DATABASE NORMALIZATION—CHAPTER EIGHT

Introduction

Normalization is a formal database design process for grouping attributes together in a data relation. Normalization takes a "micro" view of database design while entity-relationship modeling takes a "macro view." Normalization validates and improves the logical design of a data model. Essentially, normalization removes redundancy in a data model so that table data are easier to modify and so that unnecessary duplication of data is prevented.

Definitions

To fully understand normalization, relational database terminology must be defined. An **attribute** is a characteristic of something—for example, hair colour or a social insurance number. A **tuple** (or row) represents an entity and is composed of a set of attributes that defines an entity, such as a student. An **entity** is a particular kind of thing, again such as student. A **relation** or **table** is defined as a set of tuples (or rows), such as students. All rows in a relation must be distinct, meaning that no two rows can have the same combination of values for all of their attributes. A set of relations or tables related to each other constitutes a **database**.

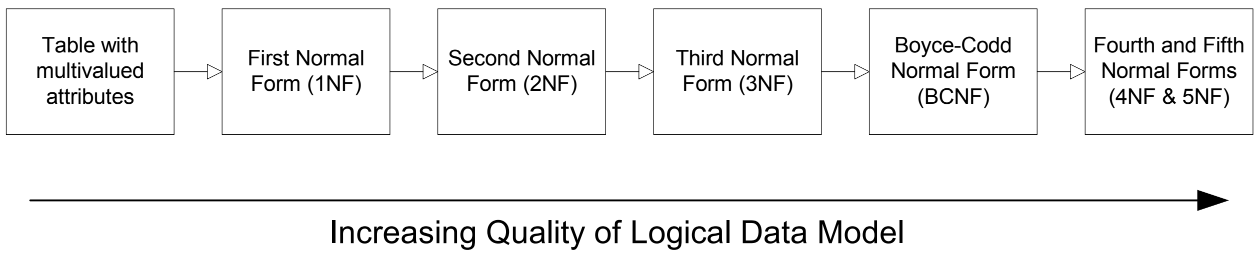
Because no two rows in a relation are distinct, as they contain different sets of attribute values, a **superkey** is defined as the set of all attributes in a relation. A **candidate key** is a minimal superkey, a smaller group of attributes that still uniquely identifies each row. One candidate key is designated as the **primary key** of the relation and is usually used to link one relation or table with another. A **prime attribute** is any attribute that is part of the table's primary key. Conversely, a **non-prime attribute** is an attribute that is not part of the primary key.

The **entity integrity constraint** states that no primary key value can be null or valueless. This is because the primary key value is used to identify individual rows in a relation. The **referential integrity constraint** is specified between two relations and is used to maintain the consistency among rows of a relation. A **functional dependency** is a constraint between two attributes or sets of attributes. For example, for any relation R, attribute B is functionally dependent on attribute A if, and only if, for every valid instance of A, the value of A uniquely determines the value of B.

Normal Forms

The normalization process can be accomplished and understood in progressive steps, or stages (Figure 1). Each stage is formally known as a **normal form**, the state of a relation that results from applying simple rules about functional dependencies to that relation. As relations reach each stage or form of normalization, the quality of the logical data model increases in terms of reduced data redundancy and fewer data anomalies.

Figure 1



First Normal Form (1NF)

A relation is considered to be in first normal form (1NF) if it contains no multi-valued attributes. To convert an unnormalized table into 1NF, replace each value of a repeating group with a row. In a new row, copy the non-repeating columns. However, tables that contain multi-valued attributes or repeating groups are not considered relations. Therefore, most tables will not need to be converted into 1NF.

Second Normal Form (2NF)

A relation is in second normal form (2NF) if it satisfies the first normal form requirements, and if every non-key attribute is fully functionally dependent on the primary key. This means that each non-key attribute in a row must be dependent on the entire primary key, not simply part of it. If a subset of the primary key determines any attribute, there is a "partial dependency," and the table is *not* in 2NF.

To convert a relation to 2NF, the relation must be decomposed into two or more relations, in which the non-key attributes are dependent on the primary key as a whole. Traditionally, three situations exist where a relation in 1NF will automatically be considered to be in 2NF:

1. The primary key consists of only one attribute.
2. No non-key attributes exist in the relation (all attributes are part of the primary key).
3. Every non-key attribute is fully functionally dependent on the full set of primary key attributes.

Third Normal Form (3NF)

A relation is in third normal form (3NF) if it satisfies the first and second normal form requirements, and if no transitive dependencies exist. A **transitive dependency** is a functional dependency between two or more non-key attributes. A relation with transitive dependencies will create insertion, deletion, and modification anomalies that can harm data structures.

Decomposing the single relation into two separate relations can eliminate transitive dependencies, thus converting the new relations into third normal form. If a non-prime attribute determines another non-prime attribute, there is a transitive dependency, and the table is *not* in 3NF.

Boyce-Codd Normal Form (BCNF)

A revised third normal form exists, known as Boyce-Codd Normal Form (BCNF). If a non-prime attribute determines another prime attribute, the table is *not* in BCNF. A relation is in Boyce-Codd normal form if it satisfies the first, second, and third normal form requirements and every determinant is a candidate key. Remember that a candidate key is a minimal superkey, a smaller group of attributes that still uniquely identifies each row. This means that every attribute that exhibits control over another attribute can uniquely define each row.

Fourth and Fifth Normal Form (4NF and 5NF)

These normal forms deal with multivalued and join dependencies, which is far beyond the scope of most data structures. Most logical data models are seen as very efficient if they have achieved 3NF or BCNF level. Therefore, these normal forms will not be discussed further in this section.