SQL Data Definition and Data Manipulation Languages (DDL and DML)

Data Definition Language.

Creating a Relation

```
CREATE TABLE Name (
    attribute-declarations
    constraint-declarations
);
```

Attribute declarations:
AttName AttType [default *expression*] [*ColConstraints*]

Constraints

Column constraints:

`[constraint <ConstName>] [NOT] NULL :` Not null constraint.

`[constraint <ConstName>] PRIMARY KEY:` Primary key constraint (when the primary key consists of exactly one attribute, otherwise, use constraint declaration).

`[constraint <ConstName>] UNIQUE :` Key constraint (when the key consists of exactly one attribute, otherwise, use constraint declaration).

`[constraint <ConstName>] REFERENCES <Table>(<AttName>) [ON DELETE CASCADE]:` Foreign key constraint (when the foreign key consists of exactly one attribute, otherwise, use constraint declaration). `ON DELETE CASCADE` specifies that all rows containing a no longer existing value for must be deleted.

`[constraint <ConstName>] CHECK (<condition>):` any additional constraint on the value of the element in the table[1].

Constraint declarations:

`[constraint <ConstName>] PRIMARY KEY (<AttNames>):` Primary key constraint. Use when the primary key includes multiple attributes.

[1] Some SQL implementations/engines will ignore the CHECK constraint. Make sure to check if your specific configuration acknowledges it.

`[constraint <ConstName>] UNIQUE (<AttNames>):` Key constraint. Use when the key includes multiple attributes.

`[constraint <ConstName>] FOREIGN KEY (<AttNames>) REFERENCES <Table> (<AttNames>):` Foreign key constraint. Use when the foreign key involve multiple attributes.

All column constraints except for not null constraint can only be used if the appropriate constraint (e.g., primary key) is associated with exactly one attribute. (i.e., if your primary key is two attributes, use the constraint declaration, rather than column constraint).

Types

Integer	`INTEGER` or `INT`
	`SHORTINT`
Real	`FLOAT` or `REAL`
	`DOUBLE PRECISION`
Fixed Point	`DECIMAL`(n, d)
	n - number of digits
	d - number of decimals
	`NUMBER`(n, d) (Oracle)
Strings	`CHAR`(n)
	n - length of string, max=255
	`VARCHAR`(n),
	`VARCHAR2(n)` (Oracle)
	n - length of string, max = 2000
Bit Strings	`BIT`(n)
	`BIT VARYING`(n)
Boolean	`BOOLEAN`
	values: `TRUE`, `FALSE`, `UNKNOWN`
Dates	`DATE`
	formatted as a string, converted to INT internally
	default format: 'DD-MON-YEAR'. e.g., '12-APR-2007'
Blob	`BLOB`
	`TEXT`

Full Reference

The complete definition for CREATE TABLE is actually quite long. For a full reference on MySQL's CREATE TBALE command, see: `http://dev.mysql.com/doc/refman/5.1/en/create-table.html`

Examples

```
CREATE TABLE Books (
    libCode INT,
    isbn CHAR(20),
    title CHAR(128),
    authors CHAR(64),
    year INT,
    publisher CHAR(32),
    purchPrice REAL,
    takeHome BOOLEAN,
    PRIMARY KEY(libCode),
    UNIQUE(isbn)
);

CREATE TABLE Employees (
    ssn INT PRIMARY KEY CHECK(ssn > 0),
    name CHAR(32) NOT NULL,
    department INT REFERENCES Departments,
    salary FLOAT NOT NULL CHECK(salary >= 20000.00),
    position CHAR(32) DEFAULT 'Not Specified',
    startYear INT CHECK(startYear > 1992)
);

CREATE TABLE Departments (
    deptID INT PRIMARY KEY,
    name CHAR(32) UNIQUE,
    head INT CHECK(head > 0),
    FOREIGN KEY(head) REFERENCES Employees(ssn)
);
```

Deleting a Table

`DROP TABLE` *Name* [CASCADE]

Example:

```
DROP TABLE Books;
```

Be aware, that referential integrity constraints (foreign keys) may prevent a table from being dropped in the incorrect order. If you have circular constraints (it is possible), then you will have to modify one of the tables to first remove the constraint before dropping the table.

Modifying a Table

- Adding an attribute

  `ALTER TABLE` *Name*
  `ADD` ([*AttName Type*]+)

  Example:

```
ALTER TABLE Books
ADD (
    genre CHAR(10),
    numPages INT
);
```

- Deleting an attribute

  `ALTER TABLE` *Name*
  `DROP` (*AttName+*)

  Example:

```
ALTER TABLE Books
DROP (year);
```

- Modifying an attribute

  `ALTER TABLE` *Name*
  `MODIFY` ([*AttName Type*]+)

  Example:

```
ALTER TABLE Books
MODIFY (genre VARCHAR(32));
```

There are **MANY** different ways that you can modify a table. See the complete MySQL reference for more details: `http://dev.mysql.com/doc/refman/5.1/en/alter-table.html`

Data Manipulation Language

Inserting a Tuple

`INSERT INTO` *TableName(AttNames)*
`VALUES(`*values*`)`

`values` — comma-separated list of values. The number of values must match the number attribute names in *AttNames*, and the types must be compatible.

`INSERT INTO` *TableName*
`VALUES(`*values*`)`

Values for all attributes must be given and in the order in which attributes were defined in `CREATE TABLE` command.

Examples:

```
INSERT INTO Books(libCode, title, year)
VALUES (12349, 'Database Management Systems', 2000);
```

```
INSERT INTO Books
VALUES (15923, '1-56592-000-7', 'Lex \& Yacc',
        'J. Levine, T. Mason, D. Brown',
        1990, 'O''Reily', 29.95, True);
```

Note that a single quote (') is escaped by using a second single quote as in the above example.

Deleting Tuples

`DELETE FROM` *TableName*
`WHERE` *Expression*

Expression identifies the properties of tuples to be removed from the table.

Examples:

```
DELETE FROM Books
WHERE year < 1950;

DELETE FROM Books
WHERE libCode = 12349;

DELETE FROM Books
WHERE
    purchPrice > 100.00
   AND year < 1950;
```

Updating Tuples

`UPDATE` *TableName*
`SET` *Assignments*
`WHERE` *Expression*

Expression identifies tuples to be updated.
Assignments specifies modifications.

Examples:

```
UPDATE Books
SET year = 2003
WHERE year > 2003;

UPDATE Books
SET
    year = year - 1,
    purchPrice = purchPrice * 1.05
WHERE year > 2000;
```

```
UPDATE Books
SET takeHome = TRUE;
```