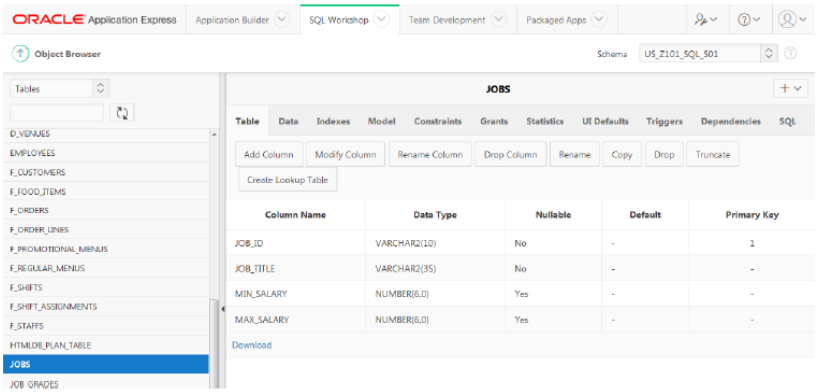
Database Programming with SQL
kurs 2017 – database design and programming with sql
students slajdovi

13-1 Creating Tables

- Treba znati kao DBA koji db objekti se najčešće koriste, kako posmatrati strukturu tabele i kako kreirati nove tabele
- Šta su to extsrene tabele – tabele koje su slične u strukturi sa normalnim Oracle db tabelama, ali pravi redovi podataka se smeštaju eksterno u flat fajlu i pristupa im se samo po potrebi

Database Schema Objects

- Jedna Oracle db može imati različite tipove objekata
- Ova sekcija prikazuje najčešće korišćene objekte a takođe kako Oracle Server koristi informacije smeštene u Data Dictionary kada radi poslove kao rezultat SQL iskaza koje programer ukucava
- Glavni db tipovi objekata su: tabele, indeksi, ograničenja, pogledi, sekvence, sinonimi (neki su nezavisni drugi ne)
- Db objekti koji uzimaju veliki deo memorijskog prostora se zovu i Segmenti
- Tabele i indeksi su segmenti
- Pogledi, ograničenja, sekvence i sinonimi su objekti ali jedini deo prostora koji zauzimaju je u samoj definiciji objekta – nijedan nema redove podataka sa kojima su povezani
- Db smešta definicije za sve db objekte u Data Dictionary, i ove definicije su dostupne za sve korisnike db kao i za same db
- Kako Oracle zna koje kolone da vrati iz upita ? Npr, ako se specificira SELECT * FROM jobs umesto SELECT job_id, job_title FROM jobs, kako Oracle zna koju kolonu da vrati?
- Db traži definiciju tabele korišćene u upitu, prevodi '!' u punu listu kolona i vraća rezultat programeru
- Db koristi Data Dictionary za sve iskaze koji se ispišu, čak ako se izlista imena kolona umesto korišćenja '*'
- To proverava da tabela na koju se referencira u iskazima postoji u db, to proverava da su kolone imena tačne, to proverava ako imate tačne privilegioje za izvođenje akcije koju tražiš, i najzad to koristi Data Dictionary za odlučivanje Execution Plan – kako će zapravo izvesti zahtev
- Sam Data Dictionary može biti pod upitom od strane svih korisnika db
- U APEX, može joj se pristupiti i preko SQL iskaza u SQL Workshop>SQL Commands interface i takođe iz SQL Workshop > Object Browser interface
- Unutar SQL Commands window treba znati imena tabele koja je pod upitom a u Object Browser interface samo se klikne na izlistane objekte za pregled njihovih detalja



- Tako ako se žele videti detalji JOBS tabele, samo se klikne na njen naslov u listingu tabele
- U primeru na prethodnoj slici, korišćenjem Object Browser, vide se detalji JOBS tabele kao i opcije za pregled podataka, indeksa, ograničenja, grantova i drugi detalji tabele
- Korišćenje SQL Commands prozora mora se tražiti DESCription tabele
- Sve dodatne opcije ponuđene od strane Objest Browser nisu dostupne u ovom interfejsu

```
DESCRIBE  jobs;
```

Results | Explain | Describe | Saved SQL | History

Object Type TABLE Object JOBS

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
JOBS	JOB_ID	VARCHAR2	10	-	-	1	-	-	-
	JOB_TITLE	VARCHAR2	35	-	-	-	-	-	-
	MIN_SALARY	NUMBER	-	6	0	-	✓	-	-
	MAX_SALARY	NUMBER	-	6	0	-	✓	-	-

- Može se koristiti DESCRIBE ili DESC za pregled opisa tabele

Table Creation

- Svi podaci u relacionoj db su smešteni u tabelama
- Pri kreiranju nove tabele, koristi sledeća pravila za imena tabela i imena kolona:
  - mora početi sa slovom
  - mora biti od 1 do 30 karaktera dugačka
  - mora sadržati samo A-Z, a-z, 0-9, _ , $, #
  - ne sme duplicirati ime drugog objekta koji je vlasništvo istog korisnika
  - ne sme biti Oracle Server rezervisana reč

Naming Conventions

- Najbolje je koristiti opisna imena za tabele i druge db objekte
- Ako će tabela smestiti informacije o studentima, imenuj ih ATUDENTS, ne PEOLE ili CHILDREN
- Imena tabela nisu osetljiva na veličinu; STUDENTS je isto kao StuDents ili students
- Imena treba da budu u množini, npr primer STUDENTS, ne student
- Kreiranje tabele su deo SQL dta definition language (DDL)
- Drugi DDL iskazi se koriste za postavljanje, izmenu i odstranjivanje strukture podataka iz tabela uključujući ALTER, DROP, RENAME i TRUNCATE

CREATE TABLE

- Za kreiranje nove tabele, prvo se moraju imati CREATE TABLE privilegije i smeštajni prostor za to
- DBA koristi DCL iskaze da bi odobrio ove privilegije korisnicima i dodelio smeštajni prostor
- Tabele koje pripadaju drugim korisnicima nisu u tvojoj schema
- Ako želiš koristiti tabelu ovo nije u tvojoj šemi, koristi ime vlasnika tabele kao prefiks za ime tabele:

```
SELECT *
FROM mary.students;
```

moraš dobiti odobrenje (grant) za pristup tabeli da bi bio u mogućnosti da biraš iz nje

CREATE TABLE Syntax

- Za kreiranje nove tabele, koristi sledeće detalje sintakse:
  - table je ime tabele, column je ime kolone
  - Data type je tip podataka kolone i dužina
  - DEFAULT izraz specificira difolt vrednost ako je vrednost izbegnuta u INSERT iskazu

```
CREATE TABLE table
(column data type [DEFAULT expression],
(column data type [DEFAULT expression],
(.....[ ] );
```

- Definicije kolona su odvojene zarezima

CREATE TABLE Example

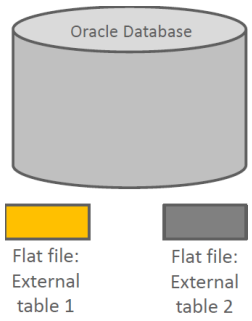
- Sledeći primer poklazuje CREATE TABLE iskaz:

```
CREATE TABLE my_cd_collection
 (cd_number NUMBER(3),
  title VARCHAR2(20),
  artist VARCHAR2(20),
  purchase_date DATE DEFAULT SYSDATE);
```

```
CREATE TABLE my_friends
 (first_name VARCHAR2(20),
  last_name VARCHAR2(30),
  email VARCHAR2(30),
  phone_num VARCHAR2(12),
  birth_date DATE);
```

External Tables

- Oracle podržava i External tabele; u eksternoj tabeli redovi podataka se ne drže unutar db fajlova već se nalaze u flat file, smeštene eksterno na db
- Tipično se jedna eksterna tabela koristi za smeštanje podataka pomerenih iz starijih verzija db korisšćenih ranije u kompaniji
- Kada kompanija implementira novu aplikaciju i db, obično treba importovati najveći deo podataka iz starog sistema u novi ali neki podaci koji se ne koriste često i treba ih imati samo za čitanje



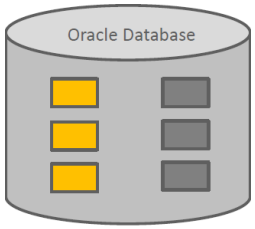
- I oni se mogu čuvati u jednoj eksternoj tabeli
- Jedna od mnogih doprinosa Oracle je da podaci smešteni u eksternim tabelama treba da se samo jednom backed up i nikada više osim ako se menja sadržaj fajlova
- Sintaksa za kreiranje eksterne tabele je veoma slična onoj za kreiranje standardne tabele, osim što ima dodatne sintakse na kraju
- Kompanije često kupuju referencirane podatke od spoljnih izvora, npr informacije o marketingu na potencijalnim kupcima ili fajlove adresa sa zip kodovima. Bolj nego da sve te podatke smeste u jednu db, kompanije češće ostavljeju podatke u flat fajlovima eksterno u odnosu na db, i pristupaju im kao External Table
- Nova sintaksa (u crvenom) na sledećim primerima nije korišćena u standardnim SQL iskazima za kreiranje tabele:
  - ORGANIZATION EXTERNAL – kaže Oracle da kreira eksternu tabelu
  - TYPE ORACLE_LOADER – tip Oracle Loadera (proizvod Oracle)
  - DEFAULT DIRECTORY def_dir1 – ime direktorijuma za fajl
  - ACCESS PARAMETERS – kako čitati fajl
  - RECORDS DELIMITED BY NEWLINE – kako identifikovati start novog reda
  - FIELDS – ime polja i tip podatka specifikacije
  - LOCATION – ime polja pravog fajla koji sadrži podatke

```
CREATE TABLE emp_load
  (employee_number CHAR(5),
   employee_dob CHAR(20),
   employee_last_name CHAR(20),
   employee_first_name CHAR(15),
   employee_middle_name CHAR(15),
   employee_hire_date DATE)
ORGANIZATION EXTERNAL
   (TYPE ORACLE_LOADER
    DEFAULT DIRECTORY def_dir1
    ACCESS PARAMETERS
   (RECORDS DELIMITED BY NEWLINE
    FIELDS (employee_number CHAR(2),
                  employee_dob CHAR(20),
                  employee_last_name CHAR(18),
                  employee_first_name CHAR(11),
                  employee_middle_name CHAR(11),
                  employee_hire_date CHAR(10) date_format DATE mask
"mm/dd/yyyy"))
   LOCATION ('info.dat'));
```

- Ovaj upit neće raditi na APEX pošto ne postoji veza sa eksternim tabelama

Data Dictionary

- Dva tipa tabela postoji u Oracle db: User i Data Dictionary
- Može se izdati SQL iskaz za pristup obe vrsti tabele – može se izabrati, uneti, updejtovati i obrisati podatak u user tabeli a izabrati podatak u DD tabeli



- User tabele koji si ti kreirao imaju podatke: employees, departments, jobs...
- DD tabele: DICTIONARY, USER_OBJECTS, USER_TABLES, USER_SEGMENTS, USER_INDEXES...
- DD tabele su vlasništvo Oracle korisnika zvanog SYS i samo SELECT iskaz se može koristiti pri radu sa ovim tabelama
- Za pravljenje ovih tabela sigurnih od slučajnog korisničkog pristupa, sve one imaju poglede (views) kreirane preko kojih se pristupa DD od strane db korisnika
- Ako bilo koji Oracle user pokuša da ubaci, updejt ili obriše bilo šta u DD tabeli, operacija nije dopuštena pošto može poremetiti integritet cele baze
- Kada se koriste DD pogledi u SQL Commands interfejsu, treba znati imena Dictionary pogleda sa kojima se radi
- U Oracle, to je jednostavno: prefiks tipa objekta koji se traži sa USER_xxx ili jedan ALL_xxx, gde je xxx tip objekta

```
SELECT table_name, status
FROM USER_TABLES;
```

```
SELECT table_name, status
FROM ALL_TABLES;
```

- Pa ako želiš istražiti indekse, onda samo izaberi iz USER_INDEXES; ako želiš informacije o sekvencama onda je tabela USER_SEQUENCES itd

```
SELECT *
FROM user_indexes;
```

```
SELECT *
FROM user_objects
WHERE object_type = 'SEQUENCE';
```

- Ako niste sigurni u DD pogled koji vam treba, možete SELECT * FROM DICTIONARY, i ovo će vratiti ime pogleda sadržanog u rečniku sa opisom njegovog sadržaja

13-2 Using Data Types

- Različiti tipovi podataka imaju različite tipove osobina, čija je svrha efikasno smeštanje podataka

Data Tyoe Overview

- Svaka vrednost koja se manipuliše sa Oracle ima svoj tip podataka
- Tip podataka neke vrednosti je u vezi sa fiksnim setom osobina te vrednosti
- Ove osobine izazivaju db da tretiraju vrednosti jednog tipa podataka drugačije od vrednosti drugog tipa podataka
- Različiti tipovi podataka nude nekoliko pogodnosti:
  - kolone jednog tipa proizvode trajne rezultate
  - npr, DATE tip podataka kolona uvek proizvodi vrednost datum
  - ne može se uneti pogrešan tip podataka u kolonu; npr, kolone tipa podataka DATE će sprečiti NUMBER tip podataka od toga da budu uneti
- Iz ovih razloga, svaka kolona u relacionoj db može čuvati samo jedan tip podataka
- Ne može se mešati tipovi podataka unutar kolone

Common Data Types

- Najčešće se koriste tipovi za znakove:
  - CHAR (fiksirana veličina, maksimum 2000 karaktera)
  - VARCHAR2 (promenjiva veličina, maksimalno 4000 karaktera)
  - CLOB (promenjiva veličina, maksimum 128 terabajtova)
- Najčešće brojne vrednosti:
  - NUMBER (promenjiva veličina, maksimalna preciznost 38 cifara)
- Najčešće vrednosti za datum i vreme:
  - DATE
  - TIMESTAMP...
  - INTERVAL
- Najčešće binarne vrednosti (npr multimedijalni formati: JPG, WAV, MP3...):
  - RAW (promenjiva veličina, maksimum 2000 bajtova)
  - BLOB (promenjiva veličina, maksimum 128 terabajtova)
- Za vrednosti karaktera, bolje je koristiti VARCHAR2 ili CLOB nego CHAR pošto se tako čuva memorijski prostor
- Npr, prezime zaposlenog je 'Chang'; u VARCHAR2(30) koloni samo 5 značajnih karaktera u smešteni: C h a n g; ali u CHAR(30) koloni, 25 pratećih praznih mesta će se smestiti takođe da bi se popunila 30 fiksnih mesta
- Brojčane vrednosti mogu biti i negativne; npr, NUMBER(6,2) može smestiti bilo koju vrednost od +9999.99 do -9999.99

DATE-TIME Data Types

- DATE tip podataka smešta vrednost od vekova sve do sekunde, ali ne može se smestiti delić sekunde
- '21-Aug-2003 17:25:30' je validna vrednost, ali '21-Aug-2003 17:25:30.255' nije
- TIMESTAMP tip podataka je ekstenzija DATE tipa podataka koji omogućava deliće sekunde
- Npr, TIMESTAMP(3) omogućava 3 cifre posle celih sekunda, što znači prikaz milisekunde
- Delići sekunde su važni za neke biznise; rad berze mora da zna precizno vreme kada se desila transakcija pošto se cene mogu promeniti hiljadu puta u sekundi, tako da treba da su sposobni za praćenje tačnog vremena transakcije/promene cena na delovima sekunde

- TIMESTAMP example:

```
CREATE TABLE time_ex1
(exact_time TIMESTAMP);
```

```
INSERT INTO time_ex1
VALUES ('10-Jun-2015 10:52:29.123456');
```

```
INSERT INTO time_ex1
VALUES (SYSDATE);
```

```
INSERT INTO time_ex1
VALUES (SYSTIMESTAMP);
```

```
SELECT *
FROM time_ex1;
```

EXACT_TIME
10-JUN-15 10.52.29.123456 AM
16-JUL-15 08.17.08.000000 AM
16-JUL-15 08.17.16.610293 AM

- Tabela time_ex1 ima jednu kolonu tipa TIMESTAMP; unosi se vrednost TIMESTAMP kao literal; koristi SYSDATE funkciju ali SYSDATE samo vraća vreme na najbližu sekundu tako da nisu zabeležene mikrosekunde; koristi SYSTIMESTAMP funkciju koja je slična sa SYSDATE ali vraća i deliče sekunde tako da je uneto vreme na najbližu mikrosekundu

TIMESTAMP ... With [LOCAL] Time Zone

- Ako je uneto vreme '17:30' i to znači pola šest posle podne ali u kojoj vremenskoj zoni?
- TIMESTAMP WITH TIME ZONE smešta vrednost vremenske zone kao displacement od Universal Coordinated Time ili UCT (Greenwitch Mean Time GMT)
- Vrednost '21-Aug-2003 08:00:00 – 5:00' znači 8:00 am 5 sati iza UTC što je US Eastern Standard Time (EST)

• TIMESTAMP WITH TIME ZONE example:

```
CREATE TABLE time_ex2
(time_with_offset TIMESTAMP WITH TIME ZONE);
```

```
INSERT INTO time_ex2
VALUES (SYSTIMESTAMP);
```

```
INSERT INTO time_ex2
VALUES ('10-Jun-2015 10:52:29.123456 AM +2:00');
```

```
SELECT *
FROM time_ex2;
```

TIME_WITH_OFFSET
16-JUL-15 08.49.47.126056 AM -07:00
10-JUN-15 10.52.29.123456 AM +02:00

- Tabela time_ex2 ima jednu kolonu tipa TIMESTAMP WITH TIME ZONE; koristi SYSTIMESTAMP funkciju za dodavanje vremena sa offset od UTC. Pošto je APEX server smešten u Kaliforniji, vremenska zona je UTC – 7:00 (Pacific Standard Time ili PST); unosi literal vrednost TIMESTAMP WITH LOCAL TIME sa lokalnim vremenom od UTC+2:00
- TIMESTAMP WITH LOCAL TIME ZONE je slično ali kada se ova kolona izabere u SQL iskazu, vreme se automatski konvertuje u korisnički izabranu vremensku zonu

• TIMESTAMP With…Time Zone Example:

```
CREATE TABLE time_ex3
( first_column TIMESTAMP WITH TIME ZONE,
  second_column TIMESTAMP WITH LOCAL TIME ZONE);
```

```
INSERT INTO time_ex3
  (first_column, second_column)
VALUES
  ('15-Jul-2015 08:00:00 AM -07:00', '15-Nov-2007 08:00:00');
```

- Obe vrednosti su smeštene sa time displacement od – 7:00 sati (PST)

- Ali korisnik u Istanbulu izvršava:

```
SELECT *
FROM time_ex3;
```

FIRST_COLUMN	SECOND_COLUMN
15-JUL-15 08.00.00.000000 AM -07:00	15-NOV-07 05.00.00.000000 PM

- Istanbulsko vreme je 9 sati ispred PST; kada je 8am u Los Anđelesu, to je 5pm u Istanbulu
- TIMESTAMP WITH LOCAL TIME ZONE tip podataka smešta timestamp bez vremenske zone informaciju. On konvertuje vreme na db vremensku zonu svaki put kada se podatak pošalje ka ili od klijenta
- Zbog limita hosted online verzije APEX, prethodni primer će uvek pokazivati lokalno vreme kao UTC – 07:00 (PST)

INTERVAl Data Types

- Ovi smeštaju proteklo vreme ili interval vremena između dva datumske vrednosti
- INTERVAL YEAR TO MONTH smešta period vremena meren u godinama i mesecima
- INTERVAL DAY TO SECOND smešta period vremena meren u danima, satima, minutama i sekundama

INTERVAL YEAR...TO MONTH

- Sintaksa: INTERVAL YEAR [(year_precision)] TO MONTH
- year_precision je maksimalan broj cifara u YEAR elementu
- Difolt vrednost od year_precision je 2
- Primer pokazuje INTERVAL YEAR TO MONTH:

```
CREATE TABLE time_ex4
(loan_duration1 INTERVAL YEAR(3) TO MONTH,
 loan_duration2 INTERVAL YEAR(2) TO MONTH);
```

```
INSERT INTO time_ex4 (loan_duration1, loan_duration2)
VALUES (INTERVAL '120' MONTH(3),
          INTERVAL '3-6' YEAR TO MONTH);
```

Assume today's date is: 17-Jul-2015

```
SELECT SYSDATE + loan_duration1 AS "120 months from now",
       SYSDATE + loan_duration2 AS "3 years 6 months from now"
FROM time_ex4;
```

120 months from now	3 years 6 months from now
17-Jul-2025	17-Jan-2019

- Tabela je kreirana sa 2 kolone tipa INTERVAL YEAR TO MONTHS
- INSERT iskaz dodaje jedan INTERVAL od 120 meseci (10 godina) u loan_duration1 i jedan interval od 3 godine i 6 meseci za loan_duration2

INTERVAL DAY..TO SECOND

- Ovo koristiti kada treba više precizna razlika između dve date-time vrednosti
- Sintaksa: INTERVAL DAY [(day_precision)] TO SECOND [(fractional_seconds_precision)]
- day_precision je maksimalan broj cifara u DAY elementu
- Difolt vrednost od day_precision je 2

- fractional_seconds_precision je broj cifara u frakcionalnom delu SECOND date-time polja
- Difolt je 6
- Primer pokazuje interval DAY TO SECOND:

```
CREATE TABLE time_ex5
(day_duration1 INTERVAL DAY(3) TO SECOND,
 day_duration2 INTERVAL DAY(3) TO SECOND);
```

```
INSERT INTO time_ex5 (day_duration1, day_duration2)
VALUES (INTERVAL '25' DAY(2), INTERVAL '4 10:30:10' DAY TO SECOND);
```

```
SELECT SYSDATE + day_duration1 AS  "25 Days from now",
   TO_CHAR(SYSDATE + day_duration2, 'dd-Mon-yyyy hh:mi:ss')
   AS  "precise days and time from now"
FROM time_ex5;
```

25 Days from now	precise days and time from now
11-Aug-2015	21-Jul-2015 01:13:17

- Tabela je kreirana sa 2 INTERVAL DAY TO SECOND kolone
- INSERT iskaz dodaje INTERVAL od 25 dana na day_duration1, a 4 dana, 10 sati, 30 minuta i 10 sekundi za day_duration2
- SELECT iskaz uzima trenutni datum i dodaje trajanje smešteno u kolonama

13-3 Modifying a Table

- Do sada su se promene dešavale u tabelama na nivou redova tabela, ali kako napraviti promene na samim tabelama ?
- DDL komande se koriste za promenu, preimenovanje, ispražnjenje ili samo eliminisanje tabele

ALTER TABLE

- ALTER TABLE iskaz se koriste za: dodaj novu kolonu, modifikovanje postojeće kolone, definiše DEFAULT vrednost za kolonu, izbacuje kolonu
- Može se dodati ili modifikovati kolona u tabeli, ali ne može se specificirati gde se kolona pojavljuje
- Nova dodata kolona uvek postaje poslednja kolona u tabeli
- Takođe, ako tabela već ima redove podataka a doda se nova kolona u tabelu, nova kolona je inicijalno null za sve pre-postojeće redove

ALTER TABLE: Adding a Column

- Za dodavanje nove kolone, koristiti SQL sintaksu:

```
ALTER TABLE tablename
ADD (column name data type [DEFAULT expression],
column name data type [DEFAULT expression], ...
```

- For example:

```
ALTER TABLE my_cd_collection
ADD (release_date DATE DEFAULT SYSDATE);
```

```
ALTER TABLE my_friends
ADD (favorite_game VARCHAR2(30));
```

- Dve ili više kolona se može dodati korišćenjem jednog ALTER iskaza odvajanjem definicija kolona sa zarezima

ALTER TABLE: Modifying a Column

- Modifikacija kolone može uključiti promene u koloni u tipu podataka, veličini i DEFAULT vrednosti
- Pravila i restrikcije pri modifikovanju kolone:
  - može se povećati širina ili preciznost numeričke kolone
  - može se povećati širina kolone karaktera
  - može se smanjiti širina NUMBER kolone ako kolona sadrži samo null vrednosti ili ako tabela nema redove
  - za VARCHAR tipove, može se smanjiti širina sve do najveće vrednosti unutar kolone
- Može se promeniti tip podataka samo ako kolona sadrži null vrednosti
- Može se konvertovati CHAR kolona u VARCHAR2 ili konvertovati VARCHAR2 COLUMN u CHAR samo ako kolona sadrži null vrednosti, ili ako se ne promeni veličina na nešto manje od bilo koje vrednosti u koloni
- Promena u DEFAULT vrednosti kolone utiče samo kasnije unose u tabelu
- Primer: kreirana je tabela sa dve kolone

```
CREATE TABLE mod_emp
  (last_name VARCHAR2(20),
   salary NUMBER(8,2));
```

- Koja od ovih modifikacija će biti dozvoljena, a koja neće ? (Uzeti u obzir odgovore sa i bez redova podataka u tabeli)

```
ALTER TABLE mod_emp
     MODIFY (last_name VARCHAR2(30));
```

```
ALTER TABLE mod_emp
     MODIFY (last_name VARCHAR2(10));
```

```
ALTER TABLE mod_emp
     MODIFY (salary NUMBER(10,2));
```

```
ALTER TABLE mod_emp
     MODIFY (salary NUMBER(8,2) DEFAULT 50);
```

- Bilo bi dozvoljeno samo ako kolone su prazne ili najveće ime je 10 ili manje karaktera

```
ALTER TABLE mod_emp
     MODIFY (last_name VARCHAR2(10));
```

- Would be permitted with or without data as column width increased.

```
ALTER TABLE mod_emp
     MODIFY (last_name VARCHAR2(30));
```

```
ALTER TABLE mod_emp
     MODIFY (salary NUMBER(10,2));
```

- Would be permitted with or without data as column precision increased.

```
ALTER TABLE mod_emp
     MODIFY (salary NUMBER(8,2) DEFAULT 50);
```

- Would be permitted with or without data as only a DEFAULT value added.

ALTER TABLE: Dropping a Column

- Pri odstranjivanju kolone važe sledeća pravila:
  - kolona koja sadrži podatke može da se odstraniti
  - samo jedna kolona se može odbaciti u jednom momentu
  - ne može se odstraniti sve kolone u tabeli, najmanje jedna kolona mora ostati
  - kada je kolona odstranjena, vrednosti podataka u njoj s ene mogu povratiti
- U velikim tabelama, DROP COLUMN može trajati dugo vremena pošto Oracle db mora modifikovati svki red za brisanje vrednosti kolona

- SQL Syntax:

```
ALTER TABLE tablename
DROP COLUMN column name;
```

- For Example:

```
ALTER TABLE my_cd_collection
DROP COLUMN release_date;
```

```
ALTER TABLE my_friends
DROP COLUMN favorite_game;
```

SET UNUSED Columns

- Odstranjivanje kolone iz velike tabele može dugo trajati
- Brža alternativa je označavanje kolone kao beskorisne (unusable)
- Vrednosti kolone ostaju u db ali im se ne može prići, pa je efekat isti kao odstranjivanje kolone
- Zapravo, može se dodati nova kolona u db sa istim imenom kao i beskorisna kolona; beskorisne kolone postoje samo su nevidljive
- Sintaksa: ALTER TABLE tablename SET UNUSED (column name);
- Primer:

```
ALTER TABLE copy_employees
   SET UNUSED (email);
```

- DROP UNUSED COLUMNS odstranjuje sve kolone trenutno označene kao nekorišćene
- Ovaj iskaz se koristi kada se želi odvojiti još prostora na disku od nekorisnih kolona u tabeli
- Primer:

```
ALTER TABLE copy_employees
   DROP UNUSED COLUMNS;
```

ALTER TABLE Summarized

- Ova tabela sumira korišćenje ALTER TABLE komande:

Syntax	Outcomes	Concerns
ALTER TABLE tablename ADD (column name data type [DEFAULT expression], column name data type [DEFAULT expression], ...	Adds a new column to a table	You cannot specify where the column is to appear in the table. It becomes the last column.
ALTER TABLE tablename MODIFY (column name data type [DEFAULT expression], column name data type, ...	Used to change a column's data type, size, and default value	A change to the default value of a column affects only subsequent insertions to the table.
ALTER TABLE tablename DROP COLUMN column name;	Used to drop a column from a table	The table must have at least one column remaining in it after it is altered. Once dropped, the column cannot be recovered.
ALTER TABLE tablename SET UNUSED (column name);	Used to mark one or more columns so they can be dropped later	Does not restore disk space. Columns are treated as if they were dropped.
ALTER TABLE tablename DROP UNUSED COLUMNS	Removes from the table all columns currently marked as unused	Once set unused, there is no access to the columns; no data displayed using DESCRIBE. Permanent removal; no rollback.

DROP TABLE

- DROP TABLE iskaz odstranjuje definiciju Oracle tabele
- Db gubi sve podatke u tabeli i sve indekse koji su povezani sa njome
- Kada se izda komanda DROPŠ TABLE svi podaci su obrisani iz tabele i opis tabele je odstranjen iz Data Dictionary
- Oracle Server ne preispituje odluku i tabela se odmah i odstranjuje
- Moguće je povratiti tabelu posle njenog odstranjivanja ali ne postoje garancije
- Samo krator tabele ili korisnik sa DROP ANY TABLE privilegijama (obično samo DBA) mogu odstraniti tabelu

- Syntax:

```
DROP TABLE tablename;
```

- Example:

```
DROP TABLE copy_employees;
```

FLASHBACK TABLE

- Ako se odstrani tabela slučajno, može se povratiti i tabela i podaci
- Svaka db koju korisnik ima ima svoj sopstveni recycle bin u kojem se sklone odstranjeni objekti, a oni se mogu oporaviti odatle sa FLASHBACK TABLE komandom
- Ova komanda se može koristiti za oporavak tabele, pogleda ili jednog indeksa koji je pogrešno odstranjen
- Sintaksa: FLASHBACK TABLE tablename TO BEFORE DROP;
- Npr, ako je odstranjena EMPLOYEES tabela greškom, može se oporaviti samo komandom: FLASHBACK TABLE copy_employees TO BEFORE DROP;
- Kao vlasnik tabele, možeš izdati flashback komandu i ako je tabela imala bilo kakve indekse oni će takođe biti oporavljeni
- Moguće je videti koji objekat se može oporaviti korišćenjem upita nad DD pogledom USER_RECYCLEBIN
- Ako je tabela odstranjena, onda bilo koji zavistan objekat, poput INDEXES su takođe odstranjeni, a ako se tabela oporavi iz kante za otpatke, onda ovi zavisni objekti su takođe oporavljeni
- USER_RECYCLEBIN pogled može biti pod upitom kao drugi DD pogledi:

```
SELECT original_name, operation, droptime
FROM user_recyclebin
```

ORIGINAL_NAME	OPERATION	DROPTIME
EMPLOYEES	DROP	2007-12-05:12.34.24
EMP_PK	DROP	2007-12-05:12.34.24

- Kada je jednom tabela oporavljena sa FLASHBACK TABLE komandom, nije više vidljiva u USER_RECYCLEBIN pogledu
- Bilo koji indeksi koji su odstranjeni sa originalnom tabelom će takođe biti oporavljeni
- Možda je neophodno (zbog sigurnosti) kompletno odstraniti tabelu, izbegavajući recycling bin
- Ovo se može uraditi dodavanjem službene reči PURGE:
  DROP TABLE copy_employees PURGE;
- Ako je tabela odstranjena korišćenjem PURGE nije moguće oporaviti je korišćenjem FLASHBACK
- Kanta za otpatke se može očistiti purgiranjem: PURGE RECYCLEBIN će trajno obrisati sve objekte u korisnikovoj kanti za otpatke

RENAME

- Za izmenu imena tabele, koristi se RENAME iskaz
- Ovo može da uradi samo vlasnik objekta ili DBA

• Syntax:

```
RENAME old_name to new_name;
```

• Example:

```
RENAME my_cd_collection TO my_music;
```

TRUNCATE

- Truncating tabelu odstranjuje sve redove iz tabele i oslobađa memoriju koju jetabela koristila
- Pri korišćenju TRUNCATE TABLE iskaza ne može se povratiti odstranjeni red i mora se biti vlasnik tabele ili dati DROP ANY TABLE sistemske privilegije
- Sintaksa: TRUNCATE TABLE tabename;
- DELETE iskaz takođe odstranjuje redove iz tabele ali ne oslobađa memoriju
- TRUNCATE je brže od DELETE pošto ne generiše rollback informaciju, a to znači da se posle TRUNCATE ništa ne može povratiti

COMMENT ON TABLE

- Može se dodati komentar do 2000 karaktera o koloni, tabeli ili pogledu korišćenjem COMMENT iskaza

• Syntax:

```
COMMENT ON TABLE tablename | COLUMN table.column
     IS 'place your comment here';
```

• Example:

```
COMMENT ON TABLE employees
     IS 'Western Region only';
```

- Za videti komentare o tabeli u DD:

```
SELECT table_name, comments
FROM user_tab_comments;
```

TABLE_NAME	COMMENTS
EMPLOYEES	Western Region Only

- If you want to drop a comment previously made on a table column, use the empty string(''):

```
COMMENT ON TABLE employees IS ' ' ;
```

FLASHBACK QUERY

- Može se desiti da je podatak u tabeli nepravilno promenjen; Oracle ima mogućnosti kojima se mogu videti podaci iz redova u određenom trenutku tako da se mogu uporediti različite verzije tokom vremena
- Npr, ako neko izvede DML na tabelu a onda izvrši COMMIT na tim promenama
- Može se koristiti FLASHBACK QUERY facility kako su redovi izgledali pre nego su ove promene primenjene; kada Oracle promeni podatak on uvek čuva kopiju kako je podatak izgledao ranije
- To znači da se čuva kopija stare kolone za updejt kolone, čuva celu kolonu za brisanje a ništa ne čuva za insert iskaz
- Stare kopije se smeštaju u posebno mesto zvano UNDO tablespace
- Korisnici mogu pristupiti ovoj posebnoj mestu u db korišćenjem flashback upita
- Može se videti starija verzija podataka korišćenjem VERSIONS iskaza u SELECT iskazu
- Primer:

```
SELECT employee_id,first_name ||' '|| last_name AS "NAME",
    versions_operation AS "OPERATION",
    versions_starttime AS "START_DATE",
    versions_endtime  AS "END_DATE", salary
FROM employees
  VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE
WHERE  employee_id = 1;
```

- SCN broj na koji se referiše u prethodnom primeru znači System Change Number i to je precizna identifikacija vremena u db; to je sekvencijalni broj inkrementiran i održavan od strane same db
- Najbolji način za demonstracije FLASHBACK QUERY je sa primerom. Sadržaj u employee_id 1 u employees tabeli:

```
SELECT employee_id,first_name ||' '|| last_name AS "NAME",
               versions_operation AS "OPERATION",
               versions_starttime AS "START_DATE",
               versions_endtime  AS "END_DATE", salary
FROM copy_employees
  VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE
WHERE  employee_id = 1;
```

no data found

- Zatim se kreira zaposleni:

```
INSERT INTO copy_employees
VALUES (1, 'Natacha', 'Hansen', 'NHANSEN', '4412312341234',
        '07-SEP-1998', 'AD_VP', 12000, null, 100, 90, NULL);
```

```
SELECT employee_id,first_name ||' '|| last_name AS "NAME",
               versions_operation AS "OPERATION",
               versions_starttime AS "START_DATE",
               versions_endtime  AS "END_DATE", salary
FROM copy_employees
  VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE
WHERE  employee_id = 1;
```

EMPLOYEE_ID	NAME	OPERATION	START_DATE	END_DATE	SALARY
1	Natacha Hansen	I	07-SEP-1998 06.51.58 AM	-	12000

- Onda se može obrisati red:

```
DELETE from copy_employees
WHERE employee_id = 1;
```

```
SELECT employee_id,first_name ||' '|| last_name AS "NAME",
               versions_operation AS "OPERATION",
               versions_starttime AS "START_DATE",
               versions_endtime  AS "END_DATE", salary
FROM copy_employees
  VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE
WHERE  employee_id = 1;
```

EMPLOYEE_ID	NAME	OPERATION	START_DATE	END_DATE	SALARY
1	Natacha Hansen	D	07-SEP-1998 07.00.10 AM	-	1
1	Natacha Hansen	U	07-SEP-1998 06.57.01 AM	07-SEP-1998 07.00.10 AM	1
1	Natacha Hansen	I	07-SEP-1998 06.51.58 AM	07-SEP-1998 06.57.01 AM	12000

- Rezultat poslednjeg upita na prethodnom primeru je vidljiv samo pri korišćenju Flashback upita, i.e. VERSIONS izraza
- Ako se pokuša normalna pretraga iz employee_id=1 pre delete iskaza, dobiće se normalna greška, No Data Found

```
SELECT employee_id, salary
FROM copy_employees
WHERE employee_id = 1;
```