LINKgroup

Distance Learning System

Database Administration and Security

Spajanje tabela

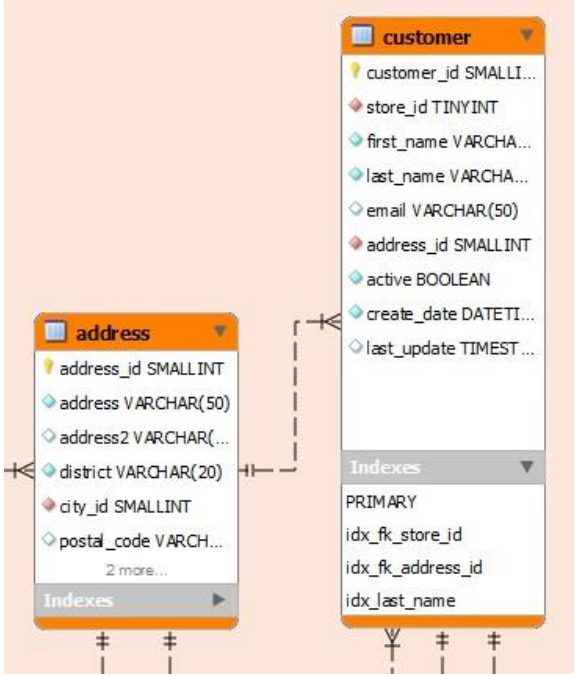
Povezivanje podataka vise tabela

U okviru tabele customer ne postoje konkretne informacije o na primer imenu ulice mušterije, već su takvi podaci grupisani u zasebnoj tabeli address.
Obe prikazane tabele poseduju atribut address_id.

Kako da dobijemo prikazane mušterije sa ulicom u kojoj žive
Prvi način koji bismo mogli da upotrebimo je da napišemo:
SELECT first_name, last_name, address
FROM customer, address;

Jedan način da ovaj upit popravimo kako bi vratio željeni rezultat je da navedemo uslov pojavljivanja rezultata, tj. da navedemo relaciju između tabela. To možemo postići korišćenjem ključne reči WHERE sa kojom smo se već upoznali:

SELECT first_name, last_name, address FROM customer, address WHERE address.address_id = customer.address_id;



JOIN

- Join je ključna reč u SQL-u, koja označava spoj između dve tabele. Ova ključna reč nikada se ne pojavljuje sama, već obavezno dolazi u kompletu sa ključnom rečju ON (ali često i nekim drugim ključnim rečima).

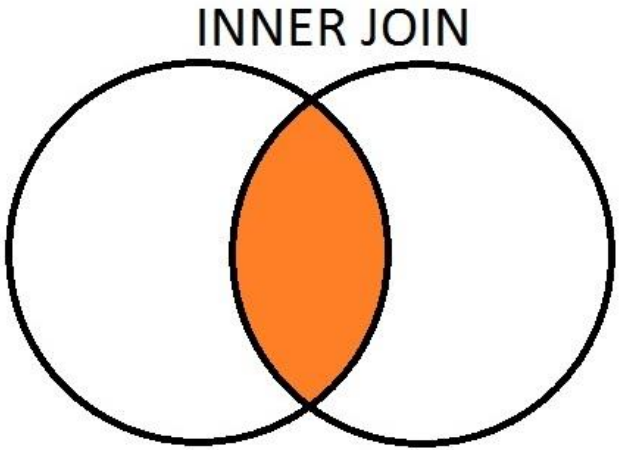
```
SELECT customer.first_name, customer.last_name,
address.address
FROM customer
JOIN address
ON address.address_id = customer.address_id
```

Vrste spajanja

- Postoje tri vrste JOIN-a:
  - **Inner Join**
  - **Left Join**
  - **Right Join**
  - **Cross Join**

Inner JOIN

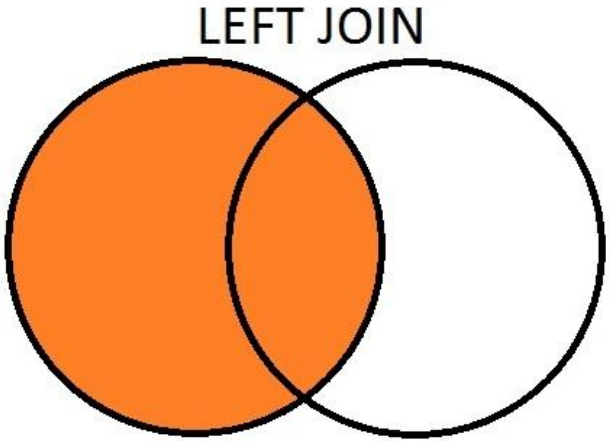
- Inner Join je zapravo Join koji smo videli na prethodnom primeru. Takoreći, potpuno identičan efekat ima korišćenje ključne reči JOIN i INNER JOIN, i u oba slučaja biće selektovani redovi koji za definisane kolone imaju vrednosti u obe tabele. Ovo znači da, ako govorimo hipotetički, ukoliko neki od naših mušterija nemaju unetu adresu, oni neće biti prikazani u rezultatima.




```
SELECT customer.first_name,
customer.last_name, address.address,
address.district
FROM customer INNER JOIN address
ON address.address_id = customer.address_id
```

Left JOIN

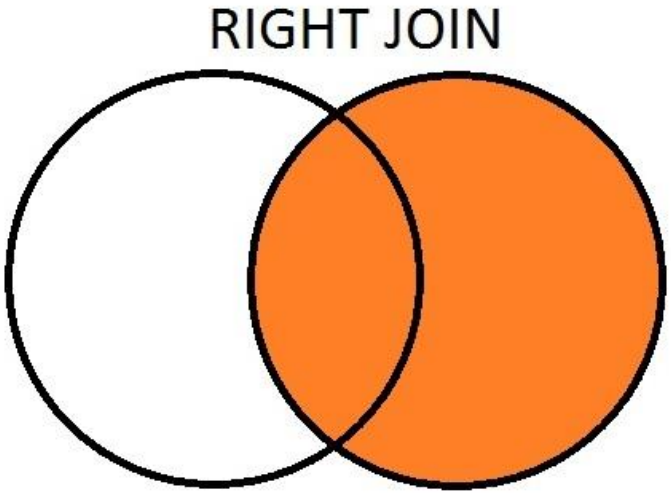
- Spajanje tabela definisano LEFT JOIN-om vratiće sve zapise iz leve tabele (tabele A), pa čak i ako neki od zapisa za tražene kolone nema vrednost




```
SELECT customer.first_name,
customer.last_name, address.address,
address.district
FROM customer LEFT JOIN address
ON address.address_id = customer.address_id
```

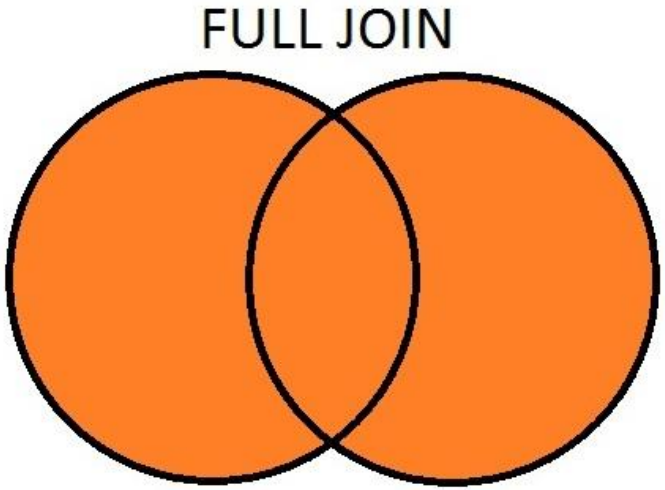
Right JOIN

- Slično kao kod prethodnog JOIN-a, sa RIGHT JOIN-om dobićemo sve zapise iz desne tabele (tabele B), i samo one koji imaju vrednosti za definisane kolone iz tabele A.



Cross JOIN

- Spajaju se sve kolone sa svim kolonama

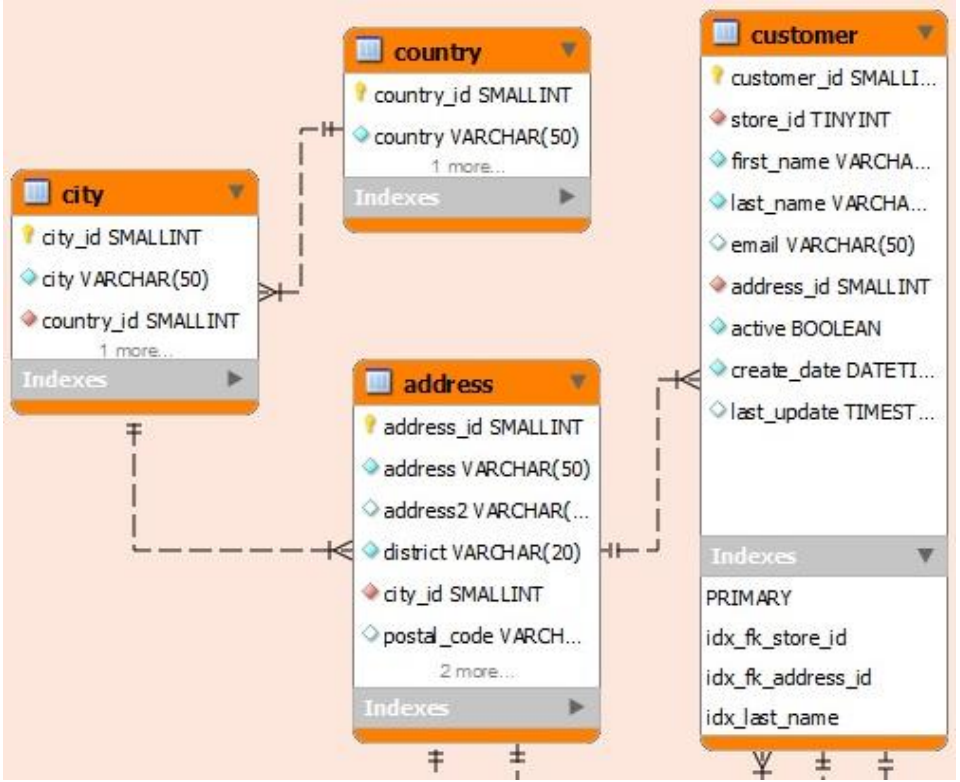


SELECT *
FROM customer cross JOIN address

Spajanje vise od dve tabele

Pomoću join-a, moguće je spojiti više od dve tabele

```
SELECT customer.first_name,
customer.last_name, address.address,
address.district, city.city,
country.country
FROM customer
JOIN address ON address.address_id =
customer.address_id
JOIN city ON city.city_id =
address.city_id
JOIN country ON country.country_id =
city.country_id
```



Naredbe za unos izmenu i brisanje podataka

- Naredbe za unos brisanje i izmenu podataka su:
- **INSERT**
- **DELETE**
- **UPDATE**

Insert

- Korišćenjem naredbe INSERT dodaju se novi zapisi u tabele baze podataka. Osnovna sintaksa ove naredbe je:

```
INSERT INTO mytable
(column1, column2...)
VALUES
(value1,value2...)
```

Insert – primeri koriscenja

- Osnovni unos:
- ***INSERT INTO country (country) VALUES ('Serbia');***

- Unos uz dobavljanje unetog id-a:
- ***INSERT INTO city(city, country_id) VALUES ('Belgrade',last_insert_id());***

Delete

- Naredba delete briše redove iz tabele, sa ili bez filtracije
- Sintaksa naredbe je:

  DELETE from TABELA
- Naredba DELETE, gotovo uvek se koristi u kombinaciji sa filtracijom. Na primer:

  DELETE FROM customer WHERE id=10

  ili

  DELETE FROM customer WHERE id BETWEEN 10 AND 20

  (BETWEEN je isto sto i id >=10 and id <=20)
- Ukoliko radimo **DELETE from TABELA**, id-ovi se ne resetuju, pa je tada moguće uraditi i: **TRUNCATE TABELA**

UPDATE

- Naredba UPDATE služi za ažuriranje određenog ili određenih redova. Ukoliko (na primer) želimo da izmenimo prezime mušterija, možemo napisati:

  UPDATE customer SET last_name = 'peterson'

- Ukoliko bismo želeli da izmenimo i ime i prezime, napisali bismo:

  UPDATE customer SET first_name = 'john', last_name = 'peterson'

UPDATE

- Kao i za naredbu DELETE i za UPDATE, skoro uvek će nam biti potrebna filtracija, odnosno navođenje uslova upita:

- **UPDATE customer SET first_name = 'john', last_name = 'peterson' WHERE customer_id=5**

- Ili

- **UPDATE customer SET first_name = 'john', last_name = 'peterson' WHERE first_name='david' AND last_name= 'miller'**

Vežba 1

- Potrebno je kreirati bazu podataka test_db. U ovoj bazi treba kreirati tabelu users, koja sadrži tri kolone id, name i password.

- Potrebno je uneti tri korisnika u ovu tabelu (Peter – sa šifrom 123, Jenny – sa šifrom 345 i John – sa šifrom 678)

Rešenje

```
create database test_db;
use test_db;
create table users ( id int primary key auto_increment, name varchar(50), password varchar(15) );
insert into users (name, password) values ('Peter','123'),('Jenny','345'),('John','567');
```

Vežba 2

- Za prethodno kreiranu tabelu (users) potrebno je kreirati upit za preuzimanje podataka. Treba preuzeti sve podatke o korisnicima čije ima počinje slovom j.

Rešenje

- select * from users where name like 'j%';

Vežba 3

- Za tabelu users potrebno je napraviti upit koji vraća sve korisnike čiji je ID veći od jedan i manji od pet.

Rešenje

- select * from users where id > 1 and id < 5;

Vežba 4

- Treba napraviti još jednu tabelu u test_db. Tabela se zove statuses i sadrži dva polja, ID i name. Potrebno je dodati tri naziva statusa u ovu tabelu. User, administrator i superadministrator.

- Treba izmeniti postojeću tabelu users dodavanjem još jedne kolone. Naziv kolone je status.

Rešenje

```
create table statuses ( id int primary key auto_increment, name varchar(30) );
insert into statuses (name) values ('user');
insert into statuses (name) values ('administrator');
insert into statuses (name) values ('superadministrator');

alter table users add column status int;
```

Vežba 5

- U tabeli users dodeliti statuse korisnicima, tako da John bude administrator, Jenny superadministrator, a Peter user.

Rešenje

```
update users set status = 2 where name = 'john';
update users set status = 3 where name = 'jenny';
update users set status = 1 where name = 'peter';
```

Vežba 6

- Potrebno je prikazati ime i naziv statusa korisnika sa ID-jem 2.

Rešenje

```
select users.name, statuses.name from users join statuses on
users.status = statuses.id where users.id = 2;
```