

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Nikolina Leskovar

ORACLE SQL DEVELOPER
ZAVRŠNI RAD

Varaždin, 2014.

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Nikolina Leskovar

Matični broj: 39065/10-R

Studij: Informacijski sustavi

ORACLE SQL DEVELOPER
ZAVRŠNI RAD

Mentor:

Izv. prof. dr. sc. Kornelije Rabuzin

Varaždin, rujan 2014.

Sadržaj

1.	Uvod.....	1
1.1.	Baza podataka.....	2
1.2.	SQL.....	3
1.3.	Povijest Oracle-a.....	2
2.	Oracle Database 11g Express Edition	4
2.1.	Instalacija OracleXE.....	4
2.2.	Arhitektura Oracle Database 11g Release 2	6
2.3.	Tipovi podataka	11
3.	Oracle SQL Developer	14
3.1.	Instalacija i spajanje na bazu podataka.....	14
3.2.	Modeliranje.....	18
3.2.1.	ER konstrukti.....	20
3.3.	Konceptualni model.....	22
3.4.	Relacijski model	26
3.5.	Implementacija tablica.....	34
3.5.1.	CREATE TABLE	38
3.5.2.	ALTER TABLE	41
3.5.3.	DROP TABLE	42
4.	Rad s podacima i upiti.....	45
4.1.	Naredba INSERT.....	45
4.2.	Klauzule SELECT , FROM, WHERE.....	47
4.2.1.	Operatori BETWEEN..AND, IN, LIKE, IS NULL	50
4.3.	Agregirajuće funkcije	51
4.4.	Klauzule GROUP BY, HAVING, ORDER BY	52
4.5.	DESCRIBE opcija	54
4.6.	Naredba UPDATE	54
4.7.	Naredba DELETE.....	55
4.8.	Spajanje tablica.....	56
4.8.1.	Spajanje tablica klauzulama FROM i WHERE	57
4.8.2.	[INNER] JOIN	58
4.8.3.	LEFT [OUTER] JOIN.....	59
4.8.4.	RIGHT [OUTER] JOIN	60
4.8.5.	FULL [OUTER] JOIN	61
4.9.	Pogledi	61
5.	Zaključak.....	66

6.	Popis slika i tablica.....	67
7.	Literatura	69

1. Uvod

Motivacija za odabir teme završnog rada bila je upoznavanje sa novim sustavom za upravljanje bazama podataka, Oracle. Ovim radom biti će prikazani principi korištenje alata Oracle SQL Developer vezano za modeliranje i implementaciju baze podataka.

Cilj rada je upoznavanje sa osnovnim funkcijama alata te prikazati korištenje istih na konkretnom primjeru.

Rad je podijeljen na pet poglavlja. Prvo poglavlje je uvodno i u njemu su spomenute osnovne informacije o bazama podataka i sustavu za upravljanje bazama podataka. Također, spomenuti je i SQL jezik koji sadrži naredbe za rad s relacijskom bazom podataka. Spomenuta je i povijest Oracle-a, prvog komercijalno dostupnog sustava za upravljanje relacijskim bazama podataka.

Na početku drugog poglavlja opisano je kako instalirati sustav Oracle Database 11g Express Edition. U nastavku je opisana njegova arhitektura te tipovi podataka koje podržava.

Treće poglavlje je posvećeno samom alatu Oracle SQL Developer. Na početku je objašnjena instalacija alata i postavljanje konekcije. Zatim se postepeno opisuje kreiranje konceptualnog i relacijskog modela te implementacija tablica. Uz izradu modela objašnjeni su i neki važni pojmovi poput entiteta, atributa, ograničenja, vanjskih i primarnih ključeva. Svaki je korak popraćen odgovarajućom slikom i primjerom.

Četvrto poglavlje opisuje rad s podacima. Nakon implementacije tablice su prazne te ih je najprije potrebno popuniti. Zatim se nad podacima izvode različiti upiti, ažuriranja, stvaraju se pogledi i slično.

Na kraju slijedi zaključak kojim se rezimira cjelokupni rad.

1.1. Baza podataka

Što su baze podataka? Postoji mnogo različitih definicija koje objašnjavaju pojam baza podataka:

„Skup trajno pohranjenih podataka informacijskog sustava naziva se bazom podataka“
[Varga, 2012, str. 17].

„Baza podataka jest model podataka poslovnog sustava odnosno segmenta stvarnog svijeta“
[Varga, 1994, str. 10].

Možemo zaključiti da je baza podataka skup podataka koji sadrži stvarne podatke nekog informacijskog sustava, ali i metapodatke koji opisuju te podatke u bazi podataka. Mnogi zamišljaju bazu podataka kao organizirani mehanizam koji ima sposobnost pohranjivanja podataka, kroz koji ih korisnici mogu dohvatiti. Ljudi svakodnevno, svjesno ili nesvjesno, koriste bazu podataka. Telefonski imenik je baza podataka. Podaci se sastoje od imena, adresa, telefonskih brojeva te su poredani abecedno kako bi korisnik mogao lakše pronaći traženi broj. Ti su podaci spremljeni negdje na računalu. Ne tipka se svake godine ručno novi telefonski imenik, već se podaci naknadno dodaju, ažuriraju ili brišu.

Današnje baze strukturiranih podataka izgrađene su prema pravilima relacijskih baza podataka. Relacijski model je razradio Edgar Ted Codd, 1970. godine. Relacijska baza se sastoji od skupa relacija, odnosno tablica. Tablica je skup redaka i stupaca, a njihov redoslijed nije bitan.

Sam rad s bazom podataka nam omogućuje sustav za upravljanje bazom podataka (eng. *Database Management System - DBMS*). To je poslužitelj baze podataka koji oblikuje fizički prikaz baze s obzirom na logičku strukturu. Sustavi za upravljanje bazom podataka su:

- *Oracle*
- *MySQL*
- *DB2*
- *MS SQL Server*

1.2. SQL

Da bi se mogli sporazumijevati sa ljudima potreban nam je odgovarajući jezik kojeg razumije druga osoba. Isto tako, za rad s relacijskom bazom podataka upotrebljavaju se neproceduralni upitni jezici. Tih jezika ima više, no danas se uglavnom koristi SQL koji je i standardiziran.

Organizacije u području standarda su ANSI (eng. *American National Standards Institute*) i ISO (eng. *American National Standards Institute*).

Rabuzin [2011] navodi da je prvu verziju standarda izdao ANSI 1986., a prihvaćen je i od ISO 1987. godine. Zadnja verzija standarda je SQL 2011.

Između pojedinih sustava za upravljanje bazama podataka i dalje postoje razlike u SQL jeziku.

SQL sadrži naredbe za potpuni rad s relacijskom bazom podataka. SQL sadrži naredbe za:

- *kreiranje objekata (eng. Data Definition Language - DDL)*
- *manipuliranje podacima (eng. Data Manipulation Language - DML)*
- *postavljanje upita (eng. Data Query Language - DQL)*

Naredbe koje spadaju u DDL naredbe su CREATE, ALTER i DROP. One služe za kreiranje i ažuriranje objekata, a objekti mogu biti tablice, pogledi i dr.

Da bi mogli unositi, mijenjati i brisati određene podatke potrebne su nam naredbe za manipuliranje podacima, a to su INSERT, UPDATE i DELETE.

Naredba SELECT je naredba koju koristimo kod upita, odnosno njome dohvaćamo podatke iz željene tablice.

1.3. Povijest Oracle-a¹

Oracle se od svog osnutka bavi informacijama, kako njima upravljati, koristiti ih i zaštititi. Sadašnja verzija Oracle baze podataka je rezultat više od trideset godina inovativnog razvoja. Godine 1977, Larry Ellison, Bob Miner i Ed Oates su osnovali tvrtku Relational Software, Inc. (RSI) sa sjedištem u Redwood Shores, Kaliforniji. Godine 1983, RSI je postao Oracle System Corporation, a kasnije Oracle Corporation.

U prijevodu, Oracle znači proročanstvo, predviđanje, odgovor na pitanja smrtnika koji dolazi od božanstva ili vrhovnog bića.²

Oracle je prvi komercijalno dostupan RDBMS (eng. *Relational Database Management System*), sustav za upravljanje relacijskim bazama podataka, odnosno sustav za upravljanje bazama podataka koji je baziran na relacijskom modelu kojega je predstavio Edgar Frank Codd, 1970. godine u svom radu "*A Relational Model of Data for Large Shared Data Banks*".

Relational Software, Inc. je 1979. godine predstavio prvu komercijalno dostupnu verziju sustava Oracle V2 (eng. *Version 2*).

Oracle Version 3 je objavljen 1983. godine te je ta verzija napisana u C programskom jeziku, što je omogućavalo korištenje na više platformi.

Oracle 4 je prva verzija koja je podržavala konzistenciju i stalnost čitanja podataka (eng. *read consistency*). Verzija Oracle 5 se počela koristiti 1985. godine te je uključivala klijent/poslužitelj arhitekturu. U sljedećoj verziji, Oracle 6 predstavljen je PL/SQL, proceduralno proširenje programskog jezika SQL.

Oracle 7 je predstavljen 1992. godine uvodeći PL/SQL pohranjene procedure i okidače. Godine 1997. Oracle je izdao Oracle 8 koji predstavlja objektno-relacijsku bazu podataka. Sljedeća verzija, Oracle 8i izašla je 1999. godine, gdje slovo *i* znači Internet. Oracle 9i

¹Oracle (2014) – Oracle Database Concepts 11g Release 2 (11.2) http://docs.oracle.com/cd/E11882_01/server.112/e40540/intro.htm#CNCPT001, preuzeto 24. srpnja 2014.

² Poslovni software- Oracle Hrvatska d.o.o. <http://www.poslovni-software.com/tvrtke/oracle-hrvatska/29/>, preuzeto 07.kolovoza 2014.

predstavlja Oracle RAC (eng. *Real Application Clusters*) 2001. godine gdje je od strane više instanci istovremeno omogućen pristup jedinstvenoj bazi podataka.

Oracle Database 10g predstavlja mrežno računalstvo 2003. godine, gdje slovo *g* znači mreža (eng. *grid*). Sljedeća verzija, Oracle Database 11g je izašla 2007. godine. Predstavljen je niz novih značajki koje omogućavaju administratorima i programerima brzo prilagođavanje poslovnim potrebama. U srpnju 2013. godine predstavljena je verzija Oracle Database 12c, prva baza podataka dizajnirana za računalstvo u oblaku (eng. *Cloud computing*).

Neka od Oracle obilježja su portabilnost na različite platforme, usvajanje industrijskih standarda i otvorenost tehnologije. Prva softverska kompanija koja je razvila i postavila internetski softver za baze podataka, poslovne aplikacije i alate za podršku i razvoj aplikacija je Oracle.

U Hrvatskoj Oracle je osnovan 1994. godine kao Oracle Software d.o.o. Godine 2002. tvrtka mijenja ime u Oracle Hrvatska d.o.o.³

³ HROUG-Oracle u Svijetu i Hrvatskoj, http://www.hroug.hr/hr/oracle_hrvatska/oracle_u_svijetu_i_hrvatskoj, preuzeto 07.kolovoza 2014.

2. Oracle Database 11g Express Edition

2.1. Instalacija OracleXE

Oracle Database 11g Express Edition ili OracleXE je besplatna verzija Oracle sustava. Jednostavna je za instaliranje i administriranje, implementaciju i razvoj. Ima svu funkcionalnost komercijalne verzije *Oracle Database 11g Release 2*. Ograničenja verzije su:

- Prostor za korisničke podatke je ograničen na 11 GB
- Može koristiti do 1 GB radne memorije
- Koristi jedan procesor, bez obzira koliko ih poslužitelj ima

Kasnije će biti prikazano spajanje na tu bazu putem alata *SQL Developer*.

Besplatna verzija se može skinuti na službenoj stranici Oracle-a.⁴

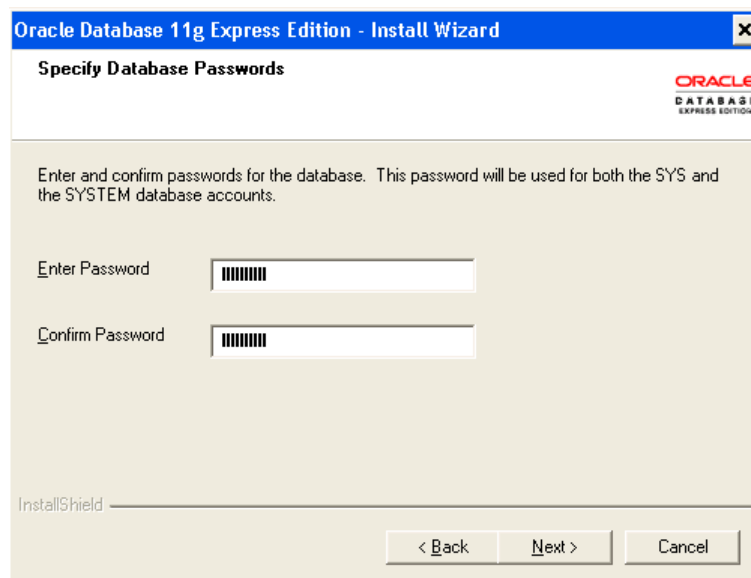
Prije samog skidanja potrebno je kreirati korisnički račun. Nakon prihvaćanja ugovora o licenci može započeti preuzimanje.



Slika 1. Instaliranje OracleXE

⁴ Oracle Database Express Edition 11g Release 2 download-
<http://www.oracle.com/technetwork/database/database-technologies/express-edition/downloads/index.html>,
preuzeto 07.kolovoza 2014.

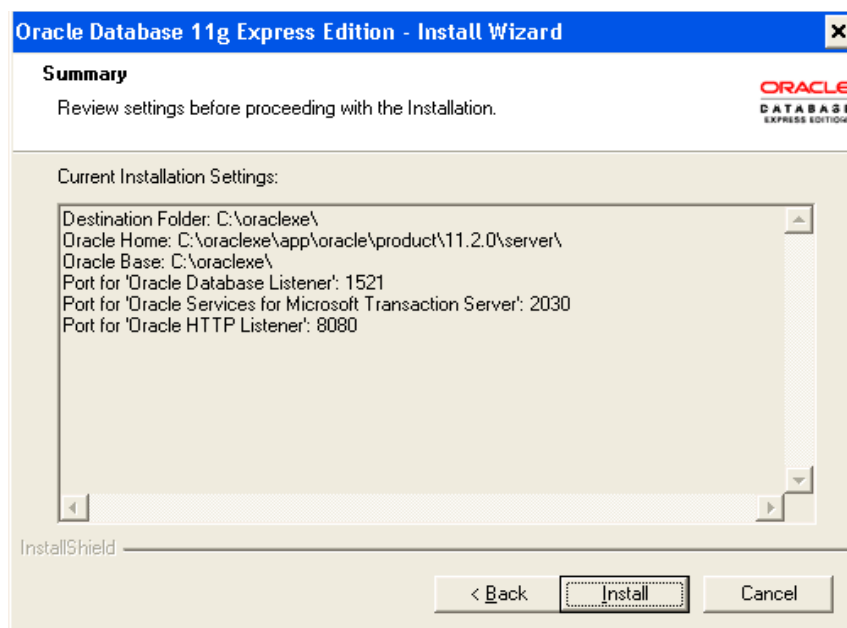
Kada se skine instalacijski paket, pokreće se *setup.exe* koji se nalazi u datoteci *DISC1*. Tokom instalacije treba odabrati lozinku za SYS i SYSTEM korisnički račun putem kojeg ćemo se povezati na bazu podataka.



Slika 2. Instaliranje OracleXE -postavljanje lozinke

Prije same instalacije potrebno je provjeriti da li je uključen vatrozid (eng. *Firewall*) jer zbog njega možda neće biti omogućen pristup do određenih portova:

- 1521:Oracle Database Listener
- 2030:Oracle Services for Microsoft Transaction Server
- 8080:Oracle HTTP Transaction Server



Slika 3. OracleXE portovi

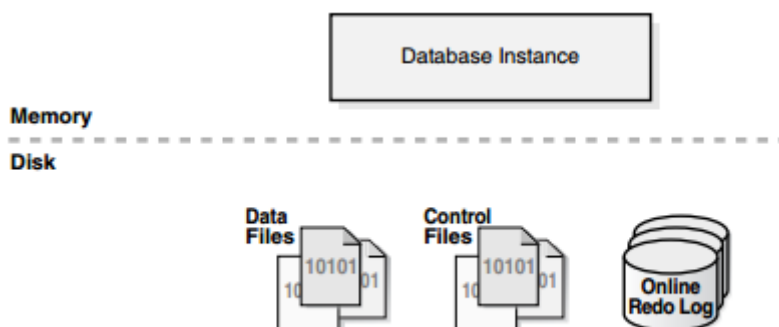
2.2. Arhitektura Oracle Database 11g Release 2⁵

Oracle sustav omogućava pohranjivanje i dohvaćanje određenih informacija. Poslužitelj upravlja velikom količinom podataka tako da više korisnika istovremeno može doći do istih podataka. Također, poslužitelj sprječava neovlašteni pristup i pruža učinkovita rješenja za neuspješni oporavak baze.

Baza podataka ima logičke i fizičke strukture koje su odvojene što znači da se fizičkom strukturom za pohranu podataka može upravljati bez utjecaja na logičku. Na primjer, preimenovanjem datoteke baze podataka neće doći do promjene naziva tablica koje se nalaze u njoj.

Fizičku strukturu Oracle baze podataka čine tri tipa datoteka:

- *Data files and temp files*
- *Online redo log files*
- *Control files*



Slika 4. Fizička struktura

[http://docs.oracle.com/cd/E11882_01/server.112/e40540.pdf]

Korisnički podaci nalaze se u datotekama koje sadrže podatke tzv. *data files*. Te su datoteke povezane samo sa jednom bazom podataka. Podaci su u podatkovnoj datoteci pohranjeni u bloku operacijskog sustava. Jedna ili više datoteka zauzimaju prostor na disku tzv. *tablespace*. Kada korisnici žele neku promjenu nad podacima učiniti trajnom, promjena se najprije zapisuje u datoteke tzv. *redo log files*, a tek nakon toga u podatkovne datoteke. Te se datoteke koriste u slučaju prekida rada baze kako bi se podaci doveli u prijašnje stanje. Kontrolna

⁵Oracle (2014) – Oracle Database Concepts 11g Release 2 (11.2)
http://docs.oracle.com/cd/E11882_01/server.112/e40540.pdf, preuzeto 10.kolovoza 2014.

datoteka tzv. *control file* sadrži informacije o podatkovnim i *redo log* datotekama. Instanca baze koristi ovu datoteku kako bi pronašla navedene datoteke na disku.

Instanca baze podataka je skup memorijskih struktura koja upravlja datotekama baze podataka.

Sistemske administratori vide datoteke na disku, a programeri segmente. Logički je prostor na disku kolekcija jednog ili više segmenata, dok se fizički sastoji od jedne ili više podatkovnih datoteka. Baza podataka mora imati SYSTEM i SYSAUX prostor. Oracle baza podataka automatski dodjeljuje prvu podatkovnu datoteku za SYSTEM prostor kod stvaranja baze podataka.

Logička struktura podataka omogućava Oracle-u kontrolu korištenja prostora na disku.

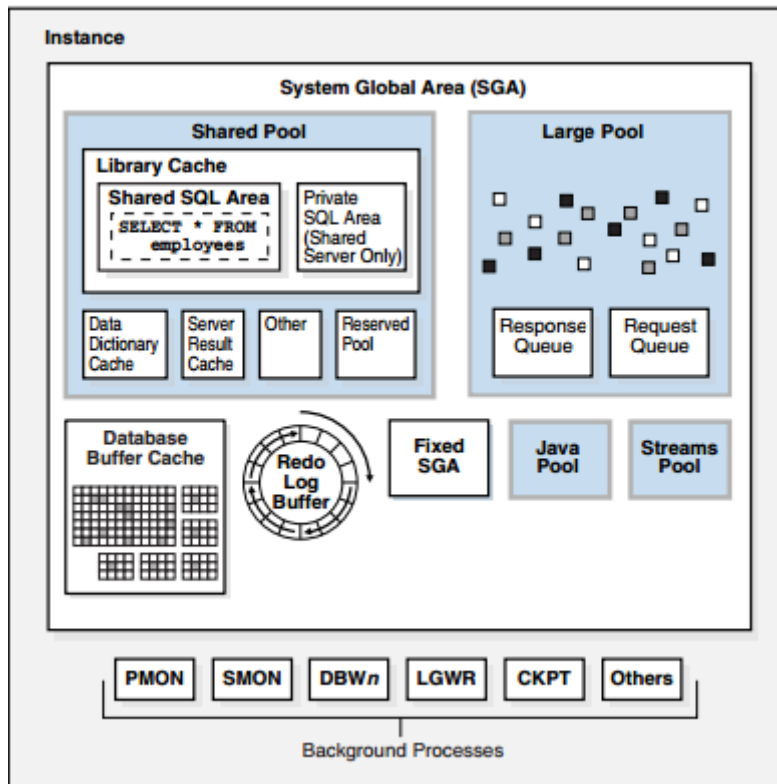
Logičke jedinice baze podataka raspodijeljene su na:

- *Data blocks*
- *Extents*
- *Segments*
- *Tablespaces*

Podaci su u logičkom smislu pohranjeni u segmentima, a segment se sastoji od blokova. Uslijed širenja segmenta je potrebno dodati nove blokove. Oni se ne dodaju jedan po jedan, već se grupiraju u tzv. *extent*-e. Veze fizičkih i logičkih struktura podataka pohranjene su u rječniku podataka.

Oracle poslužitelj se sastoji od instance i baze podataka. Svaka pokrenuta baza podataka je povezana sa najmanje jednom instancom. Baza podataka je skup fizičkih datoteka na disku kreiranih naredbom CREATE DATABASE.

Kada je instanca pokrenuta, Oracle Database alocira memorijski prostor koji se zove System Global Area (SGA) i započinje najmanje jedan pozadinski proces. Ti procesi komuniciraju sa SGA te zajedno čine instancu. Svaka instanca ima svoj vlastiti SGA. Kada je instanca zaustavljena oslobađa memoriju.



Slika 5. Instanca

[http://docs.oracle.com/cd/E11882_01/server.112/e40540.pdf]

U memoriji se pohranjuju:

- Programski kod
- Potrebne informacije tijekom izvođenja programa, primjerice trenutno stanje upita na temelju kojeg su dohvaćaju redovi i dr.

Osnovnu strukturu memorije čine:

- *System Global Area (SGA)*
- *Program Global Area (PGA)*
- *User Global Area (UGA)*

Program Global Area je memorijski prostor koji se dodjeljuje samo jednom procesu. To je nedjeljiva memorija.

User Global Area je memorija koja je alocirana za varijable koje su određene sesijom, kao što su podaci za prijavu. UGA služi za pohranu stanja sesije.

Memorijsko područje *System Global Area* je *read-write* , što znači da se iz nje može čitati i u nju upisivati te se sastoji od nekoliko memorijskih struktura:

- *Shared pool*
- *Database buffer cache*
- *Redo log buffer cache*
- *Large POOL*
- *Java POOL*
- *Streams POOL*

Shared pool sprema nedavno korištene SQL naredbe i informacije iz rječnika podataka tzv. *data dictionary*. Međuspremnik tzv. *database buffer cache* se koristi za spremanje nedavno korištenih blokova podataka. Za praćenje promjena koje u bazi izvedu procesi koristi se *redo log* međuspremnik. Također, upisuje sve promjene koje se događaju na blokovima podataka. *Large POOL* je opcionalno memorijsko područje koje služi za smanjenje opterećenja kojem je izložen zajednički *pool*. *Java POOL* je opcionalna struktura nužna u slučaju da se koristi Java. *Streams POOL* se koristi u sklopu infrastrukture za asinkrono dijeljenje podataka.

Svaka instanca mora uključiti obavezne pozadinske procese (eng. *Background processes*):

- *DBWn*
- *LGWR*
- *CKPT*
- *SMON*
- *PMON*

Database writer (DBWn) proces upisuje izmijenjene podatke iz međuspremnika u podatkovne datoteke. *Log writer (LGWR)* proces upisuje izmjene iz *redo log* međuspremnika u *redo log* datoteke. Te izmjene su nam poznate pod nazivom *redo log* podaci. *Checkpoint (CKPT)* proces je odgovoran za ažuriranje informacija o statusu baze podataka. Događa se svaki put kada su izmijenjeni podaci iz međuspremnika trajno upisani u podatkovne datoteke tokom *checkpoint* događaja ili nakon izmjene aktivne *redo log* datoteke. *System Monitor (SMON)* proces obavlja oporavak instance prilikom podizanja. Također, zadatak mu je očistiti privremene segmente koji više nisu u upotrebi te oporaviti transakcije ispuštene tokom pada sustava. *Process Monitor (PMON)* proces obavlja oporavak procesa kada dođe do neuspjeha

kod korisničkih procesa. Taj je proces odgovoran za čišćenje priručne memorije i oslobađanje resursa sustava koje je koristio korisnički proces.

Kod pokretanja Oracle-a najprije se kreira instanca u memoriji koja se zatim povezuje sa bazom podataka koja postoji na diskovima. Na kraju se baza podataka otvara od strane korisnika. Prilikom gašenja sustava, instanca se briše iz memorije, ali baza podataka i dalje postoji na disku. Nemoguće je povezati se na Oracle bazu podataka ukoliko nije pokrenuta instanca.

Administratori baza podataka moraju dobro razumjeti Oracle Database arhitekturu i princip rada.

Osnovni zadatak administratora baza podataka je omogućiti korisnicima dostupnost podataka.

Neki od mogućih zadataka administratora su:

- Instaliranje, nadogradnja Oracle Database softvera
- Dizajn baze podataka, izrada logičkog (konceptualnog) modela
- Izrada Oracle baze podataka
- Razvoj i testiranje sigurnosne kopije i oporavka u slučaju neuspjeha
- Podešavanje mrežnih postavki kako bi se klijentima omogućilo povezivanje s bazama podataka
- Pokretanje i zatvaranje baze podataka
- Upravljanje skladištem baza podataka
- Upravljanje korisnicima i sigurnošću
- Upravljanje objektima baze podataka (tablice, indeksi, pogledi)

Oracle Database programer stvara i održava aplikaciju baze podataka (eng. *Database application*). To je aplikacija čija je primarna svrha unošenje i dohvaćanje podataka iz računalne baze podataka.

Oracle programeri razvijaju nove aplikacije ili prilagođavaju postojeće kako bi funkcionirale na Oracle okruženju. Iz tih razloga programeri surađuju sa administratorima baza podataka te međusobno dijele znanje i informacije.

Neki od mogućih zadataka programera su:

- Stvaranje sheme objekata
- Osigurati integritet podataka
- Odabrati razvojno okruženje za razvoj aplikacije

- Pisati PL/SQL na strani poslužitelja i SQL naredbe na strani klijenta

Jedan od alata kojim se koriste je i SQL Developer koji će biti opisan u nastavku rada.

2.3. Tipovi podataka⁶

Svaki podatak pohranjen u tablici mora imati svoj tip podatka. Tip podatka definiramo prilikom kreiranja tablice, navodeći ga neposredno nakon naziva stupca.

Ugrađeni tipovi podataka u Oracle sustavu:

Znakovni tip podataka

CHAR	niz znakova fiksne dužine, maksimalna veličina je 2000 B
NCHAR, NVARCHAR2	Unicode znakovi
VARCHAR2	niz znakova varijabilne dužine, maksimalna veličina je 4000 B

Tabela 1. Znakovni tip podataka

CHAR tip podatka definira niz znakova fiksne duljine. Oracle osigurava da sve vrijednosti upisane u stupce imaju svoju veličinu. Ako upišemo vrijednost koja je manja od zadane veličine, sustav će dodavati praznine dok ne dođe do te veličine. Upišemo li vrijednost koja bi zauzimala više mjesta, Oracle će vratiti grešku.

VARCHAR2 tip podatka definira niz znakova varijabilne dužine. Upišemo li vrijednost koja je manja od zadane, sustav će popuniti točno toliko mjesta koliko smo znakova unijeli. Unesemo li vrijednost koja je veća od zadane, sustav će jednostavno odbaciti višak.

Numerički tip podataka

NUMBER (p,s)	1-22 B
FLOAT (p)	1-22 B
BINARY_FLOAT	4 B
BINARY_DOUBLE	8 B

Tabela 2. Numerički tip podatka

⁶ Oracle (2014) – Oracle Database SQL Language Reference 11g Release2 (11.2) http://docs.oracle.com/cd/E11882_01/server.112/e26088/sql_elements001.htm#SQLRF0021, preuzeto 23.kolovoza 2014.

NUMBER (p,s) može spremati decimalni i cijeli broj. Definira se kao preciznost koja označava ukupan broj znamenaka i skala koja označava broj znamenki od decimalnog zareza. Cijeli broj možemo definirati sa NUMBER(p).

Podatak 123.89 će biti zapisan kao 123.89 ako koristimo NUMBER. Ako koristimo NUMBER(3) zapisat će se kao 124 jer ne može imati više od 3 znamenke itd.

FLOAT tip je podtip od NUMBER. Može biti zapisan sa i bez preciznosti. Oracle sustav koristi FLOAT uglavnom kada ga pretvara iz ANSI FLOAT tipa podatka. Oracle preporuča korištenje BINARY_FLOAT i BINARY_DOUBLE.

Datum i vrijeme

DATE predstavlja tip podatka koji datum sprema u formatu dd.mm.yyyy zajedno sa vremenom.

TIMESTAMP sprema godinu, mjesec, sat, minutu i sekundu.

TIMESTAMP WITH TIME ZONE radi isto što i TIMESTAMP, jedino se još ispisuje vremenska zona.

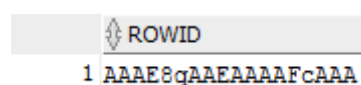
INTERVAL YEAR TO MONTH sprema interval koji predstavlja razdoblje godine i mjeseca.

ROWID tip podatka

To je oznaka dodatnog stupca koji sprema vrijednost *rowid*. To je jedinstveni identifikator koji sustav generira za svaki red. Predstavlja adresu, odnosno lokaciju podatka.

Na primjer:

```
SELECT rowid
FROM korisnici
WHERE id_korisnika=1;
```



	ROWID
1	AAAE8qAAEAAAFcAAA

Slika 6. Rezultat ROWID

Oracle sustav podržava i ANSI tipove podataka. Sustav ih automatski pretvara u ekvivalentne Oracle tipove podataka. Na slici 7. su prikazani ANSI i ekvivalentni Oracle tipovi podataka.

ANSI SQL Data Type	Oracle Data Type
CHARACTER (n) CHAR (n)	CHAR (n)
CHARACTER VARYING (n) CHAR VARYING (n)	VARCHAR2 (n)
NATIONAL CHARACTER (n) NATIONAL CHAR (n) NCHAR (n)	NCHAR (n)
NATIONAL CHARACTER VARYING (n) NATIONAL CHAR VARYING (n) NCHAR VARYING (n)	NVARCHAR2 (n)
NUMERIC [(p, s)] DECIMAL [(p, s)] (Note 1)	NUMBER (p, s)
INTEGER INT SMALLINT	NUMBER (p, 0)
FLOAT (Note 2) DOUBLE PRECISION (Note 3) REAL (Note 4)	FLOAT (126) FLOAT (126) FLOAT (63)

Slika 7. Ekvivalentni ANSI i Oracle tipovi podataka

[http://docs.oracle.com/cd/E11882_01/server.112/e26088/sql_elements001.htm#SQLRF0021]

3. Oracle SQL Developer

Oracle SQL Developer je besplatni grafički alat koji omogućuje pregledavanje, kreiranje, uređivanje i brisanje objekata baze podataka. U njemu se izvršavaju SQL naredbe i skripte, uređuje i ispravlja PL/SQL kod. Također, omogućen je izvoz podataka, migracije baza podataka te pregled metapodataka i kreiranje izvješća.

Kako bi koristili taj alat moramo imati pristup bazi podataka. Zato je potrebno instalirati Oracle Database 11g Express Edition.

U ovom radu naglasak je na modeliranju i implementaciji baze podataka. Kao primjer biti će modelirana i implementirana baza za eImenik. eImenik bi trebala biti aplikacija koja zamjenjuje papirnate imenike u osnovnoj i srednjoj školi. Namijenjena je profesorima, koji ujedno mogu biti i razrednici nekom razredu te učenicima.

3.1. Instalacija i spajanje na bazu podataka

Posljednja verzija Oracle SQL Developer-a 4.0.2. se može skinuti na službenoj stranici Oracle-a.⁷

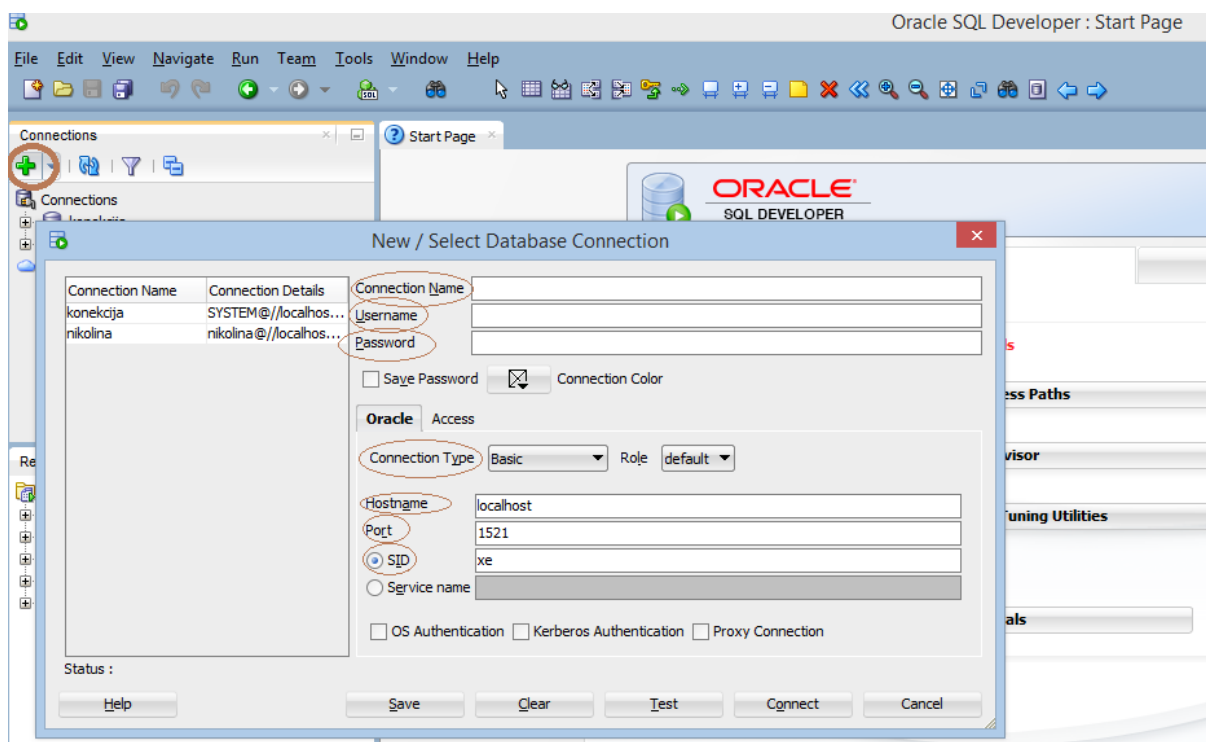


Slika 8. Skidanje Oracle SQL Developer

⁷ Oracle SQL Developer Download <http://www.oracle.com/technetwork/developer-tools/sql-developer/overview/index-097090.html>, preuzeto 15.srpnja 2014

Skinutu arhivu treba raspakirati na željenoj lokaciji. Zatim treba otvoriti raspakiranu mapu *sqldeveloper* te pokrenuti istoimenu datoteku (*sqldeveloper.exe*). Nije potrebna dodatna instalacija.

SQL Developer omogućava programerima i administratorima pretraživanje, stvaranje te ažuriranje podataka u bazi podataka. Da bi mogli izvoditi sve te radnje potrebno je stvoriti barem jednu konekciju (eng. *Connection*). Stvaranje konekcije je moguće klikom na znak plus pod *Connections*.



Slika 9. Stvaranje konekcije

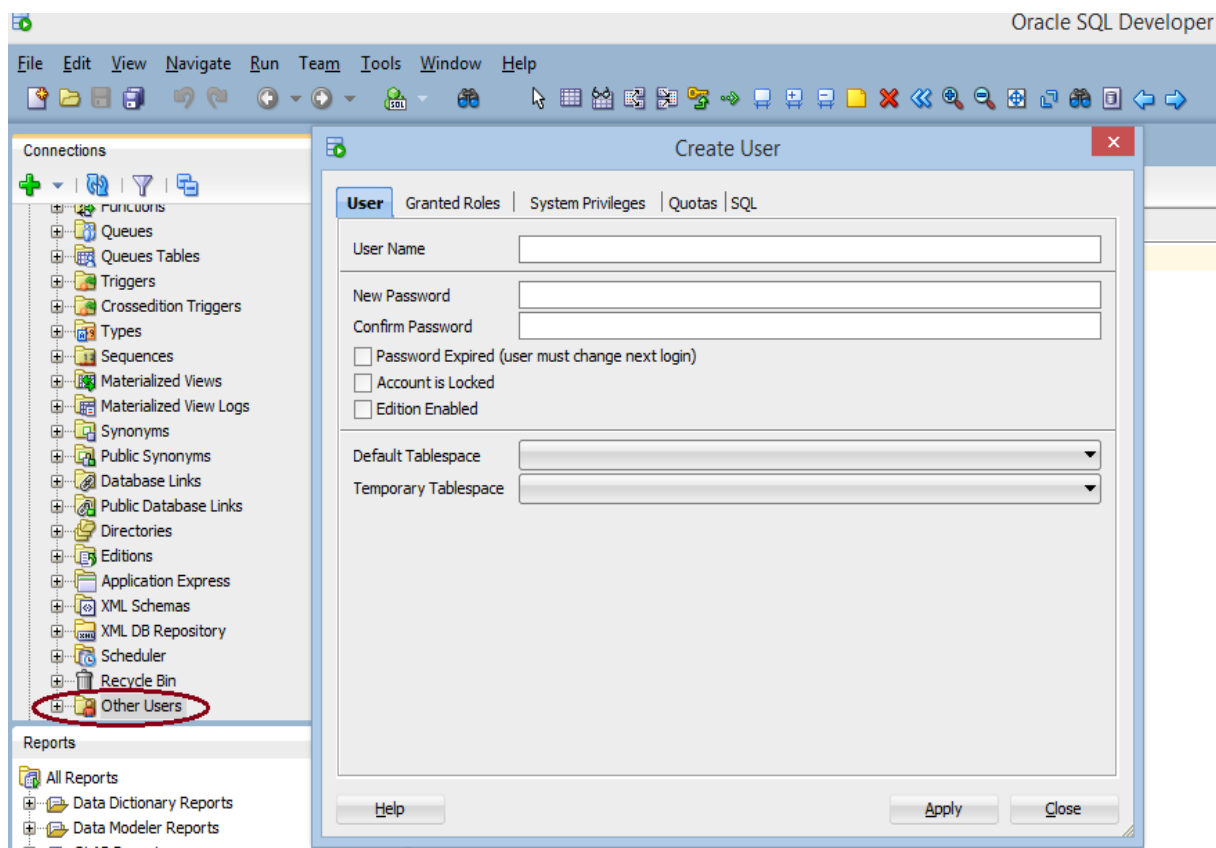
Za spajanje na bazu podataka potrebno je ispuniti sljedeća polja:

- *Connection Name*: željeni naziv konekcije
- *Username*: SYSTEM (prethodno kreiran kod instalacije OracleXE)
- *Password*: lozinka kreirana kod instalacije OracleXE
- *Connection Type*: odabrati *Basic*
- *Hostname*: localhost (kada se spaja na bazu koja se nalazi na istom računalu gdje i SQL Developer)
- *Port*: 1521-postavljeni port na kojem se osluškuje (eng. *Listener port*)

- *SID*: identifikator sustava, *orcl* (za Oracle Database 10g i Oracle Database 11g) ili *xe* (Oracle Database 10g (11g) Express Edition)

Nakon toga treba kliknuti na gumb *Test* za validaciju konekcije. Ukoliko je konekcija uspješna prikazat će se poruka "*Status:Success*" i zatim se možemo spojiti na bazu podataka.

Kada se uspješno spojimo, možemo kreirati novog korisnika (shemu). Pod izbornikom kreirane konekcije pronađemo *Other Users* i desnim klikom miša odaberemo *Create User*.

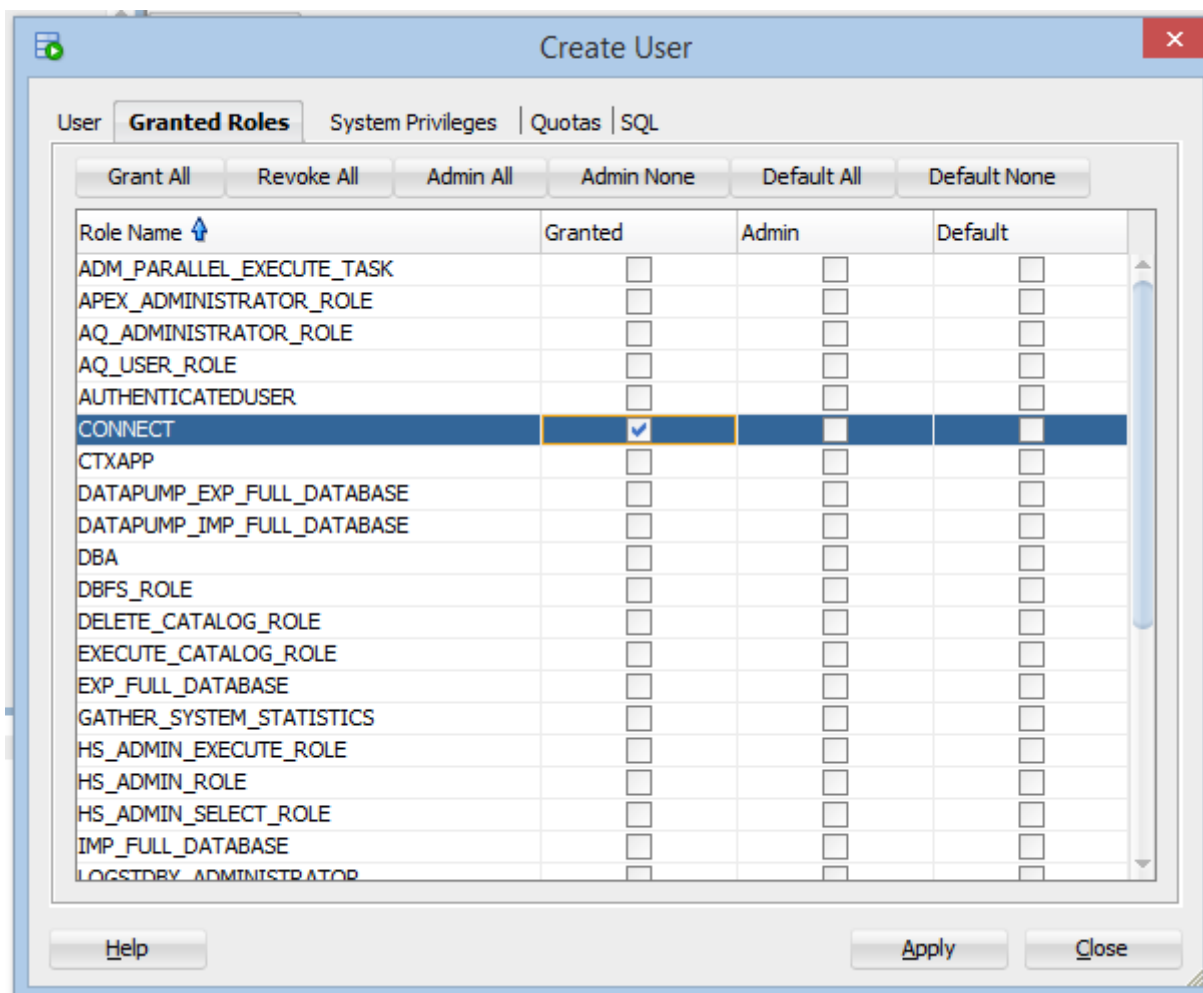


Slika 10. Stvaranje novog korisnika

Zatim treba odabrati *User Name* i *New Password* sa kojim kasnije možemo napraviti novu konekciju.

Pod *Default Tablespace* odabiremo *tablespace* u koji želimo da nam se spremaju podaci, odabiremo *USERS* ili *tablespace* koji smo sami kreirali. Pod *Temporary Tablespace* odaberemo *TEMP*.

Nakon toga odaberemo karticu *Granted Roles* gdje se dodjeljuju uloge korisnika. Da bi se korisnik mogao spojiti na bazu sa prije upisanim korisničkim imenom i lozinkom treba označiti *Granted* kod *Connect*.



Slika 11. Dodjeljivanje uloga korisniku

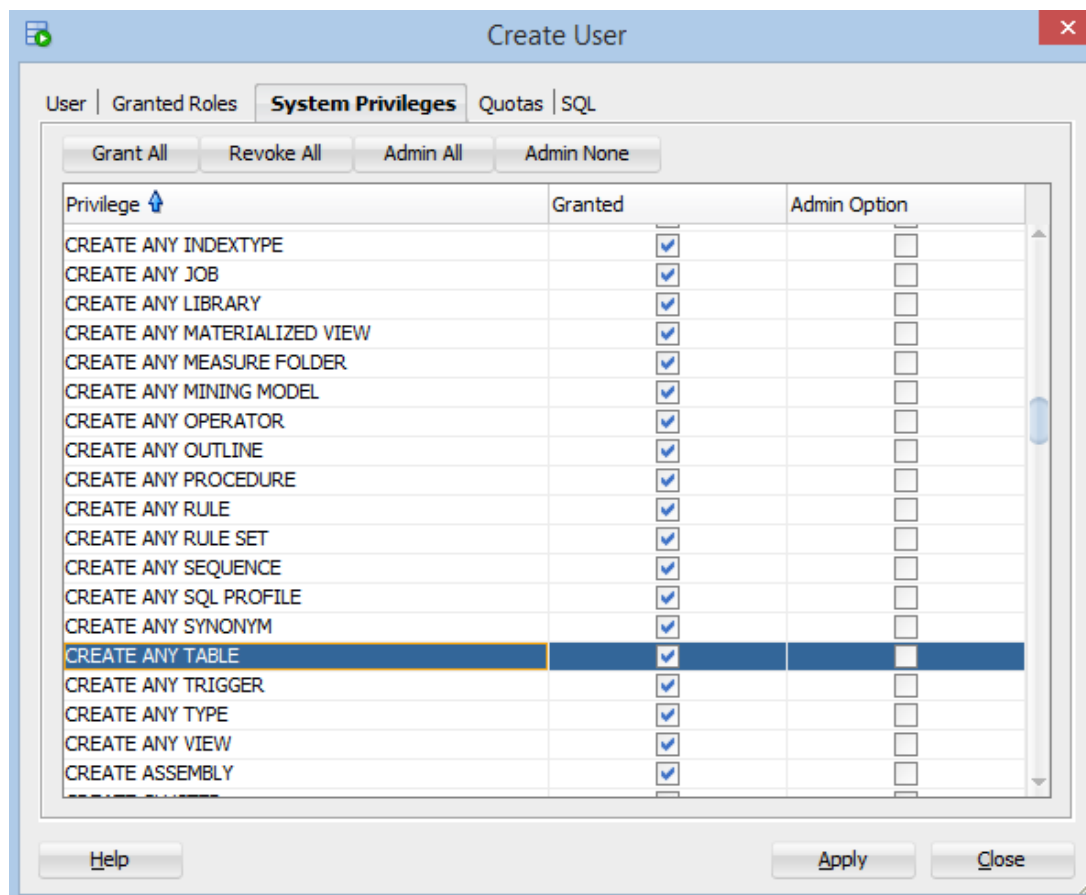
Nakon toga odaberemo karticu *System Privileges* da dodijelimo željene privilegije sustava jer inače korisnik neće moći ništa obavljati, na primjer neće moći kreirati tablicu.

Za početak možemo odabrati samo neke privilegije kao što su:

- CREATE SEQUENCE
- CREATE SESSION
- CREATE TABLE
- CREATE TRIGGER
- CREATE TYPE
- CREATE USER

- CREATE VIEW
- DELETE ANY TABLE

Privilegije se mogu naknadno dodavati, no možemo odmah sve označiti.



Slika 12. Dodjeljivanje privilegija

Kada se to završi kreiramo novu konekciju, sa parametrima korisnika kojeg smo kreirali. Kada je otvorimo, dobijemo shemu za kreiranje novih objekata kao što su tablice, pogledi, okidači i ostalo.

3.2. Modeliranje

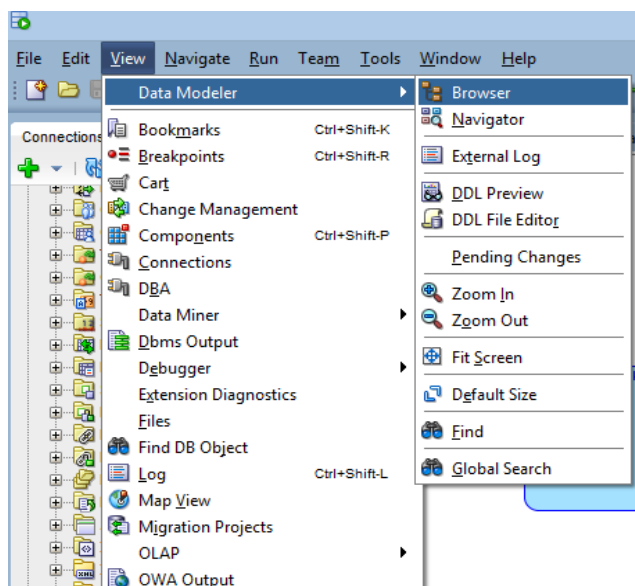
„Modeliranje podataka općenito je postupak oblikovanja podataka koji će biti smješteni u bazi podataka informacijskog sustava, a kojima se prikazuju stanja stvarnog poslovnog sustava“ [Varga, 2012, str. 100].

Modeliranje se obično dijeli u tri faze:

- Konceptualno modeliranje je modeliranje gdje je rezultat konceptualna shema baze podataka koja se sastoji od entiteta, atributa i veza. Najvažnije svojstvo konceptualne sheme je da bude razumljiva ljudima svih struka. Kod te faze modeliranja bitno je da projektanti prepoznaju sve procese i uključe sve podatke.
- Logičko modeliranje je modeliranje gdje nastaje shema sastavljena od relacija (tablica). U toj su shemi entiteti i veze pretvoreni u relacije. Najvažnije je da se ta shema može više-manje izravno implementirati pomoću sustava za upravljanje bazama podataka.
- Fizičko modeliranje gdje je rezultat fizička shema koja se kreira koristeći SQL naredbe te se relacije iz logičke sheme implementiraju kao tablice. [Manger, 2012, str. 8]

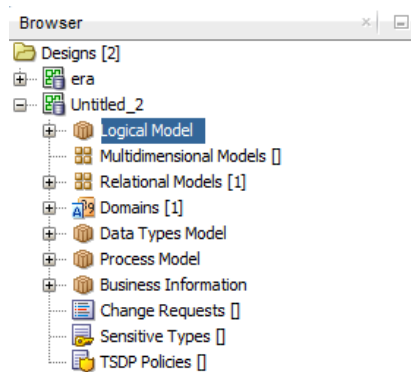
U alatu, dizajn se sastoji od konceptualnog modela, opcionalno jednog ili više relacijskih modela baziranih na konceptualnom te jednog ili više fizičkih modela s obzirom na relacijski.

Da bi krenuli sa izradom modela idemo pod *View->Data Modeler->Browser*.



Slika 13. Otvaranje Data Modeler-a

Browser se zatim prikazuje s lijeve strane gdje odabiremo model kojeg ćemo izraditi.



Slika 14. Odabir modela

U ovom radu biti će detaljno prikazano kako izraditi konceptualni (eng. *Logical Model*) i relacijski (eng. *Relational Model*) koji je potreban za implementaciju baze podataka te sama implementacija.

Prije same izrade bitno je utvrditi što je ER model i relacijski model te pojmove kojima su opisani.

3.2.1. ER konstrukti

Osnovni konstrukti svakog ER modela su entiteti, atributi i veze. Najvažnije je znati ih razlikovati.

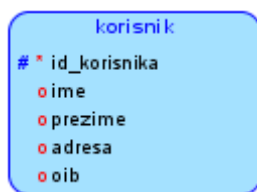
Entitet je nešto o čemu želimo spremati podatke. To je nešto što postoji i što se može identificirati. Entitet može biti stvar ili biće, događaj ili pojava, npr: *korisnik*, *razred*, *predmet*.



Slika 15. Entitet korisnik

„Entitet je stvaran ili apstraktan predmet ili događaj o kojem se u informacijskom sustavu prikupljaju podaci“ [Varga, 1994, str. 45].

Entitet je opisan atributima. Atributi su svojstva entiteta ili veza. Na primjer za entitet *korisnik* možemo uvesti attribute: *ime*, *prezime*, *adresa*, *oib*.



Slika 16. Atributi entiteta korisnik

Kada bi neki atribut imao svoje attribute, tada bi on postao novi entitet.

Atributi koji pripadaju nekom entitetu zajedno čine tip entiteta. Za tip entiteta može postojati skup primjeraka entiteta od kojih je svaki opisan drugačijim vrijednostima atributa.

Na primjer, *korisnik* je tip čiji su primjerci Ana Anić i Mara Marić. Svaki od tih primjeraka ima drugačiju kombinaciju vrijednosti za attribute ime, prezime, adresu, oib.

Svaki primjerak entiteta mora biti jednoznačno određen te zbog toga mora postojati kandidat za ključ. Ukoliko imamo više kandidata, treba odabrati jedan koji će predstavljati primarni ključ. Na primjer kandidati za *korisnik* mogu biti i oib ili korisničko ime. Odabir primarnog ključa jedna je od najvažnijih odluka jer će se koristiti kod same implementacije baze podataka [Manger, 2012, str. 20].

Između dvaju ili više entiteta koji su u odnosu mora postojati veza. Razlikujemo unarnu, binarnu i ternarnu vezu. Unarna veza uspostavlja se između jednog entiteta, binarna između dva, a ternarna između tri entiteta.

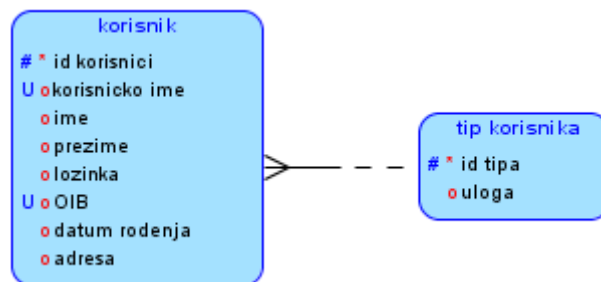
Također, postoje i načini na koji veza može povezati entitete. Zbog toga razlikujemo:

- Veza 1:1

Ta se veza danas rijetko koristi. Jedan primjerak tipa entiteta može biti povezan samo s jednim primjerkom drugog tipa entiteta. Da bi izbjegli stvaranje novog tipa entiteta, možemo sve primjerke staviti pod isti tip entiteta.

- Veza 1:N

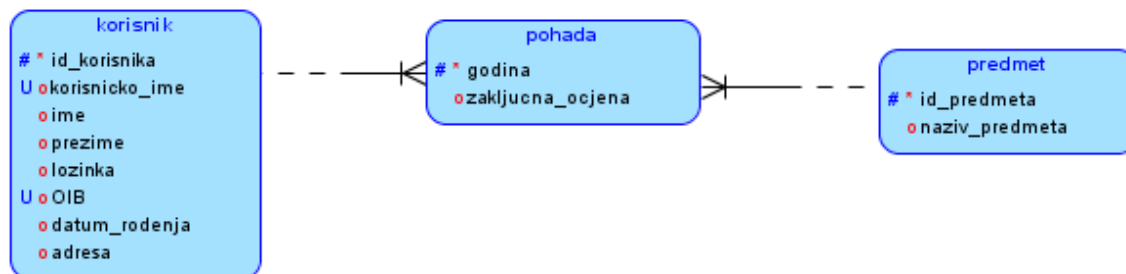
Jedan primjerak tipa entiteta može biti povezan s više primjeraka drugog tipa entiteta. Na primjeru možemo vidjeti da korisnik ima jednu ulogu, a uloga ima više korisnika.



Slika 17. Veza 1:N

- Veza N:M

Jedan primjerak tipa entiteta može biti povezan s više primjeraka drugog tipa entiteta. Kod te veze se dodaje nova tablica jer se ta veza razdvaja na dvije 1:M veze. Na primjeru možemo vidjeti da korisnik pohađa više predmeta, a predmet je slušan od strane više korisnika te je zbog toga dodana nova tablica *pohada*. U tablici *pohada* su dodani i vanjski ključevi *id_korisnika* i *id_predmeta*, no nisu prikazani u odabranoj notaciji (Barkerova). Kasnije se ti atributi dodaju sami kod pretvaranja konceptualnog modela u relacijski.



Slika 18. Veza N:M

3.3. Konceptualni model

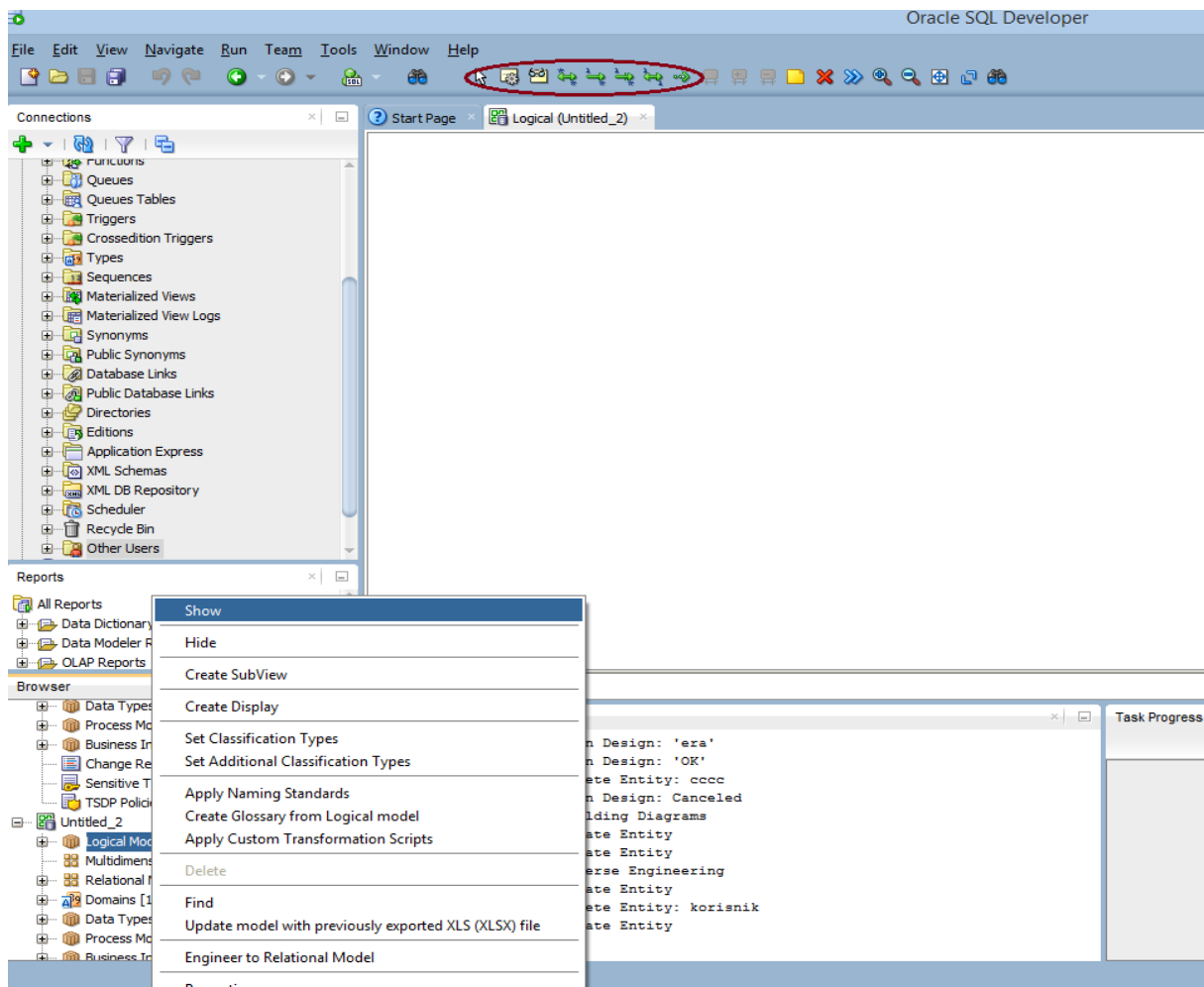
Logical model u ovom alatu predstavlja ustvari konceptualno oblikovanje baze podataka. Srž modeliranja baze podataka je model (eng. *Logical model*) koji se također naziva ER dijagram (eng. *Entity-Relationship diagram*). Konceptualni model se kasnije može pretvoriti u jedan ili više relacijskih modela. On omogućava stvaranje i upravljanje entitetima, atributima, jedinstvenim identifikatorima i vezama. Popis svih modela se nalazi u izborniku *Browser*.

Kod izrade modela moguće je odabrati notaciju:

- *Bachman notation*
- *Barker notation*
- *Information Engineering notation*

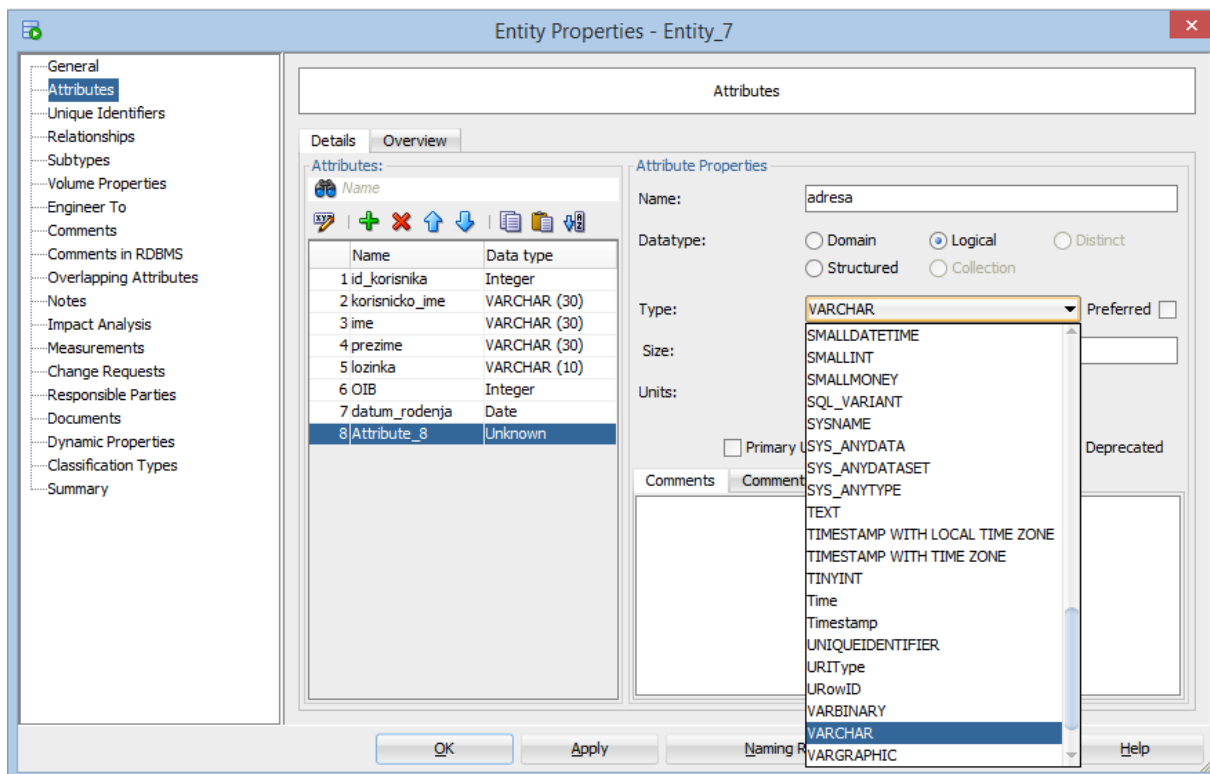
U nastavku rada korištena je *Barker notation*.

U *Browser* izborniku desnim klikom na *Logical model* odaberemo *Show* kako bi nam se otvorila prazna podloga za kreiranje modela.



Slika 19. Kreiranje konceptualnog modela

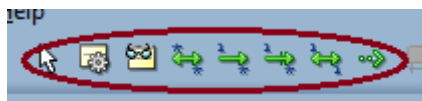
Za dodavanje entiteta koristimo ikone zaokružene na slici 19. Klikom na *New Entity* otvara nam se novi prozor u kojem definiramo novi entitet.



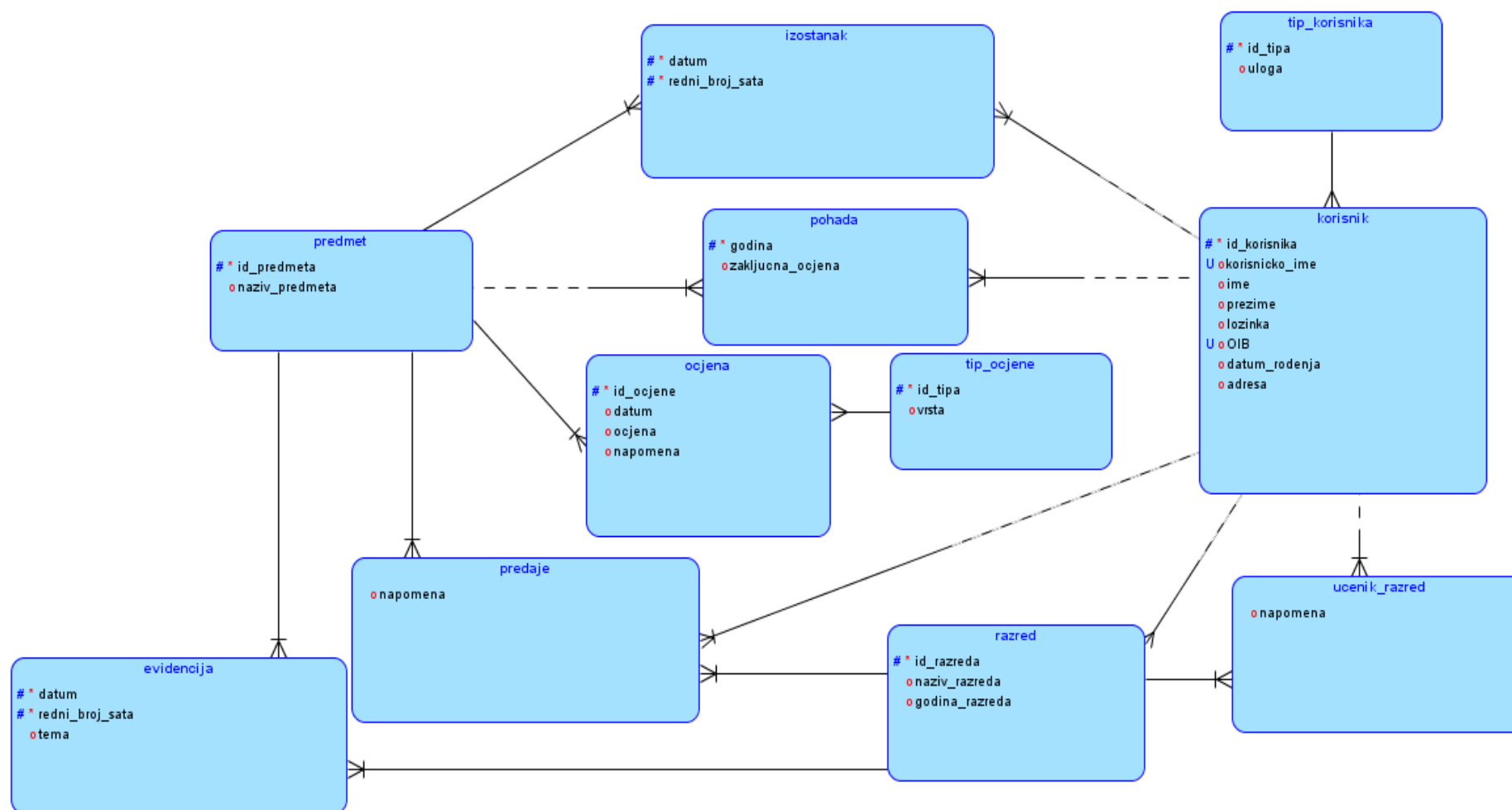
Slika 20. Kreiranje entiteta

Pod *General* upišemo samo naziv entiteta. Pod *Attributes* dodajemo attribute, a za svaki od njih definiramo tip podatka (*Datatype:Logical*).

Veze mogu biti identifikacijske i neidentifikacijske. Identifikacijske veze su prikazane punom crtom i predstavljaju vezu u kojoj se tablica dijete ne može jedinstveno identificirati bez svog roditelja. Veze možemo dodati klikom na strelice, ovisno o vrsti veze koja nam je potrebna (1:1, 1:N, N:M).



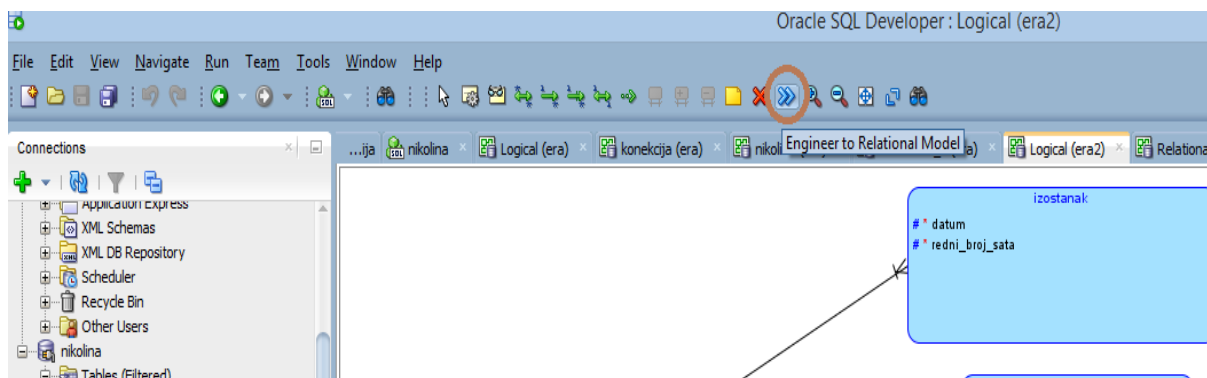
Slika 21. Dodavanje veza



Slika 22. Konceptualni model

Da bi mogli implementirati bazu podataka potrebno je konceptualni model pretvoriti u relacijski. Alat nam omogućuje automatsko pretvaranje u jedan ili više relacijskih modela i obrnuto klikom na *Engineer to Relational Model* (slika 23).

U konceptualnom modelu nisu prikazani vanjski ključevi, tipovi podataka, dodatna ograničenja te je zato potrebno izraditi relacijski model.



Slika 23. Pretvaranje konceptualnog u relacijski model

3.4. Relacijski model

Relacijski model (eng. *Relational model*) je definiran krajem 60.-ih godina 20. stoljeća u razdoblju Edgara Codd. Relacijski model zahtjeva da se baza podataka sastoji od skupa tablica, odnosno relacija [Manger, 2012, str. 37].

Svaki tip entiteta postaje relacija. Atributi tipa postaju atributi relacije. Svaka relacija ima svoje ime koje mora biti različito od ostalih. Imena relacija se pišu u množini, dok se entiteti pišu u jednini.

Tablica se sastoji od redova i stupaca. Primarni ključ tablice je jedan ili više stupaca na temelju koje se može jednoznačno identificirati svaki redak u tablici [Rabuzin, 2011, str. 5].

Ako već imamo gotov konceptualni model, moguće je isti automatski pretvoriti u relacijski. Nakon toga potrebno je preimenovati tablice da budu u množini, postaviti dodatna ograničenja, postaviti da se vrijednost primarnih ključeva automatski povećava gdje je moguće, itd. Za implementaciju baze podataka nije nužno uvijek raditi konceptualni model. Može se odmah prijeći i na izradu relacijskog modela koji je neophodan za dobru implementaciju.

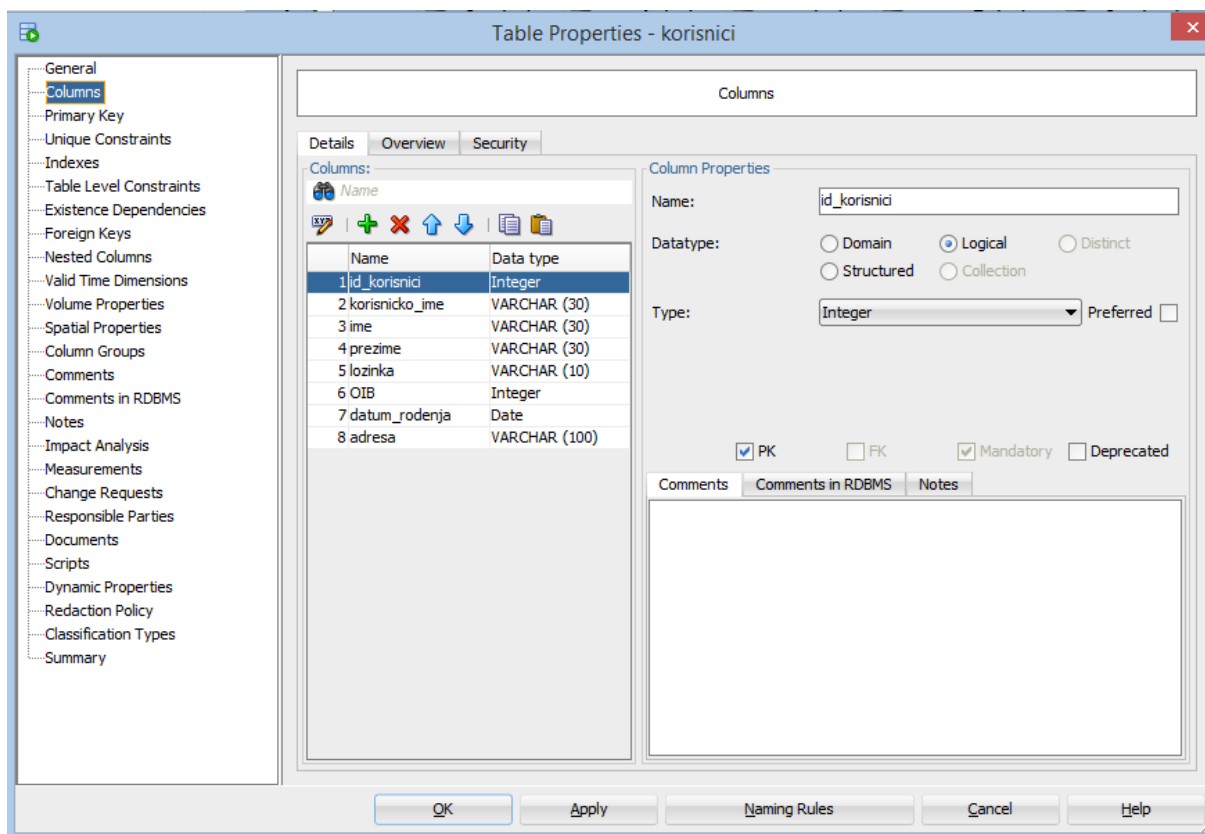
Pod izbornikom *Browser* desnim klikom odaberemo *Relational Model->New Relational Model*. Zatim nam se sa desne strane otvori prazna bijela podloga na kojem ćemo kreirati tablice.

Kliknemo na drugo ikonu *New Table* i otvorit će nam se novi prozor za stvaranje tablice.



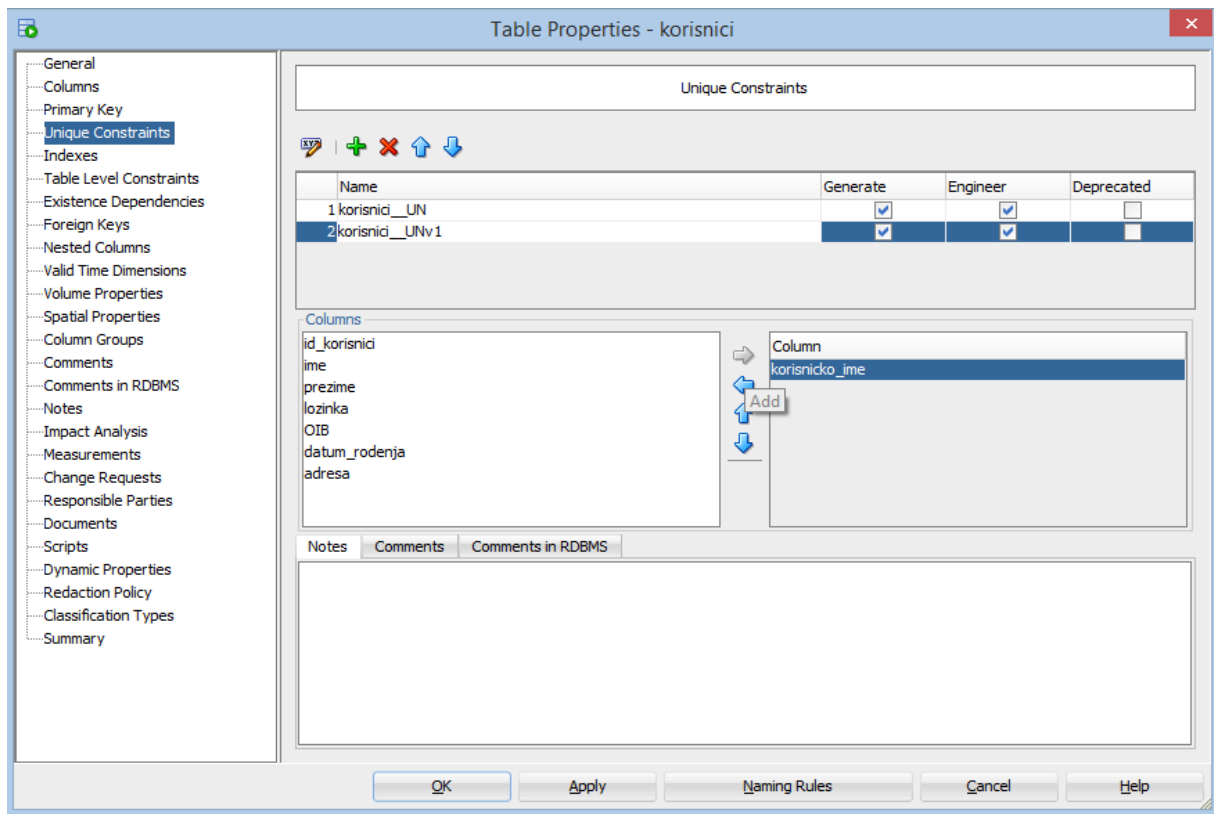
Pod karticom *General* upisujemo naziv relacije te označimo *Generate in DDL* koji će nam kasnije generirati SQL naredbe na temelju izrađenog modela.

Da bi dodali stupce moramo kliknuti na *Columns*. Tu dodajemo stupce klikom na oznaku plus. Za svaki stupac definiramo tip podatka te označimo koji nam stupac predstavlja primarni ključ.



Slika 26. Dodavanje atributa

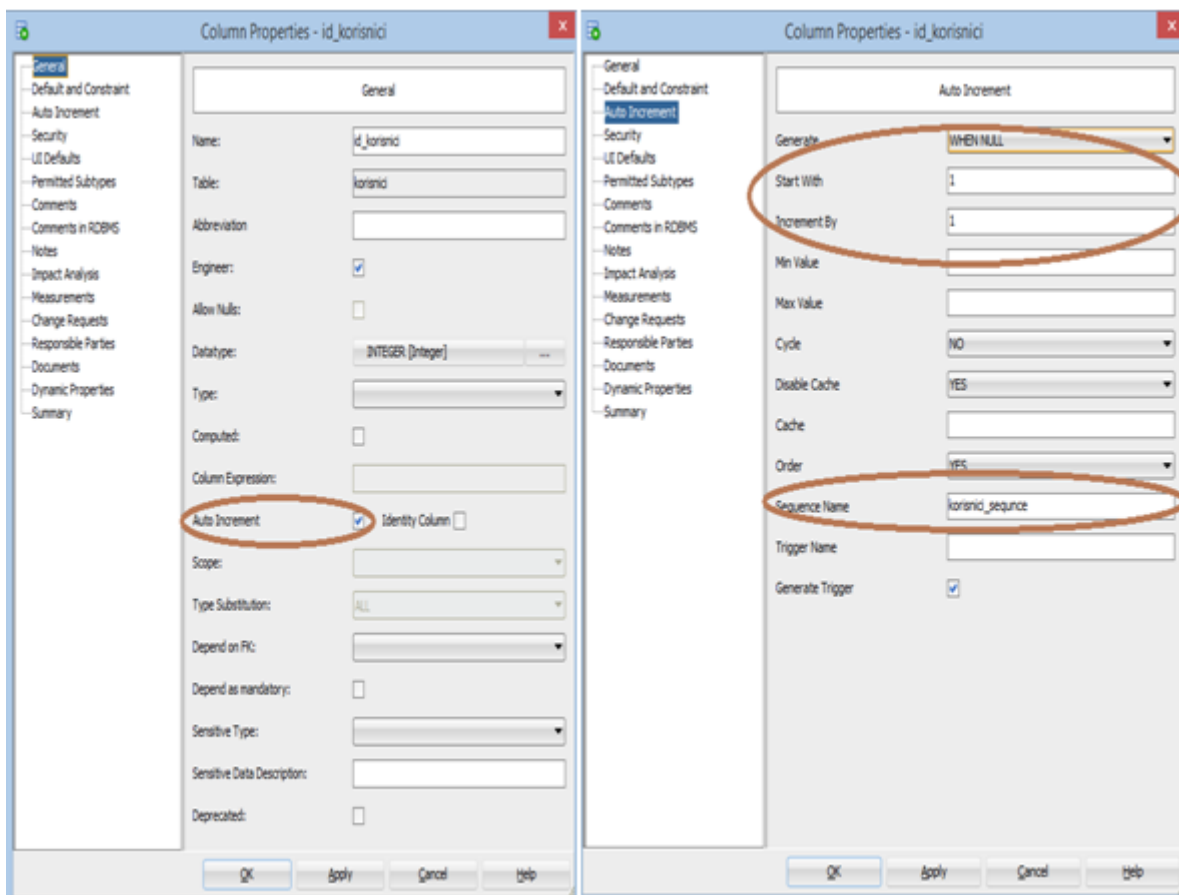
Ukoliko vrijednost nekog stupca mora biti jedinstvena, koristimo ograničenje UNIQUE (eng. *Unique Constraints*). Pod karticom *Unique Constraints* možemo odabrati attribute koji moraju biti jedinstveni, a u ovom slučaju to su *korisnicko_ime* i *OIB*.



Slika 27. Dodavanje ograničenja

Ukoliko želimo da nam se vrijednost primarnog ključa povećava automatski novim unosom, moramo kliknuti na naziv tog stupca, u ovom slučaju na *id_korisnici*. Otvara nam se novi prozor gdje pod karticom *General* označimo *Auto Increment*. Također, možemo označiti i *Allow Nulls* ukoliko je unos vrijednosti stupca opcionalan, no to nije moguće u ovom slučaju pošto vrijednost primarnog ključa ne može biti NULL.

Zatim pod karticom *Auto Increment* upišemo broj za koji želimo da bude početna vrijednost (*Start With*) te broj koji označava za koliko da nam se vrijednost povećava prilikom sljedećeg upisa (*Increment By*). Važno je da upišemo i *Sequence Name* jer Oracle nema tip podatka kojim bi automatski definirali rast vrijednosti stupca. Oracle najprije kreira sekvencu (eng. *sequence*) koja služi za generiranje brojeva, a zatim okidač (eng. *trigger*) koji automatski unosi sljedeći broj iz sekvence u *id* stupac.

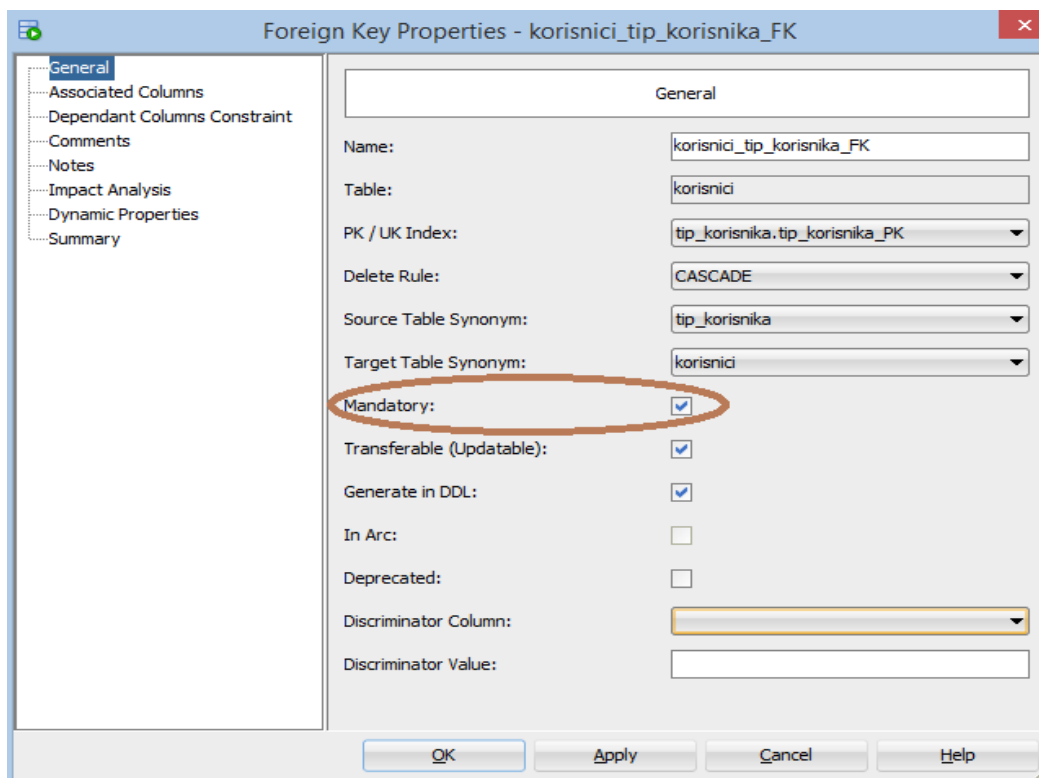


Slika 28. Auto Increment opcija

Da bi povezali tablice koriste se primarni i vanjski ključevi.

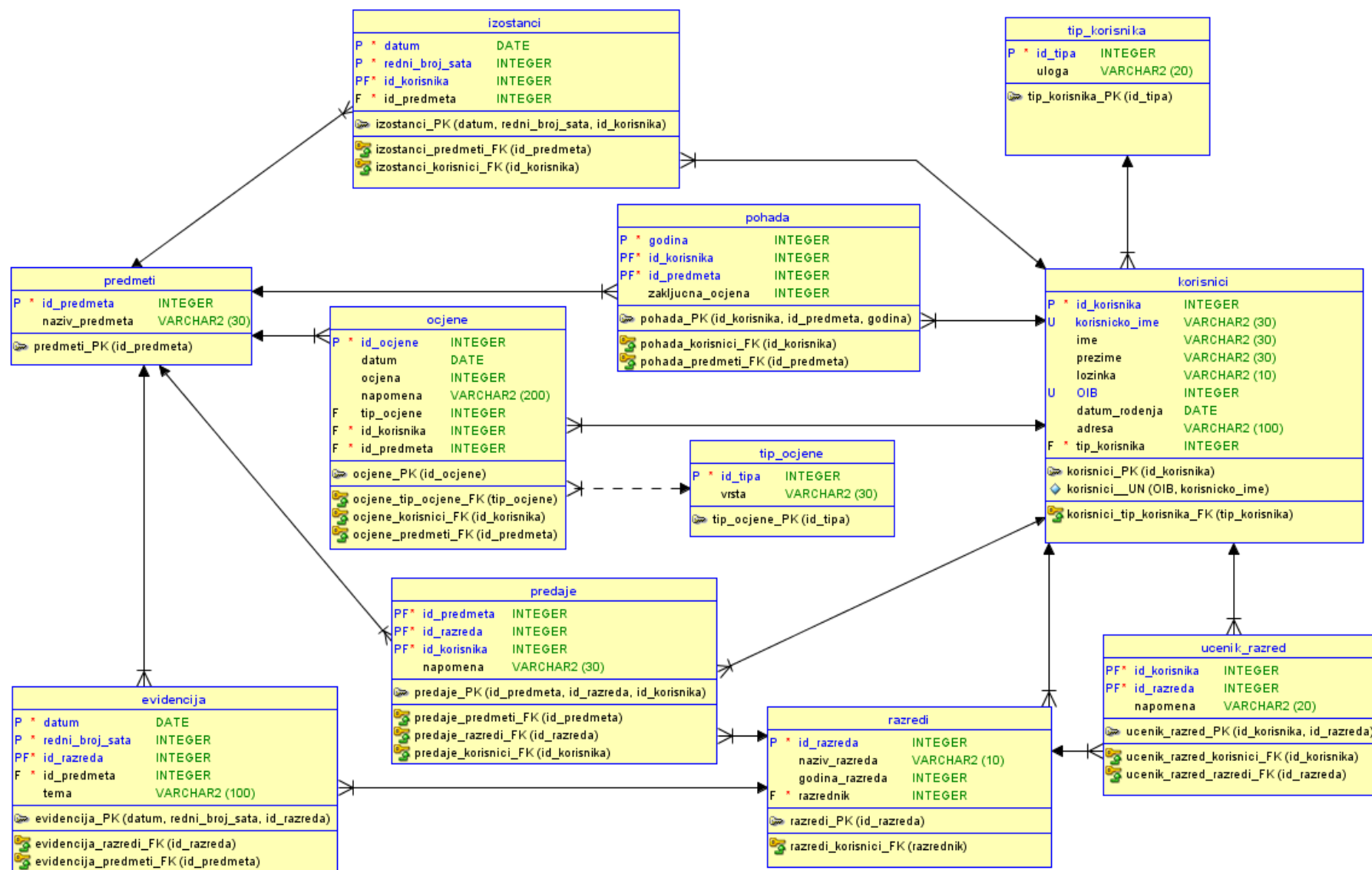
„Vanjski ključ je stupac (ili više njih) u drugoj (moguće istoj) tablici u koji pohranjujemo vrijednost(i) primarnog ključa prve tablice, s time da vrijednost vanjskog ključa mora odgovarati (nekoj od) vrijednosti primarnog ključa tablice na koju se vanjski ključ referencira, ili može biti NULL“ [Rabuzin, 2011, str. 29].

Za dodavanje vanjskog ključa na alatnoj traci odabiremo *New Foreign Key* te odaberemo tablice koje želimo povezati. Najprije odabiremo tablicu roditelj čiji će se primarni ključ nalaziti u tablici dijete kao vanjski ključ, a zatim tablicu dijete. Nakon toga otvara nam se novi prozor u kojem dodatno možemo podesiti nazive i dodatna svojstva. Ukoliko je upis vanjskog ključa obavezan moramo označiti *Mandatory*.



Slika 29. Opcionalan upis vanjskog ključa

Taj postupak ponavljamo sve dok ne kreiramo sve relacije. Na slici ispod se nalazi gotov relacijski model na temelju kojeg će se dalje vršiti implementacija.



Slika 30. Relacijski model

Relacija *korisnici* ima primarni ključ *id_korisnika*. Sadrži podatke o profesorima i učenicima te ima vanjski ključ *tip_korisnika* kojim je povezana na relaciju *tip_korisnika*. Tim atributom se određuje da li je korisnik profesor ili učenik.

Relacija *predmeti* sadrži podatke o predmetima, odnosno nazivu predmeta. Primarni ključ te relacije je *id_predmeta*.

Relacija *razredi* sadrži podatke o razredima. Ona ima primarni ključ *id_razreda*, a vanjski ključ je *razrednik* kojim je povezana na relaciju *korisnici* te određuje tko je razrednik određenom razredu.

Te tri relacije su povezane relacijom *predaje* koja se sastoji od trokomponentnog primarnog ključa (*id_korisnika*, *id_razreda*, *id_predmeta*) koji su ujedno i vanjski ključevi. Ta relacija služi da bi se znalo kojem razredu profesor predaje koji predmet.

Relacija *pohada* sadrži podatke o korisnicima te predmetima koje oni pohađaju te ustvari prikazuje vezu M:N između relacija *korisnici* i *predmeti*. Korisnik pohađa više predmeta, a predmet je slušan od strane više korisnika (učenika). Relacija ima trokomponentni primarni ključ (*godina*, *id_korisnika*, *id_predmeta*) dok su atributi *id_korisnika* i *id_predmeta* i vanjski ključevi.

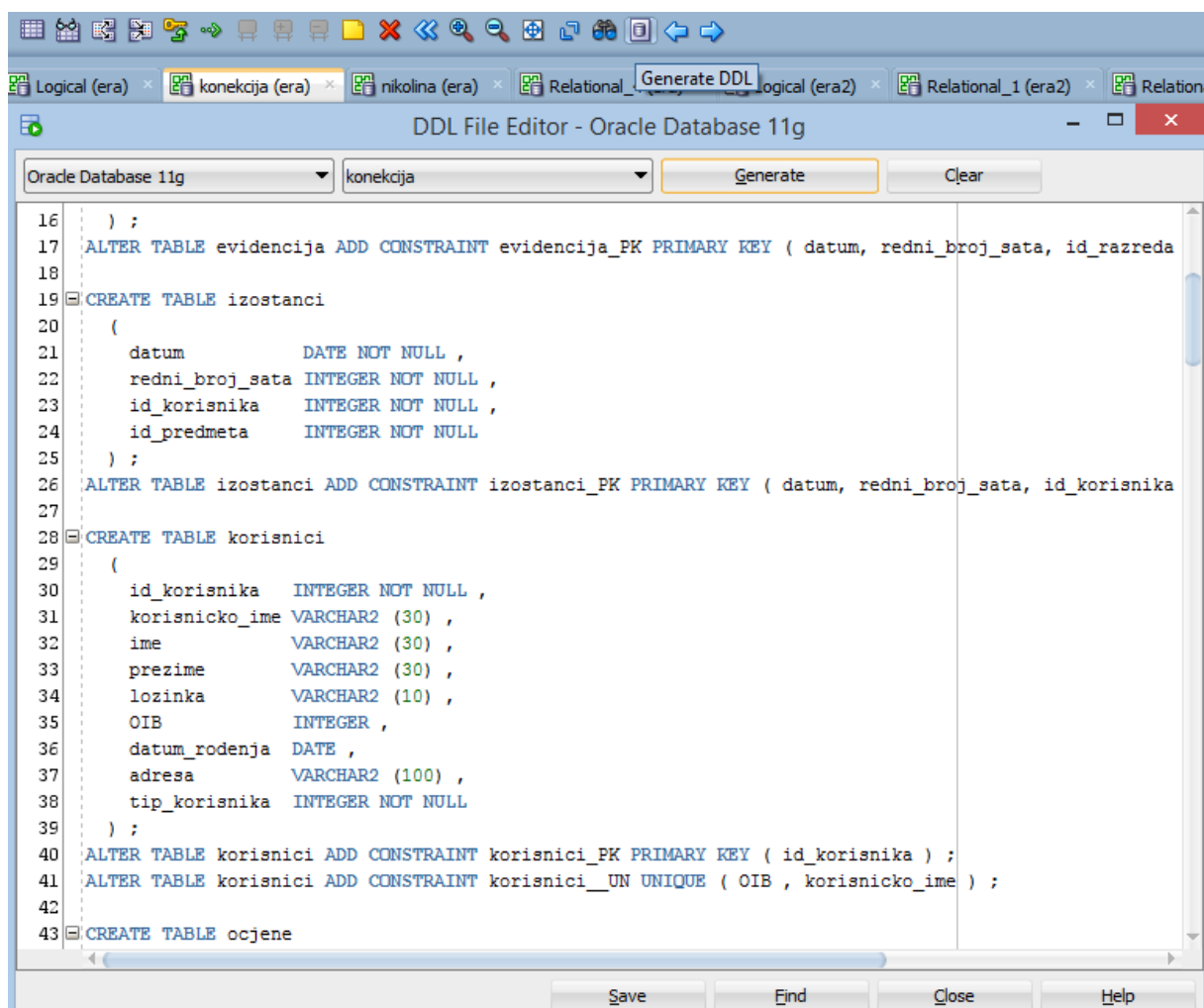
Relacija *izostanci* također predstavlja vezu M:N između relacija *korisnici* i *predmeti*. Ta relacija služi za upisivanje izostanaka učenika.

Relacija *evidencija* služi za evidentiranje razrednog sata te sadrži podatke o razredu i temi koja se obrađuje. Relacija ima primarni ključ koji se sastoji od tri atributa (*id_razreda*, *datum*, *redni_broj_sata*) od kojih je jedan i vanjski ključ (*id_razreda*). Vanjski ključ je i *id_predmeta*. On nije primarni ključ jer kombinacija primarnih ključeva mora biti jedinstvena: tako bi na primjer na isti datum, razred mogao slušati dva predmeta isti sat što nije moguće.

Relacija *ucenik_razred* sadrži podatke o učenicima i razredu kojeg taj učenik pohađa te je moguće upisati napomenu da li je učenik upisao razred prvi puta ili je ponavljač.

Relacija *ocjene* sadrži podatke o ocjenama koje pripadaju učenicima za neki predmet. Ona sadrži primarni ključ *id_ocjene* te tri vanjska ključa (*id_korisnika*, *id_predmeta*, *tip_ocjene*). Vanjskim ključem *tip_ocjene* povezana je na relaciju *tip_ocjene* i ta je veza opcionalna (na modelu označena crtikanom linijom), znači da nije obavezan unos vanjskog ključa. Tim atributom određena je vrsta ocjene, odnosno da li je učenik ocijenjen pismeno, usmeno, za aktivnost ili domaću zadaću.

Nakon gotovog modela alat nam omogućava generiranje naredbi za stvaranje objekata. Na alatnoj traci odaberemo *Generate DDL*. Zatim nam se otvori novi prozor u kojem odaberemo konekciju i zatim kliknemo na *Generate*. Generirane naredbe možemo koristiti kod implementacije tablica.

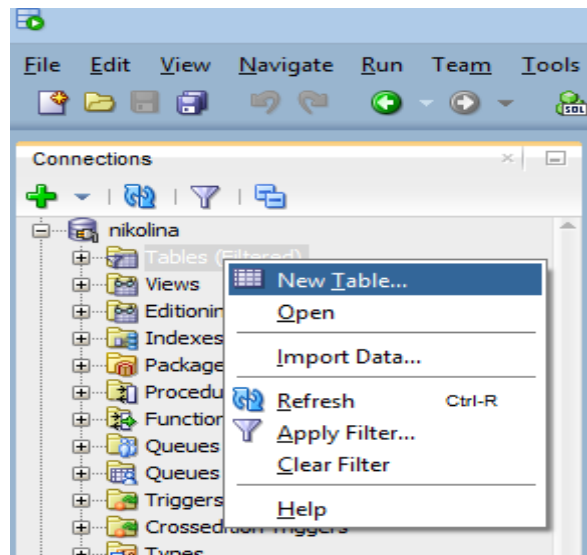


Slika 31. Generirane DDL naredbe

3.5. Implementacija tablica

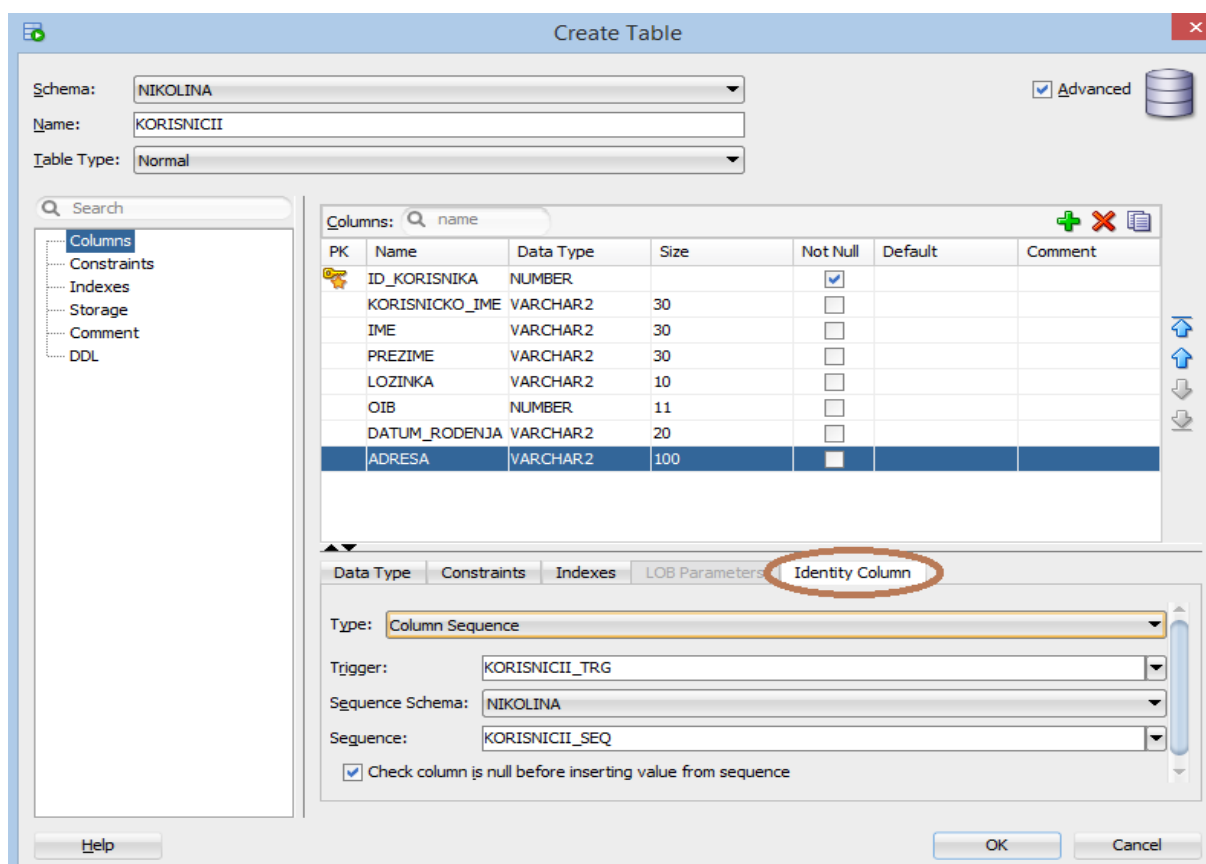
Tablice možemo implementirati putem SQL naredbi ili vizualnim putem što je prednost za osobe koje još nisu dovoljno upoznate sa samim SQL-om. Ovdje će biti prikazana oba načina implementacije.

Za kreiranje tablice pod izbornikom kreirane konekcije kliknemo na *Tables->New Table*.



Slika 32. Implementacija tablica-New Table

Zatim nam se otvara novi prozor u kojem kreiramo tablicu. Pod *Columns* dodajemo stupce, označujemo primarni ključ, itd. Zatim pod *Identity Column->Type* odaberemo *Column Sequence* kako bi se kreirala sekvenca i okidač za automatski rast vrijednosti primarnog ključa.



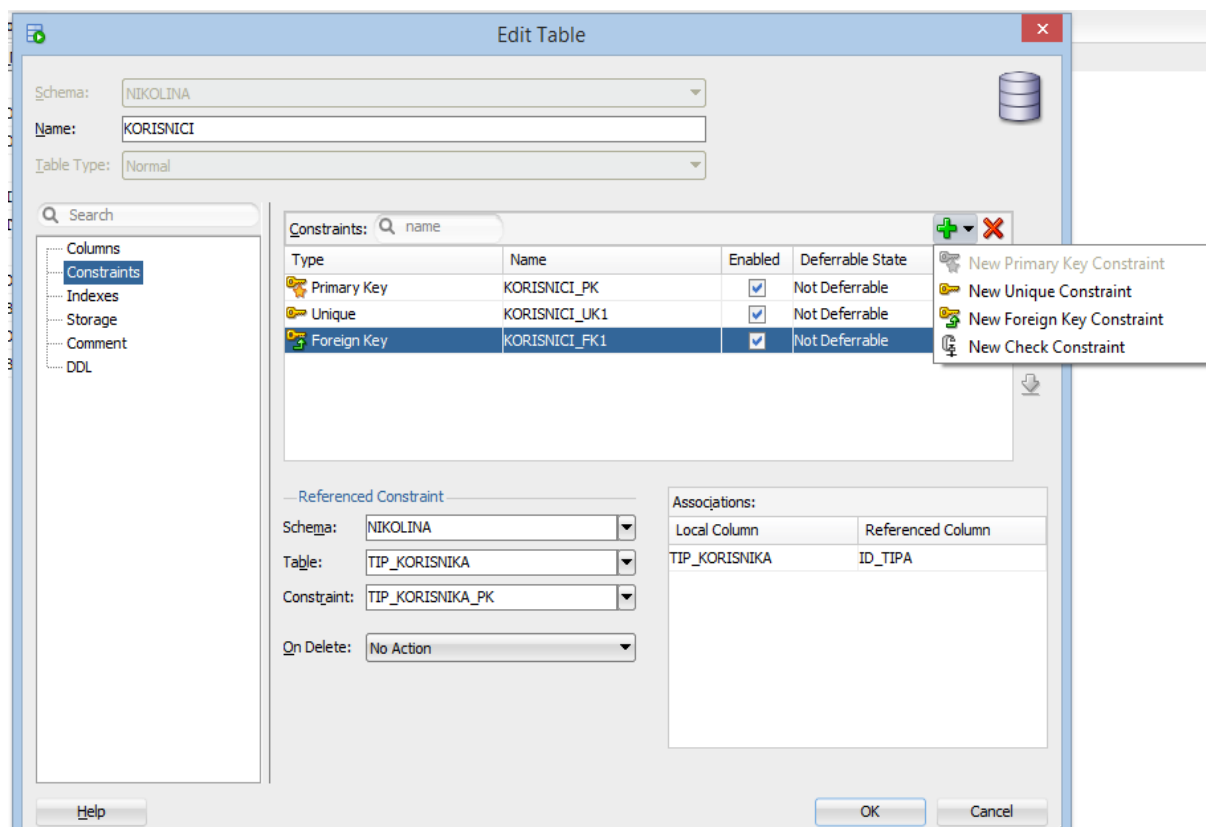
Slika 33. Implementacija tablice-Sekvence

Pod *Constraints* možemo dodati ograničenja, na primjer UNIQUE, FOREIGN KEY ili CHECK.

Pomoću CHECK definiramo uvjet koji vrijednost mora zadovoljavati kako bi bila upisana.

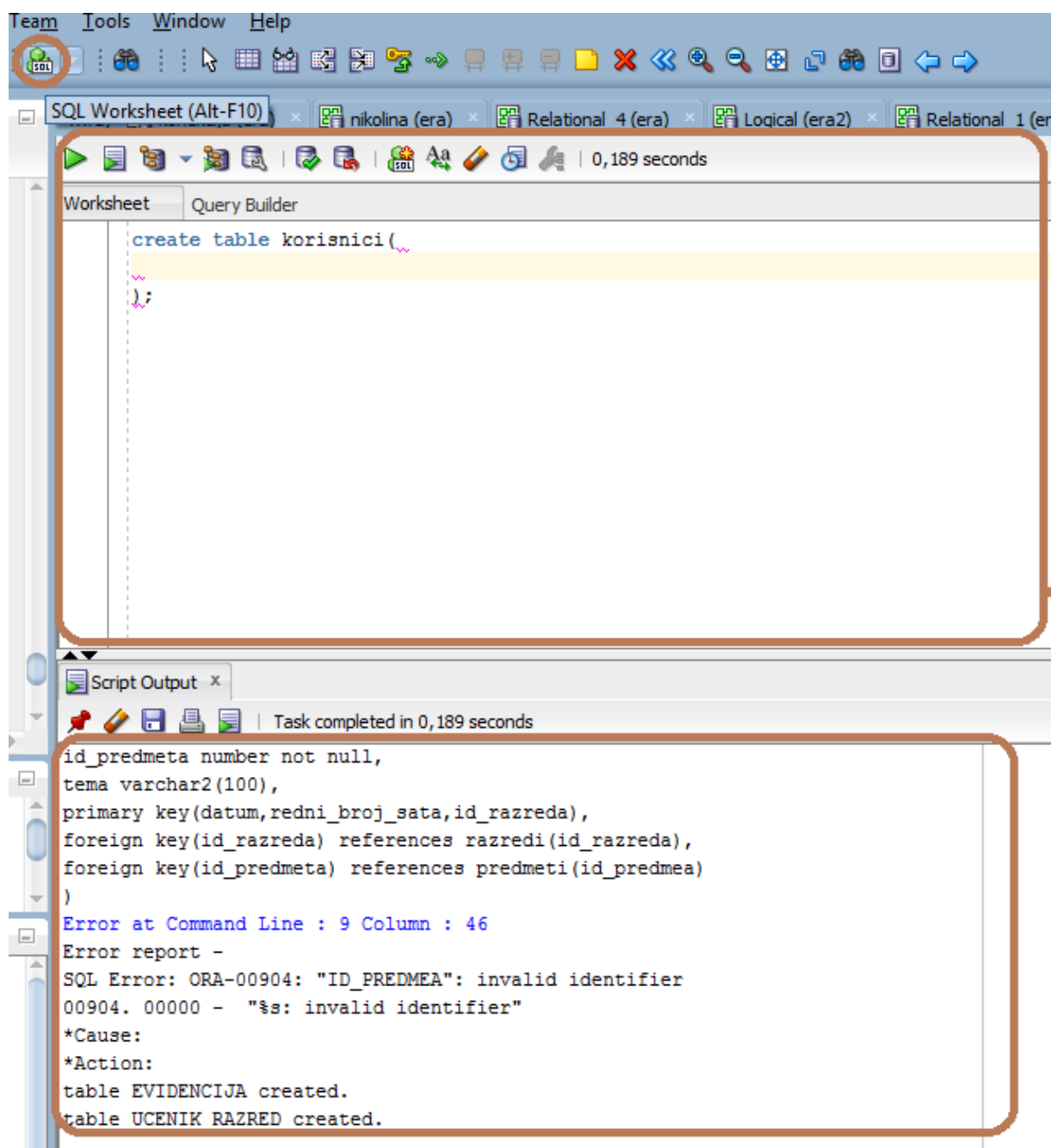
Na slici 34. je prikazano kako se dodaje vanjski ključ.

Vanjski ključ (eng. *Foreign key*) dodajemo tako da pod *Table* odaberemo tablicu sa kojom ćemo povezati trenutnu tablicu, a pod *Constraint* naziv primarnog ključa druge tablice koji će postati vanjski ključ.



Slika 34. Implementacija tablice-vanjski ključ

Tablice možemo kreirati i putem SQL naredbi. Da bi pisali SQL naredbe potrebno je kliknuti na *SQL Worksheet*. Naredbe možemo izvršavati jednu po jednu ili cijelu skriptu odjednom. Nakon izvršavanja naredbi dobijemo povratnu informaciju da li je došlo do kakvih pogrešaka ili je naredba uspješno izvršena (*Script Output*).



Slika 35. SQL Worksheet

3.5.1. CREATE TABLE

Tablicu kreiramo naredbom CREATE TABLE. U naredbi se navodi ime tablice, definira se svaki od njenih stupaca te slijede opcionalna ograničenja kojima se mogu definirati PRIMARY i FOREIGN KEY te UNIQUE i CHECK. Definicija stupca sadrži ime stupca i tip podatka. Opcijom NOT NULL ne dozvoljava se da vrijednost stupca bude NULL, odnosno da

vrijednost stupca mora biti unešena. PRIMARY KEY određuje primarni ključ te može biti sastavljen od više stupaca.

FOREIGN KEY mora sadržavati ime tablice na koju vanjski ključ pokazuje, odnosno referencira se. Definicija vanjskog ključa može opcionalno sadržavati i klauzulu ON DELETE kojom se određuje način postupanja u slučaju brisanja retka tablice na koji ukazuje vanjski ključ. Na taj se način održava integritet podataka. Njime se osiguravaju veze između zapisa u povezanim tablicama te da ne bi slučajno obrisali podatke. Ako ta klauzula nije navedena, sustav Oracle postupi prema opciji NO ACTION.

Oracle sustav podržava tri opcije:

- ON DELETE CASCADE (ako obrišemo red iz tablice roditelj, obrisat će se i potomak koji je referenciran na primarni ključ te tablice iz koje želimo obrisati podatak, npr. ukoliko želimo izbrisati korisnika jer se ispisao, pri tom ćemo izbrisati sve njegove izostanke)
- ON DELETE SET NULL (postavlja vrijednost vanjskog ključa na NULL)
- ON DELETE NO ACTION (zabranjuje promjenu primarnog ključa ako se koristi kao vanjski ključ u drugoj tablici)

U Oracle sustavu ON UPDATE je postavljen na NO ACTION. Filozofija Oracle-a je da nema smisla mijenjati vrijednost primarnog ključa. Ukoliko postoji poseban slučaj može se riješiti sa okidačima (eng. *triggers*).

Sintaksa naredbe CREATE TABLE u sustavu Oracle je:⁸

```
CREATE [GLOBAL TEMPORARY] TABLE [schema.]table (  
  column datatype [DEFAULT expr] [column constraint(s) [,...]] [,column datatype  
  [,...]] )  
[table constraint [,...]]  
[table ref constraint [,...]]...
```

Sintaksa za tablična ograničenja (eng. *table constraint*) je:

```
CONSTRAINT constrnt_name {UNIQUE|PRIMARY KEY} column constrnt_state  
  
CONSTRAINT constrnt_name CHECK(condition) constrnt_state
```

Sintaksa za referenciranje vanjskog ključa (eng. *table reference constraint*) je:

```
CONSTRAINT constrnt_name FOREIGN KEY (column,...)  
REFERENCES [schema.]table (column,...)  
[ON DELETE {CASCADE|SET NULL}] constrnt_state
```

⁸ Oracle (2014) – Oracle Database SQL Language Reference 11g Release2 (11.2)
http://docs.oracle.com/cd/E11882_01/server.112/e41084/statements_7002.htm#i2095331, preuzeto 18.kolovoza 2014.

Pogledajmo sljedeći primjer:

```
CREATE TABLE korisnici (  
  id_korisnika NUMBER NOT NULL,  
  korisnicko_ime VARCHAR2(30) NOT NULL,  
  ime VARCHAR2(30) NOT NULL,  
  prezime VARCHAR2(30) NOT NULL,  
  lozinka VARCHAR2(10) NOT NULL,  
  OIB VARCHAR2(11) NOT NULL,  
  datum_rodenja DATE NOT NULL,  
  adresa VARCHAR2(100) NOT NULL,  
  tip_korisnika NUMBER NOT NULL,  
  PRIMARY KEY (id_korisnika),  
  CONSTRAINT korisnicko_ime UNIQUE(korisnicko_ime),  
  FOREIGN KEY(tip_korisnika) REFERENCES tip_korisnika(id_tipa));
```

Oracle nema gotov tip podatka za automatski rast vrijednosti primarnog ključa. U Oracle sustavu to je malo složenije. Potrebno je samostalno kreirati sekvencu, a zatim okidač koji će vrijednosti automatski unositi kod INSERT naredbi.

Najprije kreiramo sekvencu. Sintaksa za kreiranje sekvence u sustavu Oracle je:⁹

```
CREATE SEQUENCE [ schema. ]sequence  
  [ { INCREMENT BY | START WITH } integer  
  | ...  
  | { CACHE integer | NOCACHE }  
  | { ORDER | NOORDER }  
  ...
```

Na konkretnom primjeru to bi izgledalo ovako:

```
CREATE SEQUENCE korisnici_sequence  
START WITH 1  
NOCACHE  
ORDER ;
```

START WITH označava sa kojim brojem počinje sekvenca. CACHE ili NOCACHE označava da li se vrijednosti generiraju unaprijed pa se samo dohvaćaju. ORDER nam garantira da su brojevi generirani po redoslijedu zahtjeva, važni je kod tipa podatka *timestamps*. Kod generiranja brojeva za primarne ključeve uglavnom nije potreban.

⁹ Burleson Consulting http://dba-oracle.com/t_oracle_create_sequence.htm, preuzeto 18.kolovoza 2014.

Nakon kreiranja sekvence potrebno je kreirati okidač. Sintaksa za okidač u sustavu Oracle¹⁰:

```
CREATE [ OR REPLACE ] TRIGGER [ schema. ]trigger
{ BEFORE | AFTER | INSTEAD OF }
{ dml_event_clause
| { ddl_event [ OR ddl_event ]...
| database_event [ OR database_event ]...
}
ON { [ schema. ]SCHEMA
| DATABASE
}
}
[ WHEN (condition) ]
{ pl/sql_block | call_procedure_statement } ;
```

Primjer okidača:

```
CREATE OR REPLACE TRIGGER korisnici_id_korisnika_TRG BEFORE
INSERT ON korisnici
FOR EACH ROW WHEN (NEW.id_korisnika IS NULL)
BEGIN :NEW.id_korisnika := korisnici_sequence.NEXTVAL;
END;
```

Kreirani okidač će automatski prije nego se spremi novi unos (BEFORE INSERT ON) umetati sljedeću vrijednost (NEXTVAL) iz sekvence za svaki redak (FOR EACH ROW) kada (WHEN) je, u ovom primjeru *id_korisnika* NULL. Time će se za svaki unos povećavati vrijednost primarnog ključa.

3.5.2. ALTER TABLE

ALTER TABLE naredba nam služi kako bi promijenili strukturu tablice. Tom naredbom možemo dodati stupac ili ograničenje. Sintaksa u sustavu Oracle je sljedeća:¹¹

```
ALTER TABLE [ schema. ] table
[ alter_table_properties
| column_clauses
| constraint_clauses
| alter_table_partitioning
| alter_external_table
| move_table_clause
]...
http://docs.oracle.com/cd/E11882\_01/server.112/e41084/statements\_3001.htm#CJAHHIBI
```

Sintaksa za ograničenje (eng. *constraint_clauses*) je sljedeća:

```
{ ADD { { out_of_line_constraint }...
```

¹⁰ Oracle (2014) – Oracle Database SQL Language Reference 11g Release2 (11.2) http://docs.oracle.com/cd/E11882_01/appdev.112/e17126/create_trigger.htm, preuzeto 21.kolovoza 2014.

¹¹ Oracle (2014) – Oracle Database SQL Language Reference 11g Release2 (11.2) http://docs.oracle.com/cd/E11882_01/server.112/e41084/statements_3001.htm#CJAHHIBI, preuzeto 18.kolovoza 2014.

```

        | out_of_line_REF_constraint
    }
| MODIFY { CONSTRAINT constraint_name
        | PRIMARY KEY
        | UNIQUE (column [, column ]...)
        } constraint_state [ CASCADE ]
| RENAME CONSTRAINT old_name TO new_name
| drop_constraint_clause
}

```

Na primjer, ukoliko želimo dodati ograničenje nad tablicom *korisnici*, da nam uz *korisnicko_ime* bude i *oib* UNIQUE, pišemo sljedeće:

```

ALTER TABLE korisnici
ADD CONSTRAINT oib UNIQUE(oib);

```

3.5.3. DROP TABLE

DROP TABLE naredba služi za brisanje tablice. Općenito naredbom DROP brišemo objekte.

Brisanjem tablice brišu se i svi podaci u njoj.

Sintaksa u Oracle sustavu je sljedeća:¹²

```

DROP TABLE [ schema. ] table
[ CASCADE CONSTRAINTS ] [ PURGE ] ;

```

Implementacija svih tablica prema modelu:

```

CREATE TABLE tip_korisnika (
id_tipa NUMBER NOT NULL,
uloga VARCHAR2(20) NOT NULL,
PRIMARY KEY(id_tipa)
);

CREATE SEQUENCE tip_korisnika_sequence START WITH 1 NOCACHE ;

CREATE OR REPLACE TRIGGER tip_korisnika_id_tipa_TRG BEFORE
INSERT ON tip_korisnika FOR EACH ROW WHEN (NEW.id_tipa IS NULL) BEGIN
:NEW.id_tipa := tip_korisnika_sequence.NEXTVAL;
END;

CREATE TABLE korisnici (
id_korisnika NUMBER NOT NULL,
korisnicko_ime VARCHAR2(30) NOT NULL,
ime VARCHAR2(30) NOT NULL,
prezime VARCHAR2(30) NOT NULL,
lozinka VARCHAR2(10) NOT NULL,
OIB VARCHAR2(11) NOT NULL,

```

¹² Oracle (2014) – Oracle Database SQL Language Reference 11g Release2 (11.2)
http://docs.oracle.com/cd/E11882_01/server.112/e41084/statements_9003.htm#i2061306, preuzeto 25.kolovoza 2014.


```

datum_rodenja DATE NOT NULL,
adresa VARCHAR2(100) NOT NULL,
tip_korisnika NUMBER NOT NULL,
PRIMARY KEY (id_korisnika),
CONSTRAINT oib UNIQUE (oib),
CONSTRAINT korisnicko_ime UNIQUE (korisnicko_ime),
FOREIGN KEY (tip_korisnika) REFERENCES tip_korisnika (id_tipa));

CREATE SEQUENCE korisnici_sequence START WITH 1 NOCACHE ORDER;

CREATE OR REPLACE TRIGGER korisnici_id_korisnika_TRG BEFORE
INSERT ON korisnici FOR EACH ROW WHEN (NEW.id_korisnika IS NULL) BEGIN
:NEW.id_korisnika := korisnici_sequence.NEXTVAL;
END;

CREATE TABLE predmeti (
id_predmeta NUMBER NOT NULL,
naziv_predmeta VARCHAR2(30) NOT NULL,
PRIMARY KEY (id_predmeta)
);

CREATE SEQUENCE predmeti_sequence START WITH 1 NOCACHE ORDER ;

CREATE OR REPLACE TRIGGER predmeti_id_predmeta_TRG BEFORE
INSERT ON predmeti FOR EACH ROW WHEN (NEW.id_predmeta IS NULL) BEGIN
:NEW.id_predmeta := predmeti_sequence.NEXTVAL;
END;

CREATE TABLE pohada (
godina NUMBER NOT NULL,
id_korisnika NUMBER NOT NULL,
id_predmeta NUMBER NOT NULL,
zakljucna_ocjena NUMBER,
PRIMARY KEY (godina, id_korisnika, id_predmeta),
FOREIGN KEY (id_korisnika) REFERENCES korisnici (id_korisnika),
FOREIGN KEY (id_predmeta) REFERENCES predmeti (id_predmeta)
);

CREATE TABLE tip_ocjene(
id_tipa NUMBER NOT NULL,
vrsta VARCHAR2(30) NOT NULL,
PRIMARY KEY (id_tipa)
);

CREATE SEQUENCE tip_ocjene_sequence START WITH 1 NOCACHE ORDER ;

CREATE OR REPLACE TRIGGER tip_ocjene_id_tipa_TRG BEFORE
INSERT ON tip_ocjene FOR EACH ROW WHEN (NEW.id_tipa IS NULL) BEGIN
:NEW.id_tipa := tip_ocjene_sequence.NEXTVAL;
END;

CREATE TABLE ocjene (
id_ocjene NUMBER NOT NULL,
datum DATE NOT NULL,
ocjena NUMBER,
napomena VARCHAR2(200),
tip_ocjene NUMBER,
id_korisnika NUMBER NOT NULL,
id_predmeta NUMBER NOT NULL,

```

```

PRIMARY KEY (id_ocjene),
FOREIGN KEY (tip_ocjene) REFERENCES tip_ocjene (id_tipa),
FOREIGN KEY (id_korisnika) REFERENCES korisnici (id_korisnika),
FOREIGN KEY (id_predmeta) REFERENCES predmeti (id_predmeta)
);

CREATE SEQUENCE ocjene_sequence START WITH 1 NOCACHE ORDER ;

CREATE OR REPLACE TRIGGER ocjene_id_ocjene_TRG BEFORE
INSERT ON ocjene FOR EACH ROW WHEN (NEW.id_ocjene IS NULL) BEGIN
:NEW.id_ocjene := ocjene_sequence.NEXTVAL;
END;

CREATE TABLE izostanci (
datum DATE NOT NULL,
redni_broj_sata NUMBER NOT NULL,
id_korisnika NUMBER NOT NULL,
id_predmeta NUMBER NOT NULL,
PRIMARY KEY (datum, redni_broj_sata, id_korisnika),
FOREIGN KEY (id_korisnika) REFERENCES korisnici (id_korisnika),
FOREIGN KEY (id_predmeta) REFERENCES predmeti (id_predmeta)
);

CREATE TABLE razredi(
id_razreda NUMBER NOT NULL,
naziv_razreda VARCHAR2(10) NOT NULL,
godina_razreda NUMBER NOT NULL,
razrednik NUMBER NOT NULL,
PRIMARY KEY (id_razreda),
FOREIGN KEY (razrednik) REFERENCES korisnici (id_korisnika)
);

CREATE SEQUENCE razredi_sequence START WITH 1 NOCACHE ORDER ;

CREATE OR REPLACE TRIGGER razredi_id_razreda_TRG BEFORE
INSERT ON razredi FOR EACH ROW WHEN (NEW.id_razreda IS NULL) BEGIN
:NEW.id_razreda := razredi_sequence.NEXTVAL;
END;

CREATE TABLE predaje(
id_predmeta NUMBER NOT NULL,
id_razreda NUMBER NOT NULL,
id_korisnika NUMBER NOT NULL,
napomena VARCHAR2(30),
PRIMARY KEY (id_predmeta, id_razreda, id_korisnika),
FOREIGN KEY (id_predmeta) REFERENCES predmeti (id_predmeta),
FOREIGN KEY (id_razreda) REFERENCES razredi (id_razreda),
FOREIGN KEY (id_korisnika) REFERENCES korisnici (id_korisnika)
);

CREATE TABLE evidencija (
datum DATE NOT NULL,
redni_broj_sata NUMBER NOT NULL,
id_razreda NUMBER NOT NULL,
id_predmeta NUMBER NOT NULL,
tema VARCHAR2(100),
PRIMARY KEY (datum, redni_broj_sata, id_razreda),
FOREIGN KEY (id_razreda) REFERENCES razredi (id_razreda),
FOREIGN KEY (id_predmeta) REFERENCES predmeti (id_predmeta)
);

```

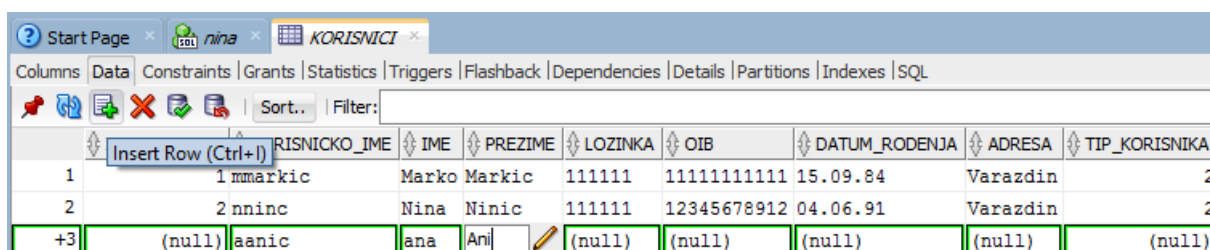
```
CREATE TABLE ucenik_razred(
id_korisnika NUMBER NOT NULL,
id_razreda NUMBER NOT NULL,
napomena VARCHAR2(20),
PRIMARY KEY (id_korisnika, id_razreda),
FOREIGN KEY (id_korisnika) REFERENCES korisnici (id_korisnika),
FOREIGN KEY (id_razreda) REFERENCES razredi (id_razreda)
);
```

4. Rad s podacima i upiti

4.1. Naredba INSERT

Nakon kreiranja tablica, one su prazne te ih je potrebno popuniti podacima. Podatke je moguće unositi, mijenjati i brisati na dva načina. Prvi način je direktnim unošenjem u tablicu, a drugi je pisanjem SQL naredbi i upita u SQL editoru gdje se upotrebljavaju standardne SQL naredbe za rad s podacima.

Sljedeća slika prikazuje unos podataka direktnim unošenjem u tablicu *korisnici*.



	RISNICKO_IME	IME	PREZIME	LOZINKA	OIB	DATUM_RODENJA	ADRESA	TIP_KORISNIKA
1	1 mmarkic	Marko	Markic	111111	11111111111	15.09.84	Varazdin	2
2	2 nninc	Nina	Ninic	111111	12345678912	04.06.91	Varazdin	2
+3	(null)	aanic	ana	Ani	(null)	(null)	(null)	(null)

Slika 36. Unos podataka

Drugi način unosa podataka je pisanje SQL naredbi. SQL naredba za unos podataka je INSERT. Skraćena sintaksa naredbe u sustavu Oracle:¹³

```
INSERT [ hint ]
{ single_table_insert | multi_table_insert } ;
```

U Oracle sustavu postoje dvije vrste INSERT naredbi, jedna za umetanje podataka u jednu (eng. *Single-table insert*), a druga za umetanje podataka u više tablica (eng. *Multi-table insert*).

¹³ Oracle (2014) – Oracle Database SQL Language Reference 11g Release2 (11.2) http://docs.oracle.com/cd/E18283_01/server.112/e17118/statements_9014.htm#i2111652, preuzeto 28.kolovoza 2014.

Primjer za unos jednog retka dan je u nastavku:

```
INSERT INTO predmeti VALUES (predmeti_sequence.NEXTVAL, 'OS2');
```

Ukoliko nemamo definiranih stupaca koje želimo unositi, moramo unijeti sve vrijednosti stupaca određene tablice. *predmeti_sequence.NEXTVAL* upisujemo za vrijednost *id_predmeta* kako bi se vrijednost primarnog ključa automatski povećala prethodno kreiranom sekvencom.

Ukoliko ne želimo unositi sve vrijednosti u tablici možemo navesti stupac ili više njih za koji želimo obaviti unos vrijednosti.

```
INSERT INTO predmeti (naziv_predmeta) VALUES ('matematika');
```

Umetanje se može izvršiti i uz korištenje podupita.

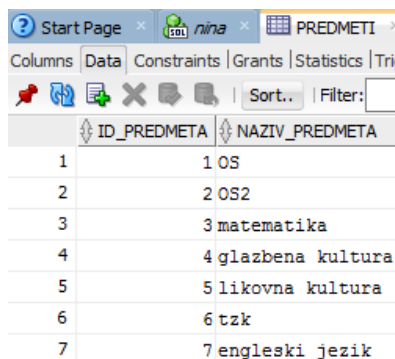
```
INSERT INTO  
  (SELECT naziv_predmeta FROM predmeti)  
VALUES ('glazbena kultura');
```

Naredbom možemo upisati i više redaka:

```
INSERT ALL  
  INTO predmeti (naziv_predmeta) VALUES ('likovna kultura')  
  INTO predmeti (naziv_predmeta) VALUES ('tzk')  
  INTO predmeti (naziv_predmeta) VALUES ('engleski jezik')  
SELECT * FROM dual;
```

DUAL je tablica koja je automatski kreirana od strane Oracle-a. DUAL je tablica koja pripada SYS korisniku, no dostupna je svim korisnicima. Sastoji se od jednog stupca DUMMY koji sadrži jedan redak sa vrijednosti X. Kod unosa više podataka, vrijednost se vraća onoliko puta koliko redaka želimo upisati.

Nakon izvršavanja gore prikazanih INSERT naredbi dobijemo popunjenu tablicu *predmeti*:



ID_PREDMETA	NAZIV_PREDMETA
1	OS
2	OS2
3	matematika
4	glazbena kultura
5	likovna kultura
6	tzk
7	engleski jezik

Slika 37. Unos podataka-predmeti

Treba voditi računa i o tipovima podataka, ograničenjima stupaca te na mjestu gdje nije definiramo NOT NULL, upisati NULL vrijednost.

Na primjer:

```
INSERT INTO pohada VALUES (1, 2, NULL);
```

4.2. Klauzule SELECT, FROM, WHERE

Za stvaranje upita i dohvaćanje podataka koristimo klauzulu SELECT. Sa klauzulom SELECT odabiremo koja ćemo polja dohvaćati iz određenih tablica koje određujemo sa klauzulom FROM.

Sintaksa u sustavu Oracle je sljedeća:¹⁴

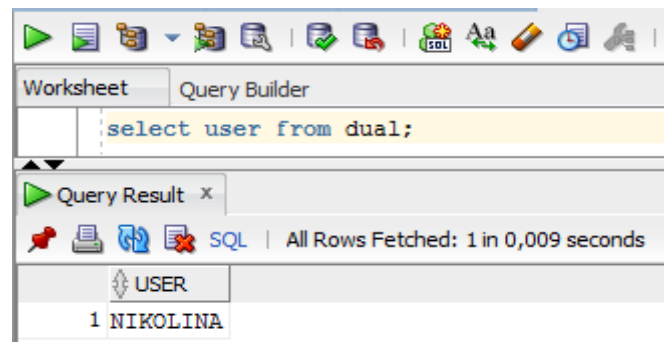
```
SELECT [DISTINCT | ALL] {* | select_list}
FROM {table_name [alias] | view_name}
[WHERE condition]
[GROUP BY condition_list]
[HAVING condition]
[ORDER BY {column_name | column_# [ASC | DESC]}]...
```

U Oracle sustavu za dohvaćanje podataka obavezne su klauzule SELECT i FROM. Ostale klauzule poput WHERE, GROUP BY, HAVING i ORDER BY su opcionalne.

Uz klauzulu SELECT se navodi ime stupca, ili naziv funkcije, ovisno o tome što želimo dobiti kao rezultat upita. Oracle sustav ima dosta gotovih funkcija, a popis svih možemo pronaći na njihovoj stranici.

Na slici je prikazan upit koji vraća ime korisnika uz gotovu funkciju *user*.

¹⁴ Tutorialspoint http://www.tutorialspoint.com/sql_certificate/the_sql_select_statement.htm, preuzeto 28.kolovoza 2014



Slika 38. Upit

Ako želimo dohvatiti sve podatke iz tablice koristimo sljedeći upit:

```
SELECT * FROM korisnici;
```

Ukoliko želimo dohvatiti samo neke podatke iz kreiranih tablica uz SELECT moramo navesti imena stupaca koje želimo dobiti kao rezultat.

```
SELECT korisnicko_ime, ime, prezime
FROM korisnici;
```

	KORISNICKO_IME	IME	PREZIME
1	zzimic	Zana	Zimic
2	mmarkic	Marko	Markic
3	nninc	ina	Ninic

Slika 39. Dohvaćanje podataka SELECT

Klauzula WHERE je opcionalna i služi za postavljanje uvjeta. Kako ne bi morali trošiti vrijeme na traženje potrebnih podataka pomaže nam klauzula WHERE.

Sljedeći upit vraća sve korisnike koji žive u Novom Marofu:

```
SELECT ime, prezime, adresa
FROM korisnici
WHERE adresa = 'Novi Marof';
```

	IME	PREZIME	ADRESA
1	Marko	Markic	Novi Marof
2	Matija	Matic	Novi Marof
3	Lovro	Lovric	Novi Marof
4	Tihomir	Tiho	Novi Marof

Slika 40. Klauzula WHERE

WHERE klauzula dolazi odmah nakon FROM klauzule. SELECT klauzula smije sadržavati samo jednu WHERE klauzulu. Za postavljanje više uvjeta koriste se operatori AND i OR.

Kada bi željeli dohvatiti korisnike koji žive u Novom Marofu ili Varaždinu upit bi bio sljedeći:

```
SELECT ime, prezime, adresa
FROM korisnici
WHERE adresa = 'Novi Marof' OR adresa = 'Varazdin';
```

U WHERE klauzuli se često koriste operatori uspoređivanja. Oracle ima devet operatora uspoređivanja:

operator	značenje
=	jednako
<	je manje
>	je veće
>=	je veće ili jednako
<=	je manje ili jednako
!=	je različito
< >	je različito
!>	nije veće od
!<	nije manje od

Tabela 3. Operatori uspoređivanja

Prema[http://www.tutorialspoint.com/sql_certificate/restricting_and_sorting_data.htm]

Ostali operatori sustava Oracle su BETWEEN..AND, IN, LIKE, IS NULL.

4.2.1. Operatori BETWEEN..AND, IN, LIKE, IS NULL

„Operator BETWEEN omogućuje definiranje raspona kojega vrijednosti moraju zadovoljavati“ [Rabuzin, 2011, str. 80].

Taj se operator koristi u klauzuli WHERE, a koristimo ga kako bi definirali uvjet za vrijednost koju tražimo. U uvjetu navodimo donju i gornju granicu, a između njih stavljamo AND. U rezultatu su uključene i gornja i donja granica.

Ako želimo vidjeti sve izostanke unutar datuma upit izgleda ovako:

```
SELECT datum, id_korisnika  
FROM izostanci  
WHERE datum BETWEEN '13-09-13' AND '15-09-13';
```

	DATUM	ID_KORISNIKA
1	13.09.13	11
2	13.09.13	11
3	13.09.13	11
4	14.09.13	11

Slika 41. Operator BETWEEN

Operator IN nam omogućava da provjerimo nalazi li se vrijednost nekog stupca u skupu vrijednosti koje su definirane uvjetom.

```
SELECT ime, prezime, adresa  
FROM korisnici  
WHERE adresa IN ('Novi Marof', 'Varazdin');
```

Ovaj upit vraća isti rezultat kao upit koji smo ranije kreirali sa operatorom OR. Umjesto IN možemo i koristiti NOT IN, ali tada ćemo dobiti rezultat koji ne uključuje vrijednosti koje se nalaze u uvjetu.

Operator LIKE koristimo kada želimo pronaći neku vrijednost koja u sebi sadrži određeni znak ili niz znakova. Uz operator LIKE moramo koristiti ('%') da bi zamijenili više znakova ili ('_') koji zamjenjuje jedan znak.

Da bi dohvatili sva imena korisnika koja započinju sa "Ma" kreiramo sljedeći upit:

```
SELECT ime  
FROM korisnici  
WHERE ime LIKE 'Ma%';
```


	IME
1	Marko
2	Martina
3	Maja
4	Matija
5	Marica
6	Margareta

Slika 42. Operator LIKE

Za dohvaćanje korisnika kojima se na trećem mjestu imena nalazi slovo "a" kreiramo upit:

```
SELECT ime
FROM korisnici
WHERE ime LIKE '__a%';
```

	IME
1	Ana
2	Ivana
3	Franjo
4	Ena

Slika 43. Operator LIKE 2

Operator IS NULL služi da bi mogli provjeriti u kojim poljima nisu upisane nikakve vrijednosti. Također postoji i operator IS NOT NULL kojim dobivamo ispis redaka koji imaju vrijednost za traženi stupac.

Ako želimo vidjeti koje ocjene imaju komentar kreiramo sljedeći upit:

```
SELECT id_korisnika, ocjena, napomena
FROM ocjene
WHERE napomena IS NOT NULL;
```

	ID_KORISNIKA	OCJENA	NAPOMENA
1	11	2	dovoljno razumijevanje
2	13	4	vrlo dobro vladanje

Slika 44. Operator IS (NOT) NULL

4.3. Agregirajuće funkcije

Agregirajuće funkcije su funkcije koje vraćaju jednu vrijednost na temelju skupa vrijednosti. Mogu se koristiti uz klauzule GROUP BY i HAVING.

Neke od agregirajućih funkcija su:

- AVG() koja vraća prosječnu vrijednost u stupcu
- COUNT() koja broji redove
- SUM() koja vraća sumu vrijednosti

4.4. Klauzule GROUP BY, HAVING, ORDER BY

Klauzula GROUP BY dolazi opcionalno nakon klauzule WHERE. Koristi se za grupiranje podataka, a može se koristiti nad stupcima (jednim ili više). Podaci se grupiraju u logičke grupe. Kada grupiramo podatke možemo koristiti agregirajuće funkcije.

Korištenje te klauzule prikazano je primjerom:

```
SELECT adresa, count(*)  
FROM korisnici  
GROUP by adresa;
```

	ADRESA	COUNT(*)
1	Novi Marof	4
2	Cakovec	5
3	Ludbreg	4
4	Varazdin	12

Slika 45. Klauzula HAVING

Ukoliko ne trebamo koristiti agregirajuće funkcije preporuča se koristiti opciju DISTINCT jer ćemo dobiti iste rezultate.

```
SELECT DISTINCT adresa  
FROM korisnici;
```

	ADRESA
1	Novi Marof
2	Cakovec
3	Ludbreg
4	Varazdin

Slika 46. Opcija DISTINCT

Nakon grupiranja podataka možemo koristiti klauzulu HAVING. Ta se klauzula koristi kako bi izbacili određene rezultate, odnosno grupe. Grupa će biti izbačena ukoliko ne zadovoljava uvjet uz klauzulu HAVING.

Korištenje klauzule HAVING prikazano je primjerom:

```
SELECT adresa, count(*)
FROM korisnici
GROUP BY adresa
HAVING count(*) > 4;
```

	ADRESA	COUNT(*)
1	Cakovec	5
2	Varazdin	12

Slika 47. Klauzula HAVING

Klauzula ORDER BY se koristi ukoliko dobivene podatke želimo poredati po vrijednosti određenog stupca. Sortiranje se može izvoditi i prema vrijednosti više stupaca. ORDER BY sortira stupce po rastućim vrijednostima (ASC). Ukoliko želimo da vrijednosti budu padajuće moramo uz ORDER BY (naziv stupca) dodati i DESC.

Sortiranje možemo obaviti i tako da uz ORDER BY stavimo broj na kojem se mjestu nalazi vrijednost stupca u klauzuli SELECT prema kojem želimo sortirati.

Klauzula ORDER BY prikazana je primjerom:

```
SELECT ime, prezime
FROM korisnici
ORDER BY 2;
```

	IME	PREZIME
1	Ana	Anic
2	Anita	Anit
3	Ante	Antic
4	Ena	Enic
5	Fabijan	Fabo
6	Franjo	Franjic
7	Margareta	Gretic
8	Ivana	Ivanic
9	Ivo	Ivic
10	Jakov	Jakiv

Slika 48. Klauzula ORDER BY

4.5. DESCRIBE opcija

DESCRIBE opcija služi da bi provjerili strukturu tablice. Tom opcijom dobivamo informaciju od kojih se stupaca sastoji tablica, njihov tip podataka te da li podaci moraju biti upisani u određeni stupac (NULL/NOT NULL).

Primjer korištenja te opcije:

```
DESC razredi;
```

Name	Null	Type
ID_RAZREDA	NOT NULL	NUMBER
NAZIV_RAZREDA	NOT NULL	VARCHAR2(10)
GODINA_RAZREDA	NOT NULL	NUMBER
RAZREDNIK	NOT NULL	NUMBER

Slika 49. Opcija DESCRIBE

4.6. Naredba UPDATE

Za ažuriranje podataka koristimo naredbu UPDATE. Nekad se može desiti da zaboravimo unijeti neke vrijednosti ili jednostavno nismo mogli znati njihovu vrijednost. Zato nam naredba UPDATE omogućava da naknadno ažuriramo vrijednost.

Sintaksa naredbe UPDATE u sustavu Oracle je:¹⁵

```
UPDATE [ hint ]
  { dml_table_expression_clause
  | ONLY (dml_table_expression_clause)
  } [ t_alias ]
update_set_clause
[ where_clause ]
[ returning_clause ]
[error_logging_clause] ;
```

Nakon riječi UPDATE potrebno je navesti ime tablice u kojoj želimo promijeniti neku vrijednost. Zatim je važna riječ SET bez koje ne možemo postaviti vrijednost. Uz SET se

¹⁵Oracle (2014) – Oracle Database SQL Language Reference 11g Release2 (11.2)
http://docs.oracle.com/cd/E11882_01/server.112/e26088/statements_10008.htm#BABGGJCE, preuzeto 29.kolovoza 2014.

navodi ime stupca kojem dodjeljujemo vrijednost. Pomoću naredbe UPDATE možemo ažurirati jedino one retke za koje je zadovoljen uvjet te se zato koristi klauzula WHERE.

Najprije pogledajmo tablicu koju ćemo ažurirati:

```
SELECT * FROM korisnici;
```

ID_KORIS...	KORISNICKO_IME	IME	PREZIME	LOZINKA	OIB	DATUM_RODENJA	ADRESA	TIP_KORISNIKA
1	1mmarkic	Marko	Markic	111111	111111111111	15.09.84	Varazdin	2
2	2nninc	Nina	Ninic	111111	12345678912	04.06.91	Varazdin	2
3	3zzimic	Zana	Zimic	1111111	12345612345	04.05.92	Varazdin	2
4	4Ana	AAnic	Ana	123456	15478541233	01.08.95	Ludbreg	1
5	5mmia	Mia	Mic	154788	45454521456	15.11.82	Ludbreg	1

Slika 50. Ažuriranje podataka1

Ukoliko je korisnik zaboravio svoju lozinku te je dobio novu, to se može realizirati sljedećim upitom:

```
UPDATE korisnici
SET lozinka = '123456'
WHERE id_korisnika = 2;
```

ID_KORIS...	KORISNICKO_IME	IME	PREZIME	LOZINKA	OIB	DATUM_RODENJA	ADRESA	TIP_KORISNIKA
1	1mmarkic	Marko	Markic	111111	111111111111	15.09.84	Varazdin	2
2	2nninc	Nina	Ninic	123456	12345678912	04.06.91	Varazdin	2
3	3zzimic	Zana	Zimic	1111111	12345612345	04.05.92	Varazdin	2
4	4Ana	AAnic	Ana	123456	15478541233	01.08.95	Ludbreg	1
5	5mmia	Mia	Mic	154788	45454521456	15.11.82	Ludbreg	1

Slika 51. Ažuriranje podataka2

4.7. Naredba DELETE

U bazi podataka su spremljene velike količine podataka. Podatke koje više ne trebamo možemo obrisati. Za to nam služi naredba DELETE. Tom naredbom brišemo redove. Koliko redova će biti obrisano ovisi o klauzuli WHERE kojom se definira uvjet.

Sintaksa naredbe DELETE u sustavu Oracle je:¹⁶

```
DELETE [ hint ]
[ FROM ]
```

¹⁶ Oracle (2014) – Oracle Database SQL Language Reference 11g Release2 (11.2) http://docs.oracle.com/cd/E11882_01/server.112/e26088/statements_8005.htm#i2112830, preuzeto 29.kolovoza 2014.

```

{ dml_table_expression_clause
| ONLY (dml_table_expression_clause)
} [ t_alias ]
[ where_clause ]
[ returning_clause ]
[error_logging_clause];

```

Nakon riječi DELETE obavezno dolazi klauzula FROM kojom definiramo tablicu iz koje želimo obrisati podatke. Zatim postavljamo uvjet sa klauzulom WHERE, odnosno definiramo točno koje podatke želimo obrisati.

Za primjer ćemo izbrisati jedan razred. Najprije ćemo pogledati popis razreda.

```
SELECT * FROM razredi;
```

	ID_RAZREDA	NAZIV_RAZREDA	GODINA_RAZREDA	RAZREDNIK
1	1	3a	2014	1
2	2	3b	2014	2
3	3	4gim	2014	3
4	4	4a	2014	4
5	5	4b	2014	5
6	6	3c	2014	9

Slika 52. Naredba DELETE 1

Ako želimo izbrisati 3c razred, naredba je sljedeća:

```
DELETE FROM razredi
WHERE id_razreda = 6;
```

	ID_RAZREDA	NAZIV_RAZREDA	GODINA_RAZREDA	RAZREDNIK
1	1	3a	2014	1
2	2	3b	2014	2
3	3	4gim	2014	3
4	4	4a	2014	4
5	5	4b	2014	5

Slika 53. Rezultat naredbe DELETE

4.8. Spajanje tablica

Najčešće su nam potrebni podaci iz dvije ili više tablica. Zato moramo znati spojiti tablice. U jednom upitu je potrebno spojiti podatke iz više tablica. Takvi se upiti zovu složeni upiti.

U nastavku su opisani različiti načini spajanja tablica:

- *Spajanje sa FROM i WHERE*
- *[INNER] JOIN*
- *[OUTER] JOIN*
 - *LEFT [OUTER] JOIN*
 - *RIGHT [OUTER] JOIN*
 - *FULL [OUTER] JOIN*

4.8.1. Spajanje tablica klauzulama FROM i WHERE

Jedan od mogućih načina spajanja tablica je korištenjem klauzula FROM i WHERE. Uz SELECT navodimo nazive stupaca iz tablica za koje želimo da nam se prikažu u rezultatu. U klauzuli FROM trebamo navesti nazive tablica iz kojih dohvaćamo podatke.

U klauzuli WHERE treba navesti uvjet na temelju kojega će tablice biti spojene. Tablice se spajaju na temelju vrijednosti primarnih i vanjskih ključeva te je u klauzuli WHERE potrebno izjednačiti njihove vrijednosti.

Sljedećim primjerom dohvaćamo ime i prezime korisnika iz tablice *korisnici* te naziv njihove uloge iz tablice *tip_korisnika*. Nazive stupaca i tablica odvajamo zarezom.

```
SELECT ime, prezime, uloga
FROM korisnici, tip_korisnika
WHERE korisnici.tip_korisnika = tip_korisnika.id_tipa;
```

	IME	PREZIME	ULOGA
1	Verica	Veric	profesor
2	Nina	Nikic	profesor
3	Marko	Markic	profesor
4	Ana	Anic	profesor
5	Krunoslav	Krunic	profesor
6	Maja	Majoc	profesor
7	Ivo	Ivic	profesor
8	Ante	Antic	profesor
9	Martina	Martic	profesor
10	Pero	Peric	profesor
11	Franjo	Franjic	ucenik
12	Marica	Marinc	ucenik
13	Mirko	Mirkic	ucenik
14	Margareta	Gretic	ucenik
15	Bernarda	Nadic	ucenik
16	Tihomir	Tiho	ucenik

Slika 54. Spajanje klauzulama FROM i WHERE

4.8.2. [INNER] JOIN

Klauzula INNER JOIN zamjenjuje spajanje tablica pomoću klauzula FROM i WHERE. Upravo čini preglednijima te olakšava pisanje složenih upita. Riječ INNER je opcionalna te se uglavnom ne koristi.

Ako želimo saznati koji učenici su izostali na određeni datum te koliko sati, potrebni su nam podaci iz tablica *korisnici* i *izostanci*. Slijedi upit koji vraća traženi rezultat korištenjem klauzule [INNER] JOIN.

```
SELECT korisnici.ime, korisnici.prezime, count(izostanci.id_korisnika) as
broj_izostanaka, izostanci.datum
FROM korisnici
JOIN izostanci ON korisnici.id_korisnika = izostanci.id_korisnika
WHERE izostanci.datum = '13-09-13'
GROUP BY izostanci.datum, korisnici.ime, korisnici.prezime;
```

	IME	PREZIME	BROJ_IZOSTANAKA	DATUM
1	Jakov	Jakiv	3	13.09.13

Slika 55. [INNER] JOIN

Kod korištenja JOIN klauzule u klauzuli FROM se prvo navodi naziv prve tablice te se sa klauzulom JOIN dodaje naziv druge tablice. Dio koji se odnosi na spajanje tablica nalazi se u ON dijelu klauzule JOIN. U WHERE klauzuli se definira uvjet. Na kraju možemo i grupirati dobivene podatke po željenom stupcu.

4.8.3. LEFT [OUTER] JOIN

LEFT [OUTER] JOIN je klauzula koja se koristi za vanjsko spajanje tablica. I sam naziv nam govori da želimo spojiti nešto sa lijeve strane. Točnije rečeno, spajanjem dvije tablice jedna se u upitu nalazi sa lijeve, a druga s desne strane. Ukoliko želimo ispisati neke podatke iz lijeve tablice bez obzira ima li pripadajuću vrijednost u desnoj, koristimo klauzulu LEFT [OUTER] JOIN. OUTER je opcionalan te ga nije potrebno navoditi.

Na primjer, imamo situaciju da se u školu upisala nova učenica te još uvijek nema dodijeljen razred. Za taj upit su nam potrebne tri tablice: *korisnici*, *razred_ucenik* i *razredi*. Upit izgleda ovako:

```
SELECT korisnici.ime, korisnici.prezime, razredi.naziv_razreda
FROM korisnici
LEFT JOIN ucenik_razred
ON korisnici.id_korisnika = ucenik_razred.id_korisnika
LEFT JOIN razredi
ON ucenik_razred.id_razreda = razredi.id_razreda;
```

	IME	PREZIME	NAZIV_RAZREDA
1	Anita	Anit	3a
2	Fabijan	Fabo	3a
3	Julija	Julic	3a
4	Lovro	Lovric	3a
5	Matija	Matic	3a
6	Ivana	Ivanic	3a
7	Monika	Monik	3a
8	Jakov	Jakiv	3a
9	Ena	Enic	3b
10	Margareta	Gretic	3b
11	Mirko	Mirkic	3b
12	Marica	Marinc	3b
13	Franjo	Franjic	3b
14	Bernarda	Nadic	3b
15	Tihomir	Tiho	3b
16	Jana	Janic	(null)

Slika 56. LEFT JOIN

Možemo vidjeti da su dohvaćeni svi korisnici koji pripadaju i ne pripadaju nekom razredu te se na mjestu naziva razreda pojavila NULL vrijednost jer učenici još nije dodijeljen razred.

4.8.4. RIGHT [OUTER] JOIN

RIGHT [OUTER] JOIN klauzula je suprotna klauzuli LEFT [OUTER] JOIN. Točnije, spajanjem dvije tablice jedna se u upitu nalazi sa lijeve, a druga s desne strane. Ukoliko želimo ispisati neke podatke iz desne tablice bez obzira ima li pripadajuću vrijednost u lijevoj, koristimo klauzulu RIGHT [OUTER] JOIN. OUTER je opcionalan te ga nije potrebno navoditi.

Kao primjer ćemo ispisati sve razrede koji nemaju upisane učenike. Upit je sljedeći:

```
SELECT korisnici.ime, korisnici.prezime, razredi.naziv_razreda
FROM korisnici
RIGHT JOIN ucenik_razred
ON korisnici.id_korisnika = ucenik_razred.id_korisnika
RIGHT JOIN razredi
ON ucenik_razred.id_razreda = razredi.id_razreda;
```

	IME	PREZIME	NAZIV_RAZREDA
3	Anita	Anit	3a
4	Fabijan	Fabo	3a
5	Julija	Julic	3a
6	Lovro	Lovric	3a
7	Matija	Matic	3a
8	Ivana	Ivanic	3a
9	Margareta	Gretic	3b
10	Mirko	Mirkic	3b
11	Marica	Marinc	3b
12	Franjo	Franjic	3b
13	Bernarda	Nadic	3b
14	Tihomir	Tiho	3b
15	Ena	Enic	3b
16	(null)	(null)	4gim
17	(null)	(null)	4a
18	(null)	(null)	4b

Slika 57. RIGHT JOIN

Možemo vidjeti da su dohvaćeni svi podaci iz tablice *razredi* bez obzira imaju li upisane učenike koji se nalaze u tablici *korisnici*.

4.8.5. FULL [OUTER] JOIN

Da bi ispisali sve korisnike bez obzira pripadaju li kojem razredu i sve razrede bez obzira imaju li upisane učenike koristit ćemo klauzulu FULL [OUTER] JOIN.

Upit je sljedeći:

```
SELECT korisnici.ime, korisnici.prezime, razredi.naziv_razreda
FROM korisnici
FULL JOIN ucenik_razred
ON korisnici.id_korisnika = ucenik_razred.id_korisnika
FULL JOIN razredi
ON ucenik_razred.id_razreda = razredi.id_razreda;
```

	IME	PREZIME	NAZIV_RAZREDA
10	Maja	Majoc	(null)
11	Pero	Peric	(null)
12	Jakov	Jakiv	3a
13	Monika	Monik	3a
14	Ivana	Ivanic	3a
15	Matija	Matic	3a
16	Lovro	Lovric	3a
17	Julija	Julic	3a
18	Fabijan	Fabo	3a
19	Anita	Anit	3a
20	Tihomir	Tiho	3b
21	Bernarda	Nadic	3b
22	Franjo	Franjic	3b
23	Marica	Marinc	3b
24	Mirko	Mirkic	3b
25	Margareta	Gretic	3b
26	Ena	Enic	3b
27	(null)	(null)	4b

Slika 58. FULL JOIN

4.9. Pogledi

Pogled je objekt koji se također može kreirati u bazi podataka. Pogledi ne sadrže podatke pa ih nazivamo i virtualnim tablicama. Pogled nastaje na temelju upita. Upiti se kreiraju isto kao i kod tablica.

Naredba za kreiranje pogleda je CREATE VIEW. Sintaksa naredbe u sustava Oracle:¹⁷

```
CREATE [OR REPLACE]  
  [[NO] FORCE] VIEW [schema.] view  
  [ ( { alias [ inline_constraint... ]  
      | out_of_line_constraint  
      }  
    [, { alias [ inline_constraint... ]  
        | out_of_line_constraint  
        }  
    ]  
  )  
  | object_view_clause  
  | XMLType_view_clause  
  ]  
  AS subquery [ subquery_restriction_clause ] ;
```

OR REPLACE koristimo kada želimo kreirati novi pogled tako da ga zamijenimo sa nekim postojećim. Da bi izbjegli pisanje složenih upita možemo koristiti poglede.

Da bi vidjeli zaključnu ocjenu učenika prema predmetu te kojem razredu pripada i tko mu je razrednik, trebali bi spojiti pet tablica i pisati dugačke upite. To možemo olakšati kreiranjem tri pogleda.

Najprije kreiramo pogled koji prikazuje imena i prezime učenika te zaključnu ocjenu iz pojedinih predmeta. Upit je sljedeći:

```
CREATE OR REPLACE VIEW korisnik_zakljucna AS  
SELECT korisnici.id_korisnika, korisnici.ime, korisnici.prezime,  
predmeti.naziv_predmeta, pohada.zakljucna_ocjena  
FROM korisnici  
JOIN pohada  
ON korisnici.id_korisnika = pohada.id_korisnika  
JOIN predmeti  
ON predmeti.id_predmeta = pohada.id_predmeta;
```

Nakon što smo kreirali pogled, podatke dohvaćamo na isti način kao i kod tablica.

```
SELECT * FROM korisnik_zakljucna;
```

¹⁷ Oracle (2014) – Oracle Database SQL Language Reference 11g Release2 (11.2)
http://docs.oracle.com/cd/B28359_01/server.111/b28286/statements_8004.htm, preuzeto 30.kolovoza 2014.

	IME	PREZIME	NAZIV_PREDMETA	ZAKLJUCNA_OCJENA
1	Jakov	Jakiv	tjelesna i zdravstvena kultura	4
2	Jakov	Jakiv	informatika	5
3	Jakov	Jakiv	fizika	3
4	Jakov	Jakiv	kemija	3
5	Jakov	Jakiv	biologija	2
6	Jakov	Jakiv	likovna kultura	2
7	Jakov	Jakiv	glazbena kultura	4
8	Jakov	Jakiv	engleski jezik	5
9	Jakov	Jakiv	hrvatski jezik	3
10	Jakov	Jakiv	matematika	3
11	Monika	Monik	tjelesna i zdravstvena kultura	5
12	Monika	Monik	informatika	5
13	Monika	Monik	fizika	2
14	Monika	Monik	kemija	3
15	Monika	Monik	biologija	4
16	Monika	Monik	likovna kultura	4
17	Monika	Monik	glazbena kultura	5
18	Monika	Monik	engleski jezik	5

Slika 59. Rezultat pogleda1

Sada još trebamo kreirati pogled koji će prikazati nazive razreda i njihove razrednike. To možemo učiniti sljedećim upitom:

```
CREATE OR REPLACE VIEW razrednik AS
SELECT k1.id_korisnika, razredi.naziv_razreda, k2.prezime
FROM korisnici k1
JOIN ucenik_razred
ON k1.id_korisnika = ucenik_razred.id_korisnika
JOIN razredi
ON razredi.id_razreda = ucenik_razred.id_razreda
JOIN korisnici k2
ON k2.id_korisnika = razredi.razrednik;

SELECT * FROM razrednik;
```

	ID_KORISNIKA	NAZIV_RAZREDA	PREZIME
1	11	3a	Anic
2	12	3a	Anic
3	13	3a	Anic
4	14	3a	Anic
5	15	3a	Anic
6	16	3a	Anic
7	17	3a	Anic
8	18	3a	Anic
9	19	3b	Markic
10	20	3b	Markic
11	21	3b	Markic
12	22	3b	Markic
13	23	3b	Markic
14	24	3b	Markic
15	25	3b	Markic

Slika 60. Rezultat pogleda2

Na kraju trebamo spojiti podatke iz kreiranih pogleda. Spojit ćemo ih prema *id_korisnika* koji pripada učeniku. Kao rezultat dobit ćemo popis učenika sa zaključnim ocjenama prema predmetu, kojem razredu pripadaju te tko im je razrednik.

```
CREATE VIEW pogled1 AS
SELECT korisnik_zakljucna.ime, korisnik_zakljucna.prezime,
korisnik_zakljucna.naziv_predmeta, korisnik_zakljucna.zakljucna_ocjena,
razrednik.naziv_razreda, razrednik.prezime as razrednik
FROM korisnik_zakljucna
JOIN razrednik
ON korisnik_zakljucna.id_korisnika = razrednik.id_korisnika
ORDER BY korisnik_zakljucna.id_korisnika;

SELECT * FROM pogled1;
```

	IME	PREZIME	NAZIV_PREDMETA	ZAKLJUCNA_OCJENA	NAZIV_RAZREDA	RAZREDNIK
1	Jakov	Jakiv	matematika	3	3a	Anic
2	Jakov	Jakiv	hrvatski jezik	3	3a	Anic
3	Jakov	Jakiv	engleski jezik	5	3a	Anic
4	Jakov	Jakiv	glazbena kultura	4	3a	Anic
5	Jakov	Jakiv	likovna kultura	2	3a	Anic
6	Jakov	Jakiv	biologija	2	3a	Anic
7	Jakov	Jakiv	kemija	3	3a	Anic
8	Jakov	Jakiv	fizika	3	3a	Anic
9	Jakov	Jakiv	informatika	5	3a	Anic
10	Jakov	Jakiv	tjelesna i zdravstvena kultura	4	3a	Anic
11	Monika	Monik	matematika	2	3a	Anic
12	Monika	Monik	hrvatski jezik	5	3a	Anic
13	Monika	Monik	engleski jezik	5	3a	Anic
14	Monika	Monik	glazbena kultura	5	3a	Anic
15	Monika	Monik	likovna kultura	4	3a	Anic

Slika 61. Rezultat pogleda3

U sustavu Oracle moguće je i ažurirati pogled. U ažuriranje pogleda spada ažuriranje, brisanje ili upisivanje podataka putem pogleda.

Koristi se opcija WITH CHECK OPTION. Tom se opcijom provjerava svaka INSERT i UPDATE naredba te se provjerava i uvjet koji se nalazi u WHERE klauzuli.

Prilikom unosa ili ažuriranja podataka, ukoliko nova vrijednost ne zadovoljava uvjet iz WHERE klauzule, upit se neće izvršiti.

Za primjer ćemo kreirati pogled kojim se dohvaćaju podaci iz tablice *korisnici* na temelju vanjskog ključa koji ukazuje na *tip_korisnika*. U klauzuli WHERE definirali smo da je *tip_korisnika=2*, što znači da je uloga korisnika profesor. Sa opcijom WITH CHECK OPTION smo ograničili upis novih korisnika jer možemo upisati samo korisnike sa *tip_korisnika=2*.

Ako pokušamo unijeti novog korisnika sa vrijednošću *tip_korisnika=1* dobit ćemo poruku kojom nam ukazuju da narušavamo WITH CHECK OPTION te upis neće biti izvršen.

```
CREATE VIEW v2 AS
SELECT *
FROM korisnici
WHERE tip_korisnika = 2
WITH CHECK OPTION;
```

```
INSERT INTO v2 (id_korisnika, korisnicko_ime, ime, prezime, lozinka, oib,
datum_rodenja, adresa, tip_korisnika)
VALUES (korisnici_sequence.nextval, 'nniin', 'nino', 'nino', '154545',
'12456985236', '14-08-82', 'cakovec', 1);
```

```
Error starting at line : 7 in command -
insert into v2 (id_korisnika,korisnicko_ime,ime, prezime,lozinka, oib, datum_rodenja, adresa, tip_korisnika)
values (korisnici_sequence.NEXTVAL, 'nniin','Nino', 'Nino', 154545, '12456985236', '14-08-82', 'Cakovec', 1)
Error report -
SQL Error: ORA-01402: view WITH CHECK OPTION where-clause violation
01402. 00000 - "view WITH CHECK OPTION where-clause violation"
```

Slika 62. WITH CHECK OPTION greška

5. Zaključak

Baza podataka je ključna za informatizaciju poslovanja jer se u nju spremaju svi važni podaci. Ti su podaci kod relacijskih baza organizirani u tablicama. Da bi se napravila dobra baza podataka potrebno je izraditi dobar model. Nakon izrađenog modela može se krenuti u implementaciju baze podataka. Za implementaciju baze potreban je sustav za upravljanje bazama podataka.

U radu je opisan alat Oracle SQL Developer koji nudi mnoge mogućnosti, no naglasak je stavljen na modeliranje i implementaciju baze podataka. Također, prikazane su i karakteristike sustava za upravljanje radom baza podataka Oracle. Opisana je njegova arhitektura te tipovi podataka koje podržava. Oracle je prvi komercijalno dostupan sustav za upravljanje relacijskim bazama podataka, odnosno sustav za upravljanje bazama podataka koji je baziran na relacijskom modelu.

Za rad s bilo kojim sustavom potrebno je poznavanje SQL jezika. Kroz rad je detaljno prikazana izrada modela korak po korak te kasnije sama implementacija baze. Na kraju su opisane SQL naredbe i upiti te su dani primjeri gdje se i kako koriste. Kvalitetna baza podataka ključ je i temelj izrade dobre aplikacije.

6. Popis slika i tablica

Slika 1. Instaliranje OracleXE.....	4
Slika 2. Instaliranje OracleXE -postavljanje lozinke	5
Slika 3. OracleXE portovi	5
Slika 4. Fizička struktura.....	6
Slika 5. Instanca	8
Slika 6. Rezultat ROWID.....	12
Slika 7. Ekvivalentni ANSI i Oracle tipovi podataka	13
Slika 8. Skidanje Oracle SQL Developer	14
Slika 9. Stvaranje konekcije	15
Slika 10. Stvaranje novog korisnika.....	16
Slika 11. Dodjeljivanje uloga korisniku	17
Slika 12. Dodjeljivanje privilegija	18
Slika 13. Otvaranje Data Modeler-a.....	19
Slika 14. Odabir modela.....	20
Slika 15. Entitet korisnik.....	20
Slika 16. Atributi entiteta korisnik	21
Slika 17. Veza 1:N	22
Slika 18. Veza N:M.....	22
Slika 19. Kreiranje konceptualnog modela	23
Slika 20. Kreiranje entiteta.....	24
Slika 21. Dodavanje veza	24
Slika 22. Konceptualni model	25
Slika 23. Pretvaranje konceptualnog u relacijski model	26
Slika 24. Alatni izbornik za relacijski model	27
Slika 25. Stvaranje tablice	27
Slika 26. Dodavanje atributa	28
Slika 27. Dodavanje ograničenja.....	29
Slika 28. Auto Increment opcija.....	30
Slika 29. Opcionalan upis vanjskog ključa	31
Slika 30. Relacijski model.....	32
Slika 31. Generiranje DDL naredbe	34
Slika 32. Implementacija tablica-New Table	35
Slika 33. Implementacija tablice-Sekvence	36
Slika 34. Implementacija tablice-vanjski ključ	37
Slika 35. SQL Worksheet.....	38
Slika 36. Unos podataka.....	45
Slika 37. Unos podataka-predmeti	46
Slika 38. Upit.....	48
Slika 39. Dohvaćanje podataka SELECT	48
Slika 40. Klauzula WHERE	49
Slika 41. Operator BETWEEN	50
Slika 42. Operator LIKE	51
Slika 43. Operator LIKE 2	51
Slika 44. Operator IS (NOT) NULL	51
Slika 45. Klauzula HAVING	52
Slika 46. Opcija DISTINCT.....	52
Slika 47. Klauzula HAVING	53
Slika 48. Klauzula ORDER BY	53

Slika 49. Opcija DESCRIBE	54
Slika 50. Ažuriranje podataka1	55
Slika 51. Ažuriranje podataka2	55
Slika 52. Naredba DELETE 1	56
Slika 53. Rezultat naredbe DELETE	56
Slika 54. Spajanje klauzulama FROM i WHERE	58
Slika 55. [INNER] JOIN	58
Slika 56. LEFT JOIN	59
Slika 57. RIGHT JOIN	60
Slika 58. FULL JOIN	61
Slika 59. Rezultat pogleda1	63
Slika 60. Rezultat pogleda2	64
Slika 61. Rezultat pogleda3	64
Slika 62. WITH CHECK OPTION greška	65
 Tabela 1. Znakovni tip podataka	 11
Tabela 2. Numerički tip podatka	11
Tabela 3. Operatori uspoređivanja	49

7. Literatura

- [1] Rabuzin K. (2011). Uvod u SQL. Varaždin : Fakultet organizacije i informatike
- [2] Varga M. (2012). Upravljanje podacima. Zagreb: Element d.o.o.
- [3] Varga M. (1994). Baze podataka. Konceptualno, logičko i fizičko modeliranje podataka. Zagreb: Društvo za razvoj informacijske pismenosti (DRIP)
- [4] Manger R. (2012). Baze podataka. Zagreb: Element d.o.o.

- [5] Oracle (2014) – Oracle Database SQL Language Reference 11g Release2 (11.2) http://docs.oracle.com/cd/E18283_01/server.112/e17118.pdf preuzeto 23.kolovoza 2014.
- [6] Oracle (2014) – Oracle Database Concepts 11g Release 2 (11.2) http://docs.oracle.com/cd/E11882_01/server.112/e40540.pdf, preuzeto 7.kolovoza 2014.
- [7] Oracle (2014) – Oracle SQL Developer User's GUIDE Release 4.0 http://docs.oracle.com/cd/E39885_01/appdev.40/e38414.pdf, preuzeto 15.srpnja 2014.
- [8] TutorialsPoint http://www.tutorialspoint.com/sql_certificate/the_sql_select_statement.htm, preuzeto 28.kolovoza 2014.
- [9] Burleson Consulting http://dba-oracle.com/t_oracle_create_sequence.htm, preuzeto 18.kolovoza 2014.
- [10] Hrvatska udruga Oracle korisnika HROUG http://www.hroug.hr/hr/oracle_hrvatska/oracle_u_svijetu_i_hrvatskoj, preuzeto 7.kolovoza 2014.
- [11] Poslovni software <http://www.poslovni-software.com/tvrtke/oracle-hrvatska/29/>, preuzeto 7.kolovoza 2014.