SQL
Integrity Constraints

Integrity Constraints (Review)

- An IC describes conditions that every legal instance of a relation must satisfy.
  - Restrictions on attribute values of tuples
- Inserts/deletes/updates that violate IC's are disallowed.
- Covered so far:
  - On individual tuples
    - Domain constraints:
      - Data type: name must be a string
      - NOT NULL
  - For relation as a whole
    - Primary Key and Unique Constraints:
      - no two tuples may have the same value
  - Across relations
    - Referential integrity through foreign key constraint:
      - sid is a foreign key in relation Participates;

Rule of thumb to classify an IC

- ❑ If you can only ”see” that tuple and the schema definition, can you say if it will fail the IC ?
  - ☆YES ➔ attribute / tuple level IC :
    - Ex: NULL value in the Tuple for an attribute defined to be NOT NULL in the schema
    - Ex: CHARACTER value in the Tuple for an attribute defined to be an INTEGER
- ❑ If you have to ”see” all the tuples already in the table and the schema definition, can you say if it will fail the IC ?
  - ☆YES ➔ Table level IC :
    - Ex: The new tuple has value for its PRIMARY KEY, which is already present in the table.

Attribute-Based Checks

- If a condition must hold for specific attribute: CHECK

```
CREATE TABLE Skaters (
   sid INTEGER PRIMARY KEY NOT NULL,
   sname VARCHAR(20),
   rating INTEGER CHECK(rating > 0 AND rating <
  11),
   age INTEGER)
```

- Condition is checked only when the associated attribute changes (i.e., an insert or update, but not delete!)
- If condition is violated the system rejects the modification
- In SQL condition can be anything that could follow WHERE clause
  - ☆ CHECK rating in (1, 2, 3, 4, 5)
  - ☆ Possibly subqueries
- Most database systems allow very restricted attribute-based check (no subqueries, no reference to other attributes, …)

Tuple-Based Checks

- If a condition covers several attributes

```
CREATE TABLE Skaters (
   sid INTEGER PRIMARY KEY NOT NULL,
   sname VARCHAR(20),
   rating INTEGER,
   age INTEGER,
   CHECK (rating <= 4 OR age > 5))
```

- Checked upon each update and insert

Naming constraints

- Problem of previous examples:
  - ☆ what if constraints change (e.g., we want to increase rating constraint to (rating <=5 OR age > 5)
- Solution: name constraints:

```
CREATE TABLE Skaters (
   sid INT NOT NULL,
   sname VARCHAR(20),
   rating INT CONSTRAINT rat CHECK
                           (rating > 0 AND rating < 11),
   age INT,
    CONSTRAINT pk PRIMARY KEY (sid),
   CONSTRAINT ratage CHECK
                     (rating <= 4 OR age > 5))
```

- This allows us to drop and recreate them later on

```
ALTER TABLE Skaters DROP CONSTRAINT ratage
ALTER TABLE Skaters ADD CONSTRAINT ratage
                              CHECK (rating <=5 OR age > 5)
```

  - what if there is already a record with rating = 5 and age = 2 ?