Theory of Normal Forms

Overview

What is a Normal Form?

A **normal form** of a database is a **collection of conditions (constraints)** on the database schema. If the database schema *satisfies* these conditions, we say that **the database is in the normal form**.

Normal Forms in a nutshell. Brief descriptions of normal forms.

- First Normal Form (1NF): no repeated attribute groups, atomic values of all attributes.
- Second Normal Form (2NF): attributes depend on entire keys.
- Third Normal Form (3NF): no inderect dependencies.
- Boyce-Codd Normal Form (BCNF): all functional dependencies involve superkeys.
- Fourth Normal Form (4NF): no redundant independent multiple many-to-many relationships.
- Fifth Normal Form (5NF)/Project-Join Normal Form (PJNF): semantically related multiple many-to-many relationships are isolated.
- Domain-Key Normal Form (DKNF): only key and domain constraints are present in the relation.

In the course, we study 1NF, 2NF, 3NF and BCNF. Higher forms of normalization (i.e., additional constraints on the structure of the database schema) are needed only on rare occasions.

Why Normal Forms?

Normal forms allow us to ensure that our database designs possess some desired properties. More importantly, normal forms *eliminate* ***database anomalies***.

Database Anomalies. A **database anomaly** is a *problem* with database maintenance/management *caused by the database schema.*

Some database anomalies are listed below.

Redundancy: Same information may be repeated multiple times. This *wastes disk space* and may lead to *wasted effort* and *inconsistencies* when database is changed.

Update anomalies: If the same data is stored in multiple places, a *database update* may change it in one place and keep it unchanged in another.

Insertion anomalies: In some situations, new information cannot be recorded at all. This may happen if the new information forms only part of a database tuple, and does not contain the tuple's full primary key.

Deletion anomalies: In some situations, deleting a tuple from the database leads to wiping out data that should have been kept.

The constraints on database schemas forced by normal forms prevent the possibility of such anomalies. Higher normal forms deal with less frequent and more "tricky" anomalies.

Dealing with Anomalies

The *standard* way of eliminating anomalies in database design is **relational table decomposition**.

Relational table decomposition: replacement of a relational table $R(A_1, \ldots, A_n)$ with relational tables R_1, $R_2, \ldots, R_k$. We want relational decomopositions to have the following properties:

P1. For all i, $R_i = \pi_L(R)$, where $L \subset \{A_1, \ldots, A_n\}$.

P2. $R_1 \bowtie R_2 \bowtie \ldots R_k = R$.

Decompositions are designed to address the following issues:

- Eliination of anomalies. Property **P1** usually takes care of that.
- Recoverability of information. This is achieved by decompositions satisfying property **P2**.
- Preservation of dependencies. This is an important challenge. It may affect how far towards normalization we can get.

Prime and Non-prime attributes

Prime attribute: an attribute of a relational table R is **prime** if *it appears in **at least one** key of R.*

Non-prime attribute: an attribute of a relational table is **non-prime** if it *does not appear in any key of R.*

Note: Intuitively, **non-prime attributes** of a table must functionally depend on some other attributes. The level of relational table normalization is determined by the structure of such dependencies.

Normal Forms

First Normal Form (1NF)

First Normal Form (1NF). A relatal table R is in **First Normal Form (1NF)** if and only if it *faithfully represents a relation.* This means the following:

1. There is no top-to-bottom ordering to the rows.
2. There is no left-to-right ordering to the columns.
3. There are no duplicate rows.
4. Every cell (row and column intersection) contains exactly one value from the applicable domain.

Implications of 1NF:

- 1NF tables must have keys — this prevents duplicate rows.
- Order of tuples in 1NF tables is irrelevant.
- So is the order of attributes. (in fact, physical representations of relational tables may rearrange the order of attributes from the one given in CREATE TABLE statements to suit the purposes of efficiency.)
- The key to remembering what 1NFs are about is: **1NF tables have atomic values in cells**.
- Condition 4. may be read to imply that 1NF tables may not have **NULL values**. This is a point of contention in the database community. We will refrain for taking a position here.
- 1NF tables prohibit repeated attribute groups.

Note: `NULL` values issue aside, Non-1NF tables are very rare. E-R models create entity sets where attributes take atomic values. Translation of E-R models into relational tables produces 1NF tables.

Second Normal Form (2NF)

Second Normal Form (2NF). A relation R is in **Second Normal Form (2NF)** if it satisfies the following conditions:

1. R is in **1NF**.
2. every *non-prime attribute* of R is functionally determined by **some full key of** R.

Intuitively: in **2NF** tables non-prime attributes *cannot depend on proper subsets of keys.*

Implications of **2NF**.

- Every 1NF table with a single key consisting of one attribute is in 2NF.
- Every 1NF table with no non-prime attributes (i.e. a table, each attribute of which is part of some key) is in 2NF.

Third Normal Form (3NF)

Third Normal Form (3NF). A relational table R is in **Third Normal Form (3NF)** if it satisfies the following conditions:

- R is in 2NF.
- Each *non-prime attribute* of R is **non-transitively** functionally determined by **every key of** R.

Intuition: In 3NF tables there cannot be *transitive FDs* for non-prime attributes. I.e., If C is non-prime, it cannot be the case, that $X \rightarrow Y$, $Y \rightarrow C$ hold, when Y itself is not a key.

Implications of 3NF:

- The idea of 3NF is that non-prime attributes must depend on keys of the table without any intermediate FDs. Whenever this is not the case, relational table will contain redundant information, which in turn would make possible anomalies of insertion, deletion and update.
- Using decomposition, every relational table can be represented as a collection of tables in 3NF. (This is also true for 2NF). This will not be true for **Boyce-Codd Normal Form (BCNF)**, the next database normal form.

3NF: Alternative definition

There is an alternative definition of 3NF, which allows for better contrast between 3NF and Boyce-Codd Normal Form:

> A table R is in **Third Normal Form (3NF)** if the following conditions are met:
>
> - R is in $2NF$.
> - For each FD $X \rightarrow A$ that holds for R, **at least one of the following** is true:
>   1. X contains A (i.e., $X \rightarrow A$ is a trivial dependency).
>   2. X is a superkey.
>   3. A is **prime**.

Boyce-Codd Normal Form (BCNF)

Boyce-Codd Normal Form (BCNF). A relational table R is in **Boyce-Codd Normal Form** if it satisfies the following conditions:

- R is in $2NF$.
- For each FD $X \rightarrow A$ that holds for R, **at least one of the following** is true:
  1. X contains A (i.e., $X \rightarrow A$ is a trivial dependency).
  2. X is a superkey.

That is, R is in BCNF if the left side of all its *nontrivial FDs* contains a key.

Observations:

- BCNF is the strongest normal form for tables which do not contain representations of multiple relational sets. (4NF and 5NF deal with such cases). Thus, a typical goal, when designing databases is to attempt to have as many tables in the database schema as possible in BCNF, and put the remaining tables in 3NF.

- Most 3NF tables will be in BCNF. In particular, if R has only one key and R is in 3NF, it will also be in $BCNF$.
- Most tables created as the result of E-R model translation will be in 3NF, and consequently, most of them will also be in BCNF.