Constraints and Triggers

Databases 2020

Keys and Foreign Keys

- SQL Primary Key declaration
    - Equivalent to NOT NULL and UNIQUE
    - Creates an index, so lookup with key are faster
- SQL Foreign Key declaration
    - Insures that a value in a foreign table exists
    - That value must be declared UNIQUE

Keys and Foreign Keys

- Two declarations in SQL

```
CREATE TABLE studio(
   name CHAR(30) PRIMARY KEY,
   address VARCHAR(255),
   presC#  INT REFERENCES MovieExec(cert#)
);
```

```
CREATE TABLE studio(
   name CHAR(30) PRIMARY KEY,
   address VARCHAR(255),
   presC#  INT,
   FOREIGN KEY (presC#) REFERENCES MovieExec(cert#)
);
```

Keys and Foreign Keys

- What happens if we try to insert into studio a president or change a presC# whose certificate number does <u>not</u> match a certificate number in movieExecs?
- What happens if we delete a row from movieExecs or update a cert# in movieExecs
  - (1) Reject modification.
  - (2) Cascade operation
  - (3) Set NULL

Keys and Foreign Keys

```
CREATE TABLE studio (
   name CHAR(30) PRIMARY KEY,
   address VARCHAR(255),
   presC# INT REFERENCES MovieExec(cert#)
      ON DELETE SET NULL
      ON UPDATE CASCADE
);
```

- If we delete a movieExec tuple with a studio president, then the presC# value in studio is replaced by NULL
- If we change a movieExec tuple with a studio president, then the presC# value gets changed as well

Keys and Foreign Keys

- A tuple with foreign key is "dangling" if the foreign key does not exist
- Similarly, a tuple that does not participate in a join is called dangling.

Keys and Foreign Keys

- A table with a foreign key needs to be populated first
- But there are examples of circular references
  - To deal with them:
    - Make the two insertions part of a single transaction
    - Tells the DBMS to not check constraints until the transaction is finished
      - Can declare deferrable
        - INITIALLY DEFERRED — check just before a transaction commits
        - INITIALLY IMMEDIATE — check after each statement is executed

Keys and Foreign Keys

```
CREATE TABLE studio (
   name CHAR(30) PRIMARY KEY,
   address VARCHAR(255),
   presC# INT UNIQUE
      REFERENCES MovieExec(cert#)
      DEFERRABLE INITIALLY DEFERRED
);
```

Keys and Foreign Keys

- Can also give constraints names
- Then change is enforcement policy

```
SET CONSTRAINT myConstraint DEFERRED;


SET CONSTRAINT myConstraint IMMEDIATE;
```

Constraints on Attributes

- NOT NULL

Constraints on Attributes

- CHECK
    - Enforces conditions on an attribute

```
CREATE TABLE movieExec (
   name CHAR(30) PRIMARY KEY,
   address VARCHAR(255),
   presC# INT REFERENCES MovieExec(cert#)
              CHECK(presC# >= 100000
);
```

Constraints on Attributes

```
CREATE TABLE movieStar(
   name VARCHAR(255) PRIMARY KEY;
   address VARCHAR(255);
   gender CHAR(1)
        CHECK(gender IN ('F', 'M', 'X'))
);
```

Constraints on Attributes

- Checks cannot be used to replace foreign keys

```
…
presC# INT CHECK
    (presC# IN (SELECT cert# FROM movieExec)
…
```

- The check is only executed by the time the tuple is inserted or changed
- If movieExec changes, our table is NOT updated
- Also, NULL values would be rejected

Constraints on Tuples

- Tuple based checks are executed on Insertion and on Update
  - Checks do not trigger checks for relations mentioned in checks

```
CREATE TABLE movieStar(
   name CHAR(30) PRIMARY KEY,
   address VARCHAR(255),
   gender CHAR(1),
   birthdate DATE,
   CHECK(gender = 'F' OR name NOT LIKE 'Ms.%')
);
```

Constraints on Tuples

- Attribute based checks are executed when
  - Attribute is changed
  - Tuple inserted
- Tuple based checks are executed when
  - Tuple changes
  - Tuple inserted

Constraint Modifications

- You should give your constraints names
  - Helps with error messages
  - Used for changing constraints

```
name CHAR(30) CONSTRAINT nameIsKey PRIMARY KEY

CONSTRAINT rightTitle
    CHECK(gender = 'F' OR name not like 'Ms.%'
```

Constraint Modifications

- Dropping constraints
  - Use ALTER table

```
ALTER TABLE movieStar DROP CONSTRAINT nameIsKey;

ALTER TABLE movieStar ADD CONSTRAINT
              nameIsKey PRIMARY KEY(name)
```

Assertions

- Assertion:
  - A boolean valued SQL expression that must be true at all times
- Trigger:
  - Series of actions associated with certain events and triggered by them

Assertion

- Creating assertions

```
CREATE ASSERTION <name> CHECK (<condition>)
```

- Should be true when you call it, unless the assertion is deferred

Assertion

- Formulating assertions
    - Unlike checks, assertions need to specify the relation

```
movieExec(name, address, cert#, netWorth)
studio(name, address, presC#
```

```
CREATE ASSOCIATION richPres CHECK(
   (NOT EXISTS
       (SELECT studio.name
        FROM studio, movieExec
        WHERE presC# = cert# AND netWorth<1000000
       )
   );
```

Assertion

- Formulating assertions
    - All studios can only produce <10000 minutes of movies

Assertion

```
CREATE ASSERTION sumLength CHECK
    (10000 >= ALL
           (SELECT SUM(length)
            FROM movies
            GROUP BY studioName
           )
    );
```

Assertion

- Assertions are always checked when there is a change in the database
- Constraints for a tuple are only checked when a tuple is updated or inserted
    - Therefore, making the previous assertion a check has a different meaning:

```
ALTER TABLE movies ADD CONSTRAINT
    maxLength CHECK (10000 >= ALL
         (SELECT SUM(length) FROM movies
            GROUP BY studioName)
    );
```

Assertion

- Dropping assertions

```
DROP ASSERTION sumLength
```

Triggers

- A Trigger is awakened at certain events
  - insert, delete, updates to a particular relation
- A Trigger then tests a condition.
  - If condition is false, nothing more happens
  - Otherwise: The action associated with trigger is executed

Triggers

- Trigger's condition and action executed either :
    - state of DB before the triggering event
    - state of DB after the triggering event
- Condition and action can refer to both the new and the old values
- Update events can be limited to certain attribute(s)
- Trigger can execute
    - once for each modified tuple — *row-level trigger*
    - once for all tuples changed — *statement level trigger*

Triggers

- Example:

```
CREATE TRIGGER netWorthTrigger
AFTER UPDATE OF netWorth ON movieExec
REFERENCING
   OLD ROW AS OldTuple,
   NEW ROW AS NewTuple
FOR EACH ROW
WHEN (OldTuple.netWorth > NewTuple.netWorth)
   UPDATE movieExec
   SET netWorth = OldTuple.netWorth
   WHERE cert# = NewTuple.cert#
```

Triggers

Triggers

Triggers

Triggers

Triggers