Databases Design. Introduction to SQL

LECTURE 3

Normalization

Review of last lecture

- Previously, we looked at designing databases using ER models

- There was one question that this process leads to:

  what constitutes a “good” database design?

Today’s lecture

- We’ll discuss the criteria used to evaluate a database design.
- We'll formalize our definition of a “good” database design using the concept of functional dependencies.
- We'll look at how to fix a “bad” database design using the process of normalization.

“Good” database design

What criteria would you say constitutes a good database design?

“Good” database design

Good database design exhibits the following characteristics:

- Tables group only related data
- Tables store no redundant data
- NULL values are minimized

“Bad” database design

Consider the following table storing teachers and departments information for a university:

Teacher_id	Last_name	Dep_id	Dep_name	Room
001	Teacher1	001	CET	409
002	Teacher2	001	CET	409
003	Teacher3	002	IS	803

Update anomalies

- Clearly, this relation is a bad design because it stores redundant data, which is one of our criteria for evaluating database design.
- Data redundancy, such as in this relation, leads to three types of **update anomalies**:
    - Modification anomaly
    - Insertion anomaly
    - Deletion anomaly

Insertion anomaly

Insertion anomaly occurs when we are prevented from inserting some data into a relation until other data can be supplied.

Insertion anomaly

There are two main types of insertion anomaly:

- To insert the details of a new teacher into the relation, we must include the details of the department at which the teacher are to be located.
- To insert details of a new department that currently has no teachers into the relation, it is necessary to enter nulls into the attributes for teachers, such as Teacher_id. However, as Teacher_id is the Primary key for the relation, attempting to enter nulls for Teacher_id violates entity integrity, and is not allowed.

Deletion anomaly

- **Deletion anomaly** occurs when a deletion leads to an unintended loss of data.

- If we delete a tuple from the relation that represents the last teacher located at a department, the details about that department are also lost from the database.
- For example, if we delete the tuple Teacher_id=003, the details relating to IS department are lost from the database.

Modification anomalies

- If a relation suffers a **modification anomaly**, it is possible that not all data that needs to be changed will always be changed.
- A modification anomaly typically leads to **inconsistent data** because of missed updates.

Modification anomalies

- If we want to change the value of one of the attributes of a particular department in the relation, for example the room for CET department, we must update the tuples of all teachers located at that department.

- If this modification is not carried out on all the appropriate tuples, the database will become inconsistent.

Functional dependencies

- A "good" database table design grouped related data.
- Previous example was a "bad" design because two sets of related data were stored in one table: teachers and departments information.

Functional dependencies

- We can formalize the concept of a table grouping only related data using functional dependencies.

- **Functional dependencies** describe relationships between attributes.
- Stated another way, functional dependencies tell which attributes are uniquely determined by the primary key.

Functional dependencies

- For example, Students relation. Each student has a student_id (PK), a last name, a birthdate and a gpa.
- Since the student_id is the PK, we can use it to uniquely identify each student – we can use it to find a unique last name, birthdate, gpa for a particular student.
- Thus, last name, birthdate and gpa are dependent on the PK.

Functional dependencies

- Functional dependencies are not limited to primary key values.
- Generally, a functional dependency may be used to describe the relationship between any two attributes (columns) in a relation.
- We denote functional dependencies as

  X -> Y

  when a value of X uniquely determines a value of Y.

Full dependencies

- A **full dependency**, X -> Y, exists in a relation if there is no attribute A that can be removed from X and the dependency still holds.

  Consider a relation Students (student_id, last_name, birthdate, gpa). The following FDs exist in this relation:

  {student_id} -> {last_name, birthdate, gpa}

  This is a full dependency because no attribute may be removed from the left hand side and the dependency still holds.

Partial dependencies

- The inverse of a full dependency is a partial dependency.

- A dependency X -> Y is a **partial dependency** if there exists an attribute A that is part of X that can be removed from X and the dependency still holds

Composite primary key

- **Composite Primary key** – a primary key that consists of two or more attributes.
- A **primary key** is defined as a key or database column which uniquely identifies each row in a database table.
  A **composite key** is a set of more than one column that, together, uniquely identifies each record.

Partial dependencies

- Let‘s discuss Teachers&Courses table.

Teacher_id	Last_name	Course_id	Course_name	Credits
001	Teacher1	001	SDP1	3
001	Teacher1	002	SDP2	3
002	Teacher2	001	SDP1	3

- Suppose, PK is {teacher_id, course_id}

Partial dependencies

- Also suppose this relation has the following dependencies:

  FD1: {teacher_id, course_id} -> {last_name, course_name, credits}

  FD2: {teacher_id} -> {last_name}

  FD3: {course_id} -> {course_name, credits}

- FD1 is a partial dependency
- FD2 and FD3 are full dependencies.

Transitive dependencies

- Suppose X, Y and Z are columns in R.
- Also suppose that X -> Y and Y -> Z are dependencies in R.
- This dependency is transitive.

Transitive dependency example

- Suppose we had the following table storing information about students and their groups.
- PK is student_id.

Student_id	Last_name	Group_id	Group_name
001	Student1	001	Group1
002	Student2	001	Group1
003	Student3	002	Group2

Transitive dependency example

- The relation has the following functional dependencies:

  {student_id} -> {last_name, group_id, group_name}

  {group_id} -> {group_name}
- This relation contains a transitive dependency because

  {student_id} -> {group_id} -> {group_name}

Decomposition

- **Decomposition** is the process of breaking down in parts or elements.
- It breaks the table into multiple tables in a database.

- It should always be lossless, because it confirms that the information in the original relation can be accurately reconstructed based on the decomposed relations.

Books

- **Connolly, Thomas M. Database Systems**: A Practical Approach to Design, Implementation, and Management / Thomas M. Connolly, Carolyn E. Begg.- United States of America: Pearson Education

- **Garcia-Molina, H. Database system**: The Complete Book / Hector Garcia-Molina.- United States of America: Pearson Prentice Hall

- **Sharma, N. Database Fundamentals**: A book for the community by the community / Neeraj Sharma, Liviu Perniu.- Canada

Questions

The column of a table is referred to as the

a) Tuple
b) Attribute
c) Entity
d) Degree

Questions

The another name for a row is

a) Tuple
b) Attribute
c) Entity
d) Degree

Questions

A primary key for an entity is

a) A tuple
b) Any attribute
c) A unique attribute
d) A relationship

Questions

In E-R Diagram relationship type is represented by

a) Ellipse
b) Dashed ellipse
c) Rectangle
d) Diamond (rhombus)

Questions

Key to represent relationship between tables is called

a) Primary key
b) Secondary Key
c) Foreign Key
d) None of these

Questions

An entity relationship diagram is a tool to represent

a) Data model
b) Process model
c) Event model
d) Customer model

Question

The FD X -> Y is a full dependency in a relation R, if there is_____attribute A that can be_____X and the dependency still holds.

a)At least one, added to

b)No, added to

c)No, removed from

d)At least one, removed from

Question

The FD X -> Y is a partial dependency in a relation R, if there is_____attribute A that can be_____X and the dependency still holds.

a)At least one, added to

b)No, added to

c)No, removed from

d)At least one, removed from