Chapter 4
The Relational Model and Normalization

Relations
- Relational DBMS products store data in the form of relations, a special type of table
- A relation is a two-dimensional table that has the following characteristics
  - Rows contain data about an entity
  - Columns contain data about attributes of the entity
  - Cells of the table hold a single value
  - All entries in a column are of the same kind
  - Each column has a unique name
  - The order of the columns is unimportant
  - The order of the rows is unimportant
  - No two rows may be identical
- Although not all tables are relations, the terms table and relation are normally used interchangeably
  - Table/row/column = file/record/field = relation/tuple/attribute

Example: Relation

Types of Keys
- A key is one or more columns of a relation that identifies a row
- A unique key identifies a single row; a non-unique key identifies several rows
- Composite key is a key that contains two or more attributes
- A relation has one unique primary key and may also have additional unique keys called candidate keys
- Primary key is used to
  - Represent the table in relationships
  - Organize table storage
  - Generate indexes

Example: Tables Not Relations

Functional Dependencies
- A functional dependency occurs when the value of one (set of) attribute(s) determines the value of a second (set of) attribute(s)
- The attribute on the left side of the functional dependency is called the determinant
  - SID → DormName, Fee
  - (CustomerNumber, ItemNumber, Quantity) → Price
- While a primary key is always a determinant, a determinant is not necessarily a primary key
Normalization

- Normalization eliminates modification anomalies
  - Deletion anomaly: deletion of a row loses information about two or more entities
  - Insertion anomaly: insertion of a fact in one entity cannot be done until a fact about another entity is added
- Anomalies can be removed by splitting the relation into two or more relations; each with a different, single theme
- However, breaking up a relation may create referential integrity constraints
- Normalization works through classes of relations called normal forms

Relationship of Normal Forms

![Diagram showing the relationship of normal forms](image)

**Figure 4.7** Relationship of Normal Forms

- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)
- Boyce-Codd Normal Form (BCNF)
- Fourth Normal Form (4NF)
- Fifth Normal Form (5NF)

* Domain/Key Normal Form (DKNF)

Normal Forms

- Any table of data is in 1NF if it meets the definition of a relation.
- A relation is in 2NF if all its non-key attributes are dependent on all of the key (no partial dependencies).
  - If a relation has a single attribute key, it is automatically in 2NF.
- A relation is in 3NF if it is in 2NF and has no transitive dependencies.
- A relation is in BCNF if every determinant is a candidate key.
- A relation is in fourth normal form if it is in BCNF and has no multi-value dependencies.

Example: 3NF

![Table showing elimination of transitive dependency](image)

**Figure 4.9a** Elimination of Transitive Dependency — Relations Eliminating the Transitive Dependency

<table>
<thead>
<tr>
<th>STUHOUSING (SID, Dorn)</th>
<th>BLDG-FEE (Dorn, Fee)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SID</td>
<td>Dorn</td>
</tr>
<tr>
<td>100</td>
<td>Randolph</td>
</tr>
<tr>
<td>150</td>
<td>Ingersoll</td>
</tr>
<tr>
<td>200</td>
<td>Randolph</td>
</tr>
<tr>
<td>250</td>
<td>Pitkin</td>
</tr>
<tr>
<td>300</td>
<td>Randolph</td>
</tr>
</tbody>
</table>

Example: BCNF

![Table showing elimination of transitive dependency](image)

**Figure 4.9b** Elimination of Transitive Dependency — Relations Eliminating the Transitive Dependency

<table>
<thead>
<tr>
<th>ADRS (SID, Major, Frame)</th>
<th>Key (candidate): (SID, Frame)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SID</td>
<td>Major</td>
</tr>
<tr>
<td>100</td>
<td>Math</td>
</tr>
<tr>
<td>150</td>
<td>Psychology</td>
</tr>
<tr>
<td>200</td>
<td>Math</td>
</tr>
<tr>
<td>250</td>
<td>Math</td>
</tr>
<tr>
<td>300</td>
<td>Psychology</td>
</tr>
<tr>
<td>350</td>
<td>Math</td>
</tr>
</tbody>
</table>
Example: BCNF

Figure 4.10a: Boyce-Codd Normal Form — Relations in Boyce-Codd Normal Form

<table>
<thead>
<tr>
<th>SID</th>
<th>Frame</th>
<th>Subject</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Cauhy</td>
<td>Math</td>
</tr>
<tr>
<td>150</td>
<td>Jung</td>
<td>Psychology</td>
</tr>
<tr>
<td>200</td>
<td>Riemann</td>
<td>Math</td>
</tr>
<tr>
<td>250</td>
<td>Cauhy</td>
<td>Psychology</td>
</tr>
<tr>
<td>300</td>
<td>Perl</td>
<td>Psychology</td>
</tr>
<tr>
<td>350</td>
<td>Riemann</td>
<td>Math</td>
</tr>
</tbody>
</table>

(b)

Example: 4NF

Figure 4.11: Relation with Multi-Value Dependencies

<table>
<thead>
<tr>
<th>SID</th>
<th>Major</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Music</td>
<td>Swimming</td>
</tr>
<tr>
<td>150</td>
<td>Accounting</td>
<td>Swimming</td>
</tr>
<tr>
<td>100</td>
<td>Music</td>
<td>Tennis</td>
</tr>
<tr>
<td>100</td>
<td>Accounting</td>
<td>Tennis</td>
</tr>
<tr>
<td>150</td>
<td>Math</td>
<td>Jogging</td>
</tr>
</tbody>
</table>

Example: 4NF

Figure 4.12: Elimination of Multi-Value Dependency

<table>
<thead>
<tr>
<th>SID</th>
<th>Major</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Music</td>
<td>Skiing</td>
</tr>
<tr>
<td>100</td>
<td>Accounting</td>
<td>Swimming</td>
</tr>
<tr>
<td>100</td>
<td>Math</td>
<td>Tennis</td>
</tr>
<tr>
<td>150</td>
<td>Math</td>
<td>Jogging</td>
</tr>
</tbody>
</table>

Example: DK/NF

• First published in 1981 by Fagin
• DK/NF has no modification anomalies; so no higher normal form is needed
• A relation is in DK/NF if every constraint on the relation is a logical consequence of the definition of keys and domains

Example: DK/NF

Figure 4.14: Example 1 of DK/NF

| STUDENT (SID, GradeLevel, Dorm, Fee) |
| Key: SID |
| Constraints: Dorm = Fee, SID must not begin with digit 1 |

Example: DK/NF

Figure 4.15: Domain Key Definition for Example 1

<table>
<thead>
<tr>
<th>Domain Name</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>SID</td>
<td>4 decimal digits, first digit not 1</td>
</tr>
<tr>
<td>GradeLevel</td>
<td>StudentID</td>
</tr>
<tr>
<td>Dorm</td>
<td>BuildingName</td>
</tr>
<tr>
<td>Fee</td>
<td>StudentFees</td>
</tr>
</tbody>
</table>

Relation and Key Definitions:

STUDENT (SID, GradeLevel, Dorm)

BLDG-FEE (Dorm, Fee)
The Synthesis of Relations

- Given a set of attributes with certain functional dependencies, what relations should we form?
- Example: A and B are two attributes
  - If \( A \rightarrow B \) and \( B \rightarrow A \)
    - A and B have a one-to-one attribute relationship
  - If \( A \rightarrow B \), but \( B \not\rightarrow A \)
    - A and B have a many-to-one attribute relationship
  - If \( A \not\rightarrow B \) and \( B \not\rightarrow A \)
    - A and B have a many-to-many attribute relationship

Types of Attribute Relationship

- The letters used in these relation definitions match those used in Figure 4.22.
One-to-One Attribute Relationships

- Attributes that have a one-to-one relationship must occur together in at least one relation.
- Call the relation R and the attributes A and B:
  - Either A or B must be the key of R.
  - An attribute can be added to R if it is functionally determined by A or B.
  - An attribute that is not functionally determined by A or B cannot be added to R.
  - A and B must occur together in R, but should not occur together in other relations.
  - Either A or B should be consistently used to represent the pair in relations other than R.

Many-to-One Attribute Relationships

- Attributes that have a many-to-one relationship can exist in a relation together.
- Assume C determines D in relation S:
  - C must be the key of S.
  - An attribute can be added to S if it is determined by C.
  - An attribute that is not determined by C cannot be added to S.

Many-to-Many Attribute Relationships

- Attributes that have a many-to-many relationship can exist in a relation together.
- Assume attributes E and F reside together in relation T:
  - The key of T must be (E, F).
  - An attribute can be added to T if it is determined by the combination (E, F).
  - An attribute may not be added to T if it is not determined by the combination (E, F).
  - If adding a new attribute, G, expands the key to (E, F, G), then the theme of the relation has been changed.
  - Either G does not belong in T or the name of T must be changed to reflect the new theme.

De-normalized Designs

- When a normalized design is unnatural, awkward, or results in unacceptable performance, a de-normalized design is preferred.
- Example:
  - Normalized relation
    - CUSTOMER (CustNumber, CustName, Zip)
  - CODES (Zip, City, State)
  - De-Normalized relations
    - CUSTOMER (CustNumber, CustName, City, State, Zip)