NAREDBE ZA definiciju podataka: FORMIRANJE BAZE PODATAKA/TABELA, MENJANJE BAZE PODATAKA/TABELA, DODAVANJE NOVIH PODATAKA, BRISANJE PODATAKA, AŽURIRANJE PODATAKA

NAREDBA ALTER DATEBASE

ALTER {DATABASE | SCHEMA} [*db_name*]
  alter_specification [*alter_specification*] ...

alter_specification:

Naredba ALTER DATABASE omogućava da promenimo strukturu i karakteristike baze podataka.

NAREDBA ALTER TABLE

ALTER [IGNORE] TABLE *tbl_name*
  alter_specification [, *alter_specification*] ...

alter_specification:
  table_option ...
| ADD [COLUMN] *column_definition* [FIRST | AFTER *col_name*]
| ADD [COLUMN] (*column_definition*,...)
| ADD {INDEX|KEY} [*index_name*] [*index_type*] (*index_col_name*,...)
| ADD [CONSTRAINT [*symbol*]]
    PRIMARY KEY [*index_type*] (*index_col_name*,...)
| ADD [CONSTRAINT [*symbol*]]
    UNIQUE [INDEX|KEY] [*index_name*] [*index_type*] (*index_col_name*,...)
| ADD [FULLTEXT|SPATIAL] [INDEX|KEY] [*index_name*] (*index_col_name*,...)
| ADD [CONSTRAINT [*symbol*]]
    FOREIGN KEY [*index_name*] (*index_col_name*,...)
    [*reference_definition*]
| ALTER [COLUMN] *col_name* {SET DEFAULT *literal* | DROP DEFAULT}
| CHANGE [COLUMN] *old_col_name column_definition*
    [FIRST|AFTER *col_name*]
| MODIFY [COLUMN] *column_definition* [FIRST | AFTER *col_name*]
| DROP [COLUMN] *col_name*
| DROP PRIMARY KEY
| DROP {INDEX|KEY} *index_name*
| DROP FOREIGN KEY *fk_symbol*
| DISABLE KEYS
| ENABLE KEYS
| RENAME [TO] *new_tbl_name*
| ORDER BY *col_name* [, *col_name*] ...
| CONVERT TO CHARACTER SET *charset_name* [COLLATE *collation_name*]
| [DEFAULT] CHARACTER SET *charset_name* [COLLATE *collation_name*]
| DISCARD TABLESPACE
| IMPORT TABLESPACE

index_col_name:
  col_name [(*length*)] [ASC | DESC]

index_type:
  USING {BTREE | HASH}

Naredba ALTER TABLE omogućava da promenimo strukturu postojeće tabele. Npr. možemo da dodamo i obrišemo kolone, indekse, premenimo ime kolone ili karaktersitike postojeće kolone.

Npr, dodaj kolonu strucna sprema u tabeli radnici :

ALTER TABLE radnici ADD (SS char(5));

- Može se koristiti više klauzula ADD, ALTER, DROP i CHANGE u jednoj ALTER TABLE naredbi, odbojene zapetom. Ovo je MySQL proširenje u odnosu na stndardni SQL, koji dozvoljava samo jednu klauzulu po ALTER TABLE naredbi. Npr. za prisanje više kolona u jednoj izjavi potrebno je napisati:
- ALTER TABLE t2 DROP COLUMN c, DROP COLUMN d;
- CHANGE *col_name*, DROP *col_name* i DROP INDEX su takođe MySQL proširenja u odnosu na standardni SQL.
- MODIFY se koristti u Oracle–ovom proširenju naredbe ALTER TABLE.
- Ime kolone se može promeniti pomoću naredbe CHANGE *old_col_name column_definition* . Npr u tabeli radnici kolona strucna sprema SS ce se zameniti u stepen strucne spreme STS i imace brojnu vrednost:
- ALTER TABLE radnici CHANGE polozaj STS INTEGER;

Za promenu atributa kolone bez promene naziva naredba je:

ALTER TABLE radnici CHANGE SS SS INTEGER NOT NULL;

Može da se koristi i komanda MODIFY za promenu atributa kolone bez promene naziva kolone:

ALTER TABLE radnici MODIFY SS INTEGER NOT NULL;

- Kod korišćenja klauzula CHANGE ili MODIFY pri čemu se skraćuje kolona u koja je indeksirana, rezulutujuća dužina kolone MySQL skraćuje i indeks automatski.
- Kod dodavanja kolone na određeno mesto u tabeli koristi se naredba BEFORE COLUMN *col_name*. Po default-u dodaje se kolona na kraju.

ALTER TABLE rednici ADD godinestz ...

- DROP INDEX briše indekse.
- Ako tabela sadrži samo jednu kolone, onda ta kolone ne može biti izbrisana. Potrebno je koristiti naredbu DROP TABLE .
- DROP PRIMARY KEY briše primarni ključ.

PRIMERI

Primeri pokazuju korišćenje naredbe ALTER TABLE. Prvo je prikazano kreiranje tabele t1 :

CREATE TABLE t1 (a INTEGER,b CHAR(10));

Za premenu imena tabele t1 u t2:

ALTER TABLE t1 RENAME t2;

Za promenu imena kolone a od INTEGER u TINYINT NOT NULL (ostaje ime isto) i za promenu imena kolone b od CHAR(10) u CHAR(20) kao i promene njenog imena iz b u c:

ALTER TABLE t2 MODIFY a TINYINT NOT NULL, CHANGE b c CHAR(20);

Za dodavanje nove kolone d:

ALTER TABLE t2 ADD d TIMESTAMP;

Za dodavanje indeksa u koloni d i a:

ALTER TABLE t2 ADD INDEX (d), ADD INDEX (a);

Za brisanje kolone c:

ALTER TABLE t2 DROP COLUMN c;

Za dodavanje kolone koja vrši automatsku numeraciju AUTO_INCREMENT integer column pod nazivom c:

ALTER TABLE t2 ADD c INT UNSIGNED NOT NULL AUTO_INCREMENT,
  ADD PRIMARY KEY (c);

Napomena: indeksirali smo po koloni c (kao PRIMARY KEY), zato što AUTO_INCREMENT **kolona mora biti indeksirana i deklarisana kao NOT NULL, jer primarni ključ ne sme biti NULL.**

NAREDBA CREATE DATABASE

CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] *db_name*
   [*create_specification* [*create_specification*] ...]

create_specification:
   [DEFAULT] CHARACTER SET *charset_name*
  | [DEFAULT] COLLATE *collation_name*

CREATE DATABASE kreira bazu podataka. CREATE SCHEMA je sinonim za CREATE DATABASE u MySQL 5.0.2.

NAREDBA CREATE INDEX

CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX *index_name*
   [*index_type*]
   ON *tbl_name* (*index_col_name*,...)

index_col_name:
   col_name [(*length*)] [ASC | DESC]

index_type:
  USING {BTREE | HASH}

CREATE INDEX part_of_name ON customer (name(10));

Primeri:

CREATE TABLE lookup (id INT) ENGINE = MEMORY;
CREATE INDEX id_index USING BTREE ON lookup (id);

NAREDBA CREATE TABLE

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] *tbl_name*
  (*create_definition*,...)
  [*table_option* ...]

ili:

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] *tbl_name*
  [(*create_definition*,...)]
  [*table_option* ...]
  select_statement

ili:

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] *tbl_name*
  { LIKE *old_tbl_name* | (LIKE *old_tbl_name*) }

create_definition:
  column_definition
 | [CONSTRAINT [*symbol*]] PRIMARY KEY [*index_type*] (*index_col_name*,...)
 | {INDEX|KEY} [*index_name*] [*index_type*] (*index_col_name*,...)
 | [CONSTRAINT [*symbol*]] UNIQUE [INDEX|KEY]
    [*index_name*] [*index_type*] (*index_col_name*,...)
 | {FULLTEXT|SPATIAL} [INDEX|KEY] [*index_name*] (*index_col_name*,...)
 | [CONSTRAINT [*symbol*]] FOREIGN KEY
    [*index_name*] (*index_col_name*,...) [*reference_definition*]
 | CHECK (*expr*)

column_definition:
  col_name data_type [NOT NULL | NULL] [DEFAULT *default_value*]
    [AUTO_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY]
    [COMMENT '*string*'] [*reference_definition*]

data_type:
  BIT[(*length*)]
 | TINYINT[(*length*)] [UNSIGNED] [ZEROFILL]
 | SMALLINT[(*length*)] [UNSIGNED] [ZEROFILL]
 | MEDIUMINT[(*length*)] [UNSIGNED] [ZEROFILL]
 | INT[(*length*)] [UNSIGNED] [ZEROFILL]
 | INTEGER[(*length*)] [UNSIGNED] [ZEROFILL]
 | BIGINT[(*length*)] [UNSIGNED] [ZEROFILL]
 | REAL[(*length*,*decimals*)] [UNSIGNED] [ZEROFILL]

| DOUBLE[(*length*,*decimals*)] [UNSIGNED] [ZEROFILL]
| FLOAT[(*length*,*decimals*)] [UNSIGNED] [ZEROFILL]
| DECIMAL(*length*,*decimals*) [UNSIGNED] [ZEROFILL]
| NUMERIC(*length*,*decimals*) [UNSIGNED] [ZEROFILL]
| DATE
| TIME
| YEAR
| CHAR(*length*)
  [CHARACTER SET *charset_name*] [COLLATE *collation_name*]
| VARCHAR(*length*)
  [CHARACTER SET *charset_name*] [COLLATE *collation_name*]
| BINARY(*length*)
| VARBINARY(*length*)
| TINYBLOB

index_col_name:
  col_name [(*length*)] [ASC | DESC]

index_type:
  USING {BTREE | HASH}

reference_definition:
  REFERENCES *tbl_name* [(*index_col_name*,...)]
    [MATCH FULL | MATCH PARTIAL | MATCH SIMPLE]
    [ON DELETE *reference_option*]
    [ON UPDATE *reference_option*]

reference_option:
  RESTRICT | CASCADE | SET NULL | NO ACTION

CREATE TABLE kreira tabelu sa datim imenom. .

Klauzula IF NOT EXISTS sprečava pojavu greške ukoliko tabela već postoji. Ipak ne postoji verifikacija da postojeća tabela ima identičnu strukturu kao u naredbi CREATE TABLE.

mysql> **CREATE TABLE test (a INT NOT NULL AUTO_INCREMENT,**
  -> **PRIMARY KEY (a), KEY(b));**

NAREDBA DROP DATABASE

DROP {DATABASE | SCHEMA} [IF EXISTS] *db_name*

DROP DATABASE briše sve tabele u bazi podataka i briše samu bazu podataka.

DROP SCHEMA je sinonim za DROP DATABASE u MySQL 5.0.2.

IF EXISTS sprečava pojavu gršeke u slučaju da već baza postoji.

NAREDBA DROP INDEX

DROP INDEX *index_name* ON *tbl_name*

DROP INDEX briše indeks *index_name* iz tabele *tbl_name*.

NAREDBA DROP TABLE

DROP [TEMPORARY] TABLE [IF EXISTS]
tbl_name [, *tbl_name*] ...
[RESTRICT | CASCADE]

DROP TABLE uklanja jednu ili više tabela.

NAREDBA RENAME TABLE

RENAME TABLE *tbl_name* TO *new_tbl_name*
[, *tbl_name2* TO *new_tbl_name2*] ...

Ova naredba se koristi za promenu imena jedne ili više tabela.

DODAVANJE VREDNOSTI U TABELU

Da bi se dodali novi redovi u tabeli koristi se naredba INSTERT INTO nazivtabele VALUES (vrednost1, vrednost2, vrednost2,..)

Npr.

INSERT INTO radnici VALUES (555, 320, 'direktor', 1, 'Kristoljub', 'Sipka', 'Mekenzijeva 10', 'Beograd', 10)

npr . sada hocemo da kolonu adresa ostavimo praznom

INSERT INTO radnici (IDRADNIKA,PLATA,POLOZAJ, IME, PREZIME, GRAD)

VALUES (12, 60000, 'admin', 'Pera', 'Peric', 'Beograd')

SLOŽENI USLOVI

Za složene upite koristimo operatore AND, OR i NOT.

Operator AND kombinuje dva ili više uslova i prikazuje neki red tabele samo ukoliko podaci tog reda zadovoljavaju **sve** navedene uslove (tj. svi uslovi su tačni). Na primer, da biste prikazali sve činovnike koji zarađuju više od 40.000, koristite naredbu:

```
SELECT IDRADNIKA
FROM TABELAPRIMANJARADNIKA
WHERE PLATA > 40000 AND POLOŽAJ = 'činovnik';
```

Operator OR kombinuje dva ili više uslova, ali prikazuje red ukoliko je **neki** navedeni uslov zadovoljen. Da biste zajedno prikazali one zaposlene koji zarađuju manje od 40.000 ili su im prinadležnosti manje od 10.000, koristite ovaj upit:

```
SELECT IDRADNIKA
FROM TABELAPRIMANJARADNIKA
WHERE PLATA < 40000 OR PRINADLEŽNOSTI < 10000;
```

Operatori AND i OR mogu da se kombinuju, na primer:

```
SELECT IDRADNIKA
FROM TABELAPRIMANJARADNIKA
WHERE POLOŽAJ = 'rukovodilac' AND PLATA > 60000 OR PRINADLEŽNOSTI > 12000;
```

SQL najpre pronalazi (odvojeno) redove u kojima zaposleni ima platu veću od 60.000 i ima rukovodeći položaj, a zatim iz ove liste redova izdvaja one koji zadovoljavaju gornji AND uslov ili uslov da su prinadležnosti veće od 12.000. Na kraju, SQL prikazuje ovu drugu novu listu, pri čemu treba znati da će svako ko prima prinadležnosti veće od 12.000 biti uključen jer operator OR uključuje red ako je bar jedan uslov tačan. Takođe obratite pažnju da se najpre izračunava rezultat operatora AND.

Ako generalizujemo ovaj proces, SQL najpre izračunava AND uslove da bi odredio redove koji zadovoljavaju AND operacije (zapamtite: svi uslovi moraju biti tačni), zatim ovi redovi se proveravaju prema OR uslovima, a na kraju prikazuje samo one preostale redove koji zadovoljavaju neki od OR uslova (pri čemu se za izračunavanje operatora OR koristi par uslova ili rezultata operatora AND, a dobija se rezultat tačno ako je bilo koji međurezultat tačan). Matematički, SQL izračunava najpre sve uslove, zatim izračunava sve AND parove i na kraju OR parove (pri čemu se oba operatora izračunavaju sa leva na desno).

Da biste primenili operatore OR pre operatora AND, kao npr. u slučaju kada želite da dobijete listu rukovodilaca koji imaju veliku platu (50.000) ili imaju velike prinadležnosti (10.000), koristite zagrade:

```
SELECT IDRADNIKA
FROM TABELAPRIMANJARADNIKA
WHERE POLOŽAJ = 'rukovodilac' AND (PLATA > 50000 OR PRINADLEŽNOSTI > 10000);
```

Operatori IN i BETWEEN

Lakši način za kombinovanje uslova je pomoću operatora IN ili BETWEEN. Na primer, ako želite da dobijete sve rukovodioce ili činovnike:

```
SELECT IDRADNIKA
FROM TABELAPRIMANJARADNIKA
WHERE POLOŽAJ IN ( 'rukovodilac', 'činovnik');
```

ili da prikažete sve zaposlene koji zarađuju više ili jednako 30.000, ali manje ili jednako 50,000:

```
SELECT IDRADNIKA
FROM TABELAPRIMANJARADNIKA
WHERE PLATA BETWEEN 30000 AND 50000;
```

Da biste prikazali sve čije plate nisu u ovom intervalu, pokušajte:

```
SELECT IDRADNIKA
FROM TABELAPRIMANJARADNIKA
WHERE PLATA NOT BETWEEN 30000 AND 50000;
```

Slično, NOT IN prikazuje sve redove koji nisu dobijeni u listi operatora IN.
Pored toga, operator NOT se može upotrebiti zajedno sa operatorima AND i OR, ali morate imati na umu da je on unarni operator (koristi se sa jednim uslovom, dajući kao rezultat njegovu suprotnu logičku vrednost, dok se operatori AND i OR koriste sa dva uslova), i da se svi operatori NOT izračunavaju pre bilo kojih operatora AND i OR.

Redosled izračunavanja logičkih operacija u SQL-u (svaka operacija izračunava se sa leva na desno) je :

1. NOT
2. AND
3. OR

Operator LIKE

Vratimo se na tabelu TabelaPrimanjaRadnika i pretpostavimo da želite da dobijete sve zaposlene čije prezime počinje sa „S“; pokušajte:

```
SELECT IDRADNIKA
FROM TABELAPRIMANJARADNIKA
WHERE PREZIME LIKE 'S%';
```

Procenat (%) se koristi ukoliko želimo da predstavimo bilo koji znak (cifru, slovo, znak interpunkcije) ili skup znakova koji mogu da slede iza slova „S“. Da biste pronašli zaposlene čija se prezimena završavaju sa „S“, koristite '%S', ili ako želite „S“ u sredini reči, pokušajte '%S%'. Znak '%' može biti upotrebljen umesto bilo kojih znakova koji se nalaze na istoj poziciji relativno od datih znakova. Operator NOT LIKE prikazuje redove koji ne zadovoljavaju dati kriterijum. Postoje i druge mogućnosti za upotrebu operatora LIKE, ili onih drugih koje smo razmatrali, ali to umnogome zavisi od konkretnog sistema za upravljanje

bazama podataka koji koristite; kao i obično, konsultujte priručnik ili administratora sistema da biste saznali koje mogućnosti postoje na vašem sistemu, ili da biste proverili da li ono što pokušavate da uradite jeste raspoloživo i dopušteno. Ova napomena važi i za mogućnosti SQL-a koje ćemo opisati u nastavku. Cilj ovog odeljka je samo da vam predoči ideju mogućnosti za upite koji se mogu pisati u SQL-u.