NULL’s

NULL is a special value:

- Many possible interpretations: value unknown, value inapplicable, value withheld, etc.
- Often used as the default value

Example:

```
INSERT INTO Student VALUES(135, 'Maggie', NULL, NULL);
```

Operations on NULL’s

- When we operate on a NULL and another value (including another NULL) using +, ×, etc., the result is NULL
- Aggregate functions ignore NULL, except COUNT(*)

Example: \( \text{AVG(GPA)} = \frac{\text{SUM(GPA)}}{\text{COUNT(*)}} \) ?

Three-Valued Logic

- TRUE = 1, FALSE = 0, UNKNOWN = 1/2
- AND = \( \min \), OR = \( \max \), \( \text{NOT}(x) = 1 - x \)
- When we compare a NULL with another value (including another NULL) using =, >, etc., the result is UNKNOWN
- SELECT clause only lists tuples if the condition evaluates to TRUE—UNKNOWN is insufficient

Example: is \( (\text{GPA} > 3.0 \ \text{OR} \ \text{GPA} \leq 3.0) \) always TRUE?
Constraints

*Integrity constraints* impose restrictions on allowable data in the database, in addition to the simple structure and type restrictions imposed by the basic schema definition

- Declared as a part of the schema
- Enforced by the DBMS: if a SQL statement causes a constraint to become violated then (in most cases) the statement is aborted and a runtime error is generated

Why use integrity constraints?

- To protect the integrity of the database (e.g., to catch data-entry errors or enforce consistency across data)
- To tell the DBMS about the data (e.g., the DBMS may choose to create indexes or optimize queries accordingly)

Types of constraints offered by SQL:

- Keys: PRIMARY KEY and UNIQUE
- NOT NULL: restricts attributes to not allow NULL values
  
  Example: CREATE TABLE Course(...,
                 title VARCHAR(100) UNIQUE NOT NULL);

- Referential integrity (a.k.a. foreign-key) constraints
- Attribute-based checks
- Tuple-based checks
- General assertions

Referential Integrity

Example:

- If an SID appears in Take then it must also appear in Student
- If an CID appears in Take then it must also appear in Course

\[ \rightarrow \text{The reverse is not necessarily true} \]

Terminology:

- Take.SID references Student.SID
- Take.CID references Course.CID
- Referential integrity means referenced value always exists

\[ \rightarrow \text{When we join the referencing table with the referenced table, there are no “dangling tuples” in the referencing table (but okay in the referenced table)} \]
Referential integrity in SQL:
- Referenced attribute must be PRIMARY KEY
- Referencing attribute is called FOREIGN KEY
- Two ways to declare referential integrity:
  - With the referencing attribute
  - Separate within the referencing table
    ↦ Necessary if the foreign key contains more than one attribute

Example:
```sql
CREATE TABLE Take
  (SID INTEGER REFERENCES Student(SID),
   CID CHAR(10),
   PRIMARY KEY(SID, CID),
   FOREIGN KEY CID REFERENCES Course(CID));
```

Referential Integrity Enforcement

Example: Take.SID references Student.SID
- Insert or update a Take tuple so it refers to a nonexistent student
  - Always reject
- Delete or update a Student tuple with a SID value referenced by some Take tuple
  - Reject (default)
  - Set NULL: set all references to NULL
  - Cascade: ripple changes to all referring tuples
    ↦ Desired policy can be specified in SQL:
    ```sql
    ON { DELETE | UPDATE } { CASCADE | SET NULL }
    ```
Which policy makes sense for Take.SID/Student.SID?

When Should Constraints Be Checked?

- Usually they are checked for each modification statement
- But sometimes deferred constraint checking is necessary
  ↦ Check only at the the end of a “transaction”

Example: the no-chicken-and-no-egg problem
```sql
CREATE TABLE Dept
  (name CHAR(20) PRIMARY KEY, chair CHAR(20) NOT NULL,
   REFERENCES Prof(name));
CREATE TABLE Prof
  (name CHAR(20) PRIMARY KEY, dept CHAR(20) NOT NULL,
   REFERENCES Dept(name));
```
Attribute-Based Check

Constraint on a single attribute:
- Syntax: follow the attribute by \texttt{CHECK (cond)}
  - Condition may involve the checked attribute
  - Other attributes and tables may be involved, but only in subqueries
- Semantics: condition is checked only when the associated attribute changes (i.e., an insert or update occurs)
Example: GPA’s must be between 0 and 4.3

Example: referential integrity constraint?

\texttt{CREATE TABLE Take}
\begin{verbatim}
  (SID INTEGER
   CHECK(SID IN (SELECT SID FROM Student)),
   ...);
\end{verbatim}
\texttt{\textcircled{~} No; not checked when a Student tuple is deleted!}

Tuple-Based Check

Constraint on a single tuple:
- Syntax: \texttt{CHECK (cond)}, not associated with any particular attribute
  - Condition may involve the all attributes of the table
  - Other tables may be involved, but only in subqueries
- Semantics: condition is checked only when a tuple of the associated table changes (i.e., an insert or update occurs)
Example: only Lisa can have a GPA higher than 4.0

General Assertion

Constraint on entire relation or entire database:
- Syntax: a stand-alone statement
  \texttt{CREATE ASSERTION assertionName CHECK (cond);}
- Semantics: condition is checked for each modification that could potentially violate it
Example: all students with GPA higher than 3.0 take CS145
Triggers

A trigger is an event-condition-action rule:

- When event occurs, test condition; if it is satisfied, execute action
- More general than constraints
- In SQL3 standard but not in SQL2

Trigger options:

- Possible events include:
  - INSERT ON table
  - DELETE ON table
  - UPDATE [OF attr] ON table
- Trigger can be:
  - Row-level: activated FOR EACH ROW modified
  - Statement-level: activated for each modification statement
- Action can be executed:
  - AFTER the triggering event
  - BEFORE the triggering event
  - INSTEAD OF the triggering event
- Condition and action can reference:
  - OLD tuple and NEW tuple in a row-level trigger
  - OLD_TABLE and NEW_TABLE in a statement-level trigger

Example: whenever there comes a new student with GPA higher than 3.0, make him/her take CS145

```
CREATE TRIGGER CS145AutoRecruit
AFTER INSERT ON Student
REFERENCING NEW AS newStudent
WHEN (newStudent.GPA > 3.0)
INSERT INTO Take VALUES(newStudent.SID, ’CS145’)
FOR EACH ROW;
```

Example: rewrite the same trigger without FOR EACH ROW

Example: maintain a list of students whose GPA dropped more than 1.0 to 2.0 or less