CS 235: Introduction to Databases
Svetlozar Nestorov

Lecture Notes #9

Outline
- Active elements
  - Maintain database integrity and consistency.
  - Part of database schema.
- Constraints
- Triggers

Constraints
- Restrictions on the data in your database.
- Commercial relational systems allow much more fine-tuning of constraints than do the modeling languages we learned earlier.
- In essence: SQL programming is used to describe constraints.

Constraint Types
1. Primary key declarations (already covered).
2. Foreign-keys = referential integrity constraints.
3. Attribute- and tuple-based checks = constraints within relations.
4. SQL Assertions = global constraints.
   - Not found in Oracle.
5. Oracle Triggers.
   - A substitute for assertions.

Foreign Keys
- In relation R a clause that attribute A references S(B) says that whatever values appear in the A column of R must also appear in the B column of relation S.
- B must be declared the primary key (or unique) for S.
  - Why is this restriction necessary?

Example
CREATE TABLE Beers (
   name CHAR(20) PRIMARY KEY,
   manf CHAR(20)
);

CREATE TABLE Sells (
   bar CHAR(20),
   beer CHAR(20) REFERENCES Beers(name),
   price REAL
);
Alternative Declaration

- Add another element declaring the foreign key, as:
  
  ```
  CREATE TABLE Sells (
  bar CHAR(20),
  beer CHAR(20),
  price REAL,
  FOREIGN KEY beer REFERENCES Beers(name)
  );
  ```

- Extra element essential if the foreign key is more than one attribute.

Foreign Key Constraint Violations

1. Insert or update a Sells tuple so it refers to a nonexistent beer.
   - Always rejected.
2. Delete or update a Beers tuple that has a beer value some Sells tuples refer to:
   a) Default: reject the modification.
   b) Cascade: Ripple changes to referring Sells tuple.
   c) Set Null: Change referring tuples to have NULL in referring components.

Example (Cascade)

- Delete Bud.
- Cascade deletes all Sells tuples that mention Bud.

- Update Bud to Budweiser.
- Change all Sells tuples with Bud in beer column to be Budweiser.

Example (Set-Null)

- Delete Bud.
- Set-null makes all Sells tuples with Bud in the beer component have NULL there.

- Update Bud to Budweiser.
- Same change as above.

Selecting a Policy

- Add `ON [DELETE, UPDATE] [CASCADE, SET NULL]` to foreign key declaration.

  ```
  CREATE TABLE Sells ( 
  bar CHAR(20),
  beer CHAR(20),
  price REAL,
  FOREIGN KEY beer REFERENCES Beers(name) 
  ON DELETE SET NULL 
  ON UPDATE CASCADE
  );
  ```

- Correct policy is a design decision.
  - E.g., what does it mean if a beer goes away? What if a beer changes its name?

Attribute-Based Checks

- Follow an attribute by a condition that must hold for that attribute in each tuple of its relation.
  - `CHECK (condition`).
    - Condition may involve the checked attribute.
    - Other attributes and relations may be involved, but only in subqueries.
    - Oracle: No subqueries allowed in condition.
  - Condition is checked only when the associated attribute changes (i.e., an insert or update occurs).
Example

CREATE TABLE Sells (
  bar CHAR(20),
  beer CHAR(20) CHECK (beer IN (SELECT name FROM Beers)),
  price REAL CHECK (price <= 5.00)
);

- Check on beer is like a foreign-key constraint, except:
  - The check occurs only when we add a tuple or change the beer in an existing tuple, not when we delete a tuple from Beers.

Tuple-Based Checks

- Separate element of table declaration.
- Form: like attribute-based check.
- But condition can refer to any attribute of the relation.
  - Or to other relations/attributes in subqueries.
  - Again: Oracle forbids the use of subqueries.
- Checked whenever a tuple is inserted or updated.

Example

- Only Ripoff bar can sell beer for more than $10.
  CREATE TABLE Sells (
    bar CHAR(20),
    beer CHAR(20),
    price REAL,
    CHECK(bar = 'Ripoff' OR price <= 10.00)
  );

SQL Assertions

- Database-schema constraint.
- Not present in Oracle.
- Checked whenever a mentioned relation changes.
- Syntax:
  CREATE ASSERTION <name>
  CHECK(<condition>);

Example

- No bar may charge an average of more than $5 for beer. Sells(bar, beer, price)
  CREATE ASSERTION NoRipoffBars
  CHECK(NOT EXISTS(
    SELECT bar
    FROM Sells
    GROUP BY bar
    HAVING 5.0 < AVG(price)
  ));

- Checked whenever Sells changes.

Example

- There cannot be more bars than drinkers.
  Bars(name, addr, license) Drinkers(name, addr, phone)
  CREATE ASSERTION FewBars
  CHECK(
    (SELECT COUNT(*) FROM Bars) <= (SELECT COUNT(*) FROM Drinkers)
  );

- Checked whenever Bars or Drinkers changes.
Triggers (Oracle Version)

- Often called event-condition-action rules.
- Event = a class of changes in the DB, e.g., insertions into Beers.
- Condition = a test as in a where-clause for whether or not the trigger applies.
- Action = one or more SQL statements.
- Differ from checks or SQL assertions in that:
  1. Triggers invoked by the event; the system doesn't have to figure out when a trigger could be violated.
  2. Condition not available in checks.

Example

- Whenever we insert a new tuple into Sells, make sure the beer mentioned is also mentioned in Beers, and insert it (with a null manufacturer) if not.

```
CREATE OR REPLACE TRIGGER BeerTrig
AFTER INSERT ON Sells
FOR EACH ROW
WHEN (new.beer NOT IN (SELECT name FROM Beers))
BEGIN
  INSERT INTO Beers(name)
  VALUES(:new.beer);
END;
```

Options

1. Can omit OR REPLACE. But if you do, it is an error if a trigger of this name exists.
2. AFTER can be BEFORE.
3. If the relation is a view, AFTER can be INSTEAD OF.
   - Useful for allowing modifications to a view; you modify the underlying relations instead.

More Options

4. INSERT can be DELETE or UPDATE OF <attribute>.
   - Also, several conditions like INSERT ON Sells can be connected by OR.
5. FOR EACH ROW can be omitted, with an important effect: the action is done once for the relation(s) consisting of all changes.

Explanation

- There are two special variables new and old, representing the new and old tuple in the change.
  - old makes no sense in an insert, and new makes no sense in a delete.
  - Notice: in WHEN we use new and old without a colon, but in actions, a preceding colon is needed.

More Explanations

- The action is a PL/SQL statement.
  - Simplest form: surround one or more SQL statements with BEGIN and END.
  - However, select-from-where has a limited form.
  - Dot and run cause the definition of the trigger to be stored in the database.
  - Oracle triggers are part of the database schema, like tables or views.
Even More Explanations

- Important Oracle constraint: the action cannot change the relation that triggers the action.
- Worse, the action cannot even change a relation connected to the triggering relation by a constraint, e.g., a foreign-key constraint.

Example

- Maintain a list of all the bars that raise their price for some beer by more than $1.

  ```sql
  Sells(bar, beer, price) RipoffBars(bar)
  CREATE TRIGGER PriceTrig
  AFTER UPDATE OF price ON Sells
  FOR EACH ROW
  WHEN(new.price > old.price + 1.00)
  BEGIN
    INSERT INTO RipoffBars VALUES(:new.bar);
  END;
  .
  run
  ```

SQL Triggers

- Covered in the book.
- Some differences, including:
  1. The Oracle restriction about not modifying the relation of the trigger or other relations linked to it by constraints is not present in SQL (but Oracle is real; SQL is paper).
  2. The action in SQL is a list of (restricted) SQL statements, not a PL/SQL statement.