CSC-430(001) / 530: DATABASE MANAGEMENT SYSTEMS / DATABASE THEORY

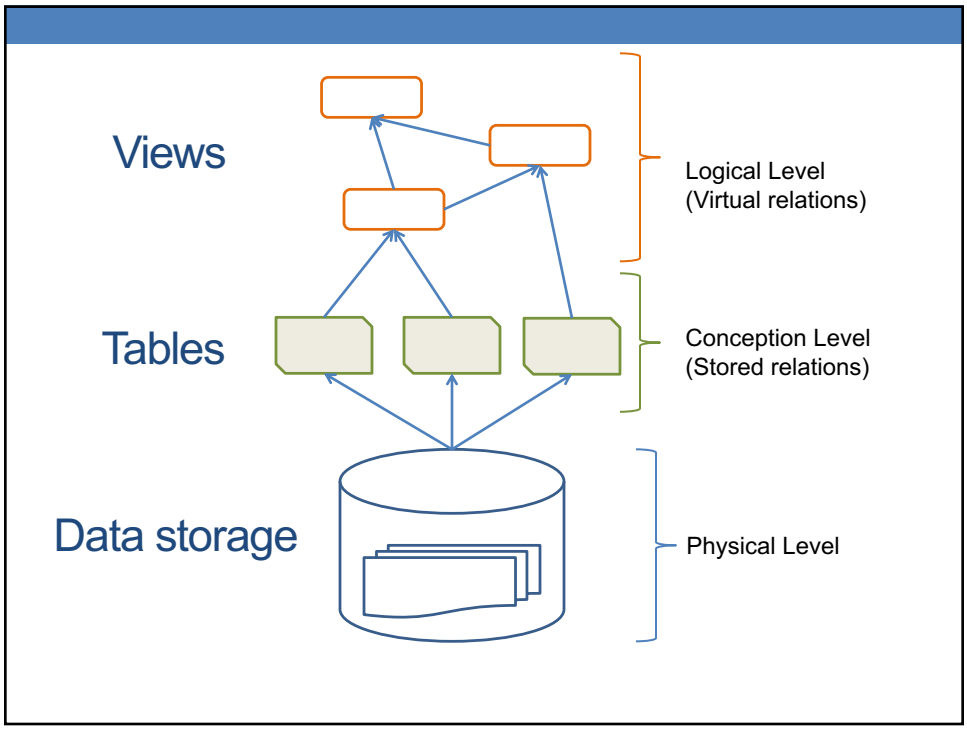
Lab 4: Introduction to Views and Triggers

1

A - Uses of views?

- To hide some data from some users.
- To make queries easier to execute by creating a virtual relations.
- For creating modules of the database and giving access to only targeted users.

2



3

Defining a View

CREATE VIEW **viewname** as
< **Select Query** >;

CREATE VIEW **viewname** (attr1, attr2)
as < **Select** (attr1, attr2) **Query** >;

ALTER VIEW **viewname** (attribute1)
as < **Select** (attribute1) **Query** >;

DROP VIEW **viewname;**

4

Modifications of views (INSERT, DELETE, UPDATE)

A View is not updatable if it contains any of the following:

- Aggregate functions (SUM(), MIN(), MAX(), COUNT())
- DISTINCT
- GROUP BY
- HAVING
- UNION or UNION ALL
- Subquery in the select list
- Non-updatable view in the FROM clause

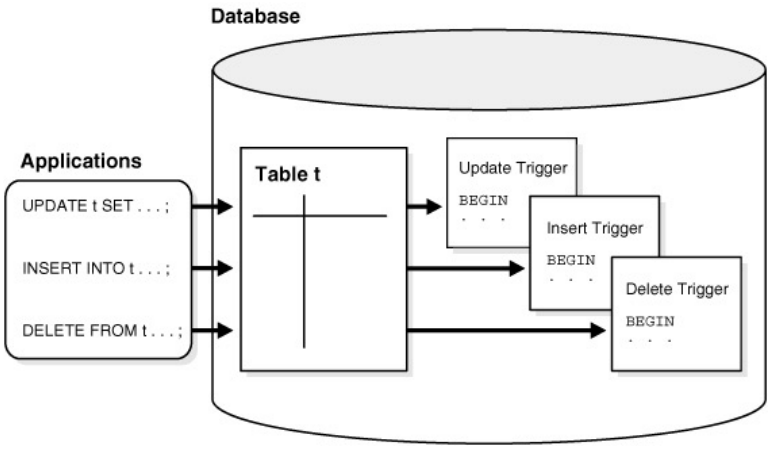
5

B - Why Learn Triggers?

- Triggers allow specified actions to be performed automatically within the database, without having to write any extra application code.
- Triggers increase the power of the database, and the power of your application.
- Triggers allow for complex application logic to be executed while being "close" to the data.

6

Triggers – close to "data"



- verify any anomalies that pertain to operations of insert, delete, and update

7

Possible Uses for Triggers

- Enhance **complex database security** rules
- Create **auditing and logging** of records automatically
- Prevent **tables from being accidentally dropped**
- Prevent **invalid DML transactions** from occurring
- Enforce **data integrity rules**

8

Trigger Syntax

- General form:
  1. ***Event*** - request to execute database **operation**
  2. ***Condition*** - predicate evaluated on database state
  3. ***Action*** - execution of procedure that might involve database updates

ON <*event*> IF <*condition*> THEN <*action*>

- **Example:**

```
ON <updating maximum course enrollment>
IF <number registered is greater than new
    max enrollment limit>
THEN <deregister students>
```

10

Trigger Syntax (cont…)

- **Events:** INSERT, DELETE, or UPDATE (alter) statements or changes to individual rows caused by these statements
- **Condition:** Anything that is allowed in a WHERE clause
- **Action:** An individual SQL statement or stored procedures

11

Trigger Syntax (cont…)

- **Consideration**:
  - Conditions can refer to the state of the affected row or table **BEFORE** *or* **AFTER** the event occurs
- **Notation**:
  - The reference to the tuple used in the transaction, before or after the execution of the triggering event is specific to the operation

INSERT	DELETE	UPDATE
NEW		NEW
	OLD	OLD

  - Use **SET** operator to modify fields of table that are activated the trigger.
- **Granularity**:
  - Both ***row-level*** and ***statement-level***

12

HANDS ON # 1

Write a trigger to **UPDATE** the address of a student with the 'Ruston' **BEFORE INSERT'**-ing the record in the students table **IF** the address is null.

```
DELIMITER $$

CREATE TRIGGER 'newdb2'.'STUDENT_BEFORE_INSERT'
BEFORE INSERT ON STUDENT
FOR EACH ROW
BEGIN
    IF (NEW.SADDRESS is null) THEN

        SET New.SADDRESS = 'Ruston';

    END IF;
end$$

DELIMITER ;
```

14

HANDS ON # 2

Write a trigger to enforce the following constraint :
Bank clients must be at least 18 years old,
If Age is lesser than 18, the client's information is discarded.

```
CREATE TABLE CLIENT (
  clientID INT NOT NULL ,
  name VARCHAR(45) NULL ,
  Age INT NULL ,
  PRIMARY KEY (`clientID`) );


INSERT INTO client (`clientID`, `Name`, `Age`) VALUES (1,
'client1', 21);

INSERT INTO client (`clientID`, `Name`, `Age`) VALUES (3,
'client3', 18);
```

15

HANDS ON # 2

Write a trigger to enforce the following constraint :
Bank clients must be at least 18 years old,
If Age is lesser than 18, the client's information is discarded.

```
DELIMITER $$

CREATE TRIGGER client_DateOfBirt
BEFORE INSERT on client
FOR EACH ROW
BEGIN
    declare msg varchar(255);
    IF (NEW.Age < 18 ) THEN
       /* Cause Error Message */
        set msg = 'Invalid Date of Birth';
        signal sqlstate '45000' set message_text = msg;
    END IF;
END
$$

DELIMITER ;
```

16

HANDS ON # 3

Write a trigger to keep track of password changes.

```
CREATE  TABLE users (
   username VARCHAR(45) NOT NULL ,
   email VARCHAR(45) NOT NULL ,
   password VARCHAR(45) NULL ,
   PRIMARY KEY (username,email) );


CREATE  TABLE users_logs (
   username VARCHAR(45) NOT NULL ,
   oldpassword VARCHAR(45) NULL ,
   newpassword VARCHAR(45) NULL);


insert into users values('user1','user1@mail.com',123);
```

17

HANDS ON # 3

Write a trigger to keep track of password changes.

```
DELIMITER $$

CREATE TRIGGER track_password
AFTER UPDATE ON Users
FOR EACH ROW
BEGIN
    IF(New.password <> Old.password) Then
        INSERT INTO users_logs VALUES(New.username,
Old.password, New.password);
    END IF;
end$$

DELIMITER ;


UPDATE users SET password='475' WHERE username='user1';
```

18