos naudojamos Nustatyti SQL*Plus parametrus, Formatuoti ataskaitas, redaguoti failus ir redaguoti komandų buferį.
SQL*Plus komandos nesąveikauja su DB.

SQL komandos – jos aprašytos aukščiau

PL/SQL blokai – jos bus aprašytos vėliau

SQL*Plus komandos naudojamos :

- formatuoti užklausos rezultatus
- komandinių failų formavimui ir jų vykdymui
- pertvarkyti SQL*PLUS aplinką (įvairūs aplinkos parametrai formuojami per SET)

SQP*PLUS komandų skirtumai nuo SQL komandų:

- jos negali susirišti su kitomis ypatingomis konstrukcijomis
- įvedamos tik vienoje eilutėje, ir jei reikia pratęsimo, komandos pratęsimui kitoje eilutėje reikia naudoti ženklą”-”
- gali būti sutrumpintos
- jų negalima nutraukti vykdymo metu.

SQL*Plus galimos komandos pateikiamos lentelėje:

ACCEPT	Priima duomenis iš vartotojo
DEFINE	Deklaruoja kintamąjį (sutrumpintai: DEF)
DESCRIBE	Pateikia lentelės ir kitų objektų atributus (sutrumpintai: DESC)
EDIT	Patalpinaį redaktorių, kur vartotojas gali redaguoti SQL komandas (sutrumpintai: ED)
EXIT or QUIT	Atjungia nuo DB ir nutraukia SQL*Plus darbą
GET	Suranda SQL filą and ir patalpina jį į SQL buferį
HOST	Issue an operating system command (short: !)
LIST	Parodo paskutinę vykdytą komandą SQL buferyje (s sutrumpintai: L)
PROMPT	Parodo tekstinę eilutę ekane. Pvz. Sveikas Pasauli!!!
RUN	Rodo ir vykdo komandas, esančias SQL buferyje (sutrumpintai: /)
SAVE	Išaugoja komandą esančią SQL buferyje į failą. Pvz "save x" sukurs komandinį failą pavadinimu x.sql
SET	Modifikuoja SQL*Plus aplinką PVZ.. SET PAGESIZE 23
SHOW	Parodo aplinkos parametrus (sutrumpintai: SHO). Pvz SHOW ALL, SHO PAGESIZE t.t.
SPOOL	Siunčia rezultatą į failą. Pvz. "spool x" išsaugos STDOUT į failą vadinamą x.lst
START	Vykdo SQL komandinį failą (sutrumpintai: @)

AFIEDT.BUF yra SQL*Plus kur pagal nutylėjimą redaktorius išsaugo failą. Kai yra vykdoma komanda "ed" ar "edit" be argumentų, paskutinė SQL ar PL/SQL komanda bus išsaugota faile AFIEDT.BUF ir atidaromaredaktoriuje pagal nutylėjimą.

Kintamųjų Nustatymas Užklausos Vykdymo Metu.

Kartais yra naudinga kurti komandinius failus su paragrafu WHERE, kuris iš lentelės gali išrinkti eilutes, patenkinančias tam tikrą sąlygą. Tačiau kyla natūralus noras tą sąlygą pakeisti arba užduoti komandos vykdymo metu. Tai galima atlikti naudojant įstatomus kintamuosius. Per juos galima keisti reikšmes lyginimui, teksto eilutę ir netgi stulpelio ar eilutės vardą. Pvz. Išvedant rezultatą galima keisti rūšiavimo tvarką. Kaskart parodyti kitą departamentą ar algos dydį ir t.t.
Tam yra naudojami simboliai:

- & - naudojamas laikinai saugoti pateiktą reikšmę. Prašys įvesti duomenis kaskart, kai vykdysime užklausą.
- && naudojamas laikinai saugoti pateiktą reikšmę Prašys įvesti duomenis tik pirmą kartą, kai vykdysime užklausą, o po to ji bus vykdoma su ta pačia reikšme.
- DEFINE- galima pateikti reikšmes per kintamąjį. Suformuojamas kintamasis, kuris saugomas kompiuterio atmintyje.
- ACCEPT Komanda priima duomenis iš vartotojo

❖ SELECT Empno,Ename,Sal,Deptno
FROM EMP
WHERE Empno=&Emploe_num

Užklausos vykdymo metu paprašys įvesti arbuotojo numerį (reikšmę) “Enter value for Emploe_num”
Nurodžius reikšmę baigiama vykdyti užklausa ir pateikiamas rezultatas.
Su operatoriumi SET VERIFY galima pareikalauti, kad būtų informuojamas vartotojas apie seną ir naują kintamojo reikšmę.

SET VERIFY ON
❖ SELECT Empno,Ename,Sal,Deptno
FROM EMP
WHERE Empno=&Emploe_num
Užklausos vykdymo metu paprašys įvesti kintamojo reikšmę
“Enter value for Emploe_num” 7369 ir išves pranešimus
Old WHERE Empno=&Emploe_num
New WHERE Empno=7369

Įvedimo metu tekstinė informacija pateikiama tarp viengubų kabučių. Tačiau jei mes tas kabutes pateiksime jau su kintamuoju, tai nebereiks teksto įvesti tarp kabučių užklausos vykdymo metu.

❖ SELECT Empno,Ename,Sal,Deptno
FROM EMP
WHERE Job=’&Job_name’

Įstatomus kintamuosius galima naudoti su paragrafais :

- WHERE nurodant paieškos sąlygą
- ORDER BY pateikiant rezultatų rūšiavimo tvarką.
- Nurodant stulpelio vardą, tuo būdu pateikiant įvairią informaciją iš lentelės
- Nurodant lentelės vardą, tuo būdu pateikiant informaciją vis iš kitos lentelės

❖ SELECT Empno, &Column_name
FROM EMP
WHERE &Sąlyga

Enter Value for Column_name: Job
Enter Value for Sąlyga: Deptn=10
Rezultate matysime darbuotojo vardą ir jo pareigas ir bus išrinkti 10 departamento darbuotojai.

❖ SELECT Empno,Ename,Job, &Column_name
FROM EMP
WHERE &Sąlyga
ORDER BY &Order_Column;

Enter Value for Column_name: Sal
Enter Value for Sąlyga: Sal>3000
Enter Value for Order_Column:Ename;

Rezultate matysime darbuotojo numerį, vardą, atlyginimą. Išrinks tik tuos darbuotojus, kur atlyginimas bus >3000 ir pateiks surūšiuotą pagal vardą.

&& Leidžia pakartotinai panaudoti įvestas reikšmes, apie tai papildomai nepranešant vartotojui.

Galima iš anksto nustatyti kintamųjų reikšmes :

DEFINE : Sukuria simbolinio tipo kintamąjį
ACCEPT: Nuskaito vartotojo įvestą reikšmę ir patalpina ją į kintamąjį.
Jei reikia apibrėžti tarpą, jis turi būti įjungtas tarp viengubų kabučių.

Komanda	Aprašymas
DEFINE Kintamasis=Reikšmė	Sukuria kintamąjį ir jam priskiria reikšmę
DEFINE Kintamasis	Parodo kintamojo reikšmę ir duomenų tipą
DEFINE	Parodo visus vartotojo kintamuosius ir jų reikšmes
ACCEPT	Nuskaito vartotojo eilutę ir patalpina į kintamąjį

ACCEPT Kintamasis [Duomenų_tipas]|[FORMAT formatas]|[PROMPT Tekstas]{HIDE}

- Kintamasis – kintamojo vardas, kuriam priskiriama įvedama reikšmė. Jei tokio vardo nėra, jis sukuriamas.
- Duomenų_tipas – Tai yra NUMBER, arba CHAR arba DATE tipo.
  CHAR eilutės ilgis ribotas ir ne didesnis kaip 240 .
  DATE priklauso nuo nustatyto formato ir jis yra simbolinis.
- FORMAT format - nustato formato modelį
- PROMPT Tekstas - pirmiausia pasirodo tekstas prieš jums įvedant duomenis
- HIDE - paslepia vartotojo įvestą tekstą. Tai galėtų būti slaptažodis.

❖ ACCEPT Deptname PROMPT ’Nurodykite departamento vardą:’
Po to kintamojo sukūrimo vykdome užklausą

❖ SELECT *
FROM DEPT
WHERE Dname=UPPER(‘&deptname’)

Užklausos vykdymo metu vartotojo paprašys pateikti informaciją.

Nurodykite departamento vardą: **Sales**

Parodys iš lentelės DEPT informaciją apie departametą , kurio pavadinimas yra **Sales**.

Define Ir Undefine

Kintamieji yra atšaukiami arba su komanda **UNDEFINE** arba su komanda **DEFINE** pateikiant naują reikšmę.Taip pat kintamieji gali būti atšaukti baigiant sesiją.

DEFINE dar yra yra naudojama konkrečiai nustatyti kintamojo reikšmę.

- DEFINE Deptname=Sales sukuriamas kintamais su konkrečia reikšme.

- DEFINE Deptname paprašo parodyti kintamojo Dname reikšmę.

Pasirodys Deptname reikšmė
DEFINE Deptname='Sales'

- SELECT *
  FROM DEPT
  WHERE Dname=UPPER('&Deptname')

Naudingų Komandų Lentele

Yra visa eilė komandų, kurios labai naudingos ir pateikiamos lentelėje:

Komanda	Aprašymas
RUNFORM	Iššaukia SQL-Forms aplikaciją jos vykdymui.
SPOOL file_vardas	Užrašo visą sekančią komandų seką į užvardintą failą. Išplėtimas formuojamas automatiškai SQLir operacinė sistema tai atpažįsta
SPOOL OFF|OUT	OFF parametras uždaro failą, OUT uždaro failą ir siunčia informaciją į printerį. OFF uždaro failą, ir jį atspausdinti galima vėliau.
DESCRIBE Lentelės_vardas	Pateikia lentelės ar kito objekto struktūrą.
HELP	Iškviečia ORACLE pagalbos priemones.
HOST komanda	Iškviečia operacinę sistemą.
CONNECT vartotojo ID/slaptažodis	Persijungia prie kito vartotojo.
PROMPT Tekstas	Išveda teksto eilutę vartotojui, kai pradedamas vykdyti komandinis failas. Naudinga pagalbinė informacija.
REMARK	Pažymi, kad ši eilutė yra nevykdoma, o tik aprašanti komentarą. Naudinga komandiniuose failuose.
PAUSE	Užšaldo komandinio failo vykdymą.
START Failo_vardas	Paleidžia komandinį failą jo vykdymui.

ORACLE APLINKOS VALDYMAS

SET komanda leidžia valdyti SQL*PLUS aplinką, kurioje yra vykdomos komandos. Yra daugybė valdymo taškų ir atitinkamai per SET komandą juos galima įjungti arba išjungti. Yra parametrai pagal nutylėjimą, užrašyti faile LOGIN.SQL. Failas yra skaitomas vartotojo prisijungimo prie sistemos metu ir tuo metu nustatomi šie parametrai. Tačiau šie priskyrimai gali būti pakoreguoti. Tie pakoregavimai veikia tik vartotojo prisijungimo metu. Kai tik atsijungiate nuo sistemos, visi pakeitimai dingsta.
SET Sistemos_kintamieji ir jų Reikšmės pateikti lentelėje:

Komandos SET parametrai	Aprašymas
ECHO ON|OFF	Nustato kurios SQL*PLUS komandos yra parodomos, kai vykdomas komandinis failas
HEADING ON|OFF	Kontroliuoja stulpelių antraštes, arba jas rodo arba ne
LINESIZE {80|n}	Nustato eilutės plotį informacijos išvedimo metu ataskaitose metu
NUMFORMAT format	Nustato to tipo laukų plotį pagal nutylėjimą
PAGESIZE {24|n}	Nustato eilučių skaičių lape. Pagal nutylėjimą yra 24.
PAUSE OFF|ON	Organizuoja pauzę
SQLPROMPT text	Pakeičia SQL standartinį pranešimą
WRAP OFF|ON	Užtikrina eilutės laužymą, jei tekstas netelpa vienoje eilutėje

Galima formuoti ataskaitą, į kurią galima įtraukti :
TTITLE Puslapio antraštę
BTITLE Puslapio apačios apiforminimą
BREAK Ataskaitos puslapiavimą
FORMAT Duomenų formatavimą
COLUMN Stulpelių antraščių modifikavimą
COMPUTE Atlikti įvairius skaičiavimus

Išvedimo Duomenų Formatavimas

COLUMN - komanda naudojama duomenų išvedimui formatuoti.
Komandos sintaksė:
COLUMN [{Stulpelio_vardas} {parametras}]
CLEAR -Panaikina stulpelio formatavimą
FORMAT Format: -Keičia duomenų atvaizdavimo formatą pagal pasirinktą modelį
HEADING Tekstas -Nustato stulpelio antraštę
JUSTFY -Išdėsto stulpelio pavadinimą pagal dešinį, kairį kraštą arba centruoja virš stulpelio.
Pateikti pavyzdžiai:

- COLUMN Ename HEADING ‘Darbuotojo pavarde’ FORMAT A15
- COLUMN Salary HEADING ‘Darbuotojo atlyginimas’ FORMAT $99,990.99

Šios komandos parametrai pateikti lentelėje 4

Lentelė 4

Parametras	Aprašymas
WRAP	Jei duomenys yra platesni, nei eilutės ilgis, tai laužome informaciją pagal nustatytą eilutės ilgį, perkeliant informaciją į kitą eilutę
TRUNC	Jei duomenys yra platesni negu nustatytas eilutės ilgis, tai galima atmesti tuos duomenis, kurie netelpa į eilutę
CLEAR	Panaikinti bet kokį formatavimą
WORD_WRAPPED	Tai kaip ir WRAP, bet užtikrina kad nepadalins žodžio į atskiras dalis
HEADING	Aprašo stulpelių antraštes paprastose kabutėse.
NULL Tekstinė eilutė	Aprašo, koks tekstas pasirodys eilutėje, jei bus sutiktas tuščias elementas
PRINT	Parodo stulpelį
NOPRINT	Nerodo stulpelio
NEW_VALUE	Pasirinktas stulpelis yra užrašoma į kintamąjį
LEFT,CENTER, RIGHT	Išlygina stulpelio antraštę pagal pasirinkimą. Bet tai neturi nieko bendro su duomenų išdėstymu stulpelyje

Formatuojant duomenis galima naudoti:

9	nurodyti skaitmeninės pozicijos	99999 12345
0	nurodyti, kad naudojam vedančius nulius	909999 01234
$	nurodyti slankiojantá dolerio þenklà	$9999 $1234
L	vietinė valiuta	L9999 L1234
B	nereikia vedančių nulių	
.	nustato dešimtainio taško vietą	9999.991234.00
,	atskiria tūkstančius	9,999 1,234
#	Pasirodo užklausos rezultate, jei neteisingai parinktas duomenų tipas, ar formatas yra per mažas.	

Galima uždrausti rodyti pasikartojančias reikšmes stulpelyje, laužant ataskaitą pagal stulpelį
BREAK ON Ename ON Job

BREAK On Report – formuoja galutines sumas

Padalinti į sekcijas pagal nustatytą formą
BREAK ON Ename SKIP 4 ON Job Skip2

Bendras komandų vaizdas:
BREAK ON Stulpelis{|eilutė] [SKIP n|DUP|PAGE] ON [ON REPOTR]
PAGE perveda į naują puslapį, kai pasikečia reikšmė
SKIP n pratraukia per n eilučių prieš spausdinant naują grupę
Laužymas galimas Stulpeliui
Eilutei
Puslapiui
Ataskaitai.

Atšaukti ataskaitos laužymą galima su CLEAR BREAK;

Nustatyti ataskaitos antraštę ir pabaigą galima su skirtingais parametrais
Puslapio antraštei formuoti naudojame parametrą

TTITLE [Tekstas|OFF|ON]. Jis automatiškai formuoja sisteminę datą ir puslapio numerį.
Formuojam tekstą, kurį norime pateikti ataskaitoje tarp kabučių.
TTITLE ‘ Atlyginimų ataskaita|pagal departamentus’
“|” reiškia kad tekstas bus padalintas per atskiras eilutes

Ataskaitos pabaigai formuoti naudojame

BITITLE ‘ Parašas’

Yra galimybė paskaičiuoti tarpines reikšmes, dalinant informaciją į puslapius. Tuomet galima naudoti komandoje COMPUTE ir paragrafus
AVG , COUNT, MAXIMUM , MINIMUM, NUMBER, STD, SUM, VARIANCE

Komandinio failo formavimas dėl ataskaitų kūrimo SQL ir SQL*PLUS operatoriais. Jo formavimui atliekame sekančius veiksmus:
1. Sukuriam bet kokią korektišką užklausą
SELECT .....
FROM ...
WHERE ....
2. Išsaugome komandą faile
Save Manoatask.SQL
3. Koreguojame failą Manoatask.SQL
EDIT Manoatask.SQL
Ten pridedame reikiamas komandas dėl stulpelių formatavimo ir naujų vardų suteikimo stulpeliui.
Būtinai paliekame Užklausos vykdymo operatorių “/”
4. Komandinio failo pabaigoje surašome komandas dėl senų parametrų atstatymo SQL*PLUS aplinkai
5. Pataisytą failą išsaugojame kaip komandinį failą (SAVE COMMAND FILE)
SAVE
6. Vykdome komandinį failą
START Manoatask

Komandinis failas **Manoatask.sql**

```
set pages 10
COLUMN ename HEADING 'Pavarde' FORMAT A12
TITLE 'Ataskaita apie darbuotojus'
SELECT ENAME
FROM Emp
ORDER BY ENAME
/
SET PAGES 30
COLUMN ename format A10
```

Gali kilti noras ataskaitą atspausdinti popieriuje. Tam galite pasinaudoti komanda SPOOL įdedant ją į failą **Manoatask.sql. Ši komanda turi eiti iš kart po užklausos, tačiau tarp paskutinės užklausos formavimo eilutės ir komandos SPOOL turi būti tuščia eilutė.**
SPOOL OUT- išveda rezultatą į failą ir jį spausdina;
SPOOL OFF- išveda rezultatą į failą kurį galima atspausdinti vėliau.

```
COLUMN ename HEADING 'Pavarde' format A12
TITLE 'Ataskaita apie darbuotojus'
SELECT ENAME
FROM Emp
ORDE BY ENAME

SPOOL Manoatask.lis
/
SPOOL OFF
SET PAGES 30
COLUMN FORMAT A10
```

Manoatask.lis - failas, kurį formuoja komanda SPOOL.
Jame matome užklausos rezultatą pagal pateiktus reikalavimus.

```
Sat Apr 04                                              page    1
                    Ataskaita apie darbuotojus

Pavarde
---------------
ADAMS
ALLEN
BLAKE
CLARK

Sat Apr 04                                              page    2
                    Ataskaita apie darbuotojus

Pavarde
---------------
FORD
JAMES
JONES
KING

Sat Apr 04                                              page    3
                    Ataskaita apie darbuotojus

Pavarde
---------------
MARTIN
MILLER
SCOTT
SMITH

Sat Apr 04                                              page    4
                    Ataskaita apie darbuotojus

Pavarde
---------------
TURNER
WARD

14 rows selected.
```

SQL*PLUS komandos naudingos formuojant pagrindines ataskaitas. Pateikiame pavyzdį komandinio failo ir jo vykdymo rezultatą

```
set echo off
set feed off
set pages 20
set lines 80
set term on
TTITLE off
rem column today new_value today1
rem select to_char(sysdate,'dd/mm/YY') today from sys.dual
rem /
TTITLE center  'Ataskaita apie darbuotojus departamentuose' skip -
left 'Spausdinimo data: ' today1 skip -
left 'Puslapio Nr.:' sql.pno
column A format A15 heading 'Departamentas'
column B format A12 heading 'Pavarde'
column C format A12 heading 'Isid/data'
column D format 9,99,999 heading 'Men.alga'
column E like D heading 'Met.alga'
column F like D heading 'Komisiniai'
break on A skip page on report
compute sum of D E F on report
set term on
select initcap(d.dname) A,initcap(e.ename) B,
to_char(e.hiredate,'dd.mm.yy') C,
e.sal D,e.sal*12 E,e.sal*12+nvl(e.comm,0) F
from Emp e,Dept d
where e.deptno=d.deptno
order by d.dname

spool sudat.lis
/
spool off
column A clear
column B clear
column C clear
column D clear
column E clear
column F clear
start login.sql
set echo on
```

Ataskaita apie darbuotojus departamentuose
Spausdinimo data: 04/04/98
Puslapio Nr.: 1

Departamentas	Pavarde	Isid/data	Men.alga	Met.alga	Komisiniai
Accounting	Clark	09.06.81	2,450	29,400	29,400
	King	17.11.81	5,000	60,000	60,000
	Miller	23.01.82	1,300	15,600	15,600

Ataskaita apie darbuotojus departamentuose
Spausdinimo data: 04/04/98
Puslapio Nr.: 2

Departamentas	Pavarde	Isid/data	Men.alga	Met.alga	Komisiniai
Research	Smith	17.12.80	800	9,600	9,600
	Adams	12.01.83	1,100	13,200	13,200
	Ford	03.12.81	3,000	36,000	36,000
	Scott	09.12.82	3,000	36,000	36,000
	Jones	02.04.81	2,975	35,700	35,700

Ataskaita apie darbuotojus departamentuose
Spausdinimo data: 04/04/98
Puslapio Nr.: 3

Departamentas	Pavarde	Isid/data	Men.alga	Met.alga	Komisiniai
Sales	Allen	20.02.81	1,600	19,200	19,500
	Blake	01.05.81	2,850	34,200	34,200
	Martin	28.09.81	1,250	15,000	16,400
	James	03.12.81	950,00	11,400	11,400
	Turner	08.09.81	1,500	18,000	18,000
	Ward	22.02.81	1,250	15,000	15,500

Ataskaita apie darbuotojus departamentuose
Spausdinimo data: 04/04/98
Puslapio Nr.: 4

Departamentas	Pavarde	Isid/data	Men.alga	Met.alga	Komisiniai
**********			---	---	---
sum			29,025	3,48,300	3,50,500

DDL (DATA DEFINITION LANGUAGE)- KOMANDOS

Šios komandos naudojamos kurti, modifikuoti ar panaikinti duomenų struktūras DB. Vartotojui turi būti suteikta privilegija (CREATE TABLE) kurti duomenų struktūras ir turi būti išskirta erdvė struktūroms (lentelėms) kurti (TABLESPACE). Šias privilegijas vartotojui suteikia DB administratorius, registruodamas vartotoją. Struktūroms taikomos taisyklės:

- Nėra jokių apribojimų laike, jos gali būti kuriamos bet kada
- Nėra apribojimų dydžiui ir jo nereikia iš anksto apibrėžti
- Struktūra taip pat bet kuriuo metu gali būti keičiama arba panaikinama.

DUOMENŲ BAZĖS OBJEKTAI IR JŲ KŪRIMAS

Objektas	**Objekto aprašymas**
Lentelė (Table)	Pagrindinis informacijos kaupėjas; sudarytas iš eilučių ir stulpelių
Vaizdas (View)	Logiškai atvaizduojamos reikšmės iš duomenų lentelėse
Seka (Sequence)	Naudojama generuoti pirminio rakto (Primary Key) reikšmes
Indeksas (Index)	Pagreitina kai kurių užklausų atvaizdavimą
Sinonimas (Synonym)	sukuria alternatyvų objekto vardą

Lentelių kūrimas.

Kuriant lentelę jūs aprašote duomenų struktūrą, nurodydami kokiuose stulpeliuose ir kaip bus saugomi duomenys. Be to būtina nurodyti lentelės vardą.
Vartotojo lentelės kuriamos SQL operatoriumi **Create Table**, kurio bendra sintaksė :

```
CREATE TABLE  [Schema.]lentelės_vardas
({stulpelio_vardas        stulpelio_dtipas(stulpelio_plotis) [DEFAULT
reikšmė][stulpelio_apribojimas] [,Kitų stulpelių aprašymas,]..}
[,lentelės_lygio_apribojimai]});
```

Operatorius Create Table veikia nedelsiant.

Schema-Tai yra tas pats kas ir vartotojo vardas. Lentelės, kurios priklauso kitam vartotojui yra kitoje schemoje. Schema, jei ji nurodoma, nuo lentelės vardo atskiriama tašku.
Lentelės_vardas - kuriamos lentelės vardas (jam taikomos bendros ORACLE vardų taisyklės). Vardas turi būti unikalus vartotojo schemoje.
Stulpelio_vardas - stulpelio vardas. Stulpelių skaičius lentelėje nuo 1 iki 254.
Stulpelio_Dtipas - stulpelių tipai- atitinka duomenų tipus juose. Galimi duomenų tipai nurodyti nurodyti žemiau.

Stulpelio_plotis - stulpelio plotis. Jame Nustatome maksimalų simbolių kiekį, įvedamą į aprašomą stulpelį. Kai kurie tipai nustato stulpelio plotį automatiškai, nors jį bet kada galima pakeisti. Tai būtų datos tipo duomenys.

Galimi duomenų tipai

Char(plotis)	Simbolinės eilutės,fiksuoto ilgio.
Varchar2(plotis)	Simbolinės eilutės , kintamo ilgio. Nurodomas maksimalus simbolių skaičius stulpelyje
Number(p,s)	Kintamo ilgio skaitmeniniai duomenys p-skaičiaus ilgis, s – tikslumas. p yra bendras plotis, įskaitant ir dešimtainį skaičių
Data	Informacija apie laiką ir datą. Pagal nutylėjimą ORACLE formuojama formatu “DD-MMM-YY”
Long	Didelės kintamo ilgio eilutės (iki 2 GB).
Row(size)	Dvejetainiai duomenys - nedidelės eilutes iki 2000 bitų.
LongRow	Dvejetainės labai didelės eilutės(iki 2GB).

Pvz. Reikia sukurti lentelę Emp

CREATE TABLE Emp-
(EMPNO NUMBER(4) NOT NULL,
ENAME VARCHAR2(10),
JOB VARCHAR2(10),
MGR NUMBER(4),
HIREDATE DATE,
SAL NUMBER(7,2),
COMM NUMBER(7,2),
DEPTNO NUMBER(2) NOT NULL);
Sukuriama lentelė su tam tikrais aprašytais reikalavimais. Stulpeliai EMPNO ir DEPTNO negali būti neužpildyti. Taip yra sukuriama nauja struktūra. Galima perkurti lentelę pasinaudojant Subužklausa.

CREATE TABLE [Schema.]lentelės_vardas
({stulpelių vardai}) AS Subužklausa;

Tuo būdu į naują lentelę pernešami senosios lentelės duomenys. Bus sukurta lentelė su nurodytais stulpeliais ir į lentelę bus patalpintos kitos lentelės reikšmės. Reikia nurodyti tik stulpelių vardus [ir reikšmes pagal nutylėjimą]. Subužklausoje stulpelių skaičius turi sutapti su stulpelių skaičiumi aprašant naują struktūrą. Jei struktūroje nenurodyti stulpelių vardai jie formuojami tokie patys kaip ir Subužklausoje.
Subužklausa formuojama per SELECT operatorių.

CREATE TABLE Empn
(Dnum, Dpavarde, Depnumer) AS SELECT EMPNO, ENAME,DEPTNO FROM Emp;

Kuriant lenteles galima nurodyti apribojimus (taisykles), kurie bus tikrinami duomenų įvedimo į lenteles metu. ORACLE numato galimybę užduoti apribojimus struktūros aprašymo metu.

Constraint (Apribojimai Arba Taisyklės)

Tai yra taisyklės, kurios vėliau taikomos sukurtai lentelei duomenų į ją įvedimo metu. Jos taip pat užkerta kelią netyčiniam duomenų pašalinimui iš lentelės jei yra numatytas ryšys su kita lentele.
Apribojimai gali būti taikomi skirtingam lygyje:
Stulpelio_lygio_apribojimas - apribojimai stulpeliui.

- Stulpelis [CONSTRAINT *Constraint_name*] *Constraint_type*,

Lentelės_lygio_apribojimas - apribojimai lentelei.

- Paskutinis aprašytas stulpelis duomenų struktūroje [CONSTRAINT *Constraint_name*] *Constraint_type*

(stulpelis ar stulpelių sąrašas)
Apribojimus galime užduoti:

- kuriant lentelę arba juos vėliau pridedant lentelės taisymo metu arba kuriant įvairias procedūras.

Vienas iš apribojimo būdų, kai apribojimai naudojami aprašant lentelių valdymo taisykles, kad -užkirsti kelią priklausomų duomenų pašalinimui iš lentelės, kai yra kelios tarpusavyje surištos lentelės. Apribojimus naudoja įvairios ORACLE priemonės kuriant aplikacijas.

- Stulpelio lygio apribojimai yra aprašomi kartu su aprašomais stulpeliais, kuriems norime pritaikyti šį apribojimą.
- Lentelės lygio apribojimai aprašomi atskirai nuo stulpelių aprašymo, po paskutinio aprašyto stulpelio.

Visos apribojimų detalės formuojamos ORACLE žodyne (**DICTIONARY**) Vėliau galima peržiūrėti žodyne suformuotus apribojimus su paprasta komanda SELECT. Kiekvienam apribojimui galima suteikti nuosavą vardą, arba jis kuriamas automatiškai. Tą patį apribojimą pagal vardą vėliau galima pritaikyti ir kitiems objektams. Apribojimo vardas reikalingas ir tam, kad vėliau būtų galima pataisyti tą apribojimą arba jį atjungti nuo lentelės tam tikram laikui. Rekomenduotina kurti savo unikalius apribojimų vardus. Pagal nutylėjimą apribojimų vardai formuojami naudojant sistemą SYS_Cn, kur n - ORACLE iš eilės formuojamas numeris.

❖ SELECT Constraint_name,Constraint_type
FROM User_constraints
WHERE Table_name='Emp'

Parodys visus apribojimus sukurtus lentelei Emp. Galima dar nustatyti ir stulpelius, kuriems šie apribojimai priskirti:

❖ SELECT Constraint_name,Column_name
FROM User_cons_columns
WHERE Table_name='Emp'

Apribojimų (Constraint) Tipai

- **NOT NULL** -rūpinasi, kad į sulplį būtų įvesti duomenys. Tai užtikrina, kad lentelės pasirinktas stulpelis visada turės tam tikrą reikšmę. Standartiškai formuojama **NULL** reikšmė, t.y. netikrinamas stulpelio užpildymas duomenų įvedimo metu. Išimtį sudaro tik pirminis raktas (PRIMARY KEY). Jam iš kart formuojamas NOT NULL apribojimas.

CREATE TABLE Emp
(EMPNO NUMBER(4) NOT NULL, stulpelio lygio
ENAME VARCHAR2(10),
JOB VARCHAR2(10),
MGR NUMBER(4),
HIREDATE DATE,
SAL NUMBER(7,2),
COMM NUMBER(7,2),
DEPTNO NUMBER(2) NOT NULL); stulpelio lygio

Abu apribojimai neturi nuosavo vardo

- **DEFAULT** -nurodo kokią reikšmę pagal nutylėjimą priskirti stulpeliui duomenų įvedimo metu. (tarkime sisteminė data formuojama užsakymo formavimo dieną kaip užsakymo data). Tai gali būti simboliai arba išraiška, bet negali būti kito stulpelio vardas. Išimtį sudaro pseudostulpelis **SYSDATE**. Apribojimas negali būti įvardintas, t.y. jam negali būti suteiktas vardas ir jis negali būti pritaikytas nei kitam stulpeliui nei lentelei.
- **UNIQUE** -kontroliuoja, kad stulpelyje arba stulpelių kombinacijoje per visą lentelę nebūtų vienodų reikšmių. Gali būti naudojama kaip unikalus raktas. Reikšmė NULL yra galima, jei apribojimas taikomas tik vienam stulpeliui. Gali būti taikomas stulpelio lygyje, jei jis taikomas vienam stulpeliui, arba gali būti lentelės lygyje, jei taikomas stulpelių kombinacijai.

Paskutinis_aprašytas_Stulpelis,[**CONSTRAINT** *Constraint_name*]
U**NIQUE**(Stulpelis1,Stulpelis2...)
pvz. sukuriame lentelę DEPT naudojant lentelės lygio apribojimą.

❖ **CREATE TABLE** DEPT
(DEPTNO NUMBER(2),
DNANE VARCHAR2(10),
LOC VARCHAP2(10),
CONSTRAINT PAV-SK-TAM-LYG **UNIQUE**(DNAME,LOC));

Kableliais skiriame apribojimo detales.

❖ **CREATE TABLE** DEPT
(DEPTNO NUMBER(2),
DNANE VARCHAR2(10)
CONSTRAINT PAV-SK-TAM-LYG **UNIQUE**,
LOC VARCHAP2(10));

Šiuo atveju kuriame stulpelio lygio apribojimą. Taip bus tikrinamas tik stulpelio DNAME unikalumas, skirtingai negu ankstesniame pavysdyje, kai unikalumo reikalavimas buvo keliamas stulpelių DNAME ir LOC kombinacijai.

- **PRIMARY KEY** - pirminis raktas. Gali būti lentelės ar stulpelio lygio apribojimas. Jis sukuria pirminį lentelės raktą. Tik vienas pirminis raktas gali būti nustatytas lentelei. Pirminis raktas gali būti sudarytas iš vieno stulpelio ar kelių stulpelių kombinacijos. Jis unikaliai identifikuoja kiekvieną eilutę lentelėje. Automatiškai sukuriamas indeksas duomenų valdymui. Negali būti NULL reikšmė šiame stulpelyje. Stulpelio lygyje aprašomas kartu su stulpeliu. Lentelės lygyje kaip paprastai, aprašomas aprašius paskutinį stulpelį.

...Paskutinis_stulpelis, [**CONSTRAINT** *Constraint_vardas*] **PRIMARY** K**EY** (Stulpelis1,Stulpelis2...)

PRIMARY KEY

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

CREATE TABLE DEPT
(DEPTNO NUMBER(2),
DNANE VARCHAR2(10)
LOC VARCHAP2(10),
CONSTRAINT DEPT-PK **PRIMARY KEY(**DEPTNO)); Čia yra lentelės lygio apribojimas stulpeliui DEPTNO. Stulpelio lygyje aprašomas prie pačio aprašomo stulpelio.

...Stulpelis [**CONSTRAINT** *Constraint_vardas*] **PRIMARY KEY**,
...Kitas stulpelis

CREATE TABLE DEPT
(DEPTNO NUMBER(2) **CONSTRAINT** DEPT-PK **PRIMARY KEY**,
DNANE VARCHAR2(10)
LOC VARCHAP2(10));
Ta pati kombinacija negali būti naudojama tuo pat metu dėl apribojimų **UNIQUE** ir **PRIMARY KEY** .

- **FOREIGN KEY** - išorinis raktas.

Jis naudojamas nustatyti vientisą ryšį vienos lentelės su kita lentele. Jis naudojamas sąryšyje su vienu iš dviejų apribojimų: **PRIMARY KEY** arba **UNIQUE**.

EMP lentelė surišta su DEPT lentele per stulpelį DEPTNO. Lentelėje Dept Stulpelis DEPTNO yra PRIMARY KEY, o lentelėje Emp jis yra FOREIGN KEY.

FOREIGN KEY

EMPNO	ENAME	JOB	…	COMM	DEPTNO
7839	KING	PRESIDENT			10
7698	BLAKE	MANAGER			30
7571	FORD	MANAGER		200	9 (bus klaida įterpimo metu, nes DEPT lentelėje nėra tokio numerio)
FORD	MANAGER			200	Bus gerai, nes lauko reikšmė neapibrėžta

Išorinis raktas turi turėti atitinkančią reikšmę “pirminio rakto reikšme arba būti NULL Tėvo”” lentelėje.
Aprašant FOREIGN KEY :

- Būtinai turi būti įvardintas.
- Gali būti panaudotas užkirsti kelią šalinant įrašus iš lentelių, kai jos tarpusavyje surištos.

Lentelės lygyje aprašomas kaip paprastai aprašius paskutinį stulpelį,
...Paskutinis_stulpelis,[**CONSTRAINT** *Constraint_vardas*] **FOREIGN KEY**(Stulpelis1,Stulpelis2....) **REFERENCES** lentelė1(Stulpelis1,Stulpelis2....)

FOREIGN KEY(Stulpelis1,Stulpelis2....) – prašo nurodyti stulpelius vaiko lentelėje, kai apribojimas aprašomas lentelės lygyje, kurie turi būti suprantami kaip išorinis raktas.
REFERENCES – nurodo lentelės stulpelius “tėvo” lentelėje, kurie yra suprantami kaip pirminis raktas.

```
CREATE TABLE Emp
(EMPNO    NUMBER(4) NOT NULL,
ENAME          VARCHAR2(10),
JOB            VARCHAR2(10),
MGR       NUMBER(4),
HIREDATE       DATE,
SAL            NUMBER(7,2),
COMM      NUMBER(7,2),
DEPTNO         NUMBER(2) NOT NULL,
CONSTRAINT EMP_DEPT_FK FOREIGN KEY (DEPTNO)
```

REFERENCES DEPT(DEPTNO));
Kreipiantis į lentelę DEPT per REFERENCES, ši lentelė jau turi būti sukurta.
Šis ryšys užtikrina, kad departamentas iš lentelės Dept negali būti ištrintas, jei bent vienas darbuotojas jame dar dirba, tai yra jei lentelėje Emp yra bent vienas susijęs įrašas. Taip pat užtikrina, kad kiekvienas darbuotojas yra priskirtas kokiam nors apibrėžtam departamentui.
FOREIGN KEY apribojimą taip pat galima aprašyti, aprašant stulpelį. Tačiau tuo atveju nenaudojame paragrafo **FOREIGN KEY** ir tai galioja tik vienam stulpeliui.

CREATE TABLE Emp
(EMPNO NUMBER(4) NOT NULL,
ENAME VARCHAR2(10),
JOB VARCHAR2(10),
MGR NUMBER(4) **CONSTRAINT** EMP_EMPNO_FK **REFERENCES** Emp(EMPNO),
HIREDATE DATE,
SAL NUMBER(7,2),
COMM NUMBER(7,2),
DEPTNO NUMBER(2) NOT NULL **CONSTRAINT** EMP_DEPT_FK **REFERENCES** Dept(DEPTNO));

- **ON DELETE CASCADE** kaskadinis įrašų šalinimas lentelėse. Apribojimas eina kartu su išoriniu raktu. Nurodo, jog pašalinant įrašą "tėvo" lentelėje, kartu su juo pašalinami ir surišti įrašai iš "vaikų" lentelės. Jei šis apribojimas nenurodytas, tai iš "tėvo" lentelės negalima pašalinti įrašo, kol nebus pašalinti atitinkami įrašai iš "vaikų" lentelės.

Stulpelio lygyje apribojimas rašomas:
....DEPTNONUMBER(2) NOT NULL **CONSTRAINT** EMP_DEPT_FK **REFERENCES** Dept(DEPTNO)) **ON DELETE CASCADE, …..**

Formuojam lentelės lygio išorinio rakto apribojimą **ON DELETE CASCADE**
CREATE TABLE Emp
(EMPNO NUMBER(4) NOT NULL,
ENAME VARCHAR2(10),
JOB VARCHAR2(10),
MGR NUMBER(4) **CONSTRAINT** EMP_EMPNO_FK **REFERENCES** Emp(EMPNO),
HIREDATE DATE,
SAL NUMBER(7,2),
COMM NUMBER(7,2),
DEPTNO NUMBER(2) NOT NULL **CONSTRAINT** EMP_DEPT_FK **REFERENCES** Dept(DEPTNO) **ON DELETE CASCADE**);

- **KOMBINUOTAS IŠORINIS RAKTAS** - Jis susideda daugiau nei iš vieno stulpelio. Galima nurodyti iki 16 stulpelių. Jis gali būti taikomas tik lentelės lygio aprašyme.

CREATE TABLE EMP
(EMPNO NUMBER(4) NOT NULL,
ENAME VARCHAR2(10),
JOB VARCHAR2(10),
MGR NUMBER(4) **CONSTRAINT** EMP_EMPNO_FK **REFERENCES** EMP(EMPNO),
HIREDATE DATE,
SAL NUMBER(7,2),
COMM NUMBER(7,2),
DEPTNO NUMBER(2) NOT NULL **CONSTRAINT** EMP_DEPT_DNAME_FK **REFERENCES** Dept(DEPTNO,DNAME));

- **CHECK** -nurodomos sąlygos, kurios yra tikrinamos duomenų įvedimo metu.

Tai yra stulpelio lygio apribojimas, t.y. jis taikomas kiekvienam stulpeliui atskirai. Negalima naudoti funkcijų, kurios keičia reikšmes darbo metu ir taip pat pseudostulpelių (SYSDATE, USER).

... DEPTNONUMBER(2) NOT NULL **CONSTRAINT CHECK** _DEPTNO **CHECK (**DEPTNO BETWEEN 10 AND 99),
DNAME VARCHAR2(10) **CONSTRAINT CHECK** _DNAME DNAME= UPPER(DNAME),...

Duomenų įvedimo metu kiekvienai įvedamai eilutei bus tikrinamos pateiktos sąlygos. Gali būti naudojamos tos pačios konstrukcijos, kaip ir vykdant užklausas.

Sukurti tris lenteles kuriose yra kaupiama informacija apie darbuotojus, departamentus ir atlyginimų gradaciją.

CREATE TABLE DEPT
(DEPTNO NUMBER(2) **PRMARY KEY**,
DNAME VARCHAR2(10),
LOC VARCHAR2(10));

CREATE TABLE EMP
(EMPNO NUMBER PRIMARY KEY,
ENAME VARCHAR2(10),
JOB VARCHAR2(10),
MGR NUMBER(4),
HIREDATE DATE,
SAL NUMBER(7,2),
COMM NUMBER(7,2),
DEPTNO NUMBER(2) NOT NULL,
FOREIGN KEY DEPT(DEPTNO));

CREATE TABLE SALGRADE
(GRADE NUMBER,
LOSAL NUMBER(7,2),
HISAL NUMBER(7,2));

Apribojimus pridėti arba panaikinti galima, bet modifikuoti juos galima taisant lentelę. Taip pat galima prijungti arba atjungti apribojimus prie lentelės bet kuriuo metu.
NOT NULL apribojimą galima pateikti per MODIFY paragrafą modifikuojant lentelę.

DEPTNO NUMBER(2) NOT NULL, CONSTRAINT CHEK_DEPTNO
CHECK (DEPTNO BETWEEN 10 AND 99) **DISABLE**,
DNAME VARCHAR2(10)
CONSTRAINT CHECK_DNAME=UPPER(DNAME)...

DISABLE- gali būti naudojamas, norint paneigti taikomą apribojimą. Vėliau per aplikacijų kodą jį galima prijungti (ENABLE) naudojant ORACLE priemones, ir taikyti apribojimus duomenų įvedimo metu.

DARBAS SU LENTELĖMIS

Yra specialūs operatoriai skirti darbui su lentelėmis.

- **DESCRIBE lentelės_vardas;** - Komanda parodo lentelės struktūrą.
- **DROP TABLE lentelės_vardas;** -Komanda panaikina nurodytą lentelę.
- **TRUNCATE TABLE lentelės_vardas;** -Ištrina visus įrašus lentelėje, bet palieka lentelės struktūrą.
- **RENAME** senas_vardas **TO** naujas vardas ;- leidžia keisti lentelės, vaizdo ar sinonimo vardą - galima naudoti objekto vardo pakeitimui. Komandą gali naudoti tik lentelės, vaizdo arba sinonimo savininkas.

Keičiant vardą , duomenys objekte nesikeičia, bet jei tas vardas naudojamas kituose objektuose (pvz. formose, užklausose), tai tuos objektus reiks atnaujinti. Kartais reikia atlikti ir lentelių perkūrimą.

❖ DESCRIBE KLIENTAI;
Rezultate matome lentelės KLIENTAI struktūrą

Name	Null?	Type
KL_ID	NOT NULL	NUMBER(3)
PAVARDE	NOT NULL	VARCHAR2(20)
VARDAS	NOT NULL	VARCHAR2(25)
TELEFONAS		NUMBER(6)
AUT_NUMERIS		VARCHAR(6)
AUTOMOBILIS	NOT NULL	VARCHAR2(30)

- DESCRIBE UZSAKYMAI;

Name	Null?	Type
KL_ID	NOT NULL	NUMBER(3)
DARBAI	NOT NULL	VARCHAR2(50)
DATA		DATE

VARTOTOJO DB IR SCHEMA

Lentelės, į kurias kreipiamės aprašant apribojimus, turi būti vartotojo schemoje. Jei jos priklauso skirtingiems vartotojams, kreipiantis į lentelę, kuri yra aprašyta ne jūsų schemoje, reikia per prefiksą nurodyti savininko vardą.
Schema yra objektų kolekcija. Schemos objektai yra loginės struktūros, kurios tiesiogiai pririštos prie DB duomenų. Schemos objektai - tai lentelės, sinonimai, sekos, patalpintos procedūros, indeksai, klasteriai ir DB ryšiai. Kartais naudinga žinoti, kokia lentelė/vaizdas ir kaip rišasi su kita lentele/vaizdu.
Naudokitės informacija, pateikta duomenų žodyne **(Data Dictionary)** .

APRIBOJIMŲ PAŽEIDIMŲ LENTELĖ

Jiems formuoti naudojame **EXCEPTIONS INTO lentelė** parametrą. Tačiau jau pažeidimų lentelė turi būti sukurta. Ši lentelė turi būti jūsų asmeninėje duomenų bazės struktūroje (schemoje). Ją galima naudoti, jei norime fiksuoti apribojimų pažeidimus, kad vėliau galėtume juos analizuoti.
Šios lentelės struktūra:

Stulpelio vardas	Duomenų tipas
ROWID	ROWID
OWNER	VARCHAR2
TABLE_NAME	VARCHAR2
CONSTRAINT	VARCHAR2

Jai formuoti rašome
CREATE TABLE PAZEIDIMAI
(ROWID ROWID
OWNER VARCHAR2(10)
TABLE_NAME VARCHAR2(10)
CONSTRAINT VARCHAR2(10));

Vėliau ši lentelė gali būti naudojama pažeidimams fiksuoti ir ji prijungiama per parametrą, formuojant apribojimus, kuriuos norime vėliau analizuoti

- **CREATE TABLE** Emp
  (EMPNO NUMBER(4) NOT NULL,
  ENAME VARCHAR2(10) **CONSTRAINT** CHK_DR CHECK
  (ENAME=UPPER(ENAME)) **EXCEPTION INTO** PAZEIDIMAI);

Sukurti lentelę su apribojimais
Sukurti lentelę naudojant eilutes iš kitos lentelės
Create Table vardas
As
Select
From kita_lentelė
Where sąlyga

LENTELIŲ IR APRIBOJIMŲ KOREGAVIMAS

ALTER TABLE Lentelės vardas -Lentelės struktūrą koreguoti galima

- Pridedant stulpelius
- Modifikuojant stulpelius
- Taip pat galima paveikti ir apribojimus
- Pridedant apribojimus stulpeliui ar lentelei
- Nuimant apribojimus
- Modifikuojant apribojimus atjungiant juos ar prijungiant

Tačiau:

- negalima panaikinti stulpelio;
- negalima pakeisti apribojimo pagal nutylėjimą;

ALTER TABLE- galima naudoti lentelės struktūros keitimui

- **ADD** žodį galima naudoti norint pridėti stulpelį ar apribojimą esamai lentelei. Mes negalime pasirinkti vietos lentelėj kur bus pridėtas stulpelis. Jis visada pridedama lentelės gale.
- **MODIFY** žodis gali būti naudojamas pakeisti esamo stulpelio aprašymą. Tačiau yra keturi pakeitimai, kurių negalima atlikti
  - Negalima pakeisti stulpelio, kuriame yra NULL reikšmė iš **NULL** į **NOT NULL**
  - Negalima pridėti naujo stulpelio su apribojimu **NOT NULL**. Tam jūs pirmiausia pridedat naują stulpelį be apribojimų, tačiau užpildote pilnai bet kokia reikšme. Po to galima jau pakeisti apribojimą į **NOT NULL**.
  - Negalima sumažinti stulpelio pločio arba pakeisti duomenų tipo, nebent stulpelyje nėra duomenų.
  - **DROP** žodį galima naudoti norint nuimti apribojimus nuo lentelės. Turite prisiminti, kad apribojimų keisti negalima. Todėl jį keičiam, vieną sunaikinant, o kitą pridedant. Čia būtina žinoti apribojimo vardą.
- **ENABLE/DISABLE** žodis naudojamas, kai norime atjungti arba prijungti apribojimus, bet nenorime jų panaikinti arba pakeisti.
  - **ENABLE -** prijungti apribojimą, susijusį su šia lentele. Jis gali būti pateiktas kartu su **CREATE TABLE** arba su **ALTER TABLE** komandomis.

- **DISABLE** - leidžia atjungti surištus su lentele apribojimus. Tai gali būti kartu su **CREATE TABLE** arba su **ALTER TABLE** komandomis.

***P AVYZDŽIAI** :*

- **ALTER TABLE Lentelės_vardas**
  ADD (Stulpelio_vardas stulpelio aprašymas);

Naujai sukurtas stulpelis pridedamas į jau sukurtos lentelės struktūros pabaigą.
Stulpelio_vardas-yra naujas vardas, kuris negali būti anksčiau toje lentelėj panaudotas.
MODIFY (**Stulpelio_vardas** stulpelio aprašymas);

- **ALTER TABLE Table_name**
  ADD (PASTABA VARCHAR 2(100) CHEK UPPER);
  DROP PRIMARY KEY ;

Su **DROP** - pašaliname apribojimus, o su **MODIFY** keičiam apribojimus pagal jų vardą.

- **DROP** [**CONSTRAINT** - Apribojimo_vardas];

- **MODIFY (CONSTRAINT** - Apribojimo_vardas);

DML-DARBAS SU DUOMENIMIS.

Duomenų manipuliavimo komandos leidžia įterpti duomenis į duomenų bazę arba pašalinti juos iš ten. Be to galima duomenis pataisyti ir ten pat juos palikti. Yra trys pagrindinės operacijos su duomenimis, kuriuos gali atlikti vartotojas, jei tam jam suteikti įgaliojimai. Prie tų komandų dar galima priskirti ir transakcijų valdymo komandas.

- **INSERT** leidžia įterpti vieną eilutę į lentelę.

INSERT INTO Lentelės_vardas[(stulpelis1[,stulpelis2, ...])]
VALUES(reikšmė1[, reikšmė2, ...]);

Komanda įterps reikšmę1 į stulpelį1, reikšmę2 į stulpelį2 ir t.t.
Stulpelių sąrašas yra nebūtinas, bet jei sąrašas nenurodytas, tai komanda nebus užbaigta tol, kol nebus įvesta tiek duomenų, kiek numatyta lentelės struktūroje. Simboliniai duomenys ir datos tipo duomenys visada nurodomi tarp viengubų kabučių.

- INSERT INTO Dept(Deptno,Dname,Loc)
  VALUES(50,'DEVELOPMENT','DETROIT');

Galima įvedimo metu taip pat suformuoti ir tuščias laukų reikšmes, Dėl to reikia norimą palikti neužpildytą stulpelį neįtraukti į sąrašą.

- INSERT INTO Dept(Deptno,Dname) VALUES(60,'MIS');

Tą patį galima padaryti ir kitaip

- INSERT INTO Dept VALUES(70,'FINANCE',NUL);

Įterpti specialius duomenis galima naudojant pseudostulpelius pvz. SYSDATE

❖ **INSERT INTO**
Emp(EMPNO,,ENAME,JOB,MGRR,HIREDATE,SAL,COMM,DEPTNO)
VALUES(7196,'GREEN','SALASMEN',7782,SYSDATE,2000,NULL,10);

Automatiškai formuojama priėmimo į darbą data tos dienos, kada buvo atlikta įterpimo operacija. Data imama iš sistemos, be to stulpelis COMM lieka neužpildytas.

Kad palengvinti įvedimą, galima reikšmes užduoti kaip įstatomus parametrus:

VALUES (&reikšmė1, &reikšmė2, ...);

" &" ženklas rodo, kad įvedimo metu bus pareikalauta įvesti tą reikšmę iš klaviatūros.

❖ INSERT INTO Dept(DEPTNO,DNAME,LOC)
VALUES(&Departamento_Nr,'&Departamento_pavad','&Departamento_vieta');

Galima bus kartoti įvedimą neribotą kiekį kartų ir kiekvieną kartą įvesti naujas eilutes į lentelę Dept, atsakant į sistemos pateiktus paklausimus. Tai bus interaktyvus informacijos įvedimas. Įterpti naujas eilutes į lentelę galima taip pat iš kitos lentelės perkopijuojant duomenis ir tam galima pasinaudoti subužklausa.

❖ INSERT INTO Managers(ID,NAME,SALARY,HIREDATE)
SELECT Empno,Ename,Sal,Hiredate
FROM EMP
WHERE Job='MANAGER';

Bus perkopijuota informacija iš lentelės Emp į lentelę Manager. Vietoje VALUES mes panaudojome SELECT operatorių kaip subužklausą. Lentelė Manager jau turi būti sukurta.

- **UPDATE** leidžia pakeisti duomenis lentelėje.

UPDATE Lentelės vardas
SET Stulpelis1=reikšmė1[,Stulpelis2=reikšmė2 …]
[**WHERE** sąlyga];

Galima iš kart atnaujinti kelias eilutes pagal nurodytą sąlygą.
Stulpelis1, Stulpelis2… Stulpelių vardai, kurių reikšmes reikia pakeisti naujomis
Reikšmė1, Reikšmė2 atitinkamos naujos stulpelių reikšmės.
WHERE užduoda sąlygą, kuriai esant patenkintai tos eilutės yra atnaujinamos. Sąlyga yra nebūtina, bet jei ji nenurodyta, pakeitimas atliekamas visoms eilutėms lentelėje.

❖ UPDATE Emp
SET DEPTNO=20
WHERE EMPNO=7782;

Bus pakeista stulpelio DEPTNO reikšmė eilutei, kur EMPNO=7782, t.y. darbuotojas pervedamas į kitą departamentą.

❖ UPDATE Emp
SET DEPTNO=20;

Visoms eilutėms lentelėje Emp pakeičiamas departamento numeris. Visi darbuotojai taps to paties departamento pareigūnais.

Taip pat lentelės įrašų atnaujinimui galima naudoti subužklausas

❖ UPDATE Emp
SET (Job,Deptno)=(SELECT Job,Deptno FROM Emp WHERE Empno=7499)
WHERE Empno=7698;

Vieno darbuotojo pareigas ir departamento numerį perduos kitam darbuotojui.

- **DELETE** galima pašalinti įrašus iš lentelės.

DELETE FROM Lentelės_vardas
[**WHERE** sąlyga];

Iš lentelės nurodytu vardu bus pašalintas įrašas, kuris patenkina sąlygą.

❖ DELETE FROM Dept
WHERE Dname=’DEVELOPMENT’;

Jei sąlyga nenurodyta, panaikinamos visos eilutės.

❖ DELETE FROM Dept;

Kaip paprastai įrašų panaikinimui galima pasinaudoti subužklausa pateikiančia duomenis iš kitos lentelės

❖ DELETE FROM EMPLOE
WHERE Deptno=(SELECT Deptno FROM Dept WHERE Dname=’SALES’);

Iš lentelės EMPLYE pašalins visus darbuotojus, kurių pareigos nurodytos sąlygoje.

Pašalinti visus įrašus galima taip pat ir su komanda

- **TRUNCATE TABLE** lentelės_vardas;

Ši komanda įrašus šalina fiziškai ir atstatyti jų neįmanoma.

- **DROP** pašalina viską: duomenis kartu su struktūra.

❖ **DROP TABLE** lentelės_vardas;

TRANSAKCIJOS (DUOMENŲ APDOROJIMAS).

DB transakcijos sudaromos naudojant vieną iš nurodytų būdų**:**

- DML komandas, kurios atlieka vieną nuoseklų duomenų pakeitimą.
- DDL komandas, kurios sukuria struktūrą.
- DCL komandas, kurios atlieka duomenų kontrolę.

ORACLE serveris užtikrina duomenų nuoseklumą per transakcijas.
Transakcijos suteikia vartotojui daugiau lankstumo ir kontrolės, keičiant duomenis, bei užtikrina duomenų pastovumą, netgi kai yra vartotojo proceso trūkis arba pačios sistemos trūkis.Transakcijos pavyzdys gali būti veiksmai, kuriuos reikia atlikti norint kliento pinigus iš vienos sąskaitos pervesti į kitą sąskaitą. Tai pirmiausia reiktų pinigus nuo vienos sąskaitos nurašyti ir po to tuos pinigus (tą pačią sumą) padėti į kitą sąskaitą. Abu veiksmai kartu baigiami sėkmingai arba jie yra nesėkmingai baigiami.

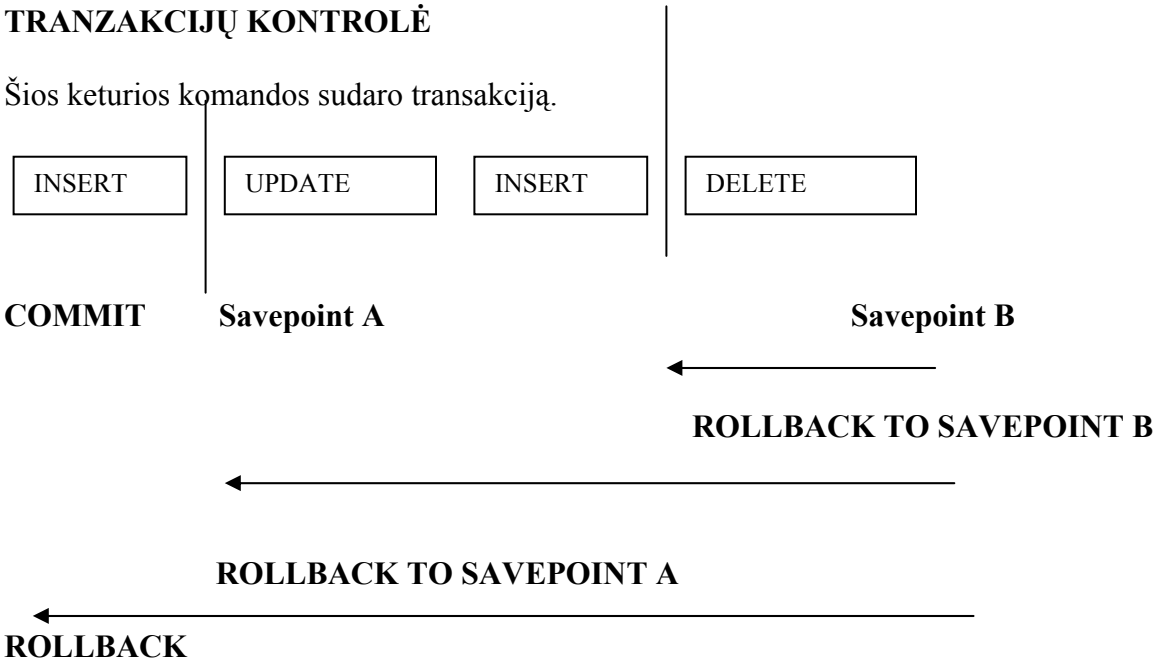
Transakcija vykdoma:

- pradedama, kai yra pirmą kartą vykdoma DML ir baigiama, kai sutinkama viena iš komandų **COMMIT** arba **ROLLBACK**.
- Pradedama DDL komanda **CREATE** ir vykdoma tuoj pat nelaukiant jokio patvirtinimo ir ji negali būti atšaukta.

Komanda **COMMIT** užbaigia transakciją t.y. patvirtina visas operacijas, kurias vartotojas vykdė seanso metu. Operacijos patvirtinamos ir tada, kai vartotojas tvarkingai atsijungia nuo sistemos.
Komanda **ROLLBACK** -atšaukia visas operacijas(veiksmus), kurias vartotojas vykdė tam tikros transakcijos metu. Tai galioja tik tuo atveju jei nebuvo prieš tai atlikta komanda **COMMIT** t.y. nebuvo patvirtinta transakcija, arba nuo paskutinės tokios komandos vykdymo.
COMMIT galime užduoti po kiekvienos DML komandos patvirtinimo. Po šios komandos prasideda nauja transakcija.

TRANZAKCIJŲ KONTROLĖ

Šios keturios komandos sudaro transakciją.



ROLLBACK TO SAVEPOINT *vardas*- Atšaukimui galime naudoti tik tuo atveju, jei kontrolinis taškas buvo sukurtas su komanda **SAVEPOINT Kontrolinio_taško_vardas**.

- **SAVEPOINT** dalina transakcijas į smulkias dalis ir duoda galimybę sustabdyti darbą bet kuriuo laiku su parametru vėlesniam patikrinimui, arba atšaukimui iki pasirinkto taško arba galutinai. Vienoje transakcijoje galima sukurti iki penkių kontrolinių taškų, kiekvienam užduodant vardą. Jeigu naudojame tą patį vardą, tai ankstesnis kontrolinis taškas panaikinamas.

TRANSAKCIJOS PAVYZDYS

❖ **UPDATE** Emp **Transakcijos pradžia**
SET SAL=SAL*1.2
WHERE JOB=’CLERK’;
COMMIT; **Transakcijos pabaiga**

- **ROOLBACK** - atšaukia transakciją iki galo.
- **ROLLBACK TO [SAVEPOINT] Kontrolinio_taško_vardas** -atšaukia visas operacijas iki kontrolinio taško arba iki paskutinės operacijos **COMMIT**;

- **STATEMENT LEVEL ROLLBACK** - gali dalinai atšaukti transakciją.

Tai gali būti vienas DML patvirtinimas kai įvyksta krizė DML lygyje, todėl atšaukus patvirtinimą duomenys lieka kokie buvo iki šio DML patvirtinimo.
Jei atliekama DDL komanda, tai transakcija yra patvirtinama nedelsiant ir jos neįmanoma atšaukti. Rekomenduojama transakcijas užbaigti su **COMMIT** arba **ROLLBACK**.

Duomenų būsena prieš COMMIT ar ROLBACK:

- Ankstesnė duomenų būsena dar gali būti atstatyta
- Esamas vartotojas gali peržiūrėti DML komandų veikimo rezultatus per SELECT operatorių, lyg transakcija būtų patvirtinta
- Kiti vartotojai negali matyti DML veikimo rezultatų, kuriuos atliko kitas vartotojas
- Naudojami įrašai yra užrakinti, kiti vartotojai negali keisti duomenų, kol juos naudoja šis vartotojas.

DUOMENŲ BŪSENA PO COMMIT

- Duomenų pakeitimai yra įvykdomi DB
- Ankstesnė duomenų būsena yra prarasta
- Visi vartotojai gali matyti pakeitimų rezultatus
- Įrašų užrakinimas yra atšaukiamas. Kiti vartotojai gali dirbti su atatinkamais įrašais.
- Visi SAVEPOINT sukurti kontroliniai taškai yra panaikinami.

DUOMENŲ BŪSENA PO ROLBACK

- Jokie pakeitimai neatlikti
- Yra atstatoma ankstesnė duomenų būsena
- Atšaukiamas įrašų užrakinimas.

PAVYZDŽIAI

- DELETE FROM EMPLOYE;
- ROLBACK Transakcija atšaukiama. Visi įrašai, kurie buvo pašalinti, yra atstatomi.

Per SQL*Plus komandas galima valdyti transakcijas, pasirenkant darbo režimą.
SET COMMIT ON - nustato automatinį transakcijų patvirtinimą po kiekvienos DML komandos vykdymo. Bet tai stabdo darbą.
SET COMMIT OFF - atšaukia automatinio patvirtinimo reikalavimą, tuomet transakciją reikia patvirtinti patiems.

SET TRANSACTION READ ONLY;

DB vartotojai gali atlikti dviejų tipų operacijas su duomenimis:

- Skaitymo operacija (**SELECT** komanda)
- Rašymo operacija (**INSERT,UPDATE,DELETE**)

Jums skaitant nuosekliai:

- DB skaitytojas ir taisytojas turi tokį pat nuoseklų duomenų vaizdą
- Skaitytojai nemato tų duomenų, kurie yra taisomi kito vartotojo, kol nebus patvirtinta transakcija.
- Taisytojai yra užtikrinti kad pakeitimai DB yra atliekami nuosekliai ir mato visus pataisymus, netgi iki transakcijos patvirtinimo.
- Visi vartotojai gali keisti duomenis nekonfliktuojant tarpusavyje.

Tam yra sukuriamas **ROLLBACK Segmentas**, kuriame prieš keičiant duomenis, serveris kitiems vartotojams kopijuoja duomenis iš DB. Visi DB skaitytojai, išskyrus taisytoją, mato informaciją iš šio segmento. Tik po **COMMIT** komandos pakeitimai tikrai yra atliekami pačioje DB. Po to šie pakeitimai matomi visiems DB vartotojams. **ROLBACK Segmentas** atlaisvinamas nuo senų duomenų. Jei nebuvo patvirtinta transakcija, duomenys iš **ROLBACK Segmento** grąžinami į DB.

Visi vartotojai mato duomenis kurie buvo užfiksuoti paskutinės COMMIT komandos metu.

DUOMENŲ PERŽIŪREJIMO BŪDAI

Duomenis peržiūrėti galima su komanda **SELECT** pateikiant užklausą serveriui**.** Tai duomenų atvaizdavimas ekrane.Tačiau galima peržiūrėti duomenis, sukuriant duomenų atvaizdą. Atvaizdas - tai virtuali lentelė į kurią duomenys patalpinami iš realios/ (ar net kelių) lentelės. Su sukurtu atvaizdu galima dirbti kaip su lentele. Atvaizdas taip pat gali būti sukurtas kito atvaizdo pagrindu.

Vaizdų (Atvaizdų) kūrimas.

Vaizdas yra kaip langas, pro kurį galima matyti duomenis iš skirtingų lentelių. Vaizdas yra patalpintas kaip užklausos dalis duomenų žodyne. Pats vaizdas neturi savo nuosavų duomenų.
Juos kuriame tam kad:

- Apriboti priėjimą prie DB
- Lengvai gauti kompleksinius duomenis. Kai yra užklausoje formuojamos sąlygos, kurios yra labai komplikuotos, tai kartą sukūrus vaizdą, tuos duomenis lengvai pateiksime vartotojui.
- Daryti duomenis nepriklausomus
- Pateikti tuos pačius duomenis skirtingais vaizdais.Vartotojų grupės tuos pačius duomenis pasiima pagal jų pačių poreikius.

Sukurti arba perkurti esamą vaizdą galima su komanda :

CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW *Vaizdo_vardas*
AS SELECT *SUBUŽKLAUSA*
[WITH CHECK OPTION [CONSTRAINT Constraint]
[WITH READ ONLY]
SUBUŽKLAUSA- konstruojama kaip paprastai, nurodant ką norime patalpinti į vaizdą, iš kokių lentelių ir kaip turi būti atrenkami duomenys. Tik čia negalima naudoti paragrafo ***ORDER BY***
[OR REPLACE] – perkuria naujai vaizdą, jei toks jau yra
FORCE –
NOFORCE kuria vaizdą, tik jei yra bazinė lentelė. Nustatomas pagal nutylėjimą.
WITH CHECK OPTION –tikrina apribojimus.
WITH READ ONLY- negalimi jokie DML veiksmai.

Vaizdas, kurio vardas užduotas komandoje, gali būti traktuojamas kaip lentelė, su kuria vėliau galima dirbti kaip su bazine lentele. Jei administratorius vartotojui leidžia dirbti su lentele, tai vartotojas gali matyti visą informaciją, kuri yra toje lentelėje, bet jei administratorius leidžia dirbti tik su vaizdu, tai vartotojas gali matyti tik tą informaciją, kuri yra perkeliama iš lentelės į vaizdą, nurodant stulpelius, kuriuos leidžiama matyti.Taigi, vaizdas apsaugo lentelę. Kuriant vaizdą, galime suformuoti papildomą informaciją, kuri yra gaunama iš skirtingų lentelių.
Pvz. turime dvi lenteles, iš kurių norime išrinkti laukus ir sukurti vaizdą:

CREAT VIEW vaizdo_ vardas

AS SELECT lentele1.stulpelis1,lentele2.stulpelis2,lentele2.stulpelis3
FROM lentele1, lentele2;

Galime nurodyti sąlygas, pagal kurias atrinksime duomenis:
Pvz. WHERE lentele1.stulpelis1= lentele2.stulpelis2;

Paprastieji ir kompleksiniai vaizdai

Savybės	Paprastieji vaizdai	Kompleksiniai vaizdai
Lentelių skaičius	Viena	Viena ar daugiau
Funkcijų panaudojimas	Negalimas	Galimas
Duomenų grupavimas	Negalimas	Galimas
DML panaudojimas vaizduose	Galimas	Ne visada

Paprastieji vaizdai.

- **CREATE VIEW** V1
  AS SELECT Empno,Ename,Sal
  FROM EMP
  WHERE DEPTNO=10;

Sukuria vaizdą, kuriame pateikiama informacija apie 10 departamento darbuotojus.
Vaizdo struktūrą galima sužinoti panaudojus komandą DESCRIBE.

- **DESCRIBE V1** parodo vaizdo V1 struktūrą.

- **SELECT** * FROM V1 pateiks vartotojui rezultatus lyg tai būtų atskira lentelė.

- **CREATE VIEW** V2
  AS SELECT Empno Darbuotojo_vardas,Ename Darbuotojo_numeris, Sal Darbuotojo_alga
  FROM Emp **WHERE** Deptno=20;

Šiame pavyzdyje sukuriamas vaizdas, kuriems stulpelių antraštės pakeičiamos kitomis.

- **SELCT** *
  FROM V2; -rodys duomenis su naujais stulpelių vardais.

Kompleksinių vaizdų kūrimas.

- **CREATE VIEW** V2 (VARDAS,MINALGA,MAXALGA,VIDALGA)
  AS SELECT D.Dname,MIN(E.Sal),MAX(E.Sal),AVG(E.Sal)
  FROM EMP E,DEPT D
  WHERE E.DEPTNO=D.DEPTNO
  GROUP BY D.Dname;

Kur MIN(),MAX(),AVG() ORACLE grupinės funkcijos. Čia grupuojame pagal departamento vardą, kad galėtume pateikti ir departamento vardą.
Vaizdą galime pakeisti prieš tai nepanaikinus seno su ta pačia komanda Bet nurodant parametrą **OR REPLACE** - senas vaizdas keičiamas nauju.

- **CREATE VIEW** V3
  AS SELECT ENAME,I2*SAL METALGA
  FROM Emp
  WHERE DEPTNO=20;

Jei užklausoje naudojamas stulpelis, tai nėra reikalo jai priskirti vardo vaizde, bet jei stulpelis išskaičiuojamas, tai rekomenduojame jam sukurti naują vardą .

Taisyklės dirbant su vaizdais.

DML operatoriai galimi tik paprastiems vaizdams. Negalima pašalinti eilutės, jei kuriant vaizdą buvo numatyta:

- Grupinės funkcijos panaudojimas su **GROUP BY** paragrafas
- DISTINCT paragrafas
- Negalima pridėti duomenų, jei buvo bent viena aukščiau išvardinta taisyklė.

DROP VIEW VARDAS;
Komanda panaikina vaizdą iš duomenų bazės. Duomenys bazėse lieka nepakeisti, nes vaizdas realiai neegzistuoja, o tik atspindi duomenis.

SPECIALIOS PASKIRTIES ŽODYNAS - DICTIONARY

Tai yra vienas iš svarbiausių ORACLE serverio komponentų. Jis susideda iš lentelių ir vaizdų ir kaupia informaciją apie duomenų bazę. Ją gali pateikti vartotojui. Duomenų žodyno lentelės yra pirmieji objektai, kurie yra sukurti duomenų bazėje, ir jos turi būti pateiktos visiems kitiems kuriamiems objektams. Duomenų žodyno lentelės yra automatiškai kuriamos SQL operatoriumi **CREATE DATABASE.** Šis žodynas priskiriamas vartotojui SYS. Bazinės lentelės tiesiai prieinamos retai, nes informacija jose nėra lengvai suprantama. Informacija iš ten pateikiama per vaizdus. Bet kada, kai tik yra kreipiamasi į DB, šis žodynas visada yra atnaujinamas. Šiuo žodynu gali naudotis visi DB vartotojai. Tai yra vertingas informacijos šaltinis vartotojui, aplikacijų kūrėjui ir DB administratoriui. Duomenų žodynas taip pat reaguoja į ORACLE Serverio operacijas, kurios realizuoja įrašymo ir tikrinimo informaciją apie duomenų bazę. Viešas priėjimas prie duomenų žodyno yra galimas per vaizdus ir vertingesnis nei per bazines lenteles. Duomenų žodyno vaizdai, atspindi informaciją lentelės formate, kuris yra lengvai suprantamas.Vartotojas SYS taip pat valdo duomenų žodyno vaizdus.

Iš šio žodyno galima sužinoti:

- ORACLE vartotojų vardus
- Vartotojų teises ir privilegijas, kurios jiems yra taikomos
- Duomenų bazių objektų vardus (lentelių, vaizdų, indeksų, sinonimų, sekų.)

- Apribojimus, taikomus ORACLE objektams
- Peržiūros informaciją, t.y. kas buvo prisijungęs arba atnaujino Serverį.

Yra du keliai prieiti prie duomenų žodyno.

- Vartotojai gali prieiti prie Duomenų žodyno objektų per **SELECT** komanda, paprastai organizuodami užklausą iš atitinkamų lentelių.
  - ORACLE Serveris automatiškai atnaujina duomenų žodyną, atspindėdamas pakeitimus duomenų struktūrose, leidimuose, peržiūroje ir kituose duomenyse. Per duomenų operacijas Serveris taip pat skaito duomenų žodyną, kad nustatyti, jog duomenų bazės objektai egzistuoja ir kad vartotojai gali prieiti prie jų.

Retai kada kreipiamasi tiesiogiai į žodyno lenteles. Paprastai yra sukurt vaizdai, kuriuose saugoma ta pati informacija, lengvai suprantama vartotojui.
Žodynui peržiūrėti galima taikyti tas pačias komandas kaip ir paprastai lentelei.

❖ **DESRIBE DICTIONARY**; parodo duomenų žodyno struktūrą, t.y. nurodo kokie stulpeliai sudaro duomenų žodyno lentelę. Rezultate bus parodyta žodyno struktūra:

TABLE_NAME Varchar2(20)
COMMENTS Varchar2(2000)

Norint sužinoti **USER_OBJECTS** vaizdo struktūrą

❖ **DESCRIBE USER_OBJECTS**;
Rezultate matysime:

OBJECT_NAME VARCHAR2(120)
OBJECT_ID NUMBER
OBJECT_TYPE VARCHAR2(13)
CREATED DATE
LAST_DDL_TIME DATE
TIMESTUP VARCHAR2(75)
STATUS VARCHAR2(7)

Toliau jau įprastu būdu, žinant lentelės ar vaizdo struktūrą, galima gauti informaciją apie objektą, naudojant komandą **SELECT**.

❖ **SELECT OBJECT_NAME,OBJECT_TYPE,CREATED**
FROM USER_OBJECTS
WHERE OBJECT_NAME=‘EMP’; galime sužinoti visą informaciją apie objektą vadu “EMP”.

❖ **SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME, DELETE_RULE**
FROM USER_CONSTRAINTS
WHERE TABLE_NAME=‘COMPANY_CARS’;

Rezultate ekrane parodys apribojimų vardus ir apribojimų tipus, kuriuos vartotojas suformavo, projektuodamas lentelę **COMPANY CARS**.

Duomenų žodyno Lentelė ir Vaizdai

Vaizdai gali būti įvardijami, kad atspindėtų vartotojo tipą, kuriam jis yra adresuotas. Vaizdai yra klasifikuojami į tris grupes, kurios yra skiriamos viena nuo kitos prefiksais USER, ALL, DBA.

USER_xxxx	Objektai, kuriuos gali pasiekti tik šių objektų savininkas. Pvz. vaizdai, kurių prefiksas nurodo vartotoją parodo informaciją apie vartotojo sukurtas lenteles ir privilegijas, suteiktas vartotojui.
ALL_xxxx	Vartotojai gali pasiekti objektus, į kuriuos jie turi teisę, papildomai prie savo nuosavų objektų.
DBA_xxxx	Vartotojai, kurie turi Duomenų Bazės administratoriaus privilegijas- gali pasiekti bet kokius objektus DB
V$xxxx	Vaizdai, rodantys duomenų bazės serverio darbą, apsaugą

Kai kurie žodyno vaizdai nenaudoja jokių prefiksų. Tai yra:

DICTIONARY	atspindi visas žodyno lenteles, vaizdus, sinonimus,prieinamas bet kokiam vartotojui
DICT_COLLUMNS	atspindi stulpelius žodyno objektuose
CONSTRAINT_DEFS	atspindi visus apribojimus aprašytus lentelėms, kurios yra prieinamos vartotojams
CONSTRAINT_COLLUMNS	atspindi stulpelius, kurie yra prieinami esamam vartotojui ir įvardintus apribojimų apibrėžimuose

- **SELECT * FROM DICTIONARY**; atvaizduoja visus duomenų žodyno vardus objektų, prieinamų vartotojui, su trumpa objekto charakteristika komentarų stulpelyje.

- **SELECT * FROM DICTIONARY**
  WHERE TABLE_NAME='ALL_USERS';

TABLE_NAME -lentelės vardas

ALL_USERS -Informacija apie visus DB vartotojus
All_OBJECTS - rodo visus vartotojui prieinamus objektus (table,view,index ir t.t.).

- **DESCRIBE ALL_OBJECTS;**

```
Name                          Null?    Type
------------------------------ -------- ----
OWNER                          NOT NULL     VARCHAR2(30)
OBJECT_NAME                    NOT NULL     VARCHAR2(30)
OBJECT_ID                      NOT NULL     NUMBER
OBJECT_TYPE                                 VARCHAR2(12)
CREATED                        NOT NULL DATE
LAST_DDL_TIME                  NOT NULL DATE
TIMESTAMP                      VARCHAR2(75)
STATUS                         VARCHAR2(7)
```

- **SELECT * FROM ALL_OBJECTS**
  WHERE OWNER='VEROBEJK';

```
OWNER     OBJECT_NAME  OBJECT_ID OBJECT_TYPE   CREATED
LAST_DDL_ TIME   TAMESTAMP   STATUS
-------------------------------------------------------------------------------------------
VEROBEJK        A              16336          TABLE       15-DEC-96       15-DEC-96
1996-12-15:17:57:58        VALID
```

ALL_USERS - informacija apie visus vartotojus (turinčius teises dirbti konkrečiame tinkle).

```
SQL> DESCRIBE ALL_USERS;
Name                          Null?    Type
------------------------------ -------- ----
USERNAME                      NOT NULL    VARCHAR2(30)
USER_ID                       NOT NULL    NUMBER
CREATED                       NOT NULL    DATE
```

- **SELECT * FROM ALL_USERS**
  WHERE USERNAME='ESPADA';

```
USERNAME                       USER_ID CREATED
------------------------------ --------- ---------
ESPADA                          75          26-AUG-96
```

USER_CATALOG - atspindi informaciją apie visas lenteles, vaizdus ir t.t..

- **DESCRIBE USER_CATALOG;**

```
Name                          Null?          Type
------------------------------ -------- ----
TABLE_NAME                    NOT NULL VARCHAR2(30)
TABLE_TYPE                             VARCHAR2(11)
```

- **SELECT * FROM USER_CATALOG ;**

```
TABLE_NAME                     TABLE_TYPE
------------------------------ -----------
A                               TABLE
B                               TABLE
KLIENTAI                        TABLE
ONE                             TABLE
UZSAKYMAI                       TABLE
VARDAS                          VIEW
VARDAS1                         VIEW
```

USER_OBJECTS - informacija apie vartotojo objektus.

```
SQL> DESCRIBE USER_OBJECTS;
Name                           Null?    T   ype
------------------------------ -------- ----
OBJECT_NAME                    VARCHAR2(128)
OBJECT_ID                      NUMBER
OBJECT_TYPE                    VARCHAR2(13)
CREATED                         DATE
LAST_DDL_TIME                  DATE
TIMESTAMP                      VARCHAR2(75)
STATUS                          VARCHAR2(7)
```

```
SQL> SELECT * FROM USER_OBJECTS;

OBJECT_NAME OBJECT_ID OBJECT_TYPE    CREATED
LAST_DDL_TIMESTAMP STATUS
--------- ------------- --------- ----------------
A                               TABLE           15-DEC-96    15-DEC-96
VALID
B                               TABLE           15-DEC-96   15-DEC-96
VALID
KLIENTAI                        TABLE           15-DEC-96   15-DEC-96
```

USER_TABLES - informacija apie vartotojo lenteles.

SQL> **DESCRIBE USER_TABLES;**

Name	Null?	Type
TABLE_NAME	NOT NULL	VARCHAR2(30)
TABLESPACE_NAME	NOT NULL	VARCHAR2(30)
CLUSTER_NAME		VARCHAR2(30)
PCT_FREE		NUMBER
PCT_USED	NOT NULL	NUMBER
INI_TRANS	NOT NULL	NUMBER
MAX_TRANS	NOT NULL	NUMBER
INITIAL_EXTENT		NUMBER
NEXT_EXTENT		NUMBER
MIN_EXTENTS		NUMBER
MAX_EXTENTS		NUMBER
PCT_INCREASE		NUMBER
FREELISTS		NUMBER
FREELIST_GROUPS		NUMBER
BACKED_UP		VARCHAR2(1)
NUM_ROWS		NUMBER
BLOCKS		NUMBER
EMPTY_BLOCKS		NUMBER
AVG_SPACE		NUMBER
CHAIN_CNT		NUMBER
AVG_ROW_LEN		NUMBER
DEGREE		VARCHAR2(10)
INSTANCES		VARCHAR2(10)
CACHE		VARCHAR2(5)
TABLE_LOCK		VARCHAR2(8)

USER_TAB_PRIVS - rodo prieinamus vartotojui objektus.

SQL> **DESCRIBE USER_TAB_PRIVS;**

Name	Null?	Type
GRANTEE	NOT NULL	VARCHAR2(30)
OWNER	NOT NULL	VARCHAR2(30)
TABLE_NAME	NOT NULL	VARCHAR2(30)
GRANTOR	NOT NULL	VARCHAR2(30)
PRIVILEGE	NOT NULL	VARCHAR2(40)
GRANTABLE		VARCHAR2(3)

USER_VIEWS - informacija apie vartotojo sukurtus vaizdus.

```
SQL>  DESCRIBE USER_VIEWS;
Name                           Null?          Type
------------------------------ -------- ----
VIEW_NAME                      NOT NULL VARCHAR2(30)
TEXT_LENGTH                             NUMBER
TEXT                                    LONG
SQL> SELECT * FROM USER_VIEWS;
```

USER_TABLESPACES - lentelei prieinamos vietos aprašymas.

```
SQL> DESCRIBE USER_TABLESPACES;
Name                           Null?          Type
------------------------------ -------- ----
TABLESPACE_NAME   NOT NULL VARCHAR2(30)
INITIAL_EXTENT                          NUMBER
NEXT_EXTENT                             NUMBER
MIN_EXTENTS           NOT NULL   NUMBER
MAX_EXTENTS           NOT NULL  NUMBER
PCT_INCREASE          NOT NULL   NUMBER
STATUS                                  VARCHAR2(9)
CONTENTS                                VARCHAR2(9)
```

SQL> SELECT * FROM USER_TABLESPACES;

Ta pati SQL kalba gali būti galinga priemonė SQL konstrukcijų generavimui,tam naudojant informaciją iš Duomenų žodyno. Taip darbuojantis galima:

- išvengiame įkyraus kodavimo
- gauname pagalbą
- galime perkurti lenteles
- formuoti “dinamines” sąlygas , kuriuose saugomi vykdymų sąlygų parametrai.

Bendrai naudojant DBA ir kūrėjams galima sukurti skriptus (komandinius failus), kurie vykdomi SQL*PLUS aplinkoje ir

- atlieka bendrą lentelių naikinimą
- perkuria DML konstravimą
- priskiria vartotojui privilegijas

Kaip pavyzdį mes galime aprašyti skriptą, kuris išveda visas vartotojo sukurtas lenteles. Tai galima atlikti dviem būdais:

- SELECT ‘DESC’||OBJECT_NAME
  FROM USER_OBJECTS
  WHERE OBJECT_TYPE=‘TABLE’;

arba

❖ SELECT 'DESC'||TABLE_NAME
FROM USER_TABLES;

Tai turėtų būti sutvarkyta taip, kad jį būtų galima įjungti į eilę, kurioje po to gali būti vykdomas skriptas.

Daugelis duomenų žodyno vaizdų turi ilgus vardus. Vieši sinonimai gali būti pasiūlyti daugeliui dažnai bendram naudojamų duomenų žodynų vaizdų.

Viešai naudojami sinonimai:

CAT	USER_CATALOG
CLU	USER_CLUSTERS
COLS	USER_TAB_COLUMNS
DICT	DICTIONARY
IND	USER_INDEXES
OBJ	USER_OBJECTS
SEQ	USER_SEQUENCES
SYN	USER_SYNONYMS
TABS	USER_TABLES

VARTOTOJAII IR DUOMENŲ APSAUGA

DCL tipo komandos

Esant daugiavartotojiškai sistemai jums reikia pasirūpinti vartotojų priėjimu prie duomenų. Duomenų apsauga rūpinasi serveris. Tam yra sukurta standartinė priėjimo prie duomenų valdymo sistema. Tai reiškia, kad duomenų bazės administratorius suteikia leidimus visiems duomenų bazės vartotojams ir suteikia jiems įgaliojimus atlikti duomenų bazėje konkrečius veiksmus ir su konkrečiais duomenimis.
Yra sukurti skirtingi duomenų apsaugos metodai. Apie apsaugos sistemą galima sakyti kad ji yra decentralizuota. Serveris :

- Kontroliuoja priėjimą prie DB
- Leidžia pasiekti specifinius DB objektus
- Patvirtina privilegijas užfiksuotas Duomenų žodyne
- Sukuria sinonimus DB objektams

DB saugumas skirstomas į dvi kategorijas. Sistemos apsauga ir duomenų apsauga. Sistemos apsauga apima priėjimą prie sistemos ir duomenų sistemos lygyje nustatant vartotojo vardą ir slaptažodį, diskinę erdvę ir sistemines operacijas leidžiamas vartotojui. Duomenų saugumas apima priėjimą prie DB objektų ir veiksmus kuriuos vartotojas gali atlikti su objektais.

Privilegijos

Privilegijos yra teisė vykdyti ypatingus SQL operatorius. DB administratorius yra aukščiausio lygio vartotojas su galimybe leisti vartotojui pasiekti DB objektus. Vartotojų reikia sisteminių privilegijų pasiekti DB ir objektų privilegijas kad manipuliuoti objektais DB-ėje. Vartotojams gali būti suteikta papildoma privilegija suteikti privilegijas kitiems vartotojams ar roles, kurios yra vadinamos grupinėmis.

Bendras leidimas prieiti prie duomenų bazės.

Šį leidimą suteikia DB administratorius, užregistruodamas vartotojus ir sukurdamas duomenų bazėje naujus vartotojus. Užregistruodamas vartotoją, administratorius nurodo jo vardą ir slaptažodį. Tik po to vartotojas gali pradėti dirbti su duomenų baze prisijungdamas prie jos, nurodant savo vardą ir slaptažodį. Tada vartotojui taikomi sisteminai reikalavimai per visą prisijungimo seansą dirbant su DB objektais. Kartą vartotojo sukurtas objektas, kuriuo vėliau vartotojas naudojasi su visom teisėm, gali būti perleistas kitiems vartotojams, nurodant privilegijas kitam vartotojui.

Įgaliojimų išplėtimas ir apribojimas.

Bazės administratorius gali vartotojui suteikti įgaliojimus dirbti tik su informacija, kuri skirta jam. Tokiu būdu vartotojas gali kreiptis tiktai į jam priskirtas lenteles ir atlikti su tom lentelėm tiktai jam skirtus veiksmus.
Pats administratorius gali kreiptis į bet kurias duomenų bazės lenteles ir gali atlikti bet kuriuos veiksmus. Administratorius gali nustatyti, kokie vartotojai gali kurti lenteles ir atvaizdus, kokie vartotojai gali kurti lentelių sritis ir jas koreguoti ir kokie gali taisyti duomenis lentelėse. Privilegijos į objektą gali būti atšauktos. Privilegijų rūšys Jų yra per 80.

DBA turi aukščiausias sistemines privilegijas

- Sukurti Vartotoją
- Panaikinti vartotoją
- Panaikinti bet kokią lentelę
    - Palaikyti bet kokias lenteles naudojant eksporto utilites

Sukurti vartotoją naudojame CREATE USER *User* IDENTITY BY *Pasword*

Kai tik vartotojas sukuriamas, jam priskiriamos ir tipinės vartotojo sisteminės privilegijos:

- CREATE SECTION prisijungti prie bazės
- CREATE TABLE sukurti lentelę savo schemoje
- CREATE SEQUENCE Sukurti seką savo schemoje
- CREATE VIEW sukurti vaizdą savo schemoje
- CREATE PROCEDURE Sukurti procedūrą.
- ALTER SESION

DBA sukurti vartotojo privilegijoms naudoja komandą **GRANT**

- GRANT Create Table,Create Sequence,Create View To SCOTT;

Naudojant komandą GRANT galima priskirti vartotojui privilegijas: Jos bendra sintaksė:

GRANT Privilegijos TO (nurodome kam [vardas] suteikiame privilegijas);

Jei norime sukurti dar papildomų privilegijų, vėl naudojamės komanda GRANT. Privilegijos atskiriamos kableliu. Pabaigoje dedame kabliataškį.

GRANT CREATE SECTION,CREATE TABLE, CREATE SEQUENCE,CREATE VIEW,ALTER SESION TO PETRAS,JONAS;

(PETRAS,JONAS- vartotojų vardai)

Rolės

Rolės - yra rinkinys bendrų privilegijų, kuriuos administratorius gali priskirti įvairiems vartotojams. Taip palengviname savo darbą, suteikiant ar atšaukiant tas pačias privilegijas keletui vartotojų.
Vartotojui gali būti priskiriamos kelios rolės, ir taip pat keletui vartotojų gali būti prieinama ta pati rolė, vadinasi tai yra patogi ir laisva apsaugos sistema.
Vartotojui registruojantis, jam yra priskiriamos rolės pagal nutylėjimą. Darbo metu galima persijungti nuo vienos rolės prie kitos, jei vartotojas turi keletą rolių. Sukurti rolę galima naudojant komandą :

- CREATE ROLE Rolės_vardas

Po to kai rolė yra sukurta, DBA gali naudoti komandą GRANT priskirti vartotoją šiai rolei taip pat kaip priskirti privilegijas rolei.

- CREATE ROLE Manager;
- GRANT Create Table,Create View TO Manager;
- GRANT Manager To BLACKE,CLARK;
-

Sukuriama rolė Manager, po to jai priskiriamos privilegijos kurti lenteles ir vaizdus, o po to ši rolė priskiriama dviem vartotojams BLACKE ir CLARK.

Įgaliojimai kreiptis į atskirus duomenų bazės objektus

Objektai - bet kas, kas yra pačioje duomenų bazėje (lentelės, atvaizdai, rolės, procedūros, vartotojai ir visi kiti).
Administratorius gali nurodyti kas gali naudotis viena ar kita lentele, taip pat administratorius gali nurodyti kaip vartotojas gali elgtis su tom lentelėm.
Administratorius gali suteikti įgaliojimus dėl komandų:

SELECT - duomenų išrinkimas iš bet kurios lentelės.
INSERT - įterpti duomenis į nurodytas lenteles.
UPDATE - galimybė pataisyti duomenis lentelėse.
DELETE - galimybė išmesti eilutę iš lentelės.
ALTER- stulpelių apibrėžimai lentelėse.
INDEX-
REFERENCES - kreiptis į lentelę per lentelės lygio ar stulpelio lygio apribojimus.
ALL - visas aukščiau išvardintas galimybes

Objektų privilegijos

Objektų privilegijos skirtingiems objektams yra skirtingos Savininkas turi visas privilegijas į savo nuosavą objektą. Savininkas gali suteikti specifines privilegijas savo nuosaviems objektams
Objekto privilegijos

Objektų privilegijos	Lentelės	Vaizdai	Sekos (Sequence)	Procedūros
ALTER	3		3	
DELETE	3	3		
EXECUTE				3
INDEX	3			
INSERT	3	3		
REFERENCES	3			
SELECT	3	3	3	
UPDATE	3	3		

Objektų privilegijos yra teisė atlikti veiksmus su objektais ar procedūrom.
GRANT Objekto_privilegijos[(stulpeliai)]
ON Objektas
TO {Vartotojas|Rolė|PUBLIC}
[WITH GRANT OPTION];
ALL reiškia visas privilegijas
PUBLIC suteikia privilegijas visiems vartotojams
WITH GRANT OPTION –leidžia garantuoti objektų privilegijų garantijas kitiems vartotojams ir rolėms.
GRANT SELECT
ON Emp TO SUE,RICH
GRANT UPDATE (Dname,Loc)
ON DEPT TO Scott,Manager;
Kad suteikti privilegijas objektams, objektas turi būti schemoje arba turi būti suteiktos privilegijos su WITH GRANT OPTION.
Savininkas gali suteikti bet kokias privilegijas bet kokiam vartotojui ar rolei.
GRANT SELECT,INSERT
ON Dept TO Scott
WITH GRANT OPTION; suteikia Scott privilegijas ir galimybę jas perduoti kitam vartotojui.

GRANT SELECT
ON Alice.DEPT TO PUBLIC . Alice- savininkė, t.y. jai priklauso DEPT lentelė.

Viešos privilegijos nuorodos naudojant PUBLIC paragrafą. Visi vartotojai turi teisę į nurodytą objektą.

- CREATE ROLE Rolės_vardas
- GRANT SELECT, INSERT, UPDATE, DELETE
ON (lentelių pavadinimai) TO (kam suteikiame rolę) ;

PVZ: Dirbame su buhalteriniais uždaviniais. Kiekvienam iš 100 vartotojų suteikiame tas pačias privilegijas. Kad to nereikėtų kartoti kiekvienam atskirai, galime sukurti vieną privilegiją aštuonioms lentelėms ir suteikti privilegiją kitiems naudotis šiomis lentelėmis.

- GRANT SELECT, INSERT
  ON LENT1
  TO buhal ;

- GRANT buhal TO VART1, VART2, VART3 ;

Tai yra skirtingos privilegijos skirtingoms lentelėms. Kad suteiki vartotojui teises, vartotojas, kuriam ji yra prieinama, negali jos perduoti kitam vartotojui. Tačiau yra galimybė ir čia pakeisti šią nuostatą per **WITH GRANT OPTION**.

Sisteminiai įgaliojimai

Sisteminiai įgaliojimai suteikiami tuo atveju, kai norime atlikti veiksmus duomenų bazės lygyje. Tai gali būti:

ALTER DATABASE - leidžia keisti duomenų bazės struktūrą, prisijungiant naujus failus.
DROP TABLESPACE - su šituo įgaliojimu galima pašalinti bet kurią lentelės sritį išskyrus sritį SYSTEM.
SELECT ANY TABLE - šis įgaliojimas duoda galimybę kreiptis į bet kurią bazės lentelę.

Kiekvienas vartotojas turi savo slaptažodį, kurį jam priskyrė DBA registruodamas vartotoją. Pats vartotojas gali pasikeisti savo slaptažodį.

- **ALTER USER** vart_vardas **IDENTIFIED BY** slaptažodis;

Kaip galima suteikti privilegijas, taip galima jas ir atšaukti. Tam naudojame komandą **REVOKE. Komanda** atšaukia visas privilegijas kaskadiškai pradedant nurodytu lygiu ir žemesnius iki pat pabaigos.

REVOKE privilegijos...
ON lentelė...
FROM vartotojas...

Informacija apie privilegijas saugoma duomenų žodyne. Galima sužinoti kurie vartotojai turi privilegijas į tavo lenteles, vaizdus. Reikia parašyti eilutę užklausai iš žodyno, naudojant vaizdus USER_TAB_GRANTS ir USER_COL_GRANTS.
Kreipiantis į lentelę, reikia nurodyti pilną jos vardą. Jei ši lentelė nėra jūsų nuosavybė, tai dar būtina nurodyti ir lentelės savininką per tašką. Galima sukurti sinonimą, kuris reiškia "kitą vardą kažkam kitam".

Nėra patartina naudoti sinonimus aplikacijose.
Kreipiamės į kito vartotojo lentelę per

- SELECT * FROM SCOTT.EMP;

Sekos, indeksai bei sinonimai

Sekų kūrimas
Kas tai yra sekos. sekų generatorius gali būti naudojamas automatiškai generuoti numerių seką lentelėse. Seka yra DB objektas, kuriamas vartotojo ir gali būti pasidalintas tarp keleto vartotojų. Tipiškas jo panaudojimas yra pirminio rakto reikšmės generavimas kiekvienai lentelės eilutei. Seka yra generuojama didinant arba mažinant numerį per vidinę ORACLE sistemą. Tai gali būti laikinai saugomas objektas, kadangi tai gali sumažinti aplikacijos kodo dydį, kuris reikalingas parašyti sekos generavimo tvarkai. Sekos numeriai yra saugomi ir generuojami nepriklausomai nuo lentelės, todėl ta pati seka gali būti panaudota keliose lentelėse.
Sekos generuojamos nes jos:
automatiškai generuoja unikalius numerius
yra pasidalijamas objektas
tipiškai yra naudojamos kuriant pirminio rakto reikšmes.
pakeičia aplikacijos kodą
pagreitina pasiekimo sekos reikšmių efektyvumą kai yra patalpintos atmintyje.

Aprašyti sekos generavimą automatiškai galime su komanda

CREATE SEQUENCE Seka
[INCREMENT BY n]
[START WITH n]
[{MAXVALUE n |NOMAXVALUE}]
[{MINVALUE n| NOMINVALUE}]
[{CYCLE|NOCYCLE}]
[{CACHE n| NOCACHE}];

Automatiškai generuojami skaičiai.
INCREMENT BY n Nustato intervalą tarp sekos elementų, kur n yra skaičius. Jei šis paragrafas yra praleistas, tai elemento reikšmės bus didinamos vienetu.

START WITH n Nustato pirmo sekos elemento reikšmę. Jei šis paragrafas yra praleistas, tai elemento reikšmės bus didinamos vienetu.
MAXVALUE n Nustato didžiausią generuojamos sekos numerio reikšmę
NOMAXVALUE Nustato maksimalią generuojamos sekos numerio reikšmę 10^27 sekos didėjimo atveju ir –1 sekos mažėjimo atveju. Tai nusistato pagal nutylėjimą.
MINVALUE n Nustato minimalią sekos elemento reikšmę
NOMINVALUE Nustato minimalią n reikšmę 1 didėjimo atveju ir - 10^26 mažėjimo atveju. Tai yra reikšmė pagal nutylėjimą.
CYCLE|NOCYCLE Nustato, jog pasiekus tam tikrą reikšmę seka toliau generuojama arba ne.
CACHE n| NOCACHE Nustato kiek reikšmių ORACLE Serveris talpins atmintyje Pagal nutylėjimą tai yra 20 reikšmių.

Pvz.
Reikia sukurti seką vardu **DEPT_DEPTNO** kuri būtų naudojama pirminio rakto DEPT lentelėje kūrimui. Nereikia naudoti CYCLE parametro.

```
CREATE SEQUENCE DEPT_DEPTNO
INCREMENT BY 1
START WITH 91
NOMAXVALUE 100
NOCACHE
NOCYCLE;
```

Šią seką bus galima naudoti Deptno generavimui pildant lentelę DEPT.
CYCLE negalima naudoti, jei seka naudojama pirminio rakto generavimui.

Sekos patvirtinimas

Kai tik seka sukuriama, ji yra dokumentuojama duomenų žodyne. Nuo to laiko kai seka tampa DB objektu, jūs galite identifikuoti jį kaip USER_OBJECTS duomenų žodyne. Taip pat galite patvirtinti priskyrimus, išrenkant tai iš DB žodyno. Galima patikrinti sekos reikšmes duomenų žodyno vaizde USER_SEQUENCES.

```
SELECT Sequence_name,Min_value,Max_value,Increment_by,Last_number
FROM User_Sequences
```

Last_number stulpelis parodo sekantį galimą sekos numerį.

Kai tik jūs sukūrėte seką, jūs galite naudoti sekos generatorių generuoti numerius ir juos panaudoti savo lentelėse. Remtis sekos reikšmėmis galima naudojant NEXTVAL ir CURVAL pseudo stulpelus.
NEXTVAL naudojama ištraukti eilinį sekos numerį iš aprašytos sekos. Jūs turite aprašyti NEXTVAL su sekos vardu. Kai kreipiatės Seka.NEXTVAL, naujas sekos numeris yra generuojamas, o esamas numeris yra perkeliamas į CURVAL pseudo stulpelį.

CURVAL nustato esamą sekos reikšmę. Ji yra naudojama nustatyti sekai numerį, kurį esamas vartotojas ką tik sugeneravo. NEXTVAL turi būti naudojamas sekos numeriui generuoti esamo vartotojo seanso metu prieš tai kai CURVAL bus galima naudoti. Jūs turite aprašyti CURVAL su sekos vardu. Kai Seka.CURVAL yra nurodoma, paskutinė reikšmė yra grąžinama tam vartotojo procesui, kuris yra rodomas.
Yra taisyklės kaip naudoti NEXTVAL ir CURVAL. Jos naudojamos sekančiais atvejais:

SELECT sąraše SELECT komandoje, kuri nėra subužklausos atveju.
SELECT sąraše subužklausoje INSERT komandoje
VALUES paragrafe komandoje INSERT
SET paragrafe komandoje UPDATE

Jūs negalite naudoti NEXTVAL ir CURVAL sekančiais atvejais:

SELECT komandoje dėl vaizdų
SELECT komandoje su DISTINCT parametru
SELECT komandoje su GROUP BY,HAVING, ar ORDER BY
Subužklausoje SELECT, DELETE, ar UPDATE.
DEFAULT išraiškose su CREATE TABLE ar ALTER TABLE.

Pavyzdys, kurį aprašysime parodys kaip įterpti naują eilutę į lentelę DEPT. Naudosime Seką DEPT_DEPTNO pirminio rakto generavimui.

INSERT INTO DEPT(Deptno,Dname,Loc)
VALUES(DEPT_DEPTNO.NEXTVAL,’MARKETING’,’SAN DIEGO’);

Galime pradžioje pasitikrinti einamąją reikšmę sekai

SELECT DEPT_DEPTNO.CURVAL FROM Dual;
Ta reikšmė yra 91. Manykime, kad jūs norite pridėti darbuotojų iš naujo departamento. INSERT komanda kuri gali būti panaudota pakartotinai naujiems darbuotojams įterpti gali įtraukti sekantį kodą:

INSERT INTO EMP …
VALUES9EMP_EMPNO.NEXTVAL,DEPT_DEPTVAL.CURVAL,…);
Manoma, kad EMP_EMPNO seka jau yra sukurta iki to. Ji skirta darbuotojo numeriui generuoti vis keičiant reikšmę, kai tuo tarpu departamento numeris naudojant kitą seką ir kitokį parametrą lieka nepakitęs. Patalpinti sekas atmintyje reiškia pagreitinti priėjimą prie sekų reikšmių. Tai atliekame pirmą kart aprašant seką. Kiekvienas reikalavimas sekančios sekos reikšmės tai yra jos pasiėmimas iš sekos atmintyje. Po paskutinio sekos panaudojimo , sekantis sekos reikalavimas įtrauks kitą seką į atmintį.

Kaip apsisaugoti nuo spragų sekoje.
Nors sekos generatoriaus rezultatas yra numeriai be tarpų, šis veiksmas yra nepriklausomas nuo COMMIT ar ROLBACK. Tačiau jei jūs grįšite atgal per komandą, kuri generuoja seką, numeris bus pamestas.
Kitas įvykis kuris gali sudaryti spragas yra sistemos žlugimas. Jei sekos reikšmės yra atmintyje, tai šios reikšmės yra pamestos per sistemos žlugimą.
Kadangi sekos nėra tiesiogiai surištos su lentelėmis, ta pati seka gali būti panaudota kelioms lentelėms. Jei tai atsitinka, kiekviena lentelė gali turėti numerių spragų sekoje.
Galima atvaizduoti sekantį galimą sekos numerį, nepadidinant jo reikšmės tik tuo atveju, jei seka buvo sukurta su paragrafu NOCACHE atvaizduojant tai iš USER_SEQUENCES.
Yra galimybė taisyti tam tikras reikšmes aprašytas sekos kūrimo metu. Tai MAXVAL, MINVAL, INCREMENT reikšmė, CYCLE ar CACH opcijas.

```
ALTER SEQUENCE DEPT_DEPTNO
INCREMENT BY 1
MAXVALUE 99999
NOCACHE
NOCYCLE;
```

Kad taisyti sekos aprašymą, jūs turite būti tos sekos savininkas arba turėti privilegijas tai atlikti. Tik naujos reikšmės bus paveiktos. Seka turi būti panaikinta arba perkurta kad pradėti seką nuo naujo numerio.

Panaikinti seką galima su komanda DROP SEQUENCE
DROP SEQUENCE DEPT_DEPTNO panaikins seką nurodytu vardu. Panaikinti seką gali tik jos savininkas, ar vartotojas kuriam suteikta privilegija .

DROP ANY SEQUENCE.

Indeksų kūrimas

Indeksas yra objektas kuris gali pagreitinti priėjimą prie duomenų, pateikiant eilutes iš lentelės vartotojui. Jie gali būti sukuriami automatiškai ar išoriškai. Jei nėra indekso priskirto stulpeliui tai pilna lentelė bus peržiūrima.

Indeksas pasirūpina tiesioginį ir greitą priėjimą prie eilučių lentelėse. Jų tikslas yra pašalinti diskų I/O operacijas, tam panaudojant indeksų dalį . Indeksus automatiškai naudoja ir palaiko ORCLE Serveris. Kai jau kartą indeksai yra sukurti, jokių vartotojo tiesioginių veiksmų nereikia.

Indeksai yra logiškai ir fiziškai nepriklausomi nuo lentelės kuriai jie sukurti. Tai reiškia kad jie gali būti sukurti ar panaikinti bet kuriuo laiku ir neturi įtakos bazinei lentelei ar kitiems indeksams. Kai lentelė panaikinama, surišti su ja indeksai taip pat panaikinami.Taigi indeksai kuriami automatiškai ar rankiniu būdu. Automatiškai jie kuriami kai jūs nustatote pirminį raktą arba stulpelio reikšmes apibūdinate kaip unikalias. Rankiniu būdu galima suskurti indeksą dėl neunikalių stulpelio reikšmių kad pagreitinti priėjimą prie eilučių.

Indeksas gali būti kuriamas vienai ar keliems stulpeliams.

CREATE INDEX Indeksas ON Lentelė (Stulpelis1[,stulpelis2]…);

Padidinti užklausos greitį galima per Ename lentelei EMP.

CREATE INDEX Emp_Ename_idx ON EMP(Ename);

Daug indeksų lentelei nereiškia kad tai pagreitins darbą. Kiekviena DML operacija kuri baigiama per COMMIT lentelei su indeksais reiškia kad ir indeksai turi būti atnaujinami. Daugiau indeksų kuriuos surišate su lentele, daugiau pastangų Serveris turi kad atnaujinti indeksus po DML.

Indeksai kuriami kai:

- Dažnai stulpeliai naudojami paragrafe WHERE ar apjungimo sąlygoje
- stulpelis talpina plačią reikšmių eilę.
- stulpelis talpina didelį kiekį NULL reikšmių
- Du ar daugiau stulpelių dažnai naudojamos kartu WHERE paragrafe ar apjungimo sąlygoje
- Lentelė yra didelė ir daugumoje užklausų yra pateikiami 2-4% visų lentelės eilučių.

Negalima kurti indeksų:

- Kai lentelė yra maža
- Stulpeliai nėra dažnai vartojami kaip sąlyga užklausoje
- Dauguma užklausų tikisi ištraukti daugiau nei 2-4% visų lentelės eilučių
- Lentelė dažnai yra atnaujinama.

Kaip paprastai vartotojo sukurti indeksai saugomi Duomenų žodyne. Juos galima rasti per USER_INDEXES vaizdą.

USER_IND_COLUMNS vaizdas saugo indekso vardą, lentelės vardą ir stulpelio vardą. Pamatyti galima paprastai naudojant komandą SELECT.

SELECT ic.Index_name, ic.column_name,ic.column_postion col_pos,ix.uniqueness FROM User_indexes ix,user_ind_column ic WHERE icindex_name= ix.index_name AND ic.table_name= ’EMP’;

Patvirtina USER_INDEXSES vaizdą iš duomenų žodyno. Taip pat galite patikrinti stulpelius, kurios apima indeksus peržiūrint USER_INDEX_COLUMNS vaizdą.

Ix ir ic čia yra kaip alternatyvūs vaizdų vardai. Pašalinti indeksus galima iš Duomenų žodyno per DROP INDEX *Indeksas.*

DROP INDEX Emp_Ename_Index;

Sinonimų kūrimas

Sinonimai supaprastina kreipimąsi į objektus.(pagal kitus vardus). Kreipiasi į lentelę kurios savininkas yra kitas vartotojas. Sutrumpina ilgį objekto vardo. Kreipiantis į lentelę, kuri priklauso kitam vartotojui, reikia per prefiksą nurodyti vartotojo vardą. Kuriant sinonimą ši būtinybė eliminuojama. Taip galima turėti alternatyvų vardą lentelei, vaizdui, sekai, procedūrai, ar kitam objektui. Tas metodas gali būti naudingas trumpinant objekto vardus.

Galima sukurti sinonimą

CREATE SYNONYM EMP
FOR SCOTT.EMP;
Tada naudojam sinonimą SELECT * FROM EMP;

Sinonimai yra naudingi ir tada, kai lentelių vardai yra ilgi.

Vieši sinonimai- juos gali sukurti tik DBA.

CREATE PUBLIC SYNONYM Sinonimo_vardas
FOR KAM_vardas
Objektų_vardai...;

CREATE [PUBLIC] SYNONIM *Sinonimas* FOR *Objektas*

PUBLIC -sukurią sinonimą, kuriuo gali naudotis bet kuris vartotojas.

Taip pat DBA gali atšaukti viešus sinonimus.

DROP [PUBLIC] SYNONYM Sinonimo_vardas;

Sukurti sinonimą vardo vaizdui DEPT_SUM_VU

CREATE D_SUM FOR DEPT_SUM_VU;

Panaikinti sinonimą galima per

DROP SYNONIM DEPT_SUM_VU;

LITERATŪRA

1. Jackie Collins, Non –Procedural Programing with SQL and SQL*Plus, 1992, ORACLE. Corporation, U.S.A