Database Systems I

Constraints & Triggers

Instructor: Ouldooz Baghban Karimi
CMPT 354 - Summer 2019
Maintaining Integrity

• Integrity Constraints
  • Referential Constraints
  • Attribute & Tuple Constraints
    • Not NULL
    • Key Constraints
    • Custom Constraints

• Actively Maintaining Integrity
  • Constraints
  • Checks
  • Deferrable Checks
  • Assertions
  • Triggers
Integrity Constraints

• Integrity Constraints
  • Referential Constraints
    • When? Insert, Delete, Update
    • Action? Default, Restrict, Cascade, Set NULL

• Attribute & Tuple Constraints
  • NOT NULL
  • Key Constraints
  • Other Constraints
Checks

• Not NULL

• Attribute-based: on only one attribute
  • Checked only when the attribute mentioned in the constraint changes
    • In principle: anything that could follow WHERE in a SQL query
    • In practice: simple limit on values
  • Attribute-based constraint can mention other attributes of the same relation in subqueries

• Tuple-based: On more than one attribute
  • Tuple-based constraints will be checked more frequently than the attribute-based constraints
    • Whenever any attribute of the tuple changes
Example: Checks

- `presCNum INT REFERENCES MovieExec(certNum) CHECK (presCNum >= 100000)`

- `gender CHAR(1) CHECK (gender IN ('F', 'M'))`

- `CREATE TABLE MovieStar (``
  name CHAR(30) PRIMARY KEY, ``
  address VARCHAR(255), ``
  gender CHAR(1), ``
  birthdate DATE, ``
  CHECK (gender = 'F' OR name NOT LIKE 'Ms.*/.'))
)`
Modification of Constraints

• Naming a constraint
  CONSTRAINT CHECK (<conditions>)

• Altering a Constraint
  • Drop
    ALTER TABLE <table name> DROP CONSTRAINT <constrain name>;
  • Add
    ALTER TABLE <table name>
    ADD CONSTRAINT <constraint name> PRIMARY KEY (<attribute(s)>);
Example: Modification of Constraints

• gender CHAR(1) CONSTRAINT GenderCheck CHECK (gender IN ('F', 'M'))

• name CHAR(30) CONSTRAINT NameIsKey PRIMARY KEY

• ALTER TABLE MovieStar DROP CONSTRAINT NameIsKey

• ALTER TABLE MovieStar ADD CONSTRAINT NameIsKey PRIMARY KEY (name)
Deferred Checking of Constraints

• Constraints of any type can be given names.

• We can change a deferrable constraint to be immediate or deferred
  \texttt{DEFERRABLE INITIALLY <DEFERRED/IMMEDIATE>}

• For named deferrable constraints we can also make this change
  \texttt{SET CONSTRAINT <constraint name> DEFERRED}
Assertions & Triggers

• An assertion is a boolean-valued SQL expression that must be true at all times.

• A trigger is a series of actions that are associated with certain events, such as insertions into a particular relation, and that are performed whenever these events arise.
Assertions

• Creating Assertions

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

• Condition
  • Combination of EXISTS, NOT, IN, ANY, ALL, >, etc
  • It is necessary to combine results in some way to make a single true/false choice

• Deleting Assertions

```
DROP ASSERTION <assertion name>
```
Example: Assertions

CREATE ASSERTION SumLength
  CHECK (10000 >= ALL
    ( SELECT SUM(length)
      FROM Movies
      GROUP BY studioName)
  );
Triggers

- Event-condition-action rules or ECA rules

- Triggers are only awakened when certain events

- Once awakened by its triggering event, the trigger tests a condition

- If the condition of the trigger is satisfied, the action associated with the trigger is performed by the DBMS
Triggers in SQL

• Creating Triggers
  `CREATE TRIGGER <trigger name>`

• The event and before and after it
  `BEFORE/AFTER INSERT/UPDATE/DELETE ON`

• The referenced tuple for condition and action
  `REFERENCING`

• Trigger execution
  `FOR EACH ROW`
  `FOR EACH STATEMENT`

• The condition
  `WHEN`

• The action
  `INSERT INTO/DELETE FROM/UPDATE/SET`
Example: Triggers

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#;
Indexes

✓ Views
  • Indexes

• Storage Management
  • Secondary Storage (Required Reading)
  • Index Structures
(Virtual) Views

• Relations that are defined by a query over other relations
  • Not stored in the database
  • Can be queried as if they existed

• Constructed periodically from the database and stored
  • Can speed up the execution of queries
  • Special Case: Indexes
Views in SQL (1)

• Declaring Views

  CREATE VIEW <view-name> AS <view-definition>;

  The view definition is a SQL query

  • Attributes of the view could be specified renaming table attributes
    Surrounded by parentheses, after the name of the view

    <view-name (attribute list)>

• Querying Views: A view may be queried exactly as if it were a stored table

• Removing Views

  DROP VIEW <view-name>;;
Views in SQL (2)

• Modifying Views
  • Limited insertions, deletions, or updates to updatable views

• Updatable Views
  • Using \textit{SELECT}, not \textit{SELECT DISTINCT}
  • The \textit{WHERE} clause must not involve \textit{R} in a subquery
  • The \textit{FROM} clause consists of only one occurrence of \textit{R}, and no other relation
  • The list in the \textit{SELECT} clause must include enough attributes that for every tuple inserted into the view, we can fill the other attributes out with NULL values or the proper default

• Instead-of Triggers on Views
  • Use instead-of instead of before and after
  • Trigger to replace an action on a view by an action on the underlying base table
Example: Instead-of Triggers on Views

CREATE TRIGGER ParamountInsert
INSTEAD OF INSERT ON ParamountMovies
REFERENCING NEW ROW AS NewRow
FOR EACH ROW
   INSERT INTO Movies(title, year, studioName)
   VALUES(NewRow.title, NewRow.year, 'Paramount');
Acknowledgements

I have used materials from the following resources in preparation of this course:

• Database Systems: The Complete Book
• Database Systems (Kiefer, Bernstein, Lewis)
• Database System Concepts: https://www.db-book.com
• Course offerings
  • W 4111 (Eugene Wu - Columbia): https://w4111.github.io/
  • CS 245 (Matei Zaharia - Stanford): http://web.stanford.edu/class/cs245/
  • CS 186 (Joe Hellerstein - Berkeley): https://sites.google.com/site/cs186fall17/