Data Integrity

Hi! Here in this lecture we are going to discuss about the data integrity part of relational data model

Data Integrity

We noted at the beginning of the previous lecture the relational model has three main components; data structure, data manipulation, and data integrity. The aim of data integrity is to specify rules that implicitly or explicitly define a consistent database state or changes of state. These rules may include facilities like those provided by most programming languages for declaring data types which constrain the user from operations like comparing data of different data types and assigning a variable of one type to another of a different type. This is done to stop the user from doing things that generally do not make sense. In a DBMS, integrity constraints play a similar role.

The integrity constraints are necessary to avoid situations like the following:

1. Some data has been inserted in the database but it cannot be identified (that is, it is not clear which object or entity the data is about).
2. A student is enrolled in a course but no data about him is available in the relation that has information about students.
3. During a query processing, a student number is compared with a course number (this should never be required).
4. A student quits the university and is removed from the student relation but is still enrolled in a course.

Integrity constraints on a database may be divided into two types:

1. *Static Integrity Constraints* - these are constraints that define valid states of the data. These constraints include designations of primary keys etc.
2. *Dynamic Integrity Constraints* - these are constraints that define side-effects of various kinds of transactions (e.g. insertions and deletions).

We now discuss certain integrity features of the relational model. We discuss the following features:

1. Primary Keys
2. Domains
3. Foreign Keys and Referential Integrity
4. Nulls

Primary Keys

We have earlier defined the concept of candidate key and primary key. From the definition of candidate key, it should be clear that each relation must have at least one candidate key even if it is the combination of all the attributes in the relation since all tuples in a relation are distinct. Some relations may have more than one candidate keys.

As discussed earlier, the primary key of a relation is an arbitrarily but permanently selected candidate key. The primary key is important since it is the sole identifier for the tuples in a relation. Any tuple in a database may be identified by specifying relation name, primary key and its value. Also for a tuple to exist in a relation, it must be *identifiable* and therefore it must have a primary key. The relational data model therefore imposes the following two integrity constraints:

(a) no component of a primary key value can be null;
(b) attempts to change the value of a primary key must be carefully controlled.

The first constraint is necessary because if we want to store information about some entity, then we must be able to identify it, otherwise difficulties are likely to arise. For example, if a relation

CLASS (STUNO, LECTURER, CNO)

has (STUNO, LECTURER) as the primary key then allowing tuples like

```
3123    NULL    CP302
NULL   SMITH    CP302
```

is going to lead to ambiguity since the two tuples above may or may not be identical and the integrity of the database may be compromised. Unfortunately most commercial database systems do not support the concept of primary key and it would be possible to have a database state when integrity of the database is violated.

The second constraint above deals with changing of primary key values. Since the primary key is the tuple identifier, changing it needs very careful controls. Codd has suggested three possible approaches:

Method 1

Only a select group of users be authorised to change primary key values.

Method 2

Updates on primary key values be banned. If it was necessary to change a primary key, the tuple would first be deleted and then a new tuple with new primary key value but same other values would be inserted. Of course, this does require that the old values of attributes be remembered and be reinserted in the database.

Method 3

a different command for updating primary keys be made available. Making a distinction in altering the primary key and another attribute of a relation would remind users that care needs to be taken in updating primary keys.

Advantages and disadvantages of each to be discussed.

<u>Domains</u>

We have noted earlier that many commercial database systems do not provide facilities for specifying domains. Domains could be specified as below:

```
create   DOMAIN   NAME1   CHAR(10)
create   DOMAIN   STUNO   INTEGER
create   DOMAIN   NAME2   CHAR(10)
etc.
```

Note that *NAME1* and *NAME2* are both character strings of length 10 but they now belong to different (semantic) domains. It is important to denote different domainsto

(a) Constrain unions, intersections, differences, and equijoins of relations.
(b) Let the system check if two occurrences of the same database value denote the same real world object.

The constrain on union-compatibility and join-compatibility is important so that only those operations that make sense are permitted. For example, a join on class number and student number would make no sense even if both attributes are integers and the user should not be permitted to carry out such operations (or at least be warned when it is attempted).

<u>Domain Constraints</u>

All the values that appear in a column of a relation must be taken from the same domain. A domain usually consists of the following components.

1. Domain Name
2. Meaning
3. Data Type
4. Size or length
5. Allowable values or Allowable range(if applicable)

Entity Integrity

The Entity Integrity rule is so designed to assure that every relation has a primary key and that the data values for the primary key are all valid. Entity integrity guarantees that every primary key attribute is non null. No attribute participating in the primary key of a base relation is allowed to contain nulls. Primary key performs unique identification function in a relational model. Thus a null primary key performs the unique identification function in a relation would be like saying that there are some entity that had no known identity. An entity that cannot be identified is a contradiction in terms, hence the name entity integrity.

Operational Constraints

These are the constraints enforced in the database by the business rules or real world limitations. For example if the retirement age of the employees in a organization is 60, then the age column of the employee table can have a constraint “Age should be less than or equal to 60”. These kinds of constraints enforced by the business and the environment are called operational constraints.

Null constraints

`NOT NULL` constraint restricts attributes to not allow `NULL` values.

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); or
INSERT INTO Student (SID, name) VALUES (135, ’Maggie’);

Foreign Keys and Referential Integrity

We have earlier considered some problems related to modifying primary key values. Additional problems arise because primary key value of a tuple may be referred in many relations of the database. When a primary key is modified, each of these references to a primary key must either be modified accordingly or be replaced by NULL values. Only then we can maintain referential integrity.

Before we discuss referential integrity further, we define the concept of a *foreign key*. The concept is important since a relational database consists of relations only (no pointers) and relationships between the relations are implicit, based on references to primary keys of other relations. These references are called foreign keys.

We now define foreign key. ***A foreign key in a relation R is a set of attributes whose values are required to match those of the primary key of some relation S.***

In the following relation the supervisor number is a foreign key (it is the primary key of *employee*)

employee (empno, empname, supervisor-no, dept)

In the following relation *Class*, *student-num* and *lecturer-num* are foreign keys since they appear as primary keys in other relations (relations student and lecturer).

Class (student-num, lecturer-num, subject)
student ( )
lecturer ( )

Foreign keys are the implicit references in a relational database. The following constraint is called referential integrity constraint:

If a foreign key F in relation R matches the primary key P of relation S than every value of F must either be equal to a value of P or be wholly null.

The justification for referential integrity constraint is simple. If there is a foreign key in a relation (that is if the relation is referring to another relation) then its value must match with one of the primary key values to which it refers. That is, if an object or entity is being referred to, the constraint ensures the referred object or entity exists in the database.

In the relational model the association between the tables is defined using foreign keys. The association between the SHIPMENT and ELEMENT tables is defined by including the Symbol attribute as a foreign key in the SHIPMENT table. This implies that before we insert a row in the SHIPMENT table, the element for that order must already exist in the ELEMENT table.

A referential integrity constraint is a rule that maintains consistency among the rows of two tables or relations. The rule states that if there is a foreign key in one relation, either each of the foreign key value must match a primary key value in the other table or else the foreign key value must be null.

When Should Constraints Be Checked?

- Usually they are checked for each modification statement.
- But sometimes deferred constraint checking is necessary.

Review Questions.

1. Define Data Integrity
2. What is a primary key
3. Define referential integrity
4. Define a foreign key
5. Define domains constraints

1. **Codd, E. F. (1974), "A Relational Model of Data for Large Shared Data Banks", Comm. ACM, Vol. 13, No. 6, pp. 377-387.**
2. **Codd, E. F. (1974), "Recent Investigations in Relational Data Base Systems", in Information Processing 74, North Holland.**
3. **Codd, E. F. (1982), "Relational Database: A Practical Foundation for Productivity", in Comm. ACM, Vol. 25, No. 2, pp. 109-117.**

4. **Codd, E. F. (1981), "The Capabilities of Relational Database Management Systems", IBM Report RJ3132.**
5. **Codd, E. F. (1971), "Relational Completeness of Data Base Sublanguages" Courant Computer Science Symposium 6, Data Base Systems, Prentice-Hall.**
6. **Codd, E. F. (1971), "Further Normalization of the Data Base Relational Model", Courant Computer Science Symposium 6, Data Base Systems, Prentice-Hall.**
7. **C. J. Date (1987) "A Guide to the SQL Standard", Addison Wesley.**

Summary

The explosion of information made available to enterprise applications by the broad-based adoption of Internet standards and technologies has introduced a clear need for an information integration platform to help harness that information and make it available to enterprise applications. The challenges for a robust information integration platform are steep. However, the foundation to build such a platform is already on the market. DBMSs have demonstrated over the years a remarkable ability to manage and harness structured data, to scale with business growth, and to quickly adapt to new requirements. We believe that a federated DBMS enhanced with native XML capabilities and tightly coupled enterprise application services, content management services and analytics is the right technology to provide a robust end-to-end solution.