CS143 Notes: Database Integrity

Book Chapters
(4th) Chapter 6.1-4
(5th) Chapter 4.2, 8.6
(6th) Chapter 4.4, 5.3

Things to Learn

- Key constraints
- Referential integrity (Foreign key constraints)
- CHECK constraints
- SQL trigger (part of SQL99)

What are integrity constraints?

- An example database with invalid entries (Show the example)

- A statement about what a valid database should look like
  - As a human being, we understand what is a “valid” database
  - The system needs an explicit specification of the semantics/rules

- Arbitrary predicate pertaining to the database (in principle)
  - In practice, only the ones that are easy to enforce

- If a SQL statement violates IC, the statement is aborted and generates an error

Q: What rules/constaints can you find from the example?
• Database constraints checks the rules in the DB (Three tier diagram)

![Three tier diagram]

• Q: Why do we check these rules in DB, not in application? Checking them at application/Web browser can be cheaper

Data validity enforcement in RDBMS

• 3 ways to enforce data validity in RDBMS
  – Domain: GPA is real
  – Constraints: Gives error. Abort statement
    * Key
    * Referential Integrity
    * CHECK constraint
  – Trigger: Event-Condition-Action rule. If a certain event happens, invoke an action to handle it

Key Constraints

• A set of attributes should be unique in a table

- Course(dept, cnum, sec, unit, instructor, title)
- Course(dept, cnum, sec, unit, instructor, title)
- Course(dept, cnum, sec, unit, instructor, title)

  – CREATE TABLE Course (  
    dept CHAR(2) NOT NULL,  
    cnum INTEGER NOT NULL,  
    sec INTEGER NOT NULL,  
    unit INTEGER,  
    instructor VARCHAR(30),  
    title VARCHAR(30),  
    PRIMARY KEY(dept, cnum, sec),  
    UNIQUE(dept, cnum, instructor),  
    UNIQUE(dept, sec, title) )

  – One primary key per table
– Unique for other keys
– Primary key, unique are enforced through index (more discussion later)

**Referential Integrity Constraints**

• **Example:**

  – If an sid appears in Enroll, it should also appear in Student
  – If an (dept, cnum, sec) appears in Enroll, it should also appear in Class
    * Q: Is the reverse true?

• **Terminology**

  – (Two table diagram: E.A references S.A)
    ![Diagram](E_A_references_S_A)
  – E.A references S.A
  – E.A: referencing attribute or **foreign key**
  – S.A: referenced attribute
  – **Referential integrity** means that referenced value always exists
    * foreign key can be NULL. When a foreign key is NULL, no constraint checking

• **Referential Integrity in SQL**

  – Example:
    ```sql
    CREATE TABLE Enroll (
        sid INTEGER REFERENCES Student(sid),
        dept CHAR(2),
        cnum INTEGER,
        sec INTEGER,
        FOREIGN KEY (dept, cnum, sec) REFERENCES Class(dept, cnum, sec) )
    ```
  – Notes:
    * Referenced attributes must be PRIMARY KEY or UNIQUE
    * Referenced attributes may be omitted if they are the same name with referencing attributes
      - e.g., sid INT REFERENCES Student
    * One attribute foreign key may be defined directly

• **Referential Integrity Violation**

  – Q: When is the RI violated (two table diagram)?
    ![Diagram](E_A_relations_S_A)
e.g., do we have to worry if a tuple is deleted from E?

- RI violation from E (insert to E or update to E.A) is **not allowed**
  * System rejects the statement
  * Always insert/update S first.

- **Q:** If a tuple in S is updated/deleted, what can we do to avoid RI violation?

**ON DELETE/UPDATE SET NULL/SET DEFAULT/CASCADE in SQL**

1. Default: disallow the statement and generate error
2. SET NULL/SET DEFAULT: Change E.A value to NULL or default value
3. CASCADE:
   * On deletion of S: delete the referencing tuples in E
   * On update of S.A: change E.A to the new S.A

- **Example:**
  ```sql
  CREATE TABLE Enroll (  
    sid INTEGER REFERENCES Student(sid)  
    ON DELETE CASCADE  
    dept CHAR(2),  
    cnum INTEGER,  
    sec INTEGER,  
    FOREIGN KEY (dept, cnum, sec) REFERENCES  
    Class(dept, cnum, sec)  
    ON DELETE CASCADE  
    ON UPDATE SET NULL )
  ```

**Comments:**
* By default, Student.sid update is not allowed if RI is violated
* Many RDBMS does not support all actions

- **Comments:** Referential integrity is the only SQL constraint that can “fix itself”
  * Other constraints simply abort and report error
- **Q:** Why should the referenced attributes be unique?

- **Self referencing table**
  - **Example:**
    | A | B |
    |---|---|
    | 1 | NULL |
    | 2 | 1   |
    | 3 | 2   |
    | 4 | 3   |
    | 5 | 4   |

  ```
  CREATE TABLE R (
    A INTEGER PRIMARY KEY,
    B INTEGER REFERENCES R(A)
    ON DELETE CASCADE
  )
  ```

  - **Comments:**
    * A table references itself: self-referecing table
    * **Q:** What will happen if we delete (1,NULL)?

- **Circular constraints**
  - **Example:** ChickenFrom cid, eid: eid became cid,
    EggFrom eid, cid: eid is born of cid
    (Chicken.eid ⊆ Egg.eid, Egg.cid ⊆ Chicken.cid) (diagram)

  - **Q:** Can we insert any tuple to Chicken? or to Egg? How can we fix it?
CHECK constraint

• Constraints attached to a table
• CHECK clause in table definition

• Example: $0 \leq GPA \leq 4.0$

    CREATE TABLE Student(
        sid int,
        ...
        GPA real CHECK(0 <= GPA and GPA <= 4.0),
        ...
    )

• Example: cnum < 600 AND unit < 10

    CREATE TABLE Enroll(
        dept CHAR(2),
        cnum INT,
        unit INT,
        title VARCHAR(50),
        CHECK (cnum < 600 AND unit < 10) )

• Constraint is checked whenever the tuple updated.
• In SQL92, conditions can be complex, e.g., with subqueries
• Q: The units of all CS classes are above 3 for Class(dept, cnum, unit, title)?

• Q: Students whose GPA is below 2.0 cannot take CS classes?

• Q: Can we express referential integrity constraint, e.g., Enroll.sid ⊂ Student.sid, using CHECK?
Triggers

Trigger

- Event-Condition-Action rule (or ECA rule)
  - We explicitly specify what events to monitor, what condition to check and what action to take if the condition is met.

- **Query 1:** All new students have to take CS143 (For every insertion to Student, add the corresponding tuple to Enroll.)

  Q: What if we insert thousands of student tuples in one insertion? Execute the trigger thousand times? Can we execute it once for all new tuples?

- **Query 2:** If a student GPA is updated to less than 2.0, revert back to the old GPA.

Comments: ROLLBACK command

- Trigger general syntax: Event-Condition-Action rule (or ECA rule)
  - CREATE TRIGGER <name>
    <event>
    <referencing clause> // optional
    WHEN (<condition>) // optional
    <action>
- **<event>**
  * BEFORE | AFTER INSERT ON R
  * BEFORE | AFTER DELETE ON R
  * BEFORE | AFTER UPDATE [OF A1, A2, ..., An] ON R

- **<action>**
  * Any SQL statement. Multiple statements should be enclosed with **BEGIN ATOMIC**
    ... **END** and be separated by **;**

- **<referencing clause>**
  * REFERENCING OLD|NEW TABLE|ROW AS <var>, ...  
  * FOR EACH ROW: row-level trigger  
  * FOR EACH STATEMENT (default): statement-level trigger

- **Query 3:** How to enforce AVG(GPA) > 3.0?

- **Q:** For, \( R(A) \), after inserting (1), what will happen? 
  CREATE TRIGGER Recursion  
  AFTER INSERT ON R  
  BEGIN INSERT INTO R VALUES (1); END

- **Action sequence**
  1. BEFORE trigger
  2. Statement
  3. AFTER trigger
  4. Constraint checking
What is supported in MySQL

- Key constraint
- Under InnoDB, most referential integrity except “ON DELETE/UPDATE SET DEFAULT"
- No CHECK constraints
- Limited trigger: does not allow updating the table that caused the trigger event
  - Generates error and rejects the statement that caused the event

Things to Remember

Constraints and Trigger

- Key constraint: PRIMARY KEY, UNIQUE

- Referential Integrity
  - Referencing attribute (foreign key), referenced attribute
    * Referenced attribute should be PRIMARY KEY or UNIQUE
  - Violation at referencing attribute not allowed
  - Violation at referenced attribute can be fixed automatically
    * ON DELETE/UPDATE SET NULL/SET DEFAULT/CASCADE

- Tuple-based CHECK constraint

- Trigger