CS 405G: Introduction to Database Systems

Topics Next
- Relational Integrity Constraint
- Reading assignment:
  - Chapter 3.2

Relational Integrity Constraints

- Constraints are conditions that must hold on all valid relation instances. There are four main types of constraints:
  1. Domain constraints
  2. Key constraints
  3. Entity integrity constraints
  4. Referential integrity constraints
Primary Key Constraints

- A set of fields is a **candidate key** for a relation if:
  1. No two distinct tuples can have same values in all key fields, and
  2. This is not true for any subset of the key.
- Part 2 false! A **superkey**.
- If there are >1 keys for a relation, one of the keys is chosen (by DBA) to be the **primary key**.
- E.g., given a schema **Student(sid: string, name: string, gpa: float)** we have:
  - sid is a key for Students. (What about name?) The set {sid, gpa} is a superkey.

---

Key Example

- **CAR** (licence_num: string, Engine_serial_num: string, make: string, model: string, year: integer)
  - What is the candidate key(s)
  - Which one you may use as a primary key
  - What are the super keys

---

Entity Integrity

- **Entity Integrity**: The primary key attributes PK of each relation schema R in S cannot have null values in any tuple of r(R).
  - Other attributes of R may be similarly constrained to disallow null values, even though they are not members of the primary key.
Foreign Keys, Referential Integrity

- **Foreign key**: Set of fields in one relation that is used to `refer` to a tuple in another relation. (Must correspond to primary key of the second relation.) Like a ‘logical pointer’.
  - E.g. `sid` is a foreign key referring to **Students**:
    - `Student(sid: string, name: string, gpa: float)`
    - `Enrolled(sid: string, cid: string, grade: string)`
  - If all foreign key constraints are enforced, **referential integrity** is achieved, i.e., no dangling references.
  - Can you name a data model w/o referential integrity?

- Links in HTML!

In-Class Exercise

(Taken from Exercise 5.16)

Consider the following relations for a database that keeps track of student enrollment in courses and the books adopted for each course:

- **STUDENT(SSN, Name, Major, Bdate)**
- **COURSE(Course#, Cname, Dept)**
- **ENROLL(SSN, Course#, Quarter, Grade)**
- **BOOK_ADOPTION(Course#, Quarter, Book_ISBN)**
- **TEXT(Book_ISBN, Book_Title, Publisher, Author)**

Draw a relational schema diagram specifying the foreign keys for this schema.
In-Class Exercise
Consider the following relations for a database that keeps track of student enrollment in courses and the books adopted for each course:

STUDENT(SSN, Name, Major, Bdate)
COURSE(Course#, Cname, Dept)
ENROLL(SSN, Course#, Quarter, Grade)
BOOK_ADOPTION(Course#, Quarter, Book_ISBN)
TEXT(Book_ISBN, Book_Title, Publisher, Author)

Draw a relational schema diagram specifying the foreign keys for this schema.

Other Types of Constraints
- Semantic Integrity Constraints:
  - based on application semantics and cannot be expressed by the model per se
  - e.g., “the max. no. of hours per employee for all projects he or she works on is 56 hrs per week”
  - A constraint specification language may have to be used to express these
  - SQL-99 allows triggers and ASSERTIONS to allow for some of these

Update Operations on Relations
- Update operations
  - INSERT a tuple.
  - DELETE a tuple.
  - MODIFY a tuple.
- Constraints should not be violated in updates
Example

- We have the following relational schemas
  - `Student(sid: string, name: string, gpa: float)`
  - `Course(cid: string, department: string)`
  - `Enrolled(sid: string, cid: string, grade: character)`

- We have the following sequence of database update operations. (assume all tables are empty before we apply any operations)
  - `INSERT('1234', 'John Smith', '3.5')` into Student
    
    | sid | name   | gpa |
    |-----|--------|-----|
    | 1234| John Smith | 3.5 |

Example (Cont.)

- `INSERT('647', 'EECS')` into Courses
- `INSERT('1234', '647', 'B')` into Enrolled
- UPDATE the grade in the Enrolled tuple with `sid = 1234` and `cid = 647` to 'A'.
- DELETE the Enrolled tuple with `sid 1234` and `cid 647`

Exercise

- `INSERT('108', 'MATH')` into Courses
- `INSERT('1234', '108', 'B')` into Enrolled
- `INSERT('1123', 'Mary Carter', '3.8')` into Student

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>John Smith</td>
<td>3.5</td>
</tr>
<tr>
<td>1123</td>
<td>Mary Carter</td>
<td>3.8</td>
</tr>
<tr>
<td>1234</td>
<td>108</td>
<td></td>
</tr>
<tr>
<td></td>
<td>647</td>
<td>EECS</td>
</tr>
</tbody>
</table>
Exercise (cont.)

- A little bit tricky
- INSERT '<1125', 'Bob Lee', 'good' > into Student
  - Fail due to domain constraint
- INSERT '<1123', NULL, 'B'> into Enrolled
  - Fail due to entity integrity
- INSERT '<1233', '647', 'A'> into Enrolled
  - Failed due to referential integrity

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>John Smith</td>
<td>3.5</td>
</tr>
<tr>
<td>1123</td>
<td>Mary Carter</td>
<td>3.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cid</th>
<th>department</th>
</tr>
</thead>
<tbody>
<tr>
<td>647</td>
<td>EECS</td>
</tr>
<tr>
<td>108</td>
<td>MATH</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sid</th>
<th>cid</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>108</td>
<td>B</td>
</tr>
</tbody>
</table>

Exercise (cont.)

- A more tricky one
- UPDATE the cid in the tuple from Course where cid = 108 to 109

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>John Smith</td>
<td>3.5</td>
</tr>
<tr>
<td>1123</td>
<td>Mary Carter</td>
<td>3.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cid</th>
<th>department</th>
</tr>
</thead>
<tbody>
<tr>
<td>647</td>
<td>EECS</td>
</tr>
<tr>
<td>108</td>
<td>MATH</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sid</th>
<th>cid</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>109</td>
<td>B</td>
</tr>
</tbody>
</table>

Update Operations on Relations

- In case of integrity violation, several actions can be taken:
  - Cancel the operation that causes the violation (REJECT option)
  - Perform the operation but inform the user of the violation
  - Trigger additional updates so the violation is corrected (CASCADE option, SET NULL option)
  - Execute a user-specified error-correction routine