The Relational Model

Chapter Objectives

- Learn the conceptual foundation of the relational model.
- Understand how relations differ from non-relational tables.
- Learn basic relational terminology.
- Learn the meaning and importance of keys, foreign keys, and related terminology.
- Understand how foreign keys represent relationships.
- Learn the purpose and use of surrogate keys.
- Learn the meaning of functional dependencies.
- Learn to apply a process for normalizing relations.
**Entity**

- An entity is something of importance to a user that needs to be represented in a database.
- An entity represents one theme or topic.
- In an entity-relationship model (discussed in Chapter 4), entities are restricted to things that can be represented by a single table.

**Relation**

- A relation is a two-dimensional table that has specific characteristics.
- The table dimensions, like a matrix, consist of rows and columns.
Characteristics of a Relation

- 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.

A Sample Relation

<table>
<thead>
<tr>
<th>EmployeeNumber</th>
<th>FirstName</th>
<th>LastName</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Mary</td>
<td>Abernathy</td>
</tr>
<tr>
<td>101</td>
<td>Jerry</td>
<td>Cadley</td>
</tr>
<tr>
<td>104</td>
<td>Alex</td>
<td>Copley</td>
</tr>
<tr>
<td>107</td>
<td>Megan</td>
<td>Jackson</td>
</tr>
</tbody>
</table>
### A Nonrelation Example

**Cells of the table hold multiple values**

<table>
<thead>
<tr>
<th>EmployeeNumber</th>
<th>Phone</th>
<th>LastName</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>335-6421</td>
<td>Abernathy</td>
</tr>
<tr>
<td></td>
<td>454-9744</td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>215-7789</td>
<td>Cadley</td>
</tr>
<tr>
<td>104</td>
<td>610-9850</td>
<td>Copley</td>
</tr>
<tr>
<td>107</td>
<td>299-9090</td>
<td>Jackson</td>
</tr>
</tbody>
</table>

**Course:** CIS2/DE4, **Course Teacher:** D. M. Akbar Hussain
**Department of Electronic Systems**

---

### A Nonrelation Example

**No two rows may be identical**

<table>
<thead>
<tr>
<th>EmployeeNumber</th>
<th>Phone</th>
<th>LastName</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>335-6421</td>
<td>Abernathy</td>
</tr>
<tr>
<td>101</td>
<td>215-7789</td>
<td>Cadley</td>
</tr>
<tr>
<td>104</td>
<td>610-9850</td>
<td>Copley</td>
</tr>
<tr>
<td>100</td>
<td>335-6421</td>
<td>Abernathy</td>
</tr>
<tr>
<td>107</td>
<td>299-9090</td>
<td>Jackson</td>
</tr>
</tbody>
</table>

**Course:** CIS2/DE4, **Course Teacher:** D. M. Akbar Hussain
**Department of Electronic Systems**
Terminology

Synonyms...

<table>
<thead>
<tr>
<th>Table</th>
<th>Row</th>
<th>Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>Record</td>
<td>Field</td>
</tr>
<tr>
<td>Relation</td>
<td>Tuple</td>
<td>Attribute</td>
</tr>
</tbody>
</table>

A Key

- A key is one (or more) columns of a relation that is (are) used to identify a row.
Uniqueness of Keys

<table>
<thead>
<tr>
<th>Unique Key</th>
<th>Non-unique Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data value is unique for each row.</td>
<td>Data value may be shared among several rows.</td>
</tr>
<tr>
<td>Consequently, the key will uniquely identify a row.</td>
<td>Consequently, the key will identify a set of rows.</td>
</tr>
</tbody>
</table>

EMPLOYEE (EmployeeNumber, FirstName, LastName, Department, Email, Phone)

A Composite Key

- A composite key is a key that contains two or more attributes
- For a key to be unique, often it must become a composite key
- Composite key can be unique or non-unique
**Composite Key Example**

To identify a family member, you need to know FamilyID, a FirstName, and a Suffix (e.g., Jr.)

**The composite key is:**
(FamilyID, FirstName, Suffix)

One needs to know the value of all three columns to uniquely identify an individual.

---

**A Candidate Key**

- A candidate key is called "candidate" because it is a candidate to become the primary key.
- A candidate key is a unique key.

EMPLOYEE (EmployeeNumber, FirstName, LastName, Department, Email, Phone)

EmployeeNumber, Email, Phone

(FirstName, LastName, Department)
A Primary Key

- A primary key is a candidate key chosen to be the main key for the relation.
- If you know the value of the primary key, you will be able to uniquely identify a single row.

Task performed by a Primary Key

The primary key is used for four primary tasks:

1. First, it is used to uniquely identify the rows in a table.
2. Second, it is used to represent rows in relationships.
3. Third, most DBMS products use the values of the primary key to organize the storage of the relation.
4. Finally, primary keys are used in indexes and other structures to improve performance for search operations.
Defining the Primary Key in Microsoft Access

Primary Key button
The key symbol indicates which column or columns are being used as the primary key.

Defining the Primary Key in Microsoft SQL Server 2005

Primary Key button
The key symbol indicates which column or columns are being used as the primary key.

The Is Identity setting
The Identity Increment setting
The Identity Seed setting
Defining the Primary Key in MySQL

The key symbol indicates which columns or columns are being used as the primary key.

A Surrogate Key

- A surrogate key is a unique, numeric value that is added to a relation to serve as the primary key.
- Surrogate key values have no meaning to users and are usually hidden on forms, queries and reports.
- A surrogate key is often used in place of a composite primary key.
More on Surrogate Key

A surrogate key is an artificial key that is created to act as the primary key for a relation.

The surrogate key is a unique, numeric value that is appended to the relation.

Surrogate keys are used in situations when no suitable primary key exists within the user data, or when all available primary keys within the data are too cumbersome for an efficient design.

Surrogate key values have no meaning to the users and are normally hidden on all forms, reports, and displays. Most DBMS products have the ability to automatically generate values for surrogate keys as needed.

Surrogate Key Example

If the Family Member primary key is FamilyID, FirstName, Suffix, it would be easier to append and use a surrogate key of FamMemberID.

FamilyID, FirstName and Suffix remain in the relation.

PROPERTY (Street, City, State, Zip, OwnerID)

Primary key is long and non-numeric.

PROPERTY (PropertyID, Street, City, State, Zip, OwnerID)
Relationships Between Tables

- A table may be related to other tables
- For example:
  - An Employee works in a Department
  - A Manager controls a Project

A Foreign Key

- To preserve relationships, you may need to create a foreign key.
- A foreign key is a primary key from one table placed into another table.
- The key is called a foreign key in the table that received the key.
Foreign Key Example I

Project

ProjID
ProjName
MgrID

Manager

MgrID
MgrName

Foreign Key

Primary Key

Foreign Key Example II

Department

DeptID
DeptName
Location

Employee

EmpID
DeptID
EmpName
Referential Integrity

- Referential integrity states that every value of a foreign key must match a value of an existing primary key.
- For example (see previous slide):
  - If EmpID = 4 in EMPLOYEE has a DeptID = 7 (a foreign key), a Department with DeptID = 7 must exist in DEPARTMENT.
  - The primary key value must exist before the foreign key value is entered.

Referential Integrity

Another perspective...

The value of the Foreign Key EmployeeID in EQUIPMENT

must exist in

The values of the Primary Key EmployeeID in EMPLOYEE
Referential Integrity

EMPLOYEE (EmployeeNumber, FirstName, LastName, Department, Email, Phone)
DEPARTMENT (DepartmentName, BudgetCode, OfficeNumber, DepartmentPhone)
EMPLOYEE (EmployeeNumber, FirstName, LastName, Department, Email, Phone)
DEPARTMENT (DepartmentName, BudgetCode, OfficeNumber, DepartmentPhone)

CUSTOMER (CustomerNumber, CustomerLastName, CustomerFirstName, Phone)
COURSE (CourseNumber, Course, CourseDate, Fee)
ENROLMENT (CustomerNumber, CourseNumber, AmountPaid)
Foreign Keys in Microsoft Access

The relationship is between CUSTOMER and ENROLLMENT.
- The foreign key CustomerNumber in ENROLLMENT references the primary key CustomerNumber in CUSTOMER.

Use this check box to enforce referential integrity in this relationship.

Foreign Keys in Microsoft SQL Server 2005

The relationship is between CUSTOMER and ENROLLMENT.
- We are enforcing the foreign key constraint, which is the referential integrity constraint.
Foreign Keys in MySQL

The relationship is between CUSTOMER and ENROLLMENT.

The Null Value

- A Null value means that no data was entered.
- This is different from a zero, space character or tab character.
The Problem of Null Values

- A Null is often ambiguous. It could mean...
  - The column value is not appropriate for the specific row
  - The column value is not decided
  - The column value is unknown
- Each may have entirely different implications.

<table>
<thead>
<tr>
<th>ItemNumber</th>
<th>ItemName</th>
<th>Color</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>110</td>
<td>T-Shirt</td>
<td>White</td>
<td>2</td>
</tr>
<tr>
<td>120</td>
<td>T-Shirt</td>
<td>Blue</td>
<td>45</td>
</tr>
<tr>
<td>200</td>
<td>T-Shirt</td>
<td>Green</td>
<td>3</td>
</tr>
<tr>
<td>230</td>
<td>Hat</td>
<td>Red</td>
<td>4</td>
</tr>
<tr>
<td>307</td>
<td>Hat</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Determinant**

- The attribute (or attributes) that we use as the starting point (the variable *(composite)* on the left side of the equation) is called a determinant.

```
(CookiePrice, Qty) → BoxPrice
```

**Candidate/Primary Keys and Functional Dependency**

- By definition...

  A **candidate key** of a relation will functionally determine all other attributes in the row

- Likewise, by definition...

  A **primary key** of a relation will functionally determine all other attributes in the row
Primary Key and Functional Dependency Example

CUSTOMER (CustomerNumber, CustomerLastName, CustomerFirstName, Phone)

CustomerNumber → (CustomerLastName, CustomerFirstName, Phone)

Normalization

- A process of analyzing a relation to ensure that it is well formed.
- More specifically, if a relation is normalized (well formed), rows can be inserted, deleted or modified without creating update anomalies.
Normalization Principles

- Relational design principles for normalized relations:
  - To be a well-formed relation, every determinant must be a candidate key.
  - Any relation that is not well formed should be broken into two or more well-formed relations.

Normalization Process

1. Identify all Candidate Keys.
2. Identify all functional dependencies.
3. Examine the determinants of functional dependencies, if any determinant is not a candidate key the relation is not well formed.
   a. Place the columns of the functional dependency in a new relation.
   b. Make the determinant the primary key of new relation.
   c. Leave a copy of determinant as a foreign key in original relation.
   d. Create a referential constraint between the original and new relation.
4. Repeat step 3 till all determinant of every relation is a candidates key.
Normalization Example

**Prescription** (PrescriptionNumber, Date, Drug, Dosage, CustomerName, CustomerPhone, CustomerEmail)

1. PrescriptionNumber is the only candidate key.
2. Functional dependency: Drug $\rightarrow\text{Dosage}$, CustomerEmail $\rightarrow\text{(CustomerName, CustomerPhone)}$
3. This means determinant (CustomerEmail) is not a candidate key so PRESCRIPTION has a normalization problem for we split the functional dependency.

**Customer** (CustomerName, CustomerPhone, CustomerEmail)

**Prescription** (PrescriptionNumber, Date, Drug, Dosage, CustomerEmail)

---

Normalization Example

**Meeting** (AttorneyID, ClientID, ClientName, MeetingDate, Duration)

AttorneyID, ClientID, MeetingDate

Or

AttorneyID, ClientName, MeetingDate

**Meeting** (AttorneyID, ClientID, MeetingDate, Duration)

**Client** (ClientID, ClientName)
Normalization Example

CUSTOMER (CustomerID, LastName, FirstName, Address, City, State, ZIP, Phone, Fax, Email)

CONTACT (ContactID, CustomerID, Data, Type, Remarks)

CustomerID in CONTACT must exist in CustomerID in CUSTOMER.

CONTACT Table Column Characteristics

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Key</th>
<th>Required</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>ContactID</td>
<td>AutoNumber</td>
<td>Primary Key</td>
<td>Yes</td>
<td>Surrogate Key</td>
</tr>
<tr>
<td>CustomerID</td>
<td>Number</td>
<td>Foreign Key</td>
<td>Yes</td>
<td>Long Integer</td>
</tr>
<tr>
<td>Date</td>
<td>Date/Time</td>
<td>No</td>
<td>Yes</td>
<td>Short Date</td>
</tr>
<tr>
<td>Type</td>
<td>Text (10)</td>
<td>No</td>
<td>Yes</td>
<td>Allowed values are Phone, Fax, Email and Meeting</td>
</tr>
<tr>
<td>Remarks</td>
<td>Memo</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>
Normalization Example

All fields from the CUSTOMER_CONTACT table appear on the form.
Normalization Example

The Email address has been updated.

Next record button.

All contact data for each customer is grouped together and sorted by date.
A modification problem has occurred. Not all records were updated with the new email address, and the database records are now inconsistent.