SCHEMA REFINEMENT and NORMAL FORMS

Session 2:FUNCTIONAL DEPENDENCIES

Functional Dependencies

- A *functional dependency (FD)* is a type of integrity constraint (IC) that plays a crucial role in the process of database schema design, particularly in the normalization process.
- Functional dependencies generalize the concept of a key by establishing relationships between different sets of attributes within a relation (table).
- Understanding FDs is essential for ensuring that the database schema is well-structured and free from redundancy and anomalies.

Functional Dependencies

- **Definition of Functional Dependency:**
- Let R be a relation schema, and let X and Y be nonempty sets of attributes in R . We say that an instance r of R satisfies the Functional Dependency X->Y if the following condition holds for every pair of tuples t1 and t2 in r :
  - If t1:X = t2:X , then t1:Y = t2:Y .
  - This means that if two tuples in the relation have the same values for the attributes in X , then they must also have the same values for the attributes in Y .
- Notation:
  - t1:X : This notation refers to the projection of the tuple t1 onto the attributes in X . In simpler terms, it means we are considering the values of the attributes in X for the tuple t1 .
  - X->Y : This notation means that X functionally determines Y . In other words, if the values of X are known, the values of Y are uniquely determined.

- Example:
- Consider the functional dependency AB ->C. This means that if two tuples in the relation agree on the values of attributes A and B , they must also agree on the value of attribute C.

- Example:

A	B	C	D
a1	b1	c1	d1
a1	b1	c1	d2
a1	b2	c2	d1
a2	b1	c3	d1

An Instance that Satisfies $AB \rightarrow C$

- In this table:
  - The first two rows show that AB is not a key because even though AB determines C , it does not uniquely determine all the attributes in the table (specifically, the attribute D varies while AB remains the same.
  - The third and fourth rows show that if A or B differs, then C can also differ without violating the FD.
  - If we add a new tuple (a1, b1, c2, d1), the resulting instance would violate the FD : AB -> C , because for the pair of tuples (a1, b1), (C) would have different values c1 and c2 , which contradicts the functional dependency.

Legal Instances and Integrity Constraints:

- A legal instance of a relation must satisfy all specified integrity constraints, including all specified functional dependencies.
- Integrity constraints are derived from the semantics of the real-world scenario being modeled by the database.
- By looking at an instance of a relation, we might be able to tell that a certain FD does *not* hold.
- However, we can never deduce that an FD *does* hold by looking at one or more instances of the relation because an FD, like other ICs, is a statement about *all* possible legal instances of the relation.
- It is important to understand that functional dependencies cannot be deduced by just looking at a few instances of the data.
- An FD, like other integrity constraints, applies to all possible legal instances of the relation.
  - For instance, just because a set of data currently satisfies A ->B , it doesn't mean A always determines B unless it’s explicitly defined as an FD based on the business rules or logic.

Functional Dependencies and Keys

- A primary key is a special case of a functional dependency.
  - In a primary key constraint, the attributes in the key play the role of X , and the set of all attributes in the relation plays the role of Y . This means the key uniquely determines all other attributes in the table.
- The definition of a functional dependency does not require that the set X be minimal. For X to be a key, the following conditions must be met:
  - Minimality: There should not be a subset V of X such that V->Y. In other words, no proper subset of the key should functionally determine all attributes.
  - Superkey vs. Key: If X ->Y holds for all attributes, but there exists a subset V of X such that V->Y, then X is a superkey, but it is not a minimal key.