Assertions and Triggers in SQL
Assertions

• How to test if and FD is satisfied?

• ASSERTIONS:

CREATE ASSERTION assertion_name CHECK predicate

Example: We require that the sum of sailors and boats is smaller than 100. Here we need to specify a constraint between two tables:

CREATE ASSERTION SmallClub
CHECK ((SELECT COUNT(S.sid) FROM Sailors S) +
    (SELECT COUNT(B.bid) FROM Boats B) < 100)
Also

- You can do also that:
  
  ```sql
  CREATE TABLE Sailors(
    sid INTEGER,
    sname CHAR(10),
    rating INTEGER,
    age REAL,
    PRIMARY KEY (sid),
    CONSTRAINT smallClub
    CHECK( (SELECT COUNT (S.sid) FROM Sailors S)
    + (SELECT COUNT (B.bid) FROM Boats B) < 100 ));
  ```

  But it is wrong! Only checks Sailors, not Boats. Assertion is more appropriate.
Assertions

Constraint: A customer with a loan should have an account with at least 1000 dollars.

create assertion balance_constraint check (not exists (select * from loan L
   where not exists (select *
       from borrower B, depositor D, account A
       where L.loan_no = B.loan_no
       and B.cname = D.cname
       and D.account_no = A.account_no
       and A.balance >= 1000 )))
Another example

customer(customer_name, customer_street, customer_city)

Constraint: Customer city is always not null.
Can enforce it with an assertion:

**Create Assertion** CityCheck Check
( NOT EXISTS ( Select * From customer Where customer_city is null));
How to enforce FDs using Assertions

• Consider the relation:

\[
\text{Prof}(\text{pid}, \text{pname}, \text{rank}, \text{salary})
\]

and the FD $\text{rank} \rightarrow \text{salary}$

How can you enforce this?

CREATE ASSERTION salary_rank
CHECK ( NOT EXISTS(SELECT *
FROM Prof P1, Prof P2
WHERE P1.rank = P2.rank AND
P1.salary <> P2.salary))
Triggers

- **Trigger**: A procedure that starts automatically if specified changes occur to the DBMS

- Analog to a "daemon" that monitors a database for certain events to occur

- Three parts:
  - Event (activates the trigger)
  - Condition (tests whether the triggers should run) [Optional]
  - Action (what happens if the trigger runs)

- Semantics:
  - When event occurs, and condition is satisfied, the action is performed.
Trigger: 3 parts

• Event (activates the trigger)
  – A specified modification to the DB
    • May be an insert, deletion, or change
    • May be limited to specific tables
    • The trigger may fire before or after the transaction

• Condition

• Action

Slides partially based on http://cs.gmu.edu/~jessica/cs450_s12/cs450_SQL3.pdf
Trigger: 3 parts

• Event
• Condition(tests whether the triggers should run)
  – A Boolean expression or a query
    • If the query answer set is non-empty it evaluates to true, otherwise false
    • If the condition is true the trigger action occurs
• Action
Trigger: 3 parts

• Event
• Condition
• Action

– A trigger's action can be very far-ranging, e.g.
  • Execute queries
  • Make modifications to the DB
  • Create new tables
  • Call host-language procedures
When and how many times

• Synchronization of the Trigger with the activating statement (DB modification)
  – Before
  – After
• Number of Activations of the Trigger
  – Once per modified tuple
  (FOR EACH ROW)
  – Once per activating statement
  (FOR EACH STATEMENT)
Two different levels

- **Statement-level trigger**: executed once for all the tuples that are changed in one SQL statement.

  REFERENCING  NEW TABLE AS  newtuples,  
  OLD TABLE AS  oldtuples

- **Row-level trigger**: executed once for each modified tuple.

  REFERENCING  OLD AS  oldtuple,  
  NEW AS  newtuple

  newtuples, oldtuples, newtuple oldtuples can be used in the CONDITION and ACTION clauses

- The action of a trigger can consist of multiple SQL statements, surrounded by BEGIN . . .  
  END.
Examples (in Oracle):

CREATE TRIGGER youngSailorUpdate
  AFTER INSERT ON SAILORS /* Event */
  REFERENCING NEW TABLE NewSailors
  FOR EACH STATEMENT
    INSERT /* Action */
    INTO YoungSailors(sid, name, age, rating)
    SELECT sid, name, age, rating
    FROM NewSailors N
    WHERE N.age <= 18;

• This trigger inserts young sailors into a separate table.
• It has no (i.e., an empty, always true) condition.
Another Example

CREATE TRIGGER notTooManyReservations
  AFTER INSERT ON Reserves /* Event */
  REFERENCING NEW ROW NewReservation
  FOR EACH ROW
  WHEN (10 <= (SELECT COUNT(*)
       FROM Reserves
       WHERE sid = NewReservation.sid)) /* Condition */

  DELETE FROM Reserves R
  WHERE R.sid = NewReservation.sid /* Action */
  AND day = (SELECT MIN(day) FROM Reserves R2 WHERE R2.sid = R.sid);

• This trigger makes sure that a sailor has less than 10 reservations, deleting the oldest reservation of a given sailor, if necessary.
• It has a non-empty condition (WHEN).
Another Example

Assume our DB has a relation schema:

Professor (pNum, pName, salary)

We want to write a trigger that:

Ensures that any new professor inserted has salary $\geq 60000$
Example Trigger

CREATE TRIGGER minSalary BEFORE INSERT ON Professor

   for what context  ?

BEGIN

   check for violation here  ?

END;
Example Trigger

CREATE TRIGGER minSalary BEFORE INSERT ON Professor

FOR EACH ROW

BEGIN

Violation of Minimum Professor Salary?

END;
CREATE TRIGGER minSalary BEFORE INSERT ON Professor
FOR EACH ROW
BEGIN

    IF (:new.salary < 60000)
    THEN RAISE_APPLICATION_ERROR (-20004, 'Violation of Minimum Professor Salary');

    END IF;

END;

(Oracle example. In Postgres use RAISE EXCEPTION)
Example trigger

CREATE TRIGGER minSalary BEFORE INSERT ON Professor
    FOR EACH ROW

DECLARE temp int; -- dummy variable not needed

BEGIN
    IF (:new.salary < 60000)
        THEN RAISE_APPLICATION_ERROR (-20004,'Violation of Minimum Professor Salary');
    END IF;

    temp := 10; -- to illustrate declared variables

END;

.
Row vs Statement Level Trigger

• Example: Consider a relation schema

  Account (num, amount)

  where we will allow creation of new accounts only during normal business hours.
CREATE TRIGGER MYTRIG1 
BEFORE INSERT ON Account 
FOR EACH STATEMENT --- is default 
BEGIN 
  IF (TO_CHAR(SYSDATE,'dy') IN ('sat','sun')) OR 
    (TO_CHAR(SYSDATE,'hh24:mi') NOT BETWEEN '08:00' AND '17:00') 
  THEN 
    RAISE_APPLICATION_ERROR(-20500,'Cannot create new account now !!'); 
  END IF; 
END;
When to use BEFORE/AFTER

• Based on efficiency considerations or semantics.

• Suppose we perform statement-level after insert, then all the rows are inserted first, then if the condition fails, and all the inserted rows must be “rolled back”

• Not very efficient !!
Syntax of creating a Trigger

CREATE TRIGGER <triggerName>
BEFORE|AFTER INSERT|DELETE|UPDATE
  [OF <columnList>] ON <tableName>|<viewName>
  [REFERENCING [OLD AS <oldName>] [NEW AS <newName>]]
  [FOR EACH ROW] (default is “FOR EACH STATEMENT”)
  [WHEN (<condition>)]
<PSM body>;

• Syntax in PostgreSQL:
http://www.postgresql.org/docs/9.4/static/sql-createtrigger.html