3.1 A Logical View of Data

- Relational model
  - View data logically rather than physically
- Table
  - Structural and data independence
  - Resembles a file conceptually
- Relational database model easier to understand than hierarchical and network models

Tables and Their Characteristics

- Logical view of relational database based on relation
  - Relation thought of as a table
- Table: two-dimensional structure composed of rows and columns
  - Persistent representation of logical relation
- Contains group of related entities = an entity set
### 3.2 Keys

- Each row in a table must be uniquely identifiable.
- **Key** is one or more attributes that determine other attributes.
- Key’s role is based on determination.
  - A determines B (represented as $A \rightarrow B$)
    - If you know the value of attribute A, you can determine the value of attribute B
- **Functional dependence:**
  - Attribute B functionally dependent on A if all rows in table that agree in value for A also agree in value for B
Keys (continued)

- **Composite key (組合鍵)**
  - Composed of more than one attribute

- **Key attribute**
  - Any attribute that is part of a key

- If attribute B is functionally dependent on a composite key A but not on any subset of A, B is **fully functionally dependent** (完全功能相依) on A

- **Superkey (超級鍵)**
  - Any key that uniquely identifies each row

- **Candidate key (候選鍵)**
  - A superkey without unnecessary attributes
  - A table may have several candidate keys

- **Primary key (主索引鍵)**
  - A table has only one primary key, which is chosen from candidate keys

---

- **Nulls (空值, 虛值):**
  - No data entry
  - Not permitted in primary key
  - Should be avoided in other attributes
  - Can represent
    - An unknown attribute value
    - A known, but missing, attribute value
    - A “not applicable” condition
  - Can create problems when functions such as COUNT, AVERAGE, and SUM are used
  - Can create logical problems when relational tables are linked

---

- **Controlled redundancy (受控的重複性):**
  - Makes the relational database work
  - Tables within the database share **common attributes**
    - Enables tables to be linked together
  - Multiple occurrences of values not redundant when required to make the relationship work
  - Redundancy exists only when there is **unnecessary duplication** of attribute values
A relational schema is a representation of database tables where each table is listed by its name followed by its attributes in parentheses:

VENDOR(VEND_CODE, VEND_CONTACT, VEND_AREACODE, VEND_PHONE)

Keys (continued)

- **Foreign key (FK)**
  - An attribute whose values match primary key values in the related table
- **Referential integrity**
  - FK contains a value that refers to an existing valid tuple (row) in another relation
- **Secondary key**
  - Key used strictly for data retrieval purposes
  - A table may have many secondary keys

3.3 Integrity Rules

- Many RDBMs enforce integrity rules automatically
- Safer to ensure application design conforms to entity and referential integrity rules
- Designers may use flags to avoid nulls
  - Flags indicate absence of some value

<table>
<thead>
<tr>
<th>KEY TYPE</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Superkey</td>
<td>An attribute (or combination of attributes) that uniquely identifies each row in a table.</td>
</tr>
<tr>
<td>Candidate key</td>
<td>A minimal (reducible) superkey. A superkey that does not contain a subset of attributes that is itself a superkey.</td>
</tr>
<tr>
<td>Primary key</td>
<td>A candidate key selected to uniquely identify all other attribute values in any given row. Cannot contain null entries.</td>
</tr>
<tr>
<td>Secondary key</td>
<td>An attribute (or combination of attributes) used strictly for data retrieval purposes.</td>
</tr>
<tr>
<td>Foreign key</td>
<td>An attribute (or combination of attributes) in one table whose values must either match the primary key in another table or be null.</td>
</tr>
</tbody>
</table>
### Integrity Rule Example

<table>
<thead>
<tr>
<th>Requirement</th>
<th>All primary key entries are unique, and no part of a primary key may be null.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Each row will have a unique identity, and foreign key values can properly reference primary key values.</td>
</tr>
<tr>
<td>Example</td>
<td>No invoice can have a duplicate number, nor can it be null. In short, all invoices are uniquely identified by their invoice number.</td>
</tr>
</tbody>
</table>

---

### 3.4 Relational Set Operators

- **Relational algebra** (關聯式代數)
  - Defines theoretical way of manipulating table contents using relational operators
  - Use of relational algebra operators on existing relations produces new relations. This is called the property of closure (封閉性).
  - **UNION**
  - **INTERSECT**
  - **DIFFERENCE**
  - **PRODUCT**
  - **SELECT**
  - **PROJECT**
  - **JOIN**
  - **DIVIDE**
- Combines all rows from two tables, excluding duplicate rows
- Tables must have the same attribute characteristics (union-compatible, 聯集相容)

**Union**

- Yields all possible pairs of rows from two tables
- Also known as the Cartesian product (卡笛生乘積)

**Product**

**Intersection**

**Difference**

- Yields a horizontal (水平的) subset of a table
**Relational Set Operators (continued)**

- **Natural Join**
  - Links tables by selecting rows with common values in common attributes

- **Equijoin**
  - Links tables on the basis of an equality condition that compares specified columns

- **Theta join**
  - Any other comparison operator is used

- **Outer join**
  - Matched pairs retained and any unmatched values in other table left null

**Join**

- Allows information to be combined from two or more tables
- Real power behind the relational database, allowing the use of independent tables linked by common attributes
Natural Join Step 2: Select

<table>
<thead>
<tr>
<th>CUS_CODE</th>
<th>CUS_LNAME</th>
<th>CUS_ZIP</th>
<th>CUSTOMER.AGENT_CODE</th>
<th>AGENT.AGENT_CODE</th>
<th>AGENT_PHONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>27278</td>
<td>Adaree</td>
<td>32145</td>
<td>125</td>
<td>125</td>
<td>6152439887</td>
</tr>
<tr>
<td>1321242</td>
<td>Rodriguez</td>
<td>37134</td>
<td>125</td>
<td>6152439887</td>
<td></td>
</tr>
<tr>
<td>1312243</td>
<td>Rakowski</td>
<td>34129</td>
<td>167</td>
<td>6153426778</td>
<td></td>
</tr>
<tr>
<td>1132445</td>
<td>Walker</td>
<td>32145</td>
<td>231</td>
<td>6152431124</td>
<td></td>
</tr>
<tr>
<td>1867399</td>
<td>Vanloo</td>
<td>32145</td>
<td>231</td>
<td>6152431124</td>
<td></td>
</tr>
</tbody>
</table>

Natural Join Step 3: Project

<table>
<thead>
<tr>
<th>CUS_CODE</th>
<th>CUS_LNAME</th>
<th>CUS_ZIP</th>
<th>AGENT_CODE</th>
<th>AGENT_PHONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>27278</td>
<td>Adaree</td>
<td>32145</td>
<td>125</td>
<td>6152439887</td>
</tr>
<tr>
<td>1321242</td>
<td>Rodriguez</td>
<td>37134</td>
<td>125</td>
<td>6152439887</td>
</tr>
<tr>
<td>1312243</td>
<td>Rakowski</td>
<td>34129</td>
<td>167</td>
<td>6153426778</td>
</tr>
<tr>
<td>1132445</td>
<td>Walker</td>
<td>32145</td>
<td>231</td>
<td>6152431124</td>
</tr>
<tr>
<td>1867399</td>
<td>Vanloo</td>
<td>32145</td>
<td>231</td>
<td>6152431124</td>
</tr>
</tbody>
</table>

Left Outer Join

<table>
<thead>
<tr>
<th>CUS_CODE</th>
<th>CUS_LNAME</th>
<th>CUS_ZIP</th>
<th>AGENT_CODE</th>
<th>AGENT_PHONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>127278</td>
<td>Adaree</td>
<td>32145</td>
<td>125</td>
<td>6152439887</td>
</tr>
<tr>
<td>1321242</td>
<td>Rodriguez</td>
<td>37134</td>
<td>125</td>
<td>6152439887</td>
</tr>
<tr>
<td>1312243</td>
<td>Rakowski</td>
<td>34129</td>
<td>167</td>
<td>6153426778</td>
</tr>
<tr>
<td>1132445</td>
<td>Walker</td>
<td>32145</td>
<td>231</td>
<td>6152431124</td>
</tr>
<tr>
<td>1867399</td>
<td>Vanloo</td>
<td>32145</td>
<td>231</td>
<td>6152431124</td>
</tr>
<tr>
<td>1542311</td>
<td>Smithson</td>
<td>37134</td>
<td>421</td>
<td></td>
</tr>
</tbody>
</table>

Right Outer Join

<table>
<thead>
<tr>
<th>CUS_CODE</th>
<th>CUS_LNAME</th>
<th>CUS_ZIP</th>
<th>AGENT_CODE</th>
<th>AGENT_PHONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>127278</td>
<td>Adaree</td>
<td>32145</td>
<td>125</td>
<td>6152439887</td>
</tr>
<tr>
<td>1321242</td>
<td>Rodriguez</td>
<td>37134</td>
<td>125</td>
<td>6152439887</td>
</tr>
<tr>
<td>1312243</td>
<td>Rakowski</td>
<td>34129</td>
<td>167</td>
<td>6153426778</td>
</tr>
<tr>
<td>1132445</td>
<td>Walker</td>
<td>32145</td>
<td>231</td>
<td>6152431124</td>
</tr>
<tr>
<td>1867399</td>
<td>Vanloo</td>
<td>32145</td>
<td>231</td>
<td>6152431124</td>
</tr>
<tr>
<td>1867399</td>
<td>Vanloo</td>
<td>32145</td>
<td>333</td>
<td>9041234445</td>
</tr>
</tbody>
</table>

3.5 The Data Dictionary and System Catalog

- **Data dictionary (資料字典)**
  - Provides detailed accounting of all tables found within the user/designer-created database
  - Contains (at least) all the attribute names and characteristics for each table in the system
  - Contains metadata: data about data
  - Sometimes as “the database designer’s database” because it records the design decisions about tables and their structures
• **System catalog (系統目錄)**
  – Contains metadata
  – Detailed system data dictionary that describes all objects within the database
  – System catalog allows RDBMS to check for and eliminate homonyms () and synonyms (同義字)

### 3.6 Relationships within the Relational Database

- **1:M relationship**
  – Relational modeling ideal
  – Should be the norm in any relational database design
- **1:1 relationship**
  – Should be rare in any relational database design
- **M:N relationships**
  – Cannot be implemented as such in the relational model
  – M:N relationships can be changed into two 1:M relationships

---

### The 1:M Relationship

- **Relational database norm**
- **Found in any database environment**

---

*TABLE NAME* | **ATTRIBUTE NAME** | **CONTENTS** | **TYPE** | **FORMAT** | **RANGE** | **REQUIRED**<br>**PK**<br>**FK** | **RENDERED TABLE**
---|---|---|---|---|---|---|---
CUSTOMER | CUS_CODE | Customer account code | CHAR(5) | | 99999 | Y<br>PK | AGENT | CUSTOMER | CUS_NAME | Customer last name | VARCHAR(20) | | Y<br> Y | CUSTOMER | CUS_INITIAL | Customer first name | VARCHAR(20) | | Y<br> Y | CUSTOMER | CUS_BIRTHDATE | Customer birth date | DATE | | Y<br>Y | AGENT_CODE | Agent code | CHAR(6) | | 999 | Y<br>PK | AGENT | AGENT_PHONE | Agent telephone number | CHAR(20) | | Y<br>Y | AGENT_LASTNAME | Agent last name | VARCHAR(20) | | Y<br>Y | AGENT_VISSIT | Agent visits-to-date | NUMBER(9,2) | | Y<br>Y | AGENT_CUSTOMER | Customer code | CHAR(20) | | Y<br>Y | AGENT_CUSTOMER | Customer name | VARCHAR(20) | | Y<br>Y | AGENT_CUSTOMER | Customer account code | CHAR(5) | | Y<br>Y | AGENT_CUSTOMER | Customer last name | VARCHAR(20) | | Y<br>Y | AGENT_CUSTOMER | Customer first name | VARCHAR(20) | | Y<br>Y | AGENT_CUSTOMER | Customer birth date | DATE | | Y<br>Y | AGENT_CUSTOMER | Agent code | CHAR(6) | | Y<br>PK | AGENT | Agent area code | CHAR(5) | | Y<br>Y | AGENT | Agent name | VARCHAR(20) | | Y<br>Y | AGENT | Agent telephone number | CHAR(20) | | Y<br>Y | AGENT | Agent last name | VARCHAR(20) | | Y<br>Y | AGENT | Agent visits-to-date | NUMBER(9,2) | | Y<br>Y | AGENT_CUSTOMER | Customer code | CHAR(20) | | Y<br>Y | AGENT_CUSTOMER | Customer name | VARCHAR(20) | | Y<br>Y | AGENT_CUSTOMER | Customer account code | CHAR(5) | | Y<br>Y | AGENT_CUSTOMER | Customer last name | VARCHAR(20) | | Y<br>Y | AGENT_CUSTOMER | Customer first name | VARCHAR(20) | | Y<br>Y | AGENT_CUSTOMER | Customer birth date | DATE | | Y<br>Y | Agent | Agent code | CHAR(6) | | Y<br>PK | Agent | Agent area code | CHAR(5) | | Y<br>Y | Agent | Agent name | VARCHAR(20) | | Y<br>Y | Agent | Agent telephone number | CHAR(20) | | Y<br>Y | Agent | Agent last name | VARCHAR(20) | | Y<br>Y | Agent | Agent visits-to-date | NUMBER(9,2) | | Y<br>Y | "PK" = Foreign key<br>"FK" = Primary key<br>CHAR = Fixed character length data (1–255 characters)<br>VARCHAR = Variable character length data (1–2,000 characters)<br>NUMBER = Numeric data (NUMBER, DECIMAL) is used to specify numbers with two decimal places and up to nine digits, including the decimal places. Some RDBMSs permit the use of a MONEY or CURRENCY data type.
The 1:1 Relationship

- One entity related to only one other entity, and vice versa
- Sometimes means that entity components were not defined properly
- Could indicate that two entities actually belong in the same table
- Certain conditions absolutely require their use

The M:N Relationship

- Implemented by breaking it up to produce a set of 1:M relationships
- Avoid problems inherent to M:N relationship by creating a composite entity (組合實體) (also referred to as a bridge entity (橋接實體))
  - Includes as foreign keys the primary keys of tables to be linked
**Figure 3.25** The M:N relationship between STUDENT and CLASS

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>321452</td>
<td>10014</td>
</tr>
<tr>
<td>321452</td>
<td>10018</td>
</tr>
<tr>
<td>324257</td>
<td>10021</td>
</tr>
</tbody>
</table>

Table name: STUDENT
Primary key: STU_NUM
Foreign key: CLASS_CODE

Table name: CLASS
Primary key: CLASS_CODE
Foreign key: STU_NUM

**Composite Entity**

**Figure 3.27** Changing the M:N relationship to two 1:M relationships

**Figure 3.28** The expanded entity relationship model

**Table name: STUDENT**
Primary key: STU_NUM
Foreign key: none

<table>
<thead>
<tr>
<th>STU_NUM</th>
<th>STU_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>321452</td>
<td>Bowser</td>
</tr>
<tr>
<td>324257</td>
<td>Sethro</td>
</tr>
</tbody>
</table>

**Table name: ENROLL**
Primary key: CLASS_CODE, STU_NUM

<table>
<thead>
<tr>
<th>CLASS_CODE</th>
<th>STU_NUM</th>
<th>ENROLL_GRADE</th>
</tr>
</thead>
<tbody>
<tr>
<td>10014</td>
<td>321452</td>
<td>C</td>
</tr>
<tr>
<td>110014</td>
<td>324257</td>
<td>B</td>
</tr>
<tr>
<td>10018</td>
<td>321452</td>
<td>A</td>
</tr>
<tr>
<td>110018</td>
<td>324257</td>
<td>B</td>
</tr>
<tr>
<td>10021</td>
<td>321452</td>
<td>C</td>
</tr>
</tbody>
</table>

**Table name: CLASS**
Primary key: CLASS_CODE
Foreign key: CRS_CODE

<table>
<thead>
<tr>
<th>CRS_CODE</th>
<th>CLASS_CODE</th>
<th>CLASS_SECTION</th>
<th>CLASS_TIME</th>
<th>CLASS_ROOM</th>
<th>PROF_NUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCT-211</td>
<td>10014</td>
<td>3</td>
<td>MWF 9:00-9:50 a.m.</td>
<td>BUS252</td>
<td>289</td>
</tr>
<tr>
<td>CSCI-220</td>
<td>10018</td>
<td>2</td>
<td>MWF 9:00-9:50 a.m.</td>
<td>BUS252</td>
<td>114</td>
</tr>
<tr>
<td>GM-261</td>
<td>10021</td>
<td>1</td>
<td>MWF 9:00-9:50 a.m.</td>
<td>BUS252</td>
<td>119</td>
</tr>
</tbody>
</table>
3.7 Data Redundancy Revisited

- Data redundancy leads to data anomalies
  - Such anomalies can destroy the effectiveness of the database
- Foreign keys
  - Control data redundancies by using common attributes shared by tables
  - Crucial to exercising data redundancy control
- Sometimes, data redundancy is necessary
3.8 Indexes (索引)

- Orderly arrangement to logically access rows in a table
- **Index key (索引鍵)**
  - Index’s reference point
  - Points to data location identified by the key
- **Unique index (唯一索引)**
  - Index in which the index key can have only one pointer value (row) associated with it
- Each index is associated with only one table

3.9 Codd’s Relational Database Rules

- In 1985, Codd published a list of 12 rules to define a relational database system
  - Products marketed as “relational” that did not meet minimum relational standards
  - Even dominant database vendors do not fully support all 12 rules
Summary

- Tables are basic building blocks of a relational database
- Keys are central to the use of relational tables
- Keys define functional dependencies
  - Superkey
  - Candidate key
  - Primary key
  - Secondary key
  - Foreign key

Summary (continued)

- Each table row must have a primary key that uniquely identifies all attributes
- Tables linked by common attributes
- The relational model supports relational algebra functions
  - SELECT, PROJECT, JOIN, INTERSECT, UNION, DIFFERENCE, PRODUCT, DIVIDE
- Good design begins by identifying entities, attributes, and relationships
  - 1:1, 1:M, M:N