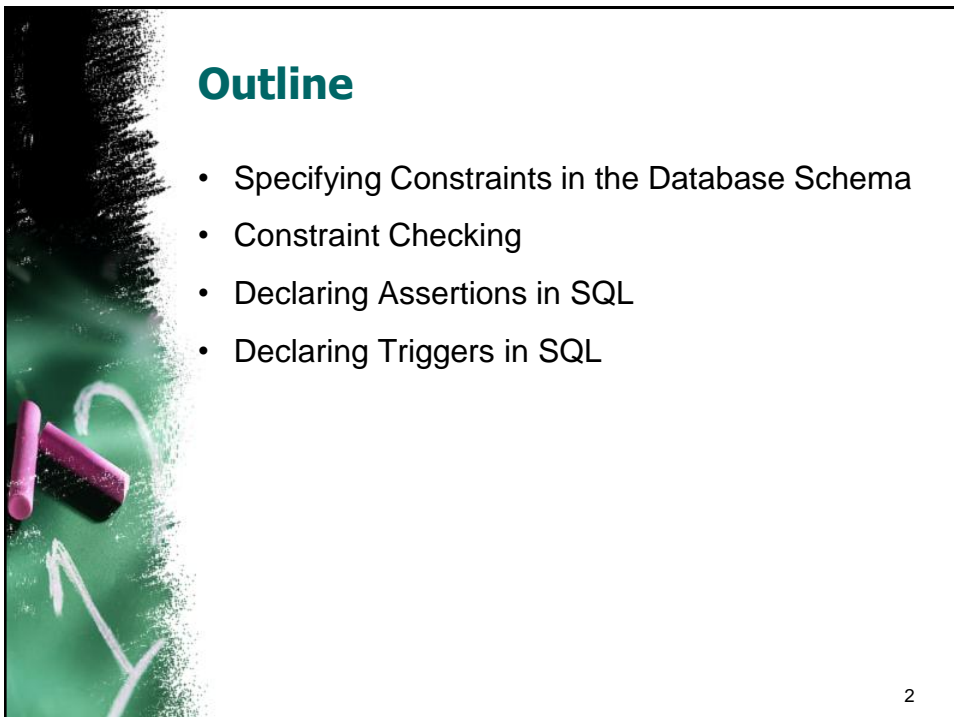




CSI 2132 Lab 5

Assertions and Triggers in SQL

1



Outline

- Specifying Constraints in the Database Schema
- Constraint Checking
- Declaring Assertions in SQL
- Declaring Triggers in SQL

2

Integrity Constraints: a Recap

To ensure the validity of the data held at the database schema, the relational model defines different types of **integrity constraints**:

- **Primary key** constraints
- **Foreign key** constraints
- **Referential integrity** constraints
- **Domain** constraints
- **General** constraints

Operations that violate any integrity constraint at the tuple level are disallowed.

3

Integrity Constraints: a Recap

Primary key constraints: By default, DBMS checks that the combination of values for those attributes declared as primary key remains unique in the relation and that none of them are null.

```
CREATE TABLE artist
(
  aname character varying(20) NOT NULL,
  birthplace character varying(20),
  style character varying(20),
  dateofbirth date,
  country character varying(20),
  CONSTRAINT "pk of artist" PRIMARY KEY (aname)
)
```

4

Integrity Constraints: a Recap

Foreign key constraints: Control what attribute values can be stored in the relation holding the foreign key field. It can point to a primary key attribute in another relation or to a non-PK attribute having the UNIQUE constraint.

```
CREATE TABLE artwork
(
  title character varying(20) NOT NULL,
  "year" integer,
  "type" character varying(20),
  price numeric(8,2),
  aname character varying(20),
  CONSTRAINT artwork_pkey PRIMARY KEY (title),
  CONSTRAINT artwork_aname_fkey FOREIGN KEY (aname)
  REFERENCES artist (aname) MATCH SIMPLE
  ON UPDATE CASCADE ON DELETE CASCADE
)
```

5

Integrity Constraints: a Recap

Referential integrity constraints: Specifies what happens to the tuples in the foreign relation when a deletion or update of a primary key attribute value is about to occur in the main table.

```
CREATE TABLE artwork
(
  title character varying(20) NOT NULL,
  "year" integer,
  "type" character varying(20),
  price numeric(8,2),
  aname character varying(20),
  CONSTRAINT artwork_pkey PRIMARY KEY (title),
  CONSTRAINT artwork_aname_fkey FOREIGN KEY (aname)
  REFERENCES artist (aname) MATCH SIMPLE
  ON UPDATE CASCADE ON DELETE CASCADE
)
```

6

Integrity Constraints: a Recap

Domain constraints: Restricts the set of values an attribute can take to lie within a particular domain.

A **CHECK** clause is added to the attribute definition.

```
CREATE TABLE customer
(
  custid integer NOT NULL,
  "name" character varying(20),
  address character varying(20),
  amount numeric(8,2),
  rating integer,
  CONSTRAINT customer_pkey PRIMARY KEY (custid),
  CONSTRAINT customer_rating_check CHECK (rating >= 1 AND rating <= 10)
)
```

7

Integrity Constraints: a Recap

General constraints: Additional constraints applicable to the environment being modeled.

They are highly model-specific and cannot be captured by any of the previous types of constraints.

The way to do this in SQL is through declarative assertions.

CREATE ASSERTION <name>

CHECK (<condition>)

8

Integrity Constraints: a Recap

CREATE ASSERTION <name>

CHECK (<condition>)

<name> is a mandatory identifier for the constraint.

It can be used later on to modify or drop the constraint.

<condition> can be written as in the WHERE clause

If it holds true, the assertion is not violated and the integrity of the data is guaranteed.

9

Integrity Constraints: a Recap

How to declare assertions?

- Write a query that highlights (selects) the tuples that violate the condition.
- Then use the NOT EXISTS clause to make sure the assertion yields true whenever the tuple set returned by the query is empty.

10

Declaring Assertions: Example

First, make sure you have the following tuples in each relation of the Sailors schema.

If not, create them accordingly.

| Sailors | | | |
|---------|----------|-----|--------|
| sid | sname | age | rating |
| 1 | Salvador | 26 | good |
| 2 | Rafael | 28 | good |
| 3 | John | 10 | good |
| 4 | Bruce | 18 | fair |
| 5 | James | 17 | fair |
| 6 | Smith | 18 | poor |
| 7 | Peter | 22 | poor |

| Boats | |
|-------|-------|
| bid | color |
| 1 | red |
| 2 | green |
| 3 | blue |

| Reserves | |
|----------|-----|
| sid | bid |
| 1 | 1 |
| 1 | 2 |
| 2 | 1 |
| 7 | 1 |
| 7 | 2 |
| 7 | 3 |
| 4 | 1 |
| 3 | 3 |

Create a new schema named: "sailors"

Download and execute all the queries within:

"Create Sailors DB" file

11

Declaring Assertions: Example

Limit the number of sailors and boats to 100 in total

```
create assertion smallClub
check (
    (select count(*) from sailors s) +
    (select count(*) from boats b) < 100
)
```

12

Declaring Assertions: Example

There must never be more than two sailors traveling in the green boat because it is fragile and could sink.

```
create assertion fragile_green_boat
check (
  (
    select count(*)
    from sailors sl, reserves rl, boats bl
    where sl.sid = rl.sid and bl.bid = rl.bid
          and bl.color = 'green'
  ) <= 2
)
```


13

Your turn

The blue boat can only be booked by an inexperienced sailor if he travels along with an expert sailor.

- Inexperienced sailor = poor rating
- Expert sailor = good rating


14



Your turn: Artist DB

It is impossible to admire Caravaggio and not Josefa.

15



Your turn: Artist DB

No modern artwork is worth more than half of the minimum price artwork from the Renaissance period.

16



Triggers in SQL

- It is important to **monitor** the database schema and take appropriate action upon the occurrence of an event.
- For example, if an artwork is appraised in over five million dollars, we notify the manager of the museum where it is located so as to enforce tighter anti-theft security measures.
- Or when an artist is removed from the database, we execute a stored procedure that forwards his/her customer list to a foreign server for cross-validation.
- In such cases, we rely on **triggers** to notify about the event and take further action.

17



Components of a Trigger

- **Event(s)**: An INSERT, UPDATE or DELETE operation on a particular tuple.
- **Condition**: Determines whether the action should be executed. If no condition is specified, the trigger is executed once the event takes place.
- **Action**: Usually a sequence of SQL statements, but could be also a database transaction or running an external program.

18

Triggers in PostgreSQL

```
CREATE TRIGGER name {BEFORE | AFTER}
{ event [OR... ] } [OF attribute] ON table
[ FOR [EACH] {ROW | STATEMENT} ]
[ WHEN (condition) ]
```

```
EXECUTE PROCEDURE funcname(arguments)
```

BEFORE = constraints are checked before the operation is attempted

AFTER = constraints are checked after the operation has been carried out

OF = Column associated with the UPDATE operation

19

Triggers in PostgreSQL

```
CREATE TRIGGER name {BEFORE | AFTER}
{ event [OR... ] } [OF attribute] ON table
[ FOR [EACH] {ROW | STATEMENT} ]
[ WHEN (condition) ]
```

```
EXECUTE PROCEDURE funcname(arguments)
```

ROW = The trigger is invoked once per row affected by the underlying operation. Individual attribute values per row are available.

STATEMENT = The trigger is invoked only once for the entire operation no matter how many rows are affected. No attribute values are available.

20

Trigger Examples

Execute the `check_sailor_rating_age()` function whenever a row of the SAILORS table is about to be updated.

```
CREATE TRIGGER check_sailor
  BEFORE UPDATE ON sailors
  FOR EACH ROW
  EXECUTE PROCEDURE check_sailor_rating_age();
```

21

Trigger Examples

Same as before, but only if the sailor's `rating` is to be updated.

```
CREATE TRIGGER check_sailor
  BEFORE UPDATE OF rating ON sailors
  FOR EACH ROW
  EXECUTE PROCEDURE check_sailor_rating_age();
```

22

Trigger Examples

Same as before, but only if the sailor's **age** will in fact change its value. Notice the **WHEN** clause.

```
CREATE TRIGGER check_sailor
  BEFORE UPDATE ON sailors
  FOR EACH ROW
  WHEN (OLD.age IS DISTINCT FROM NEW.age)
  EXECUTE PROCEDURE check_sailor_rating_age();
```

23

Trigger Examples

Call a function to **log** any sailors' updates, but only if **something** changed:

```
CREATE TRIGGER log_sailor_update
  AFTER UPDATE ON sailors
  FOR EACH ROW
  WHEN (OLD.* IS DISTINCT FROM NEW.*)
  EXECUTE PROCEDURE log_sailor_update();
```

24

Trigger Procedures (functions)

- Must be **defined** before the CREATE TRIGGER statement can execute.
- Must be declared as a **function** taking **no arguments** and returning type **trigger**.
- Can be written in C (**low-level**) or PL-PGSQL (**high-level**)
- We will use PL-PGSQL (Procedural Language for PostgreSQL) from pgAdmin

25

Writing Trigger Procedures with PL-PGSQL

- Open the **Query Editor** and type the following:

```
CREATE FUNCTION check_sailor_name_age()
  RETURNS trigger AS
  $BODY$
BEGIN

  -- Check sailor's name
  IF NEW.sname IS NULL THEN
    RAISE EXCEPTION 'The sailor must have a name';
  END IF;

  -- Check sailor's age
  IF NEW.age > 50 THEN
    RAISE EXCEPTION 'The sailor must be 50 or below';
  END IF;

  RETURN NEW;

END
$BODY$ LANGUAGE plpgsql;
```

26

Writing Trigger Procedures with PL-PGSQL

- Open the **Query Editor** and type the following:

```
CREATE TRIGGER check_sailor
BEFORE UPDATE ON sailors
FOR EACH ROW
EXECUTE PROCEDURE check_sailor_name_age()
|
```

27

Writing Trigger Procedures with PL-PGSQL

Some variables that are often needed:

NEW = Holds the contents of the row to insert or update.

OLD = Holds the contents of the original row.

TG_NARGS = Number of input arguments passed on to the trigger procedure.

TG_ARGV[] = Text array containing the arguments, accessed as \$1, \$2, etc.

28

Writing Trigger Procedures with PL-PGSQL

Testing the Trigger Procedure

Go to the Edit Data window of the Sailors table.

Try to update Peter's age to 51. What do you get?

Now update it to 23. Was the operation successful?

29

Implementing Assertions As Triggers

Let us implement the assertion below as a trigger.

There must never be more than two sailors traveling in the green boat because it is fragile and could sink.

```
create assertion fragile_green_boat
check (
  (
    select count(*)
    from sailors sl, reserves rl, boats bl
    where sl.sid = rl.sid and bl.bid = rl.bid
      and bl.color = 'green'
  ) <= 2
)
```

30

Implementing Assertions As Triggers

When does the event below happen?

There must never be more than two sailors traveling in the green boat because it is fragile and could sink.

- When a new tuple in the RESERVES relation is inserted or an existing one is updated.
- Then we create triggers accordingly.
- But first, we define the trigger procedure.

31

A Glimpse Into PL-PGSQL

Structure of the Main Code Block

```
CREATE FUNCTION identifier (arguments) RETURNS type AS '
DECLARE
  declaration; ← variables and constants here
  [...]
BEGIN
  statement;
  [...]
  RETURN { variable_name | value }
END;' LANGUAGE 'plpgsql';
```

Examples

```
id INTEGER;
title VARCHAR(10);
price FLOAT;
six CONSTANT INTEGER := 6;
ten INTEGER NOT NULL := 10;
```

32

A Glimpse Into PL-PGSQL

Assigning Query Results to Variables

```
SELECT INTO target_variable [, ...] target_column [, ...] select_clauses;
```

- Using the `SELECT INTO` statement

```
DECLARE
  customer_fname varchar(100);
  customer_lname varchar(100);
BEGIN
  SELECT INTO customer_fname, customer_lname
             first_name, last_name
  FROM customers;
  IF NOT FOUND THEN
    return -1;
  END IF;
  return 1;
END;
```

← variables
← attributes

33

The Fragile Green Boat Procedure

Type it in the Query Editor and run it (F5).

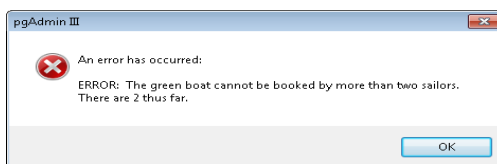
```
CREATE OR REPLACE FUNCTION check_green_boat()
  RETURNS trigger AS
$BODY$
DECLARE
  howMany INTEGER;
BEGIN
  IF NEW.bid = 2 THEN
    SELECT COUNT(*) INTO howMany FROM reserves WHERE bid = 2;
    IF howMany >= 2 THEN
      RAISE EXCEPTION 'The green boat cannot be booked by more than two sailors. There are % thus far.', howMany;
    END IF;
  END IF;
  RETURN NEW;
END
$BODY$
LANGUAGE plpgsql VOLATILE;
```

The Fragile Green Boat Procedure

Then declare the corresponding **trigger**

```
CREATE TRIGGER green_boat
BEFORE INSERT OR UPDATE
ON reserves
FOR EACH ROW
EXECUTE PROCEDURE check_green_boat();
```

Now try it. Go to the Edit Data window for the RESERVES table and try to break the green boat rule by inserting a new record or updating an existing one in such a way that its capacity is overloaded.



35

Your turn: The Blue Boat Rule

Write the trigger and its corresponding PL-PGSQL procedure to enforce the blue boat rule.

The blue boat can only be booked by an inexperienced sailor if he travels along with an expert sailor.

- Inexperienced sailor = poor rating
- Expert sailor = good rating

36

Your turn: Artist DB

It is impossible to admire Caravaggio and not Josefa.

It happens when

- A Caravaggio fan is inserted → add Josefa tuple
- A Caravaggio fan is deleted → delete Josefa tuple
- A Josefa fan is inserted → disallow insertion if no Caravaggio tuple for that fan is found
- A Josefa fan is updated → disallow update if a Caravaggio tuple for that fan is found.
- A Josefa tuple is deleted → disallow deletion if a Caravaggio tuple for that fan is found.

37

References

- Triggers in PostgreSQL

<http://www.postgresql.org/docs/8.1/static/triggers.html>

- Quick Intro to PL-PGSQL

http://www.codeproject.com/KB/database/howto_write_pl_pgsql_func.aspx

- Basic PL-PGSQL statements

<http://developer.postgresql.org/pgdocs/postgres/plpgsql-statements.html>

- Creating trigger procedures with PL-PGSQL

<http://www.postgresql.org/docs/8.1/static/plpgsql-trigger.html>

38