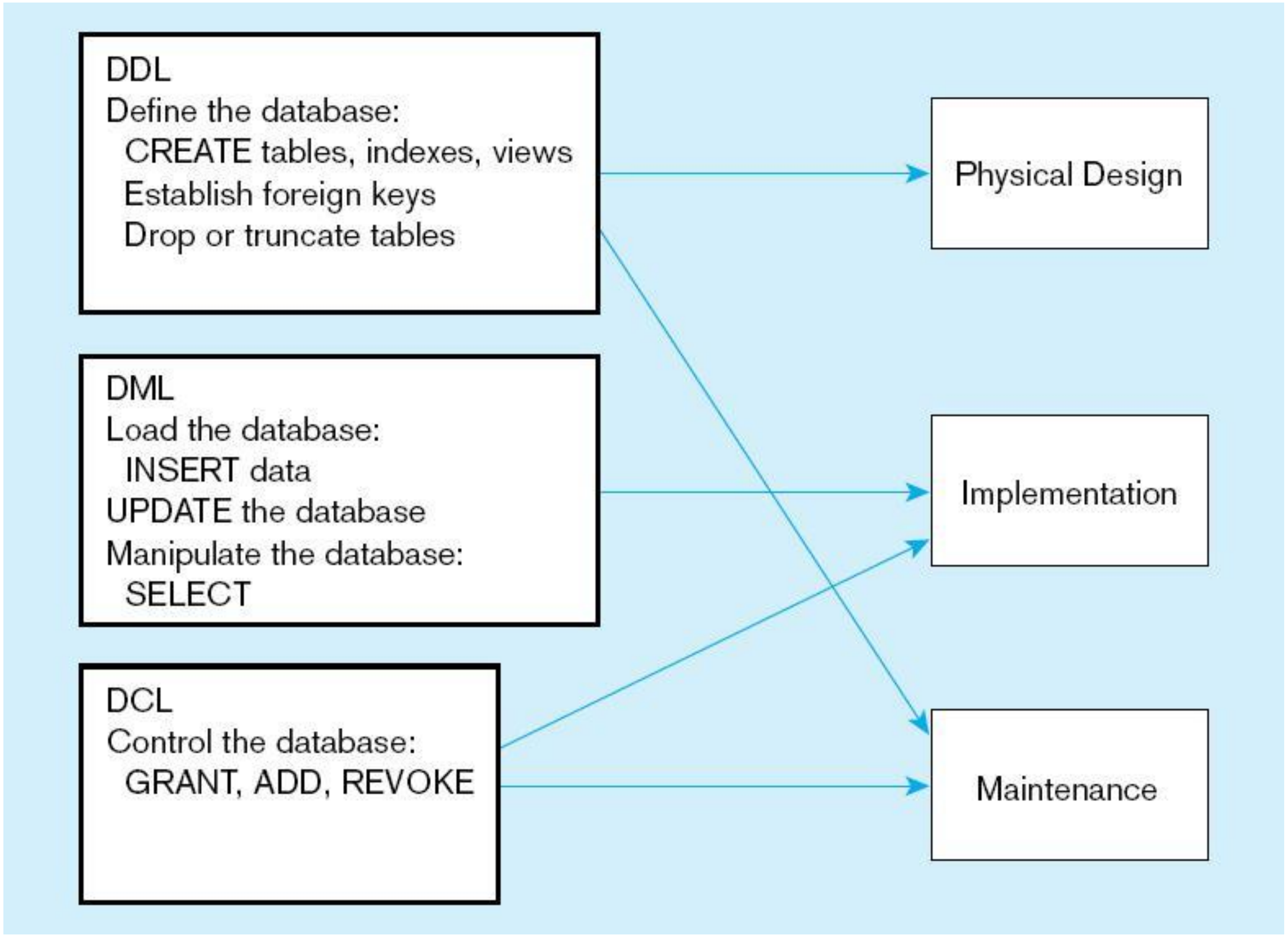
Modifying Records using Data Manipulation Language (DML)

Database System Concepts, 7th Ed.

DDL, DML (focus in this course), DCL, and the database development process



Modifying Records

Insert Statement

- **Insert:**
  - Allows you to add new records to the Table
- **Syntax:**
  - Insert into table_name[(column_list)] values (value_list)
- **Example:**

```
INSERT INTO studios
VALUES (1, ‘Giant’, ‘Los Angeles’, ‘CA’);

INSERT INTO studios
(studio_city, studio_state, studio_name, studio_id)
VALUES (‘Burbank’, ‘CA’, ‘MPM’, 2);
```

- **Notes1:** If the columns are not specified as in the first example the data goes in the order specified in the table
- **Notes2:** There are two ways of inserting Null values
  1. If the field has a default value of Null, you can use an Insert statement that ignores the column where the value is to be Null.
  2. You can specify the column in the column list specification and assign a value of Null to the corresponding value field.

Modifying Records

Select & Insert

- **Select & Insert:**
  - A select query can be used in the insert statement to get the values for the insert statement

- **Example:**

```
INSERT INTO city_state
SELECT studio_city, studio_state FROM studios
```

- This selects the corresponding fields from the studios table and inserts them into the city_state table.

- **Example:**

```
INSERT INTO city_state
SELECT Distinct studio_city, studio_state FROM studios
```

- This selects the corresponding fields from the studios table, deletes the duplicate fields (because of distinct) and inserts them into the city_state table. Thus the final table has distinct rows

Insert Example

```
CREATE TABLE clients (
    client_id NUMBER GENERATED BY DEFAULT AS IDENTITY,
    first_name VARCHAR2(50) NOT NULL,
    last_name VARCHAR2(50) NOT NULL,
    company_name VARCHAR2(255) NOT NULL,
    email VARCHAR2(255) NOT NULL UNIQUE,
    phone VARCHAR(25));

INSERT INTO clients(first_name,last_name, email, company_name, phone)
VALUES('Christene','Snider','christene.snider@abc.com', 'ABC Inc', '408-875-
6075’);

INSERT INTO clients(first_name,last_name, email, company_name, phone)
VALUES('Sherly','Snider','christene.snider@abc.com', 'ABC Inc', '408-875-
6076’);
```

→ Gives ERROR (Unique constraint is violated)

Alter Table

- Table constraints can be altered using ALTER TABLE clause

E.g.

ALTER TABLE clients

ADD CONSTRAINT unique_company_phone UNIQUE(company_name, phone);

Now combination of of company_name and phone number is unique

INSERT INTO clients(first_name,last_name, email, company_name, phone)

VALUES('Sherly','Snider','christene.snider@abc.com', 'ABC Inc', '408-875-6076'); → Executed!

Enable/Disable/Drop Constrains Using Alter Table

To disable the unique constraint in the previous example such as unique_company_phone, you use the following statement:

Syntax:

ALTER TABLE table_name

DROP/DISABLE/ENABLE CONSTRAINT constraint_name;

To disable:

ALTER TABLE clients

DISABLE CONSTRAINT unique_company_phone;

And to enable it:

ALTER TABLE clients

ENABLE CONSTRAINT unique_company_phone;

Or to drop it permanently:

ALTER TABLE clients

DROP CONSTRAINT unique_company_phone;

Drop/Enable/Disable Primary Key Using Alter Table

- You can drop/enable/disable a primary key using the ALTER TABLE statement.

Syntax:

ALTER TABLE table_name

DROP/DISABLE/ENABLE CONSTRAINT constraint_name;

Ex.

ALTER TABLE supplier

DROP CONSTRAINT supplier_pk;

Ex.

ALTER TABLE supplier

DISABLE CONSTRAINT supplier_pk;

Ex.

ALTER TABLE supplier

ENABLE CONSTRAINT supplier_pk;

Modifying Records

More Alter Statements

- **Alter Statements:**
  - used to make changes to the schema of the table. Columns can be added and the data type of the columns changed as long as the data in those columns conforms to the data type specified.

- **Syntax:**

ALTER TABLE table_name
ADD (column datatype [Default Expression])
[REFERENCES table_name (column_name)’
[CHECK condition]

- **Example:**

ALTER TABLE studios
ADD (revenueNumber INTEGER DEFAULT 0); → Adds a new column to studios table

Modifying Records

Alter Statement

Add table level constraints using alter table statement:

- **Syntax:**

ALTER TABLE table_name
ADD ([CONSTRAINT constraint_name CHECK
    comparison]
[columns REFERENCES table_name (columns)]

- **Example:**

ALTER TABLE studios
ADD (CONSTRAINT check_state CHECK (studio_state in
    (‘TX’, ‘CA’, ‘WA’));

Modifying Records

Alter Statement

Modify Columns:

- **Syntax:**

ALTER TABLE table_name
MODIFY column [data type]
[Default Expression]
[REFERENCES table_name (column_name)’
[CHECK condition]

- **Example:**

ALTER TABLE People
MODIFY person_union varchar(10) NOT NULL; → modifies the datatype and not null value

- **Notes1:** Columns can not be removed from the table using alter. If you want to remove columns you have to drop the table and then recreate it without the column that you want to discard

Modifying Records

Alter Statement

- **Modify Column Visibility:**

```
ALTER TABLE accounts
MODIFY password VISIBLE;

ALTER TABLE accounts
MODIFY password INVISIBLE;
```

Modifying Records

Delete Statement

- **Delete Statement:**
  - is used to remove records from a table of the database. The where clause in the syntax is used to restrict the rows deleted from the table otherwise all the rows from the table are deleted.

- **Syntax:** DELETE FROM table_name [WHERE Condition]

- **Example:**

```
DELETE FROM City_State
WHERE state = ‘TX’
```

- Deletes all the rows where the state is Texas keeps all the other rows.

Modifying Records

Update Statement

- **Update Statement:**
  - used to make changes to existing rows of the table. It has three parts. First, you ,must specify which table is going to be updated. The second part of the statement is the set clause, in which you should specify the columns that will be updated as well as the values that will be inserted. Finally, the where clause is used to specify which rows will be updated.

- **Syntax:**

```
UPDATE table_name
SET column_name1 = value1,  column_name2 = value2, …..
[WHERE Condition]
```

- **Example:**

```
UPDATE studios
SET studio_city = ‘New York’, studio_state = ‘NY’
WHERE studio_id = 1
```

- **Notes1**: If the condition is dropped then all the rows are updated. (WHERE CLAUSE)

Modifying Records

Truncate Statement

- **Truncate Statement:**
  - Used to delete all the rows of a table. Delete can also be used to delete all the rows from the table. The difference is that delete performs a delete operation on each row in the table and the database performs all attendant tasks on the way. On the other hand the Truncate statement simply throws away all the rows at once and is much quicker. The note of caution is that truncate does not do integrity checks on the way which can lead to inconsistencies on the way. If there are dependencies requiring integrity checks we should use delete.

- **Syntax:** TRUNCATE TABLE table_name

- **Example:**

TRUNCATE TABLE studios

- This deletes all the rows of the table studios

Modifying Records

Drop Statement

- **Drop Statement:**
  - used to remove elements from a database, such as tables, indexes or even users and databases. Drop command is used with a variety of keywords based on the need.

- **Drop Table Syntax:** DROP TABLE table_name
- **Drop Table Example:** DROP TABLE studios

- **Drop Index Syntax:** DROP INDEX table_name
- **Drop Index Example:** DROP INDEX movie_index

W3C SQL Tutorial

- Please look at more examples regarding alter statements from

https://www.w3schools.com/sql/sql_alter.asp