Chapter 6
Introduction to Databases
Chapter 6: outline

6.1 Getting started
6.2 Entity and Relations
Purpose of a Database

- The purpose of a database is to keep track of things.
- Unlike a list or spreadsheet, a database may store information that is more complicated than a simple list.
Problems with Lists: Redundancy

- In a list, each row is intended to stand on its own. As a result, the same information may be entered several times.
  - For Example: A list of Projects may include the Project Manager’s Name, ID, and Phone Extension. If a particular person is managing 10 projects, his/her information would have to be entered 10 times.
Problems with Lists: Multiple Themes

- In a list, each row may contain information on more than one theme. As a result, needed information may appear in the lists only if information on other themes is also present.
  - For Example: A list of Projects may include Project Manager information (Name, ID, and Phone Extension) and Project information (Name, ID, StartDate, Budget) in the same row.
List Modification Issues

- Redundancy and multiple themes create modification problems:
  - Deletion problems
  - Update problems
  - Insertion problems
List Modification Issues (Cont’d)

If Adviser *Baker* is changed to *Taing*, we need to change *AdviserEmail* as well. If changed to *Valdez*, we need to change *AdviserEmail*, *Department*, and *AdminLastName*.

<table>
<thead>
<tr>
<th></th>
<th>LastName</th>
<th>FirstName</th>
<th>Email</th>
<th>AdviserLastName</th>
<th>AdviserEmail</th>
<th>Department</th>
<th>AdminLastName</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Andrews</td>
<td>Matthew</td>
<td><a href="mailto:Matthew.Andrews@ourcampus.edu">Matthew.Andrews@ourcampus.edu</a></td>
<td>Baker</td>
<td><a href="mailto:Linda.Baker@ourcampus.edu">Linda.Baker@ourcampus.edu</a></td>
<td>Accounting</td>
<td>Smith</td>
</tr>
<tr>
<td>3</td>
<td>Brisbon</td>
<td>Lisa</td>
<td><a href="mailto:Lisa.Brisbon@ourcampus.edu">Lisa.Brisbon@ourcampus.edu</a></td>
<td>Valdez</td>
<td><a href="mailto:Richard.Valdez@ourcampus.edu">Richard.Valdez@ourcampus.edu</a></td>
<td>Chemistry</td>
<td>Chaplin</td>
</tr>
<tr>
<td>4</td>
<td>Fischer</td>
<td>Douglas</td>
<td><a href="mailto:Douglas.Fischer@ourcampus.edu">Douglas.Fischer@ourcampus.edu</a></td>
<td>Baker</td>
<td><a href="mailto:Linda.Baker@ourcampus.edu">Linda.Baker@ourcampus.edu</a></td>
<td>Accounting</td>
<td>Smith</td>
</tr>
<tr>
<td>5</td>
<td>Hwang</td>
<td>Terry</td>
<td><a href="mailto:Terry.Hwang@ourcampus.edu">Terry.Hwang@ourcampus.edu</a></td>
<td>Taing</td>
<td><a href="mailto:Susan.Taing@ourcampus.edu">Susan.Taing@ourcampus.edu</a></td>
<td>Accounting</td>
<td>Smith</td>
</tr>
<tr>
<td>6</td>
<td>Lai</td>
<td>Tzu</td>
<td><a href="mailto:Tzu.Lai@ourcampus.edu">Tzu.Lai@ourcampus.edu</a></td>
<td>Valdez</td>
<td><a href="mailto:Richard.Valdez@ourcampus.edu">Richard.Valdez@ourcampus.edu</a></td>
<td>Chemistry</td>
<td>Chaplin</td>
</tr>
<tr>
<td>7</td>
<td>Marino</td>
<td>Chip</td>
<td><a href="mailto:Chip.Marino@ourcampus.edu">Chip.Marino@ourcampus.edu</a></td>
<td>Taing</td>
<td><a href="mailto:Ken.Taing@ourcampus.edu">Ken.Taing@ourcampus.edu</a></td>
<td>InfoSystems</td>
<td>Rogers</td>
</tr>
<tr>
<td>8</td>
<td>Thompson</td>
<td>James</td>
<td><a href="mailto:James.Thompson@ourcampus.edu">James.Thompson@ourcampus.edu</a></td>
<td>Taing</td>
<td><a href="mailto:Susan.Taing@ourcampus.edu">Susan.Taing@ourcampus.edu</a></td>
<td>Accounting</td>
<td>Smith</td>
</tr>
<tr>
<td>9</td>
<td>????</td>
<td>????</td>
<td>????</td>
<td>????</td>
<td>????</td>
<td>Biology</td>
<td>Kelly</td>
</tr>
</tbody>
</table>

Deleted row—Student, Adviser, and Department data lost

Inserted row—both Student and Adviser data missing

Introduction to Databases 6-7
Addressing Information Complexities

- Relational databases are designed to address many of the information complexity issues.
Relational Databases

- A **relational database** stores information in tables. Each informational topic is stored in its own table.
- In essence, a relational database will break-up a list into several parts—one part for each theme in the list.
- A Project List would be divided into a CUSTOMER Table, a PROJECT Table, and a PROJECT_MANAGER Table.
In our relational database, we broke our list into several tables. Somehow the tables must be joined back together.

In a relational database, tables are joined together using the value of the data.

If a PROJECT has a CUSTOMER, the Customer_ID is stored as a column in the PROJECT table. The value stored in this column can be used to retrieve specific customer information from the CUSTOMER table.
Sounds Like More Work, Not Less

- A relational database is more complicated than a list.
- However, a relational database minimizes data redundancy, preserves complex relationships among topics, and allows for partial data.
- Furthermore, a relational database provides a solid foundation for user forms and reports.
Relational Database Example

<table>
<thead>
<tr>
<th>STUDENT data linked to ADVISER data via AdviserLastName</th>
</tr>
</thead>
</table>

Introduction to Databases 6-12
A Relational Database Solves the Problems of Lists

- Changed data—data remains consistent
- Inserted data—no STUDENT data required
- Deleted data—no ADVISER data lost
The Department, Advisor and Student Tables

Can insert DEPARTMENT data as needed—no ADVISER or STUDENT data required.

Can change STUDENT Adviser name as needed—new value is linked to its own data.

Can delete STUDENT data as needed—no DEPARTMENT or ADVISER data lost.

Introduction to Databases 6-14
The Art Course Database Tables

- Can change COURSE and CourseDate without problems
- Can insert new COURSE data as needed
- Can delete ENROLLMENT rows as needed—no adverse consequences
The Project Equipment Tables

Introduction to Databases 6-16
Structured Query Language (SQL)

- **Structured Query Language (SQL)** is an international standard for creating, processing and querying databases and their tables.

- Many database applications use SQL to retrieve, format, report, insert, delete, and/or modify data for users.
SQL Example

- We can use SQL to combine the data in the three tables in the Art Course Database to recreate the original list structure of the data.
- We do this by using an SQL SELECT statement.
SQL Example (Cont’d)

```sql
SELECT CUSTOMER.CustomerLastName,
      CUSTOMER.CustomerFirstName,
      CUSTOMER.Phone,
      COURSE.CourseDate,
      ENROLLMENT.AmountPaid,
      COURSE.Course, COURSE.Fee
FROM CUSTOMER, ENROLLMENT, COURSE
WHERE CUSTOMER.CustomerNumber
      = ENROLLMENT.CustomerNumber
AND COURSE.CourseNumber
      = ENROLLMENT.CourseNumber;
```

Introduction to Databases 6-19
### SQL Example Results

<table>
<thead>
<tr>
<th>CustomerLastName</th>
<th>CustomerFirstName</th>
<th>Phone</th>
<th>CourseDate</th>
<th>AmountPaid</th>
<th>Course</th>
<th>Fee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Johnson</td>
<td>Ariel</td>
<td>206-567-1234</td>
<td>10/1/2013</td>
<td>$250.00</td>
<td>Adv Pastels</td>
<td>$500.00</td>
</tr>
<tr>
<td>Johnson</td>
<td>Ariel</td>
<td>206-567-1234</td>
<td>3/15/2013</td>
<td>$350.00</td>
<td>Int Pastels</td>
<td>$350.00</td>
</tr>
<tr>
<td>Green</td>
<td>Robin</td>
<td>425-678-8765</td>
<td>9/15/2013</td>
<td>$350.00</td>
<td>Beg Oils</td>
<td>$350.00</td>
</tr>
<tr>
<td>Jackson</td>
<td>Charles</td>
<td>360-789-3456</td>
<td>10/1/2013</td>
<td>$500.00</td>
<td>Adv Pastels</td>
<td>$500.00</td>
</tr>
<tr>
<td>Pearson</td>
<td>Jeffery</td>
<td>206-567-2345</td>
<td>10/1/2013</td>
<td>$500.00</td>
<td>Adv Pastels</td>
<td>$500.00</td>
</tr>
<tr>
<td>Sears</td>
<td>Miguel</td>
<td>360-789-4567</td>
<td>9/15/2013</td>
<td>$350.00</td>
<td>Beg Oils</td>
<td>$350.00</td>
</tr>
<tr>
<td>Kyle</td>
<td>Leah</td>
<td>425-678-7654</td>
<td>11/15/2013</td>
<td>$250.00</td>
<td>Adv Pastels</td>
<td>$500.00</td>
</tr>
<tr>
<td>Myers</td>
<td>Lynda</td>
<td>360-789-5678</td>
<td>10/15/2013</td>
<td>$0.00</td>
<td>Beg Oils</td>
<td>$350.00</td>
</tr>
</tbody>
</table>

Introduction to Databases 6-20
The four components of a database system are:

- Users
- Database Application
- Database Management System (DBMS)
- Database
Components of a Database System
Users

- **A user** of a database system will
  - use a database application to track things;
  - use forms to enter, read, delete and query data; and
  - produce reports.
The Database

- A database is a *self-describing* collection of *related* records.
  - The database itself contains the definition of its structure.
  - Metadata is data describing the structure of the database data.
- Tables within a relational database are related to each other.
Example Database Metadata: A Relationship Diagram
Database Contents

- User data
- Metadata
- Indexes and other overhead data
- Application metadata
A database management system (DBMS) serves as an intermediary between database applications and the database.

The DBMS manages and controls database activities.

The DBMS creates, processes and administers the databases it controls.
Functions of a DBMS

- Create databases
- Create tables
- Create supporting structures
- Read database data
- Modify database data (insert, update, and delete)
- Maintain database structures
- Enforce rules
- Control concurrency
- Provide security
- Perform backup and recovery
Referential Integrity Constraints

- The DBMS will enforce many constraints.
- **Referential integrity constraints** ensure that the values of a column in one table are valid based on the values in another table.
  - If a 5 was entered as a CustomerID in the PROJECT table, a Customer having a CustomerID value of 5 must exist in the CUSTOMER table.
Database Applications

A **database application** is a set of one or more computer programs that serves as an intermediary between the user and the DBMS.
Functions of Database Applications

- Create and process forms
- Process user queries
- Create and process reports
- Execute application logic
- Control database applications
Database Applications:
Example Data Entry Form

Customer Data Entry Form

- CustomerNumber
- CustomerLastName: Johnson
- CustomerFirstName: Ariel
- Phone: 206-567-1234

Course Enrollment Data

<table>
<thead>
<tr>
<th>Course</th>
<th>CourseDate</th>
<th>Fee</th>
<th>AmountPaid</th>
<th>AmountDue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Int Pastels</td>
<td>3/15/2013</td>
<td>$350.00</td>
<td>$350.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>Adv Pastels</td>
<td>10/1/2013</td>
<td>$500.00</td>
<td>$250.00</td>
<td>$250.00</td>
</tr>
</tbody>
</table>

Records: 1 of 2

No Filter

Search

Introduction to Databases 6-32
Database Applications: Example Query

(a) Query Parameter Form

(b) Query Results
# Database Applications: Example Report

## Course Enrollment Report

<table>
<thead>
<tr>
<th>Course</th>
<th>CourseDate</th>
<th>CustomerLastName</th>
<th>CustomerFirstName</th>
<th>Phone</th>
<th>Fee</th>
<th>AmountPaid</th>
<th>AmountDue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adv Pastels</td>
<td>10/1/2013</td>
<td>Jackson</td>
<td>Charles</td>
<td>350-789-3456</td>
<td>$500.00</td>
<td>$500.00</td>
<td>$0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Johnson</td>
<td>Ariel</td>
<td>206-567-1234</td>
<td>$500.00</td>
<td>$250.00</td>
<td>$250.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pearson</td>
<td>Jeffery</td>
<td>206-567-2345</td>
<td>$500.00</td>
<td>$500.00</td>
<td>$0.00</td>
</tr>
<tr>
<td></td>
<td>11/15/2013</td>
<td>Kyle</td>
<td>Leah</td>
<td>425-678-7654</td>
<td>$500.00</td>
<td>$250.00</td>
<td>$250.00</td>
</tr>
<tr>
<td>Beg Oils</td>
<td>9/15/2013</td>
<td>Green</td>
<td>Robin</td>
<td>425-678-8765</td>
<td>$350.00</td>
<td>$350.00</td>
<td>$0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sears</td>
<td>Miguel</td>
<td>360-789-4567</td>
<td>$350.00</td>
<td>$350.00</td>
<td>$0.00</td>
</tr>
<tr>
<td></td>
<td>10/15/2013</td>
<td>Myers</td>
<td>Lynda</td>
<td>360-789-5678</td>
<td>$350.00</td>
<td>$0.00</td>
<td>$350.00</td>
</tr>
<tr>
<td>Int Pastels</td>
<td>3/15/2013</td>
<td>Johnson</td>
<td>Ariel</td>
<td>206-567-1234</td>
<td>$350.00</td>
<td>$350.00</td>
<td>$0.00</td>
</tr>
</tbody>
</table>

*Introduction to Databases 6-34*
Personal Database Systems

- Personal database systems typically
  - have one application.
  - have only a few tables.
  - are simple in design.
  - involve only one computer.
  - support one user at a time.
Personal Database Systems

User

Database application

Database management system (DBMS)

Microsoft Access or other personal DBMS

Database

Introduction to Databases 6-36
Personal Database Systems: An SQL Query in Microsoft Access

The SQL has been arranged to make it easy to read.

```sql
SELECT CUSTOMER.CustomerLastName, CUSTOMER.CustomerFirstName, CUSTOMER.Phone, COURSE.CourseDate, ENROLLMENT.AmountPaid, COURSE.Course, COURSE.Fee
FROM CUSTOMER, ENROLLMENT, COURSE
WHERE (((CUSTOMER.CustomerNumber)=[ENROLLMENT].[CustomerNumber])
AND (((COURSE.CourseNumber)=[ENROLLMENT].[CourseNumber]));
```
Personal Database Systems:
SQL Query Results in Microsoft Access
Enterprise-Class Database Systems

- Enterprise-Class database systems typically:
  - support several users simultaneously,
  - include more than one application,
  - involve multiple computers,
  - are complex in design,
  - have many tables, and
  - have many databases.
Organizational Database Systems

Database application A
Java code

Database application B
C# code

Database application C
HTML and ASP.NET

Database management system (DBMS)

SQL Server (Microsoft)
Oracle Database (Oracle)
MySQL (Oracle)
Others

Database
Commercial DBMS Products

- Example of Desktop DBMS Products
  - Microsoft Access

- Examples of Organizational DBMS Products
  - Microsoft’s SQL Server
  - Oracle’s Oracle
  - Sun Microsystem’s MySQL
  - IBM’s DB2
Enterprise-Class Database Systems: Microsoft SQL Server 2012

Click this button to run the SQL query

The database object Art-Course-Database is displayed in the Object Explorer

The table object CUSTOMER is displayed under the Art-Course-Database object

The SQL query

The query results in table format

Introduction to Databases 6-42
Enterprise-Class Database Systems: Oracle Database 11g Release 2

The database object **Art-Course-Database** is displayed in the Connections browser.

Click this button to run the SQL query.

The table object **CUSTOMER** is displayed under the Tables objects.

The SQL query:

```sql
SELECT CUSTOMER.CustomerFirstName, CUSTOMER.LastName, 
      COURSE.CourseName, ENROLLMENT.CourseName, 
      AMOUNTPAID, COURSE.Price 
FROM CUSTOMER, ENROLLMENT, COURSE 
WHERE CUSTOMER.CustomerID = ENROLLMENT.CustomerID; 
```

The query results in table format.

Introduction to Databases 6-43
Enterprise-Class Database Systems:
Oracle MySQL 5.5

Click this button to run the SQL query

The database object art-course-database is displayed in the Object Browser

The table object CUSTOMER is displayed under the art-course-database object

The SQL query

The query results in table format

Introduction to Databases 6-44
Chapter 6: outline

6.1 Getting started

6.2 Entity and 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.
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

1. Rows contain data about an entity.
2. Columns contain data about attributes of the entity.
3. Cells of the table hold a single value.
4. All entries in a column are of the same kind.
5. Each column has a unique name.
6. The order of the columns is unimportant.
7. The order of the rows is unimportant.
8. 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, 454-9744</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>107</td>
<td>299-9090</td>
<td>Jackson</td>
</tr>
</tbody>
</table>
Example of a Nonrelational Table

<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>

*No two rows may be identical*
### 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) column(s) of a relation that is (are) used to identify a row.
## Uniqueness of Keys

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

- A composite key is a key that contains two or more attributes.
- For a key to be unique, it must often become a composite key.
Composite Key Characteristics

- To identify a family member, you need to know a 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.
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.
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 2012

**Primary Key** button

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

- **Is Identity** setting
- **Identity Increment** setting
- **Identity Seed** setting
Defining the Primary Key in Oracle Database 11g Release 2

This constraint creates the primary key for CUSTOMER

The constraint type Primary_Key indicates which constraints are being used to create the primary key

The columns used in the primary key constraint CUSTOMER_PK are shown in the Columns pane

Sequences used to define primary key values are shown here

Introduction to Databases 6-61
Defining the Primary Key in Oracle MySQL 5.5

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

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

The PK check box selects which column or columns are being used as the primary key.

The AI check box indicates that AUTO_INCREMENT is 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.
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.
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

<table>
<thead>
<tr>
<th>Project</th>
<th>Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProjID</td>
<td>MgrID</td>
</tr>
<tr>
<td>ProjName</td>
<td>MgrName</td>
</tr>
<tr>
<td>MgrID</td>
<td></td>
</tr>
</tbody>
</table>

Primary Key

Foreign Key
Foreign Key Example II

<table>
<thead>
<tr>
<th>Department</th>
<th>Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeptID</td>
<td>EmpID</td>
</tr>
<tr>
<td>DeptName</td>
<td>DeptID</td>
</tr>
<tr>
<td>Location</td>
<td>EmpName</td>
</tr>
</tbody>
</table>

- **Primary Key**
  - DeptID
- **Foreign Key**
  - DeptID

Introduction to Databases 6-68
Referential Integrity

- **Referential integrity** states that every value of a foreign key must match a value of an existing primary key.

- **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 (Cont’d)

- Another perspective…

  *The value of the Foreign Key EmployeeID in EQUIPMENT*

  *must exist in*

  *The values of the Primary Key EmployeeID in EMPLOYEE*
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.

Introduction to Databases 6-71
Foreign Keys in Microsoft SQL Server 2012

The relationship is between CUSTOMER and ENROLLMENT.

We are enforcing the foreign key constraint—which is the referential integrity constraint.
Foreign Keys in Oracle Database 11g Release 2

- This constraint creates the foreign key relationship between CUSTOMER and ENROLLMENT.
- The constraint type Foreign_Key indicates which constraints are being used to create the foreign keys.
- The columns used in the foreign key constraint ENROLL_CUST_FK are shown in the Columns pane.

Introduction to Databases 6-73
Foreign Keys in Oracle MySQL 5.5

The foreign key between CUSTOMER and ENROLLMENT

The foreign key between CUSTOMER and ENROLLMENT

The relationship is between CUSTOMER and ENROLLMENT

Introduction to Databases 6-74
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.
Functional Dependency

- Functional Dependency—A relationship between attributes in which one attribute (or group of attributes) determines the value of another attribute in the same table

- Illustration...
  - The price of one cookie can determine the price of a box of 12 cookies.

(CookiePrice, Qty) → BoxPrice
The attribute (or attributes) that we use as the starting point (the variable on the left side of the equation) is called a determinant.

\[(\text{CookiePrice, Qty}) \rightarrow \text{BoxPrice}\]
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

(EmployeeID) → (EmpLastName, EmpPhone)

(ProjectID) → (ProjectName, StartDate)
Normalization

- 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 Example

(StudentID) → (StudentName, DormName, DormCost)

However, if...

(DormName) → (DormCost)

Then DormCost should be placed into its own relation, resulting in the relations:

(StudentID) → (StudentName, DormName)

(DormName) → (DormCost)
Normalization Example (Cont’d)

(AttorneyID, ClientID) → (ClientName, MeetingDate, Duration)

However, if...

(ClientID) → (ClientName)

Then ClientName should be placed into its own relation, resulting in the relations:

(AttorneyID, ClientID) → (MeetingDate, Duration)

(ClientID) → (ClientName)
Chapter 6: summary

- basic concepts
- relational databases
- SQL
- entities
- relations
- keys