Relational Integrity Rules

- Why Integrity Rules?
- The Entity Integrity Rule
- The Referential Integrity Rule

Why Integrity Rules?

- A database can be viewed as a data model of an enterprise
- There are many business rules, regulations, and constraints
- Data integrity refers to the correctness of the database
- Data integrity is critical to a DBMS

- A DBMS should accept specification of integrity rules
- A DBMS should enforce the rules automatically
- Different DBMS understand different integrity rules

Entity Integrity Rule

- The primary key of a base relation can not contain any null value.

Notes:
- If the key is composite, none of its component can contain a null value.
- This rule is applied to the primary key, not to other alternate keys if any.

- Enforcement of this rule:
  An update which results in null values in the primary key of a base relation must be rejected.
  - To enforce this rule, the DBMS must understand the primary key concept
  - Otherwise, this rule can only be enforced by an application program
Referential Integrity Rule

- If FK is the foreign key of R2 which matches the primary key PK of R1, then either
  1. Every value of FK must be equal to a value of PK in some tuple of R1, or
  2. It is a null value if FK is not the primary key of its containing relation.

- Enforcement of this Rule:
  An update on either a referenced primary key or on a foreign key must satisfy this rule. Otherwise, it is rejected.

Which operation on the primary key may violate this rule?
Which operation on the foreign key may violate this rule?

Referential Integrity Enforcement

- The first question that needs to be asked is whether or not the foreign key may accept null values? The solution to this question depends if a null value may make sense for a portion of the real world that is being modeled.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Restricted</th>
<th>Cascades</th>
<th>Nullifies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete FK</td>
<td>If it is referenced, reject the deletion</td>
<td>In addition, if it is referenced, delete all tuples that reference to it</td>
<td>If it is referenced, set all references to null. But, if any FK is a key attribute of containing relation, reject the deletion.</td>
</tr>
<tr>
<td>Modify FK (includes Insert &amp; Update)</td>
<td>If its old value is referenced, reject the modification</td>
<td>In addition, If its old value is referenced, change all foreign key references to its new value</td>
<td>If its old value is reference, perform the same action as above.</td>
</tr>
</tbody>
</table>

A Pseudo-DDL For Integrity

Foreign Key (foreign-key) IDENTIFIES target
NULLS [NOT] ALLOWED
DELETE OF target effect
UPDATE OF target:primary-key effect

Example:

CREATE TABLE SHIPMENT
  (S# INTEGER, P# INTEGER, QTY INTEGER)
  PRIMARY KEY (S#,P#)
  FOREIGN KEY (S#) IDENTIFIES SUPPLIER
  NULLS NOT ALLOWED
  DELETE OF SUPPLIER RESTRICTED
  UPDATE OF SUPPLIER.S# CASCADES
  FOREIGN KEY (P#) IDENTIFIES PART
  NULLS NOT ALLOWED
  DELETE OF PART RESTRICTED
  UPDATE OF PART.P# RESTRICTED
Exercise

Using the given STUDENT and CLASS relations, discuss how to enforce the referential integrity constraint for the following operations according to the three schemes.

1. Delete the 's1' tuple from STUDENT.
2. Delete the 's5' tuple from STUDENT.
3. Replace 's1' by 's6' in STUDENT.
4. Replace 's5' by 's8' in STUDENT.
5. Insert a new 's7' tuple into STUDENT.
6. Insert a new 's10' tuple into CLASS.
7. Insert a new 's5' tuple into CLASS.
8. Delete the 's2' tuple from CLASS.
9. Replace 's2' by 's5' in CLASS.
10. Replace 's3' by 's11' in CLASS.