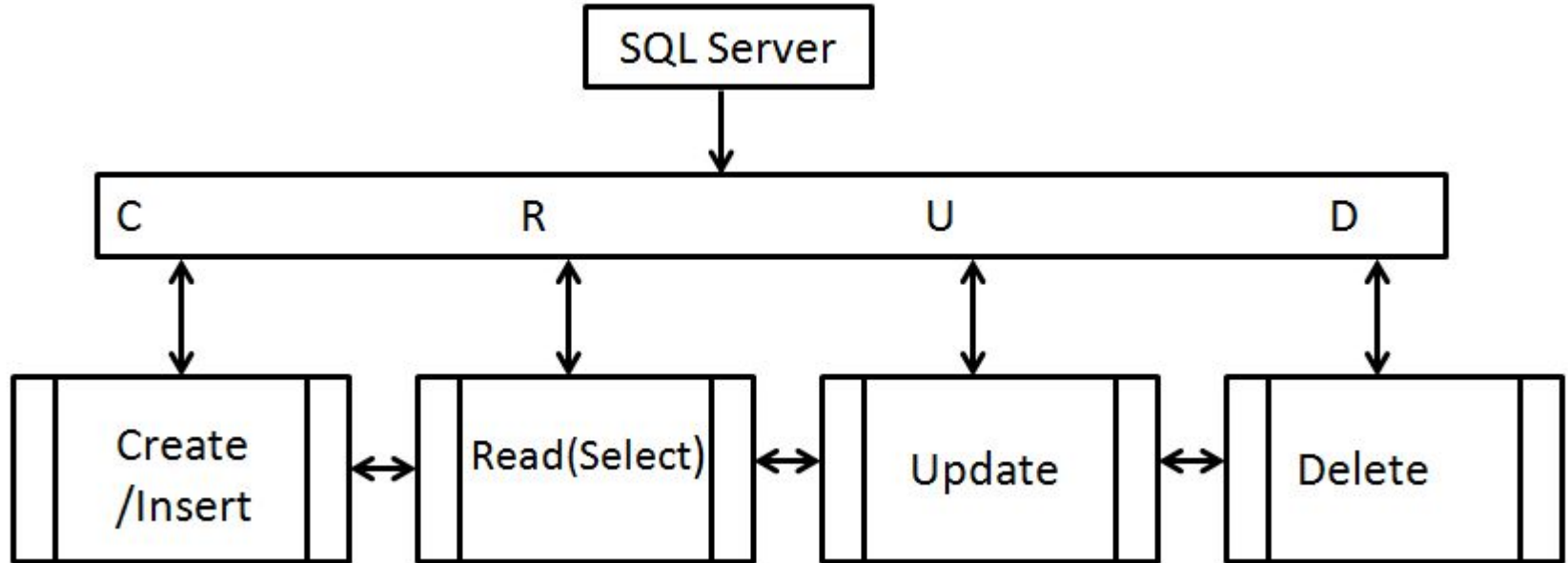


Module 2-4

INSERT, UPDATE, DELETE

SQL Operations: CRUD



Changing data

The row data for each table in a database can be changed or deleted. New rows of data can also be added. There are 3 types of statements we will cover today:

- **INSERT**: Adds a new row to the table.
- **UPDATE**: Changes the column value for an existing row or rows.
- **DELETE**: Permanently removes a row from the table.

INSERT statements

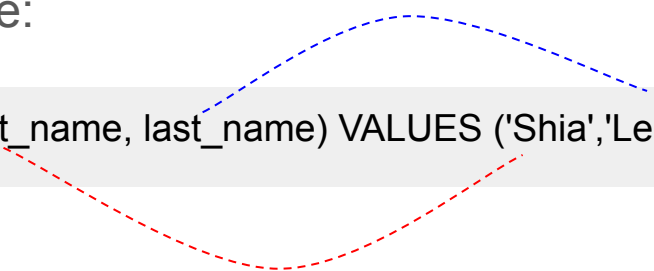
You can use the INSERT statement to insert a row into the database.
Here is the statement syntax used:

```
INSERT INTO [Name of Table] ([name of col 1], [name of col 2])  
  
VALUES ([value for col 1], [value for col2]);
```

INSERT statements example

Consider the following example:

```
INSERT INTO actor (first_name, last_name) VALUES ('Shia','LeBouf');
```



In English, this translates to insert a new row in the table actor, on this new row the value for first_name is going to be “Shia” and the value for the last_name is going to be “LeBouf”.

INSERT Statement Part II: What about the ID?

Note that in the previous example, we only specified two out of three columns and did not specify that a value be inserted for actor_id.

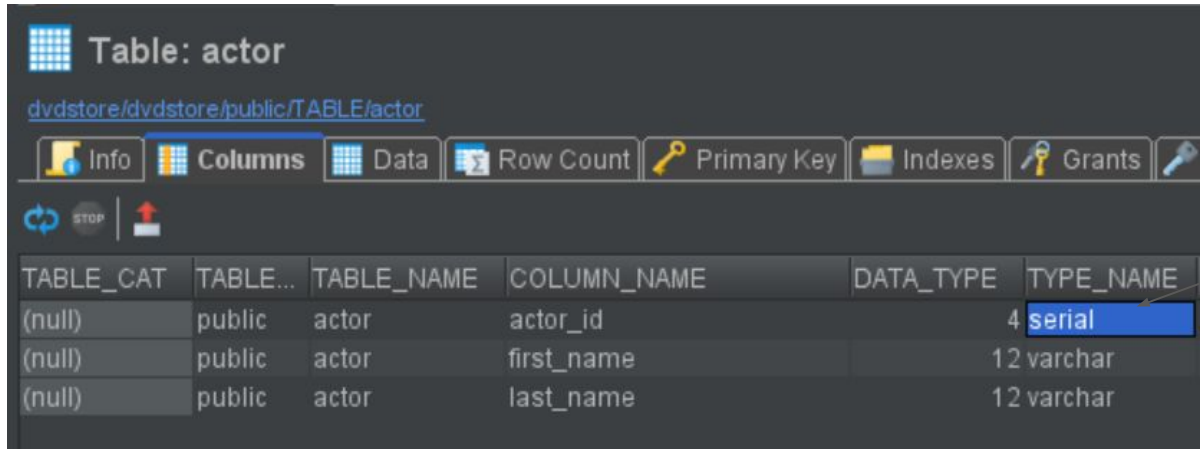


Table: actor

[dvdstore/dvdstore/public/TABLE/actor](#)

Info Columns Data Row Count Primary Key Indexes Grants

TABLE_CAT	TABLE...	TABLE_NAME	COLUMN_NAME	DATA_TYPE	TYPE_NAME
(null)	public	actor	actor_id	4	serial
(null)	public	actor	first_name	12	varchar
(null)	public	actor	last_name	12	varchar

- actor_id is of a special data type called **serial**.
- A column marked as serial will automatically increase in value with each new row.
- Columns marked as serial should not be included in the INSERT.

UPDATE statements

An update statement changes the values in specified columns.

- 1) It will execute against against the ALL rows in the table.
- 2) Updates are only limited by JOINS(more later) and WHERE clause filters!

UPDATE [table name]

SET [col 1 name] = <literal value, expression, or scalar sub-select>

WHERE ...

UPDATE: A Basic Example

Consider the following example:

```
UPDATE actor
SET
first_name = 'Nicholas',
last_name = 'Wahberg'
WHERE
actor_id = 2;
```

In the SET clause, we are changing the value for 2 columns (first_name and last_name)

We can separate multiple columns that need updating with a comma.

The syntax for structuring the WHERE statement remains unchanged. Here it is limiting the UPDATE to only the rows with an actor_id of 2.

DELETE statements

A delete statement removes row or rows from the table. It follows this format:

DELETE FROM [table name]

WHERE ...


Just as in the UPDATE statement, **the absence of a WHERE statement will affect every row in the database**; they will all be deleted!

DELETE statements example

Consider the following example.

```
DELETE FROM film_actor  
WHERE  
actor_id = 2;
```

Here, we are deleting every row that has an actor_id of 2.



Constraints

Constraints are rules imposed on the table, upon creation, that limits the ability to change the data.

- **NOT NULL:** A value must be specified
- **PRIMARY KEY:** Define that certain column/columns are part of the key
 - **A primary key value cannot be NULL.**
- **FOREIGN KEY:** Defines a foreign key based on a primary key from a different table
- **CHECK:** Only certain values can be inserted or updated

What Constitutes a Transaction: The ACID Test

A - Atomicity

All or Nothing Transactions

C - Consistency

Guarantees Committed Transaction State

I - Isolation

Transactions are Independent

D - Durability

Committed Data is Never Lost

(c) <http://blog.sqlauthority.com>

Transactions

A large number of SQL statements can be rolled into a single transaction.

The following syntax is observed:

BEGIN TRANSACTION;

// Lots of SQL statements.

COMMIT TRANSACTION;

Your INSERT or UPDATE SQL statements **will only commit (permanently save in the database) if all the SQL statements in the transaction end successfully.**