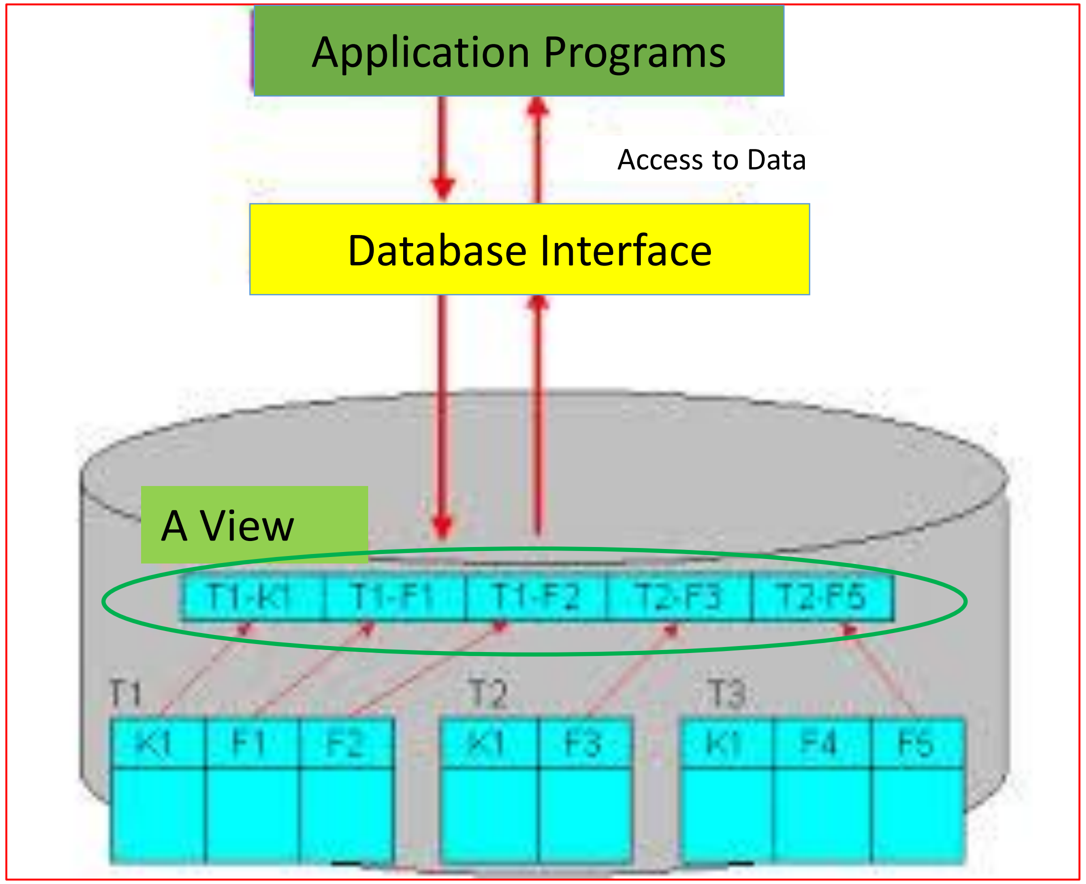
CPSC 5021: Database Systems

Views, Triggers

Lin Li

View



A view is created by applying a relational algebra expression to the base relations (tables).

A view is a virtual relation (table). No physical table is stored in the database for a view. The actual data are stored in base relations.

View

- Example:
  - ✓ Base relations

    -- Students(studentID, firstName, lastName, gender, dateofBirth)

    -- Courses(courseCode, courseName, level, credits)

    -- Registration(studentID, courseCode, grade)
  - ✓ A new table

    -- Course_Info(courseCode, numberofEnrollment)

After Course_Info is created, insert a new record in Registration. Will Course_Info be updated automatically?

How to create this table?

NO!

```
CREATE TABLE Course_Info AS SELECT courseCode, count(distinct studentID) AS
numberofEnrollment FROM Registration GROUP BY courseCode
```

View

- Example:
  - ✓ Base relations

    -- Students(studentID, firstName, lastName, gender, dateofBirth)

    -- Courses(courseCode, courseName, level, credits)

    -- Registration(studentID, courseCode, grade)
  - ✓ A view

    -- Course_Enroll(courseCode, numberofEnrollment)

How to create this view?

After Course_Info is created, insert a new record in Registration. Will Course_Enroll be updated automatically?

YES!

```
CREATE VIEW Course_Enroll AS SELECT courseCode, count(distinct studentID) AS
numberofEnrollment FROM Registration GROUP BY courseCode
```

Create a View

- Example:

What happens if we drop table t?

```
CREATE TABLE t (qty INT, price INT);

INSERT INTO t VALUES(3, 50);

CREATE VIEW v AS SELECT qty, price, qty*price AS value FROM t;

SELECT * FROM v;
```

```
CREATE VIEW PRICEGT50  AS
        SELECT PORD_DESCRIPT, PORD_QOH, PROD_PRICE
        FROM PRODUCT
        WHERE PORD_PRICE > 50.00;

SELECT * FROM PRICEGT50;
```

Alter/Drop a View

- Example:

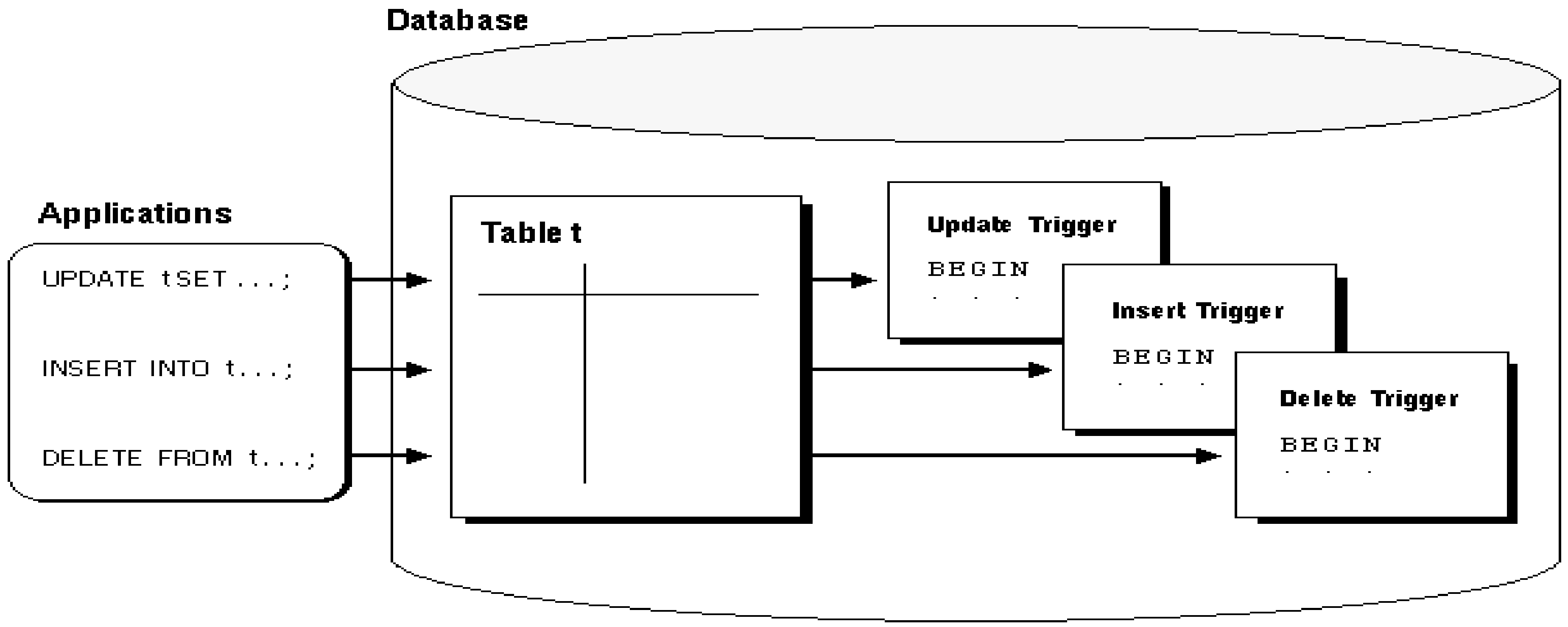
```
ALTER VIEW v AS SELECT qty, price FROM t;

SELECT * FROM v;
```

```
DROP VIEW v
```

Trigger

- A trigger is a series of actions that are implicitly executed when an INSERT, UPDATE or DELETE statement is issued against the associated table.



Trigger

- Syntax:

```
CREATE
   TRIGGER trigger_name trigger_time trigger_event
   ON tbl_name FOR EACH ROW trigger_stmt
```

-- trigger_time is the trigger action time. It can be BEFORE or AFTER to indicate that the trigger activates before or after the statement that activated it.

-- trigger_event indicates the kind of statement that activates the trigger

-- *trigger_stmt* is the statement to execute when the trigger activates. If you want to execute multiple statements, use the BEGIN ... END compound statement construct.

Trigger Event

- The trigger_event can be one of the following:

  -- **INSERT:** The trigger is activated whenever a new row is inserted into the table.

  -- **UPDATE:** The trigger is activated whenever a row is modified.

  -- **DELETE:** The trigger is activated whenever a row is deleted from the table.

- See details on

http://dev.mysql.com/doc/refman/5.7/en/create-trigger.html

Trigger

- Example:

```
CREATE TABLE  test1(a1 INT NOT NULL AUTO_INCREMENT PRIMARY KEY, b1 INT);
CREATE TABLE  test2(a2 INT NOT NULL AUTO_INCREMENT PRIMARY KEY, b2 INT);
CREATE TABLE  test3(a3 INT NOT NULL AUTO_INCREMENT PRIMARY KEY, b3 INT);
CREATE TABLE  test4(a4 INT NOT NULL AUTO_INCREMENT PRIMARY KEY, b4 INT DEFAULT 0);

DELIMITER |
CREATE TRIGGER testref BEFORE INSERT ON test1
     FOR EACH ROW BEGIN
        INSERT INTO test2 SET b2 = NEW.b1;
        DELETE FROM test3 WHERE b3 = NEW.b1;
        UPDATE test4 SET b4 = b4 + 1 WHERE a4 = NEW.b1;
      END;
|

delimiter ;
```

Trigger

- Example (Cont.)

```
INSERT INTO test3 (b3) VALUES (2), (4), (6), (8), (10), (12);

INSERT INTO test4 (a4) VALUES (0), (0), (0), (0), (0), (0), (0), (0), (0), (0);
```

Suppose that you insert the following values into table test1:

```
INSERT INTO test1(b1) VALUES (2), (3), (1), (7), (1), (8), (4), (4);
```

```
SELECT * FROM test1;
SELECT * FROM test2;
SELECT * FROM test3;
SELECT * FROM test4;
```

What is the output?

Try it!

OLD and NEW keywords

- In an INSERT trigger, only NEW.col_name can be used.
- In a DELETE trigger, only OLD.col_name can be used.
- In an UPDATE trigger, you can use OLD.col_name to refer to the columns of a row before it is updated and NEW.col_name to refer to the columns of the row after it is updated.

Drop a Trigger

- Syntax:

```
DROP TRIGGER [IF EXISTS] [schema_name.]trigger_name
```

- Example:

```
DROP TRIGGER testref
```

More Example

- Example:

```
CREATE TABLE account (acct_num INT PRIMARY KEY, amount DECIMAL(10,2));

CREATE TRIGGER ins_sum BEFORE INSERT ON account
FOR EACH ROW SET @sum = @sum + NEW.amount;

SET @sum = 0;
INSERT INTO account VALUES(137,14.98),(141,1937.50),(97,-100.00);
SELECT @sum AS 'Total amount inserted';
```

Exercise

- See in-class exercise handout