

## 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.

Constraint declarations:

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

[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 or DOUBLE PRECISION
Fixed Point	DECIMAL( <i>n</i> , <i>d</i> ) <i>n</i> - number of digits <i>d</i> - number of decimals NUMBER( <i>n</i> , <i>d</i> ) (Oracle)
Strings	CHAR( <i>n</i> ) <i>n</i> - length of string, max=255 VARCHAR( <i>n</i> ), VARCHAR2( <i>n</i> ) (Oracle) <i>n</i> - length of string, max = 2000
Bit Strings	BIT( <i>n</i> ) BIT VARYING( <i>n</i> )
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'

---

## Examples

```
CREATE TABLE Books (  
    LibCode    INT,  
    ISBN      CHAR(20),  
    Title     CHAR(80),  
    Authors   CHAR(60),  
    Year      INT,  
    Publisher  CHAR(20),  
    PurchPrice REAL,  
    TakeHome  BOOLEAN,  
    PRIMARY KEY (LibCode),  
    UNIQUE (ISBN)  
);
```

```
CREATE TABLE Employees (  
    SSN INT PRIMARY KEY CHECK(SSN > 0),  
    Name CHAR(30) NOT NULL,  
    Department INT REFERENCES Departments,  
    Salary FLOAT NOT NULL CHECK(Salary >= 20000.00)  
    Position CHAR(30) DEFAULT 'Not Specified',  
    StartYear INT CHECK(StartYear > 1992)  
);
```

```
CREATE TABLE Departments (  
    DeptID INT PRIMARY KEY,  
    Name CHAR(30) UNIQUE,  
    Head INT CHECK(Head > 0),  
    FOREIGN KEY(Head) REFERENCES Employees  
);
```

## Deleting a Table

```
DROP TABLE Name [CASCADE CONSTRAINTS]
```

Example:

```
DROP TABLE Books;
```

```
DROP TABLE Departments CASCADE CONSTRAINTS
```

In the latter case, all referential integrity constraints (foreign keys) are **dropped** from their respective tables, after **Departments** table is deleted.

## 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 VARCHAR2(30));

```

## 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);

```

### 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;
```