

DM_L - INSERT

MAD 202



INSERT

- Similar to the CREATE DDL statement because it creates a new object in the database, but while CREATE creates a new table, the INSERT creates a new row into an existing table.
- Several variations on the INSERT command
 - Insert row by using the column positions
 - Insert row by using column names
 - Insert rows from one table into another table
- It inserts ENTIRE rows.



INSERT statement

```
INSERT INTO tablename  
(column1, column2, column3,...)  
VALUES  
(value1, value2, value3....)
```

The columns in the row

The values



INSERT STATEMENT

```
INSERT INTO tablename  
(column1, column2, column3,...)  
VALUES  
(value1, value2, value3....)
```

- The first column will receive the first value.
- The second column will receive the second value....
- There has to be the same number of columns listed in the first list as there are values in the second list, or you will get an error
- Each inserted value MUST have the same data type or must be convertible to the same type as its corresponding column



INSERT STATEMENT

```
INSERT INTO tablename  
(column3, column2, column1,...)  
VALUES  
(value3, value2, value1....)
```

- Can do this BUT common practice is to list columns in order that they appear in the table.



INSERT STATEMENT

Insert by Column Positions -

```
INSERT INTO tablename  
VALUES  
(value1,value2,value3)
```

- This format is allowed ONLY when the values listed are in the matching order of the columns AND the number of values equal the number of columns.
- It is recommended that you do not use this format since the table may at some point be changed which can result in this method no longer working



INSERT SELECT

- Insert statements can also be used to insert information that is retrieved from another table
- We will take a closer look at this statement and its use when we start working with SELECT statements

```
INSERT INTO tablename  
SELECT  
    Column1,  
    Column2,  
    Column3  
FROM tablename2  
WHERE column = value;
```



INSERT STATEMENT in action

```
CREATE TABLE teams  
(  
    team_id INTEGER NOT NULL PRIMARY KEY,  
    team_name VARCHAR(30) NOT NULL,  
    city    VARCHAR(20) NOT NULL,  
    conference VARCHAR(10) NOT NULL  
) ;
```



INSERT

The process of adding information is called populating the table

```
INSERT INTO teams
  (team_id, name, city, conference)
VALUES
  (1, 'Red Wings', 'Detroit', 'Eastern');
```



INSERT

- INSERT INTO teams
VALUES
(2, 'Canadiens', 'Montreal', 'Eastern');



INSERT

```
INSERT INTO teams
```

- VALUES
- (2, 'Capitals', 'Washington');

Error

SQL query:

```
INSERT INTO teams
(2, 'Capitals', 'Washington')
```

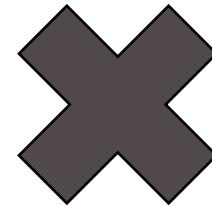
MySQL said: 

```
#1064 - You have an error in your SQL syntax; check the manual that corresponds to your MariaDB
server version for the right syntax to use near '2, 'Capitals', 'Washington')' at line 2
```



INSERT

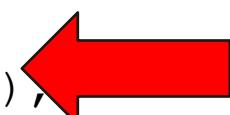
```
INSERT INTO teams  
  (team_name, city, team_id, conference)  
VALUES  
  (2, 'Capitals', 'Washington', 'Eastern');
```



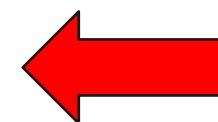
```
INSERT INTO teams  
  (team_name, city, team_id, conference)  
VALUES  
  ('Capitals', 'Washington', '3', 'Eastern');
```



INSERT multiple

```
INSERT INTO teams
(team_id, team_name, city, conference)
VALUES
(4, 'Maple Leafs', 'Toronto', 'Eastern'), MySQL
(5, 'Blackhawks', 'Chicago', 'Western');
```

```
INSERT INTO teams
(team_id, team_name, city, conference)
VALUES
(4, 'Maple Leafs', 'Toronto', 'Eastern')
;
INSERT INTO teams
(team_id, team_name, city, conference)
VALUES
(5, 'Blackhawks', 'Chicago', 'Western')
;
```

 SQL Server

Additional Examples

```
CREATE TABLE superheroes
(
    hero_id INTEGER NOT NULL PRIMARY KEY,
    hero_name VARCHAR(30) NOT NULL,
    secret_identity VARCHAR(30) NOT NULL,
    sidekick VARCHAR(30) NOT NULL DEFAULT 'None',
    age SMALLINT NULL
);
```



Knowing your columns..

```
INSERT INTO superheroes
(hero_id, hero_name, secret_identity)
VALUES
(1, 'Superman', 'Clark Kent');
```



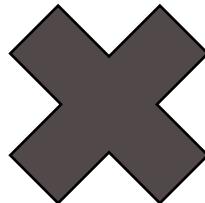
Shortened insert..

```
INSERT INTO superheroes  
VALUES  
(2, 'Batman', 'Bruce Wayne', 'Robin', 35);
```



Will this work?

```
INSERT INTO superheroes  
VALUES  
(3, 'Wonder Woman');
```



???

```
INSERT INTO superheroes  
(hero_id, hero_name, secret_identity)  
VALUES  
(3 , 'Wonder Woman', 'Linda Carter');
```



Inserting information into primary keys

- When inserting information into our sample tables we provided the primary key values
- These values we obtained by looking into the table to determine what the next available value would be
- In a real world application we would not want to first search every table to insert information
- Assign the surrogate key autonumber to the primary key column
- Value will automatically increase



ID's and auto incrementing

```
CREATE TABLE counting
(
    id INTEGER AUTO_INCREMENT NOT NULL PRIMARY KEY,
    name VARCHAR(10)
);
```

```
INSERT INTO counting
(name)
VALUES
('first');
```

```
INSERT INTO counting
(id, name)
VALUES
(3, 'second').....
```

What happens if you delete a record and then add another?



INSERTING into tables with a foreign key relationship

```
CREATE TABLE authors
(
    id INTEGER AUTO_INCREMENT NOT NULL PRIMARY KEY,
    fname VARCHAR(20) NOT NULL,
    lname VARCHAR(25) NOT NULL
);

CREATE TABLE books
(
    id INTEGER AUTO_INCREMENT NOT NULL PRIMARY KEY,
    title VARCHAR(25),
    author_id INTEGER NOT NULL,
    CONSTRAINT author_fk
        FOREIGN KEY (author_id)
        REFERENCES authors(id)
);
```



INSERT some authors

```
INSERT INTO authors
```

```
    (fname, lname)
```

```
VALUES
```

```
    ('Dean', 'Koontz'),
```

```
    ('Jim', 'Butcher'),
```

```
    ('Harlen', 'Coben'),
```

```
    ('Jim', 'Delaney')
```

```
INSERT INTO books
```

```
    (title, author_id)
```

```
VALUES
```

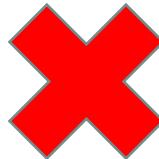
```
??????
```



INSERT some books

```
INSERT INTO books  
(title, author_id)  
VALUES  
( 'Watchers' , 1 );
```

```
INSERT INTO books  
(title, author_id)  
VALUES  
( 'Booky' , 6 );
```



```
INSERT INTO books  
(title, author_id)  
VALUES  
( 'Side Jobs' , 2 );
```



Preserving referential integrity

- The DBMS will **NOT** allow you to create orphan rows or make existing rows orphans
- When you INSERT, UPDATE, DELETE a row with a FOREIGN KEY column that references a PRIMARY KEY column in a parent table the following are checked:



Foreign-key table changes

- Inserting a row into the foreign-key table
 - The system checks that the new FOREIGN KEY value matches a PRIMARY KEY value in the parent table. If no match exists, the INSERT will not be allowed

```
INSERT INTO books  
(title, author_id)  
VALUES  
(‘Booky’,6)
```

- Updating a row in the foreign-key table
 - The system checks that the updated FOREIGN KEY value matches a PRIMARY KEY value in the parent table. If no match exists, the UPDATE will not be allowed

```
UPDATE books  
SET title='New Book', author_id='6'  
WHERE id=1;
```

- Deleting a row in the foreign-key table.
 - No check is required



Parent table changes

- Inserting a row into the parent table
 - No check is required
- Updating a row in the parent table.
 - The system checks that none of the FOREIGN KEY values matches the PRIMARY KEY value to be updated. If a match exists, the UPDATE is not allowed

```
UPDATE authors
SET id=6
WHERE fname='Dean';
```
- Deleting a row from the parent table
 - The system checks that none of the FOREIGN KEY values matches the PRIMARY KEY values to be deleted. If a match exists, the DBMS is not allowed.

```
DELETE FROM authors
WHERE id=1;
```



Drop tables with relationships

To remove a table – have to remove the primary relationship

DROP TABLE authors;



DROP TABLE books;



DROP TABLE authors;

