Limbajul de definitie a datelor DDL

- Definirea bazelor de date (creare, modificare, stergere)
- Definirea tabelelor (creare, modificare, stergere)
- Tipuri de date
- Constrangeri de integritate
- Constrangeri referentiale
- Indecsi
- View-uri

SQL DDL

Definirea bazelor de date

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

create_specification:
[DEFAULT] CHARACTER SET [=] charset_name | [DEFAULT]
COLLATE [=] collation_name
```

Exemple:

```
CREATE DATABASE test
```

```
CREATE DATABASE IF NOT EXISTS test
```

```
CREATE DATABASE IF NOT EXISTS test
CHARACTER SET = 'ascii';
```

```
CREATE DATABASE IF NOT EXISTS test
CHARACTER SET = 'utf8';
COLLATE = 'utf8_bin';
```

```
SHOW COLLATION;
```

```
SHOW CHARACTER SET;
```

Modificare baza de date

```
ALTER {DATABASE | SCHEMA} [db_name]
alter_specification ...
ALTER {DATABASE | SCHEMA} db_name
UPGRADE DATA DIRECTORY NAME

alter_specification:
[DEFAULT] CHARACTER SET [=] charset_name |
[DEFAULT] COLLATE [=] collation_name
```

Exemple:

```
ALTER DATABASE
CHARACTER SET = 'ascii'
COLLATE = 'ascii_bin';
```

Stergere baza de date

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

Exemple:

```
DROP DATABASE test
```

```
DROP DATABASE IF EXISTS test
```

SQL DDL

Definirea tabelelor

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

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

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

unde:

```
create_definition:

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

SQL DDL

si:

```
column_definition:

data_type [NOT NULL | NULL] [DEFAULT default_value]
[AUTO_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY]
[COMMENT 'string'] [reference_definition]
[COLUMN_FORMAT {FIXED|DYNAMIC|DEFAULT}]
[STORAGE {DISK|MEMORY|DEFAULT}]
```

si:

```
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 | TIMESTAMP | DATETIME | YEAR
| CHAR[(length)] [CHARACTER SET charset_name] [COLLATE collation_name]
| VARCHAR(length) [CHARACTER SET charset_name] [COLLATE collation_name]
| BINARY[(length)] | VARBINARY(length)
| TINYBLOB | BLOB | MEDIUMBLOB
| LONGBLOB
| TINYTEXT [BINARY] [CHARACTER SET charset_name] [COLLATE collation_name]
| TEXT [BINARY] [CHARACTER SET charset_name] [COLLATE collation_name]
| MEDIUMTEXT [BINARY] [CHARACTER SET charset_name] [COLLATE collation_name]
| LONGTEXT [BINARY] [CHARACTER SET charset_name] [COLLATE collation_name]
| ENUM(value1,value2,value3,...) [CHARACTER SET charset_name] [COLLATE collation_name]
| SET(value1,value2,value3,...) [CHARACTER SET charset_name] [COLLATE collation_name]
| spatial_type
```

Tipuri de date in SQL

- Reprezentarea numerelor intregi:
  - TINYINT ($-2^7 < x < 2^7-1$)
  - SMALLINT ($-2^{15} < x < 2^{15}-1$)
  - INTEGER

- Pentru reprezentarea numere reale:
  - DECIMAL: Se poate preciza precizia si numarul de cifre dupa virgula; Ex: DECIMAL (10,4) va avea maximum 6 cifre inainte de virgula si 4 dupa virgula; DECIMAL poate fi areviat DEC; NUMERIC este un echivalent pentru DECIMAL
  - FLOAT: reprezentare de numere in virgula flotanta
  - DOUBLE PRECISION: reprezentare de numere in virgula flotanta cu dubla precizie

- Reprezentarea valorilor alfanumerice:
  - CHARACTER: tip de date alfanumeric utilizat pentru reprezentarea de cuvinte, text, coduri. CHARACTER(10): sir de caractere de lungime 10; daca nu se specifica, default este lungimea 1; max. lungimii este 255; poate fi abreviat CHAR;

SQL DDL

- Reprezentarea valorilor alfanumerice (continuare)
  - VARCHAR: este folosit la fel ca si CHARACTER, pentru a stoca valori de tip text; lungimea maxima este tot de 255 de caractere; diferenta fundamentala tine de modul de stocare: in cazul CHAR, daca se intializeaza coloana cu dimensiunea 16 si se atribuie 4 caractere, se vor stoca 16, ultimele 12 fiind completate cu spatii; in cazul VARCHAR, se va stoca numai un sir de caractere de dimensiunea atribuita;
  - LONG VARCHAR
- Reprezentarea valorilor temporale:
  - DATA: pentru reprezentarea datelor calendaristice, se compune din (an, luna, zi)
  - TIME: stocheaza timpul (ora, minut, secunda)
  - TIMESTAMP: cominatie de data si timp;
- Reprezentare date binare:
  - BINARY;
  - VARBINARY;
  - LONG VARBINARY;

SQL DDL

Observatii:

Folosind cuvantul cheie TEMPORARY, la crearea unei tabele, tabela va exista numai in timpul sesiunii/conexiunii curente la baza de date;

Cuvantul cheie IF NOT EXISTS impiedica aparitia unei erori, in cazul in care tabela declarata exista deja; pe de alta parte, nu se verifica daca tabela existenta are aceeasi structura cu cea indicata de CREATE TABLE;

Daca nu se specifica atributul NULL sau NOT NULL, coloana e tratata ca si cum s-ar fi specificat atributul NULL;

Atributul AUTO_INCREMENT nu se poate atribui decat unei singure coloane intr-o tabela; acest atribut nu se aplica decat tipurilor intregi sau reale (float, double).

Tipurile de tip caracter (char, varchar, text) pot avea atribuite CHARACTER SET – setul de caractere atribuit acelei coloane;

Clauza DEFAULT permite setarea unei valori default pentru o coloana; de exemplu, pentru un tip data, se poate folosi o functie de tip NOW() sau CURRENT_TIME;

KEY e in mod normal un sinonim pentru INDEX; PRIMARY KEY poate fi simplu KEY atunci cand este folosit in definirea unei coloane;

PRIMARY KEY este un index pentru care toate coloanele care intra in definirea lui trebuie sa fie NOT NULL: daca nu sunt astfel, sunt definite implicit (si tacit).

Se poate creea o tabela din alta utilizand clauza SELECT la sfarsitul comenzii CREATE TABLE;

Folosind clauza LIKE, se poate creea o tabela goala folosind structura tabelei originale invocate dupa clauza LIKE;

SQL DDL

Exemple:

```
 9  create database d1;
10  USE d1;
11  CREATE TABLE t1
12  (c1 INT UNIQUE AUTO_INCREMENT,
13   c2 INT DEFAULT 1,
14   c3 DATE,
15   c4 TIME,
16   c5 FLOAT,
17   c6 DOUBLE,
18   c7 CHAR(20),
19    PRIMARY KEY (c1));
20
21  #Prezentarea structurii unei tabele
22  describe t1;
```

Field	Type	Null	Key	Default	Extra
c1	int(11)	NO	PRI	NULL	auto_increment
c2	int(11)	YES		1	
c3	date	YES		NULL	
c4	time	YES		NULL	
c5	float	YES		NULL	
c6	double	YES		NULL	
c7	char(20)	YES		NULL	

```
24  #inserarea (adaugarea) de inregistrari intr-o tabela
25  INSERT INTO t1 SET c2 = 10, c3='2008-02-20', c4='12:30:20', c5=1.2, c6=0.02, c7='cuvant';
26  INSERT INTO t1 SET c3='2008/02/12', c4='8:56:20', c7='curs';
27
28  select * from t1;
```

c1	c2	c3	c4	c5	c6	c7
1	10	2008-02-20	12:30:20	1.2	0.02	cuvant
2	1	2008-02-12	08:56:20	NULL	NULL	curs
NULL	NULL	NULL	NULL	NULL	NULL	NULL

SQL DDL

Exemple:

c:\wamp\bin\mysql\mysql5.0.51b\bin\mysql.exe

```
mysql>
mysql> CREATE TABLE d2.persoana
    -> (nume char(10),
    ->  prenume char(10),
    ->  adresa char(30),
    ->  telefon int,
    ->  PRIMARY KEY (nume, prenume));
Query OK, 0 rows affected (0.01 sec)

mysql> describe d2.persoana;
+---------+----------+------+-----+---------+-------+
| Field   | Type     | Null | Key | Default | Extra |
+---------+----------+------+-----+---------+-------+
| nume    | char(10) | NO   | PRI |         |       |
| prenume | char(10) | NO   | PRI |         |       |
| adresa  | char(30) | YES  |     | NULL    |       |
| telefon | int(11)  | YES  |     | NULL    |       |
+---------+----------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

```
INSERT INTO d2.persoana SET nume='Preda', prenume='Gabriel', adresa='Bucuresti S6';
INSERT INTO d2.persoana SET nume='Preda', prenume='Cristian', adresa='Bucuresti S4';
INSERT INTO d2.persoana SET nume='Preda', prenume='Caterina', adresa='Bucuresti S1';
INSERT INTO d2.persoana SET nume='Popescu', prenume='Gabriel', adresa='Bucuresti S4';
INSERT INTO d2.persoana SET nume='Cristian', prenume='Preda', adresa='Brasov';
```

c:\wamp\bin\mysql\mysql5.0.51b\bin\mysql.exe

```
mysql> select * from persoana;
+----------+----------+--------------+---------+
| nume     | prenume  | adresa       | telefon |
+----------+----------+--------------+---------+
| Cristian | Preda    | Brasov       |    NULL |
| Popescu  | Gabriel  | Bucuresti S4 |    NULL |
| Preda    | Caterina | Bucuresti S1 |    NULL |
| Preda    | Cristian | Bucuresti S4 |    NULL |
| Preda    | Gabriel  | Bucuresti S6 |    NULL |
+----------+----------+--------------+---------+
```

SQL DDL

Exemple:

```
c:\wamp\bin\mysql\mysql5.0.51b\bin\mysql.exe
mysql> CREATE TABLE traducere
    -> (token_id INT UNIQUE AUTO_INCREMENT PRIMARY KEY,
    ->  token_nume char(20),
    ->  engleza char(50) CHARACTER SET'ascii',
    ->  romana char(50) CHARACTER SET 'utf8',
    ->  chineza char(50) CHARACTER SET 'big5');
Query OK, 0 rows affected (0.02 sec)
```

```
INSERT INTO traducere set token_nume='OPEN_FILE', engleza='Open', romana='Deschide', chineza='開';
INSERT INTO traducere set token_nume='SAVE_FILE', engleza='Save', romana='Salvează', chineza='除';
```

```
c:\wamp\bin\mysql\mysql5.0.51b\bin\mysql.exe
mysql> select * from traducere;
+----------+------------+---------+----------+---------+
| token_id | token_nume | engleza | romana   | chineza |
+----------+------------+---------+----------+---------+
|        1 | OPEN_FILE  | Open    | Deschide | ?       |
|        2 | SAVE_FILE  | Save    | Salvează | ?       |
+----------+------------+---------+----------+---------+
2 rows in set (0.00 sec)
```

Creare tabela folosind SELECT (se copiaza si datele):

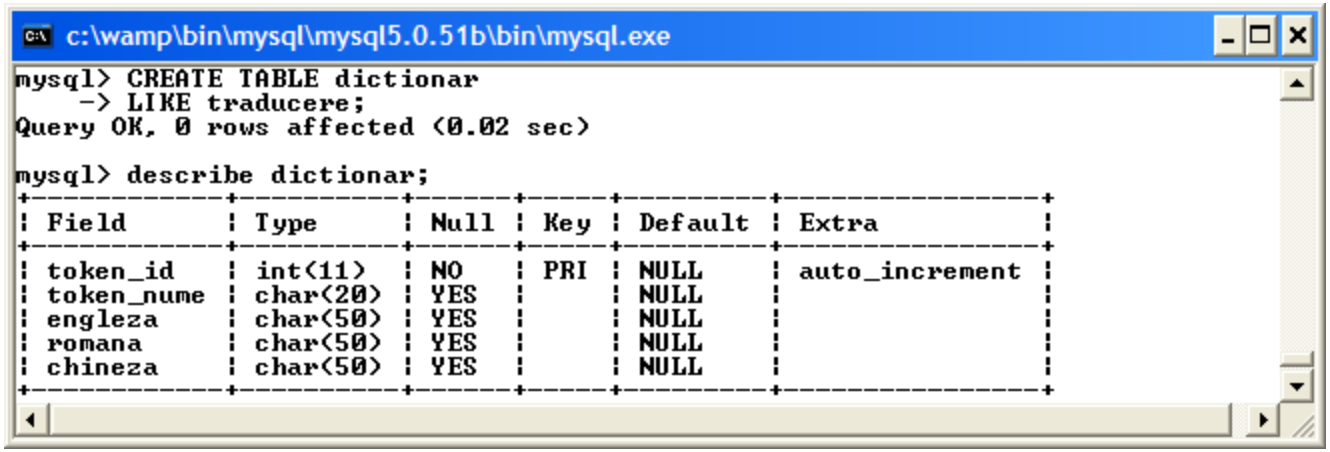
```
c:\wamp\bin\mysql\mysql5.0.51b\bin\mysql.exe
mysql> CREATE TABLE translation
    -> SELECT token_nume as token_name, engleza as English, romana AS Romanian
    -> FROM traducere;
Query OK, 2 rows affected (0.02 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> select * from translation;
+------------+---------+----------+
| token_name | English | Romanian |
+------------+---------+----------+
| OPEN_FILE  | Open    | Deschide |
| SAVE_FILE  | Save    | Salvează |
+------------+---------+----------+
2 rows in set (0.00 sec)
```

SQL DDL

Creare tabela folosind SELECT (se copiaza si datele):

Creare tabela folosind LIKE (Se copiaza numai structura tabelei, nu se pastreaza datele):



SQL DDL

Modificare tabela

```
ALTER [ONLINE | OFFLINE] [IGNORE] TABLE tbl_name
alter_specification [, alter_specification] ...
```

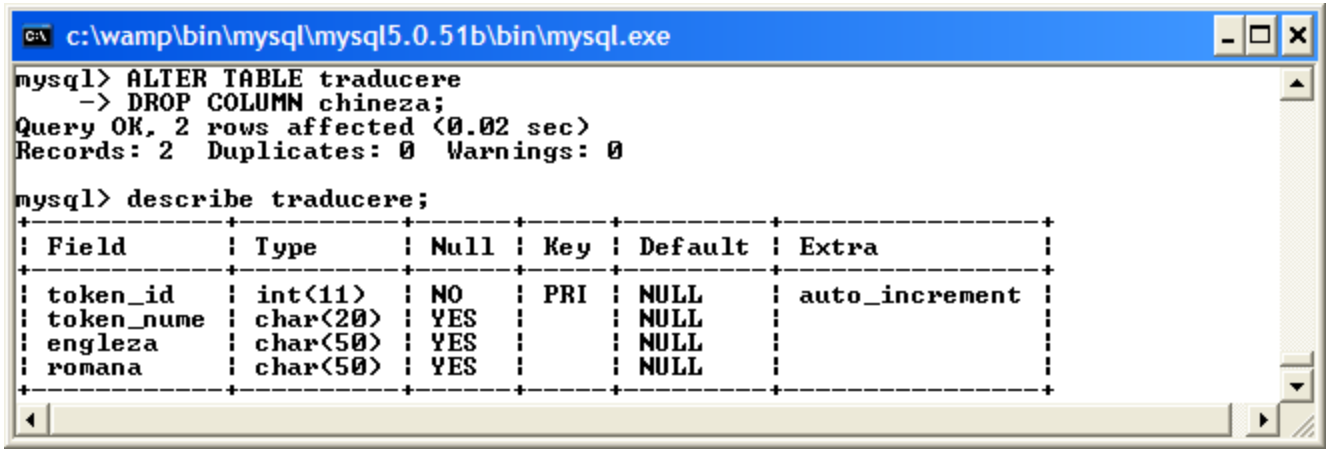
```
alter_specification:

table_option ...
| ADD [COLUMN] col_name column_definition [FIRST | AFTER col_name ]
| ADD [COLUMN] (col_name column_definition,...)
| ADD {INDEX|KEY} [index_name] [index_type] (index_col_name,...) [index_option] ...
| ADD [CONSTRAINT [symbol]] PRIMARY KEY [index_type] (index_col_name,...) [index_option] ...
| ADD [CONSTRAINT [symbol]] UNIQUE [INDEX|KEY] [index_name] [index_type] (index_col_name,...)
[index_option] ...
| ADD FULLTEXT [INDEX|KEY] [index_name] (index_col_name,...) [index_option] ...
| ADD SPATIAL [INDEX|KEY] [index_name] (index_col_name,...) [index_option] ...
| 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 new_col_name column_definition [FIRST|AFTER col_name]
| MODIFY [COLUMN] col_name 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]
[...]
```

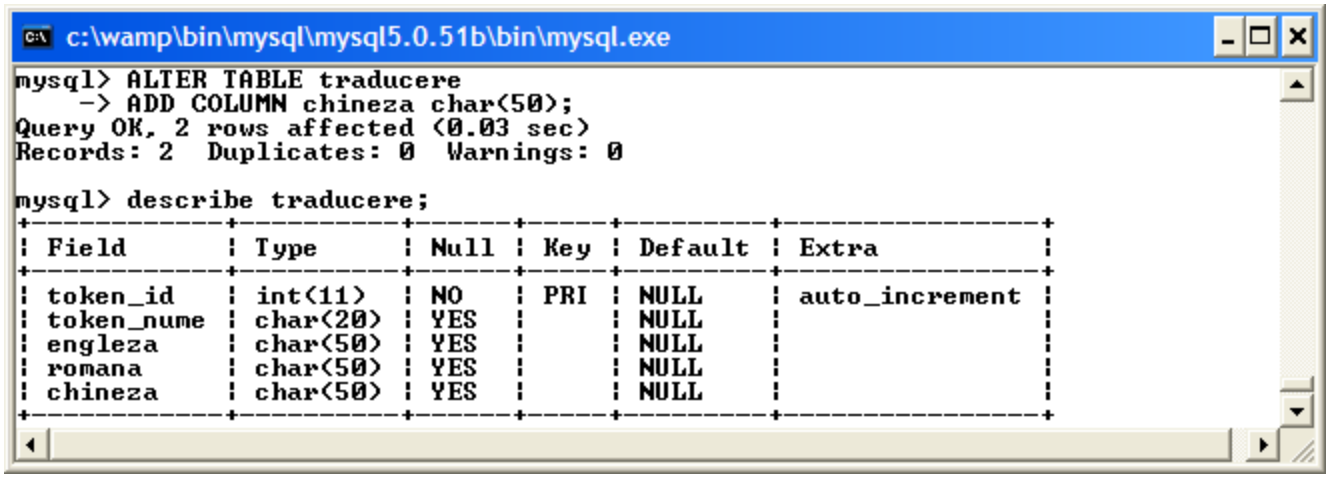
SQL DDL

Exemple:

Stergerea unei coloane



Adaugarea unei coloane



SQL DDL

Exemple:

Modificarea unei coloane

c:\wamp\bin\mysql\mysql5.0.51b\bin\mysql.exe

```
mysql> ALTER TABLE traducere
    -> ALTER COLUMN chineza SET DEFAULT 'Necompletat';
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select * from traducere;
+----------+------------+---------+----------+---------+
| token_id | token_nume | engleza | romana   | chineza |
+----------+------------+---------+----------+---------+
|        1 | OPEN_FILE  | Open    | Deschide | NULL    |
|        2 | SAVE_FILE  | Save    | Salvează | NULL    |
+----------+------------+---------+----------+---------+
2 rows in set (0.00 sec)

mysql> INSERT INTO traducere set token_nume='SAVE_AS_FILE', engleza='Save As', r
omana='Salvează ca';
Query OK, 1 row affected (0.02 sec)

mysql> SELECT * FROM traducere;
+----------+--------------+---------+-------------+-------------+
| token_id | token_nume   | engleza | romana      | chineza     |
+----------+--------------+---------+-------------+-------------+
|        1 | OPEN_FILE    | Open    | Deschide    | NULL        |
|        2 | SAVE_FILE    | Save    | Salvează    | NULL        |
|        3 | SAVE_AS_FILE | Save As | Salvează ca | Necompletat |
+----------+--------------+---------+-------------+-------------+
```

c:\wamp\bin\mysql\mysql5.0.51b\bin\mysql.exe

```
mysql> ALTER TABLE traducere
    -> ALTER COLUMN chineza DROP DEFAULT;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> INSERT INTO traducere set token_nume='PRINT_FILE', engleza='Print',
a='Tipăreşte';
Query OK, 1 row affected, 1 warning (0.02 sec)

mysql> SELECT * FROM traducere;
+----------+--------------+---------+-------------+-------------+
| token_id | token_nume   | engleza | romana      | chineza     |
+----------+--------------+---------+-------------+-------------+
|        1 | OPEN_FILE    | Open    | Deschide    | NULL        |
|        2 | SAVE_FILE    | Save    | Salvează    | NULL        |
|        3 | SAVE_AS_FILE | Save As | Salvează ca | Necompletat |
|        4 | PRINT_FILE   | Print   | Tipăreşte   | NULL        |
+----------+--------------+---------+-------------+-------------+
```

SQL DDL

Exemple:

Modificarea unei coloane

```
c:\wamp\bin\mysql\mysql5.0.51b\bin\mysql.exe
mysql> ALTER TABLE traducere
    -> CHANGE COLUMN chineza japoneza char(50);
Query OK, 4 rows affected (0.03 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM traducere;
+----------+--------------+---------+-------------+-------------+
| token_id | token_nume   | engleza | romana      | japoneza    |
+----------+--------------+---------+-------------+-------------+
|        1 | OPEN_FILE    | Open    | Deschide    | NULL        |
|        2 | SAVE_FILE    | Save    | Salvează    | NULL        |
|        3 | SAVE_AS_FILE | Save As | Salvează ca | Necompletat |
|        4 | PRINT_FILE   | Print   | Tipăreşte   | NULL        |
+----------+--------------+---------+-------------+-------------+
4 rows in set (0.00 sec)
```

```
c:\wamp\bin\mysql\mysql5.0.51b\bin\mysql.exe
mysql> ALTER TABLE translation
    -> MODIFY COLUMN French char(60) AFTER Romanian;
Query OK, 4 rows affected (0.03 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM translation;
+--------------+---------+-------------+--------+
| token_name   | English | Romanian    | French |
+--------------+---------+-------------+--------+
| OPEN_FILE    | Open    | Deschide    | NULL   |
| SAVE_FILE    | Save    | Salvează    | NULL   |
| SAVE_AS_FILE | Save As | Salvează ca | NULL   |
| PRINT_FILE   | Print   | Tipăreşte   | NULL   |
+--------------+---------+-------------+--------+
```

Redenumirea tabelei

```
c:\wamp\bin\mysql\mysql5.0.51b\bin\mysql.exe
mysql> ALTER TABLE translation
    -> RENAME TO translate;
Query OK, 0 rows affected (0.00 sec)

mysql> _
```

SQL DDL

Stergerea unei tabele

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

Exemple:

```
DROP TABLE IF EXISTS traducere;
DROP TABLE IF EXISTS translation;
```

SQL DDL

Specificarea constrangerilor

In SQL, se impune integritatea datelor prin impunerea unor constrangeri.
Integritatea datelor se refera la consistenta si corectitudinea datelor.
Consistenta se refera la lipsa contradictiilor intre date individuale.
Corectitudinea se refera la respectarea tuturor regulilor relevante pentru anumite date.

Constrangerile utilizate in SQL:

- Chei primare
- Chei alternative
- Chei straine
- Actiunea referentiala

SQL DDL

Chei primare

O cheie primara este definita ca o coloana sau un grup de coloane pentru care valorile sunt unice intotdeauna; o coloana definita ca si cheie primara trebuie sa fie definita ca NOT NULL; se pot defini in doua feluri: imediat dupa definirea unei coloana si la sfarsitul definitiei unei tabele.

- Pentru o tabela se poate defini o singura cheie primara;
- Modelul relational impune (in teorie) definirea unei chei primare pentru fiecare tabela; SQL nu impune insa aceasta constrangere;
- Valorile coloanei definita cheie primara trebuie sa fie unice pentru fiecare inregistrare (regula unicitatii);
- Daca prin eliminarea unei coloane dintr-o cheie primara compusa cheia rezultanta asigura conditia de unicitate, cheia initiala nu era corect definita (regula minimalitatii);
- Numele unei coloane poate sa apara o singura data in definitia unei chei primare;
- Coloanele care compun o cheie primara nu pot avea valori nule.

SQL DDL

Chei alternative

O cheie alternativa (selectata dintre cheile candidate) are aceleasi caracteristici ca si o cheie primara, cu doua constrangeri mai putin: poate avea valori NULL si pot exista, pentru o tabela, mai multe chei alternative. Cheile alternative se introduc folosind clauza UNIQUE dupa definitia unei coloane.

Chei straine

Cheile straine se folosesc pentru a impune integritatea referentiala.

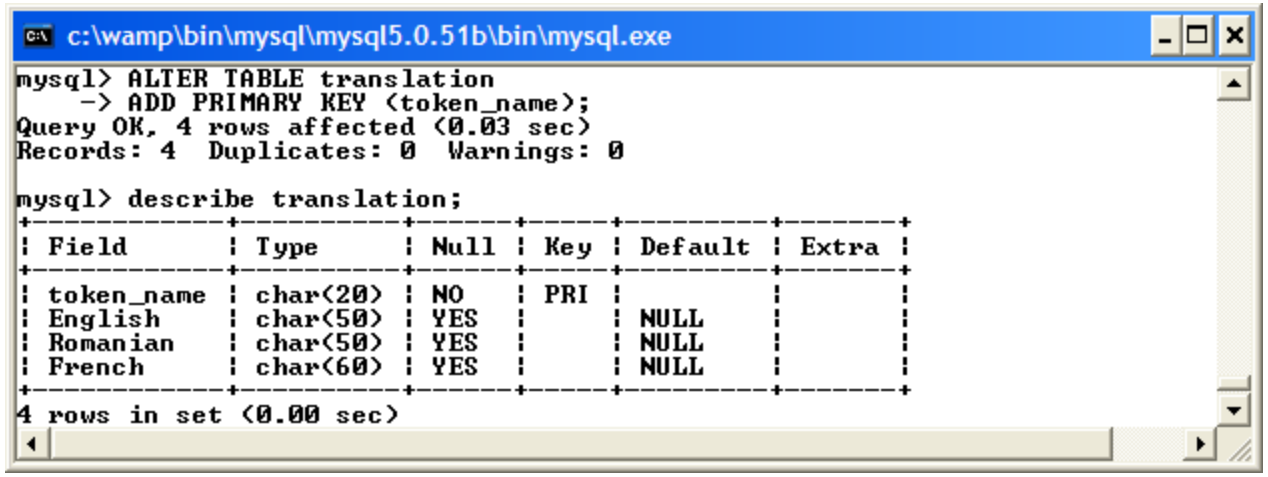
Cheia straina va face referinta la o cheie primara dintr-o alta tabela (uneori, poate fi referinta la cheia primara din propria tabela).

Cum functioneaza aceasta contrangere privind integritatea referintei: dupa definirea unei chei straine, SQL va garanta ca in coloana definita cheie straina nu vor putea fi introduse valori nenule diferite de valorile din coloana referita (cheia primara);

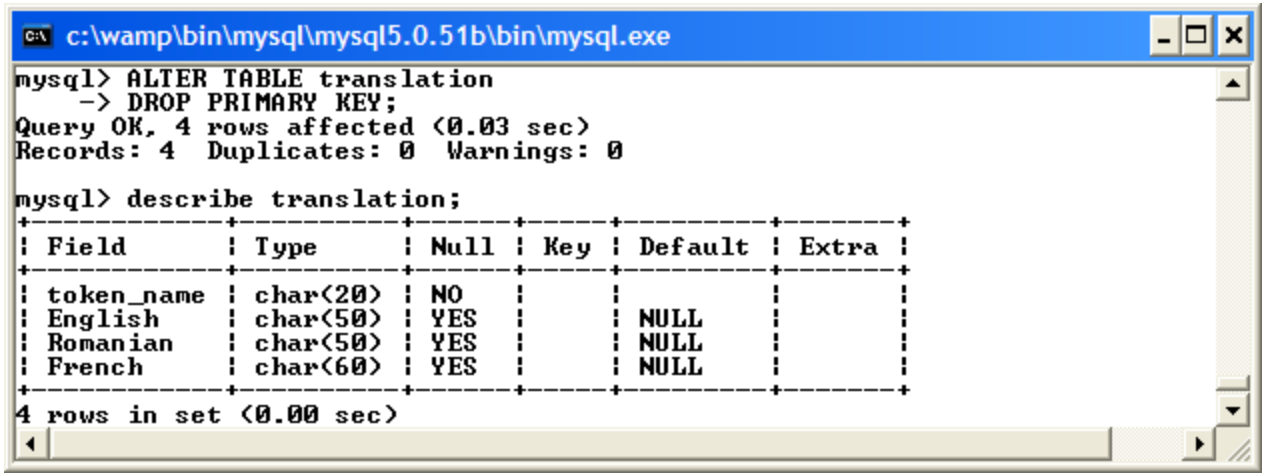
SQL DDL

Exemple:

Adaugarea unei chei



Stergerea unei chei



SQL DDL

Actiunea referentiala

Actiunile referentiale restrang sau, dimpotriva, permit, cascadarea operatiilor de DELETE sau UPDATE pentru valorile coloanelor folosite in contrangerile referentiale (chei straine).

ON UPDATE RESTRICT

ON DELETE RESTRICT

ON UPDATE CASCADE

ON DELETE CASCADE

Exemple:

```
5  CREATE TABLE t1 (s1 INT UNIQUE PRIMARY KEY, s2 INT, s3 INT);
6  CREATE TABLE t2 (s1 INT, s2 INT,
7    FOREIGN KEY (s2) REFERENCES t1 (s1) ON UPDATE CASCADE);
8
```

SQL DDL

Storage engines (motoare de stocare):

Exemplu:

CREATE TABLE t (i INT) ENGINE = ‘engine_name’;

Motor	Limita stocare	Tranzactii	B-tree index	Hash-index	Granularitate blocare
MyISAM	256TB	NU	DA	NU	Tabela
InnoDB	64TB	DA	DA	DA	Inregistrare
MEMORY	RAM	NU	DA	DA	Tabela

De ce sa utilizam totusi engine-uri netranzactionale ?

-mult mai rapide;

-mai putina memorie necesara (RAM si HD);

SQL DDL

Indecsi

PRIMARY KEY
UNIQUE
INDEX
FULLTEXT

Majoritatea indecsilor in MySQL- implementati folosind B-trees, accesul fiind de cca. 100 de ori mai rapid decat accesul direct.

Indecsii pentru date spatiale folosesc R-trees

Tabelele de MEMORY folosesc hash-indecsi

Cand se folosesc B-tree index

- Pot fi folositi in comparatii in expresii care folosesc =,>, >=,<,<= sau **BETWEEN**;
- In comparatiile cu **LIKE** daca argumentul este un string constant fara "*"

Cand se folosesc Hash index

- Utilizati pentru comparatii cu = sau <=>
- Cand doar intreaga cheie poate fi folosita pentru cautari
- Cand MySQL nu poate determina cate inregistrari sunt intre 2 valori

SQL DDL

Crearea unui view

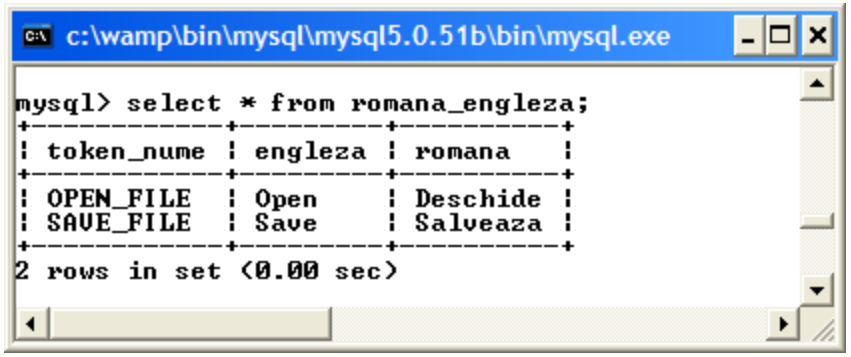
```
CREATE [OR REPLACE]
    [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    [DEFINER = { user | CURRENT_USER }]  [SQL SECURITY { DEFINER | INVOKER }]
    VIEW view_name [(column_list)]
    AS select_statement [WITH [CASCADED | LOCAL] CHECK OPTION]
```

Mai simplu:

```
CREATE VIEW view_name AS select_statement
```

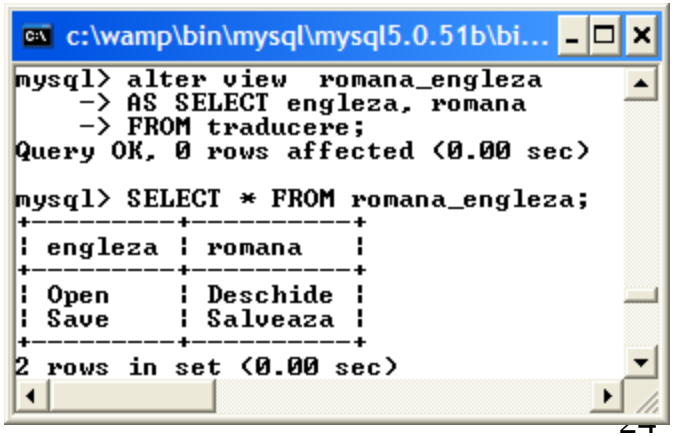
Exemplu:

```
CREATE VIEW romana_engleza
   AS SELECT token_nume, engleza, romana from traducere;
```



Modificarea unui view

```
ALTER
    [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    [DEFINER = { user | CURRENT_USER }]
    [SQL SECURITY { DEFINER | INVOKER }]
    VIEW view_name [(column_list)]
    AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```



Stergerea unui view

```
DROP VIEW [IF EXISTS]
      view_name [, view_name] ...
    [RESTRICT | CASCADE]
```