LINKgroup

Distance Learning System

MySql programiranje i administracija

Uvod u SQL

Šta je SQL

- SQL je jedini jezik baza podataka
- SQL je u širokoj primeni. Još od kako je predstavljen, ranih 70-ih godina prošlog veka, bio je kritikovan, menjan, proširivan, ali na kraju i prihvaćen od strane svih relevantnih činilaca na tržištu baza podataka.
- SQL je postao standard Američkog Nacionalnog Instituta za standarde (ANSI) 1986. godine i Internacionalne Organizacije za standarde (ISO) 1987. godine.
- Uprkos standardizaciji, SQL kod različitih sistema za upravljanje bazama podataka nije u potpunosti kompatibilan, pošto svaki proizvođač implemetira SQL sa određenim odstupanjima i primenom specifičnih proširenja.
- SQL se primarno koristi za formulisanje instrukcija sistemu baze podataka, koje obuhvataju naredbe za manipulaciju bazama i podacima.

SQL

SQL Sintaksa

- SQL nije osetljiv na velika i mala slova;
- Svaki SQL upit (ukoliko ih ima više) mora da se završi oznakom ; (ovaj separator se može promeniti);
- Stringovi se pišu isključivo u navodnicima; MySQL prihvata i jednostruke i dvostruke navodnike za string, ali je najbolja praksa koristiti jednostruke;
- Objekti u SQL-u se mogu staviti u Backtick navodnike (select * from `my table`);
- promenljive u SQL-u uvek počinju oznakom @.

Tekstualne konstante

- String predstavlja sekvencu bajtova ili karaktera, koji su ograničeni ili jednostrukim navodnicima (') ili dvostrukim navodnicima (").
- Generalno, postoje dva tipa stringova u MySQL-u i to su **binarni stringovi** i **nebinarni stringovi**, odnosno stringovi koji su sastavljeni od karaktera. Razlika između ove dve vrste stringova je u osnovnoj gradivnoj jedinici koja je za binarne stringove, naravno, bajt, dok je za nebinarne stringove to karakter. Još jedna razlika je veoma bitna, a to je činjenica da se za binarne stringove ne može definisati set karaktera ili kolacija, s obzirom da se čuvaju kao sekvenca bajtova

Komentari

- MySQL server podržava tri vrste komentara:
- jednolinijski komentar, navođenjem karaktera **#**
- jednolinijski komentar navođenjem karaktera **–**
- višelinijski komentar koji je kreiran na isti način kao i u programskom jeziku C, korišćenjem karaktera **/* i */** gde sve između ovih oznaka predstavlja komentar
- MySQL server poznaje i jednu vrstu specijalnih komentara, koji se nazivaju uslovni komentari. **/*! MySQL-specific code */**

SQL Naredbe

- SQL naredbe se dele na tri seta
  - Data Definition Language **(DDL)**
    - CREATE, ALTER DROP
  - Data Manipulation Language **(DML)**
    - SELECT, INSERT, UPDATE i DELETE
  - Data Control Language **(DCL)**
    - GRANT i REVOKE

Naredbe za definiciju

- Definicione SQL naredbe (DDL) čine tri naredbe: **CREATE**, **ALTER** i **DROP**. Isključivo se tiču strukture, a ne samih podataka, što znači da, uz pomoć njih nikada nećete dodavati ni jedan jedini podatak u bazu.

- Kada želite da kroz SQL skriptu stvorite bazu podataka, ili neke njene entitete, koristićete upravo ovaj set naredbi

CREATE

- Svaki objekat koji se pravi u bazi, pravi se ovom naredbom, počevši od same baze, preko korisnika, pa do tabela, ključeva, indeksa...
- Sintaksa naredbe je:

  CREATE object_type object_name

- Primer:
  - ***CREATE DATABASE my_database;***
  - ***CREATE TABLE mytable (id int primary key auto_increment, name varchar(40))***

Vežba

- Potrebno je kreirati tabelu (**cars**), koja će imati kolone: **car_id, car_make, car_model, car_price**
- Kolona car_id mora biti primarni ključ, i imati auto inkrement
- kolone **car_make i car_model** moraju biti tekstualnog tipa
- Kolona car_price mora biti decimal sa mogućnošću unosa šestocifrenog broja sa dve decimale

- sačuvati kreiranu tabelu za sledeći primer

ALTER

- Naredba ALTER služi da neki, već napravljeni, sadržaj zamenimo. Kada kažemo napravljeni, podrazumevamo da je napravljen naredbom CREATE

- ALTER naredba se obično koristi u jednom od dva oblika. U jednom, ona se ponaša kao naredba CREATE i prihvata kompletnu novu definiciju objekta, sa izmenama. U drugom, potrebno je eksplicitno naglasiti izmene i samo njih uneti kroz odgovarajuće modifikatore. Sami oblici zavise od tipa objekta nad kojim se alter izvršava.

CREATE VIEW my_view AS select 'my_view'
ALTER VIEW my_view as select 'my_edited_view'

CREATE TABLE my_table (id int, name varchar(50))
ALTER TABLE my_table ADD COLUMN surname varchar(50)

Vežba

- Tabeli kreiranoj u prethodnom primeru, treba dodati kolonu stock, koja predstavlja broj automobila na stanju
-

DROP

- DROP se koristi za brisanje objekta i kao parametre prihvata tip i naziv objekta:

```
DROP TABLE my_table
DROP DATABASE my_first_db
DROP INDEX my_idx
```

- Često se koristi u kombinaciji sa IF EXISTS:

```
DROP DATABASE IF EXISTS my_database;
CREATE DATABASE my_database;
```

Vežba

- Obrisati tabelu **cars** iz prethodnog primera

Vežba

- Potrebno je kreirati tabelu users koja će pamtiti sledeće podatke: id korisnika koji će se povećavati automatski za jedan prilikom svakog unosa i koji će biti primarni ključ tabele, ime i prezime korisnika.

- Potrebno je kreirati i dodatni upit koji kreiranoj tabeli dodaje još jedno polje sa šifrom korisnika.

Rešenje

http://pastie.org/pastes/10628628/text

```
create table users
(
        user_id int primary key auto_increment,
        user_firstName varchar(50),
        user_lastName varchar(50)
)

alter table users add column password varchar(15)
```

Vežba

- Potrebno je kreirati skriptu za kreiranje baze podataka application_06. Ova skripta će brisati postojeću bazu ukoliko postoji. Takođe, potrebno je kreirati skriptu za kreiranje tabele computer_parts, tako da obriše postojeću istoimenu tabelu ukoliko postoji. Ova tabela treba da sadrži podatke o ID-u uređaja, šifri uređaja, tipu uređaja, nazivu uređaja i ceni uređaja.

Rešenje

http://pastie.org/pastes/10628627/text

```
drop database if exists application_06;
create database application_06;
drop table if exists computer_parts;
create table computer_parts
(
        computer_part_id int primary key auto_increment,
        computer_part_code varchar(5),
        computer_part_type int,
        computer_part_name varchar(256),
        computer_part_price decimal(7,2)
)
```

Osnovne naredbe za pretragu

(za pracenje neophodna baza sakila)

- **SELECT**

- Select naredba je naredba koja emituje tabelarni rezultat na osnovu unetih parametara:

  SELECT 'hello'

- Iz primera možemo zaključiti da SELECT, iako se često koristi za prikazivanje podataka, nije zadužena za logiku preuzimanja tih podataka, već samo za njihov prikaz.

Preuzimanje podataka iz tabele

- SELECT može takođe posedovati i treći deo, a to je izvor podataka. Preuzimanje rezultata iz izvora podrazumeva i ključnu reč from, kojoj sledi izvor (tabela):

SELECT * FROM film

- Ovaj upit vraća sve redove i sve kolone tabele film

Restrikcija kolona

- Često, nećemo biti zainteresovani za prikaz svih kolona jedne tabele. Jer, ukoliko unesemo upit:

  SELECT * FROM actor

  dobijamo podatke iz svih kolona o svim glumcima upisanim u tabelu actor

- Ukoliko bismo želeli da dobijemo samo imena svih glumaca u bazi, napisali bismo:

  SELECT first_name FROM actor

- Ukoliko bismo želeli da prikažemo i imena i prezimena, napisali bismo:

  SELECT first_name, last_name FROM actor

- Još jednom da pomenem savet za dobavljanje podataka iz baze: uvek dobavljati podatke samo iz onih kolona čiji podaci će zapravo biti potrebni. Na taj način se rasterećuje baza podataka, klijentski program, kao i sama mreža kroz koju putuju podaci.

Utvrdjivanje broja unosa

- Često se može javiti potreba da utvrdite ukupan broj zapisa u nekoj tabeli. Jednostavno, nisu Vam potrebni nikakvi detalji, već samo ukupan broj zapisa, tj. redova ili linija jedne tabele.

- Ovo se postiže sledećom naredbom:

SELECT count(film_id) FROM film

- U navedenom upitu može se koristiti naziv bilo koje kolone iz tabele film umesto korišćene film_id kolone. Može se koristiti i zvezdica (*) za sve kolone. Rezultat je uvek isti.

Utvrđivanje broja jedinstvenih zapisa

- Ukoliko je potrebno da utvrdimo koliko određenih jedinstvenih zapisa postoji, potrebno je koristiti ključnu reč DISTINCT. Na primer, želimo da znamo koliko u bazi ima glumaca koji imaju jedinstven id broj:

```
SELECT COUNT(DISTINCT actor_id) FROM actor
```

- Prethodni primer nije efikasan, ali ako napišemo nešto ovako:

```
SELECT COUNT(DISTINCT first_name) FROM actor
```

- Na ovaj način utvrđujemo koliko glumaca postoji u bazi koji imaju jedinstveno ime. Naravno, više glumaca može imati identično ime, pa će broj glumaca sa jedinstvenim imenom biti manji od ukupnog broja glumaca u bazi.

Limitiranje rezultata

- Da bismo limitirali broj rezultata koje želimo da nam baza podataka vrati, koristimo ključnu reč LIMIT:

  SELECT first_name, last_name FROM actor LIMIT 2

- Naredba LIMIT može se koristiti i sa definisanjem takozvanog offset. Pomoću offset se definiše od kojeg unosa će baza započeti pretragu. Ako napišemo:

  SELECT first_name, last_name FROM actor LIMIT 2, 2

- Ukoliko napišemo:

  SELECT first_name, last_name FROM actor LIMIT 10, 5

- biće preskočeno prvih 10 unosa, i vraćeno narednih 5.

Stranicenje

- Od MySQL edicije 4.0 može se koristiti specijalna opcija **SQL_CALC_FOUND_ROWS** koja se koristi u paru sa funkcijom **FOUND_ROWS** i pomaže prilikom straničenja
- Prvo se napiše upit za selektovanje dela zapisa, ali korišćenjem opcije **SQL_CALC_FOUND_ROWS**:

SELECT SQL_CALC_FOUND_ROWS first_name, last_name FROM actor LIMIT 10;

- Zatim se, jednostavno, pozove funkcija **FOUND_ROWS**:

SELECT FOUND_ROWS();

Sortiranje rezultata

- Osnovno sortiranje, obavlja se parametrom **ORDER BY**
- Ukoliko bismo hteli da dobijemo imena i prezimena svih glumaca, a uz to i još sortirana po imenu od A do Z, napisali bismo ovako:
- **SELECT first_name, last_name FROM actor ORDER BY first_name;**
- Podrazumevano sortiranje, prilikom upotrebe ključne reči ORDER BY, je sortiranje od manje vrednosti ka većoj i, ukoliko želimo da to sortiranje bude obrnuto (od veće vrednosti ka manjoj), moramo to eksplicitno naglasiti. Za to koristimo ključne reči **DESC** (od većeg ka manjem) i **ASC** (od manjeg ka većem).
- **SELECT first_name, last_name FROM actor ORDER BY first_name DESC;**

Selektovanje specifičnih zapisa (WHERE)

- Najčešće nam neće biti potrebni svi zapisi iz neke tabele, pa čak ni deo njih, što smo postizali upotrebom limitiranja. Nekada će nam biti potrebni zapis(i) koji će zadovoljiti tačno definisane kriterijume. Ovi kriterijumi navode se ključnom rečju WHERE.

- Na primer, želimo da prikažemo nazive svih filmova iz tabele film, koji imaju vrednost kolone rating PG (Parental Guidance). To ćemo postići na sledeći način:

```
SELECT title FROM film WHERE rating='PG';
```

- Za poređenje se ne mora koristiti operator jednakosti. Na primer, ukoliko želimo da prikažemo imena svih filmova koji su duži od 2 sata, napisali bismo:

```
SELECT title FROM film WHERE length>120;
```

Priblizno podudaranje teksta (LIKE)

- Ukoliko je potrebno da izvršimo pretragu na osnovu samo dela nekog podatka, možemo koristiti ključnu reč LIKE. Na primer želimo da prikažemo sve glumce sa imenom koje počinje karakterima an:

SELECT first_name, last_name FROM actor WHERE first_name LIKE 'an%';

- Na sličan način možemo da utvrdimo da li se zapis neke kolone završava na određeni način:

SELECT first_name, last_name FROM actor WHERE first_name LIKE '%en';

- Možemo utvrditi i da li zapis neke kolone uopšte sadrži neku sekvencu karaktera bilo gde (na početku, na kraju, u sredini):

SELECT first_name, last_name FROM actor WHERE first_name LIKE '%en%';

Povezivanje filtera logickim operatorima

- Često ćemo, u uslovnom izrazu, koristiti više od jednog uslova, povezanih logičkim operatorima. Na primer, ako bismo želeli da vidimo sve filmove čiji naziv počinje slovom b i čije je trajanje duže od 3 sata, napisali bismo ovako:

- **SELECT title, length FROM film WHERE title LIKE 'B%' AND length > 180;**

- U uslovni izraz čak možemo staviti i sam upit:

- **SELECT title, length FROM film WHERE title LIKE 'B%' AND length > (SELECT 180);**

Grupisanje rezultata

- SQL omogućava sintaksu po kojoj je moguće vršiti grupisanje rezultata po određenim kolonoma. To se postiže upotrebom ključne reči GROUP BY. Ipak, ako bismo ovu naredbu upotrebili na sledeći način:

```
SELECT rating FROM sakila.film
GROUP BY rating;
```

- Naredba GROUP BY, gotovo uvek se koristi sa **agragatnim funkcijama** (funkcije će biti detaljno objašnjene u narednom modulu)
- Evo, na primer, kako bi prethodni primer imao mnogo više smisla:

```
SELECT rating, count(title) FROM sakila.film
GROUP BY rating;
```

Filtracija nakon grupisanja (HAVING)

- Da bi se izvršila filtracija rezultata nakon grupisanja, koristi se ključna reč HAVING
- Na primer:

```
SELECT rating, count(title) FROM sakila.film
GROUP BY rating HAVING count(title) > 200;
```

Vezba

- Potrebno je selektovati sve nazive filmova koji imaju rental_rate veći od 3, a uz to su duži od 2, ali kraći od tri sata.

Resenje

- Potrebno je selektovati sve nazive filmova koji imaju rental_rate veći od 3, a uz to su duži od 2, ali kraći od tri sata.

SELECT title FROM film WHERE rental_rate>3 AND length>120 AND length<180

Vezba

- Potrebno je preuzeti sve podatke, o svim filmovima koji u svom nazivu sadrže reč bird, a uz to imaju rental_duration veći od 5.

Resenje

- Potrebno je preuzeti sve podatke, o svim filmovima koji u svom nazivu sadrže reč bird, a uz to imaju rental_duration veći od 5.

```
SELECT * FROM sakila.film WHERE title LIKE
'%bird%' AND rental_duration>5
```

Vezba

- Potrebno je utvrditi koliko različitih rejtinga filmova postoji. Reč je o Parental Guidance rejtingu, koji je definisan u koloni rating

Resenje

- Potrebno je utvrditi koliko različitih rejtinga filmova postoji. Reč je o Parental Guidance rejtingu, koji je definisan u koloni rating.

SELECT COUNT(DISTINCT rating) FROM film