Module 4: Entity Relationship Modeling

OBJECTIVES:

In this chapter, you will learn:

• The main characteristics of entity relationship components
• How relationships between entities are defined, refined, and incorporated into the database design process
• How ERD components affect database design and implementation
• That real-world database design often requires the reconciliation of conflicting goals
The Entity Relationship Model

• The ERM forms the basis of an ERD.

• The ERD represents the conceptual database as viewed by the end user.

• ERDs depict the database’s main components: entities, attributes, and relationships.

• There are various notations used with ERDs—the original Chen notation and the newer Crow’s Foot and UML notations.
• The first two notations are used at the beginning of this chapter to introduce some basic ER modeling concepts.

• Some conceptual database modeling concepts can be expressed only using the Chen notation.

• However, because the emphasis is on design and implementation of databases, the Crow’s Foot and UML class diagram notations are used.
Because of its implementation emphasis, the Crow’s Foot notation can represent only what could be implemented. In other words:

- The Chen notation favors **conceptual** modeling.
- The **Crow’s Foot** notation favors a more **implementation-oriented** approach.
- The UML notation can be used for both **conceptual and implementation** modeling.
Entities

- Recall that an entity is an object of interest to the end user.

- An entity actually refers to the entity set and not to a single entity occurrence.

- In other words, the word entity in the ERM corresponds to a table—not to a row—in the relational environment.

- The ERM refers to a table row as an entity instance or entity occurrence.
In both the Chen and Crow’s Foot notations, an entity is represented by a rectangle containing the entity’s name.

The entity name, a noun, is usually written in all capital letters.

Fig. 4.1 A representation of the STUDENT entity
Attributes

- Attributes are characteristics of entities.

- For example, the STUDENT entity may include the attributes: STU_LNAME, STU_FNAME, STU_INITIAL, STU_EMAIL, STU_PHONE.

- In the original Chen notation, attributes are represented by ovals and are connected to the entity rectangle with a line. Each oval contains the name of the attribute it represents.

- In the Crow’s Foot notation, the attributes are written in the attribute box below the entity rectangle.

- Because the Chen representation is rather space-consuming, software vendors have adopted the Crow’s Foot attribute display.
CHEN MODEL

CROW’S FOOT MODEL

Fig 4.2 The attributes of the STUDENT entity: Chen and Crow’s Foot
Required and Optional Attributes

• A **required attribute** is an attribute that **must have a value**; in other words, it cannot be left empty.

• In Fig. 4.2, there are two boldfaced attributes in the Crow’s Foot notation. This indicates that a data entry will be required.

• In this example, STU_LNAME and STU_FNAME require data entries because of the assumption that all students have a last name and a first name.
• But students might not have a middle name, and perhaps they do not (yet) have a phone number and an e-mail address.

• Therefore, those attributes are not presented in boldface in the entity box.

• An **optional attribute** is an attribute that **does not require a value**; therefore, it can be left empty.
Domains

• Attributes have a domain.

• A **domain** is the set of possible values for a given attribute.

• For example, the domain for the grade point average (GPA) attribute is written (1,5) because the highest possible GPA value is 1 and the lowest possible value is 5.

• The domain for the gender attribute consists of only two possibilities: M or F (or some other equivalent code)
• Attributes may share a domain.

• For instance, a student address and a professor address share the same domain of all possible addresses.

• In fact, the data dictionary may let a newly declared attribute inherit the characteristics of an existing attribute if the same attribute name is used.

• For example, the PROFESSOR and STUDENT entities may each have an attribute named ADDRESS and could therefore share a domain.
Identifiers

• The ERM uses identifiers, that is, one or more attributes that uniquely identify each entity instance.

• In the relational model, such identifiers are mapped to primary keys (PKs) in tables.

• Identifiers are underlined in the ERD. Key attributes are also underlined in a frequently used table structure shorthand notation using the format:

   TABLE NAME (KEY ATTRIBUTE 1, ATTRIBUTE 2, ATTRIBUTE 3, ... ATTRIBUTE K)
Fig. 4.3 The CLASS table (entity) components and contents

<table>
<thead>
<tr>
<th>CLASS_CODE</th>
<th>CRS_CODE</th>
<th>CLASS_SECTION</th>
<th>CLASS_TIME</th>
<th>ROOM_CODE</th>
<th>PROF_NUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>10012</td>
<td>ACCT-211</td>
<td>1</td>
<td>M/W/F 8:00-8:50 a.m.</td>
<td>BUS311</td>
<td>105</td>
</tr>
<tr>
<td>10013</td>
<td>ACCT-211</td>
<td>2</td>
<td>M/W/F 9:00-9:50 a.m.</td>
<td>BUS200</td>
<td>105</td>
</tr>
<tr>
<td>10014</td>
<td>ACCT-211</td>
<td>3</td>
<td>TTh 2:30-3:45 p.m.</td>
<td>BUS252</td>
<td>342</td>
</tr>
<tr>
<td>10015</td>
<td>ACCT-212</td>
<td>1</td>
<td>M/W/F 10:00-10:50 a.m.</td>
<td>BUS311</td>
<td>301</td>
</tr>
<tr>
<td>10016</td>
<td>ACCT-212</td>
<td>2</td>
<td>Th 6:00-8:40 p.m.</td>
<td>BUS252</td>
<td>301</td>
</tr>
<tr>
<td>10017</td>
<td>CIS-220</td>
<td>1</td>
<td>M/W/F 9:00-9:50 a.m.</td>
<td>KLR209</td>
<td>228</td>
</tr>
<tr>
<td>10018</td>
<td>CIS-220</td>
<td>2</td>
<td>M/W/F 9:00-9:50 a.m.</td>
<td>KLR211</td>
<td>114</td>
</tr>
<tr>
<td>10019</td>
<td>CIS-220</td>
<td>3</td>
<td>M/W/F 10:00-10:50 a.m.</td>
<td>KLR209</td>
<td>228</td>
</tr>
<tr>
<td>10020</td>
<td>CIS-420</td>
<td>1</td>
<td>V/W 6:00-8:40 p.m.</td>
<td>KLR209</td>
<td>162</td>
</tr>
<tr>
<td>10021</td>
<td>QM-261</td>
<td>1</td>
<td>M/W/F 8:00-8:50 a.m.</td>
<td>KLR200</td>
<td>114</td>
</tr>
<tr>
<td>10022</td>
<td>QM-261</td>
<td>2</td>
<td>TTh 1:00-2:15 p.m.</td>
<td>KLR200</td>
<td>114</td>
</tr>
<tr>
<td>10023</td>
<td>QM-362</td>
<td>1</td>
<td>M/W/F 11:00-11:50 a.m.</td>
<td>KLR200</td>
<td>162</td>
</tr>
<tr>
<td>10024</td>
<td>QM-362</td>
<td>2</td>
<td>TTh 2:30-3:45 p.m.</td>
<td>KLR200</td>
<td>162</td>
</tr>
<tr>
<td>10025</td>
<td>MATH-243</td>
<td>1</td>
<td>Th 6:00-8:40 p.m.</td>
<td>DRE155</td>
<td>325</td>
</tr>
</tbody>
</table>
Composite Identifiers

- Ideally, an entity identifier is composed of only a single attribute.

- However, it is possible to use a composite identifier, that is, a primary key composed of more than one attribute.

- For instance, the Tiny College database administrator may decide to identify each CLASS entity instance (occurrence) by using a composite primary key composed of the combination of CRS_CODE and CLASS_SECTION instead of using CLASS_CODE.

- The combination of CRS_CODE and CLASS_SECTION is a proper candidate key.

- The candidate key (CRS_CODE and CLASS_SECTION) is an acceptable composite primary key.
• If the \texttt{CLASS\_CODE} is used as the primary key, the \texttt{CLASS} entity may be represented in shorthand form by:

\begin{verbatim}
CLASS (\texttt{CLASS\_CODE}, \texttt{CRS\_CODE}, \texttt{CLASS\_SECTION},
\texttt{CLASS\_TIME}, \texttt{ROOM\_CODE}, \texttt{PROF\_NUM})
\end{verbatim}

• On the other hand, if \texttt{CLASS\_CODE} is deleted, and the composite primary key is the combination of \texttt{CRS\_CODE} and \texttt{CLASS\_SECTION}, the \texttt{CLASS} entity may be represented by:

\begin{verbatim}
CLASS (\texttt{CRS\_CODE}, \texttt{CLASS\_SECTION}, \texttt{CLASS\_TIME},
\texttt{ROOM\_CODE}, \texttt{PROF\_NUM})
\end{verbatim}

• Note that both key attributes are underlined in the entity notation.
Composite and Simple Attributes

- Attributes are classified as simple or composite.

- A **composite attribute**, not to be confused with a composite key, is an attribute that can be further subdivided to yield additional attributes.

- For example, the attribute ADDRESS can be subdivided into street, city, state, and zip code. Similarly, the attribute PHONE_NUMBER can be subdivided into area code and exchange number.

- A **simple attribute** is an attribute that cannot be subdivided.

- For example, age, sex, and marital status would be classified as simple attributes.

- To facilitate detailed queries, it is wise to change composite attributes into a series of simple attributes.
Single-Valued Attributes

• A **single-valued** attribute is an attribute that can have only a single value.

• For example, a person can have only one Social Security number, and a manufactured part can have only one serial number.

• Keep in mind that a single-valued attribute is not necessarily a simple attribute.

• For instance, a part’s serial number, such as SE-08-02-189935, is single-valued, but it is a composite attribute because it can be subdivided into the region in which the part was produced (SE), the plant within that region (08), the shift within the plant (02), and the part number (189935).
Multivalued Attributes

- **Multivalued attributes** are attributes that can have many values.

- For instance, a person may have several college degrees, and a household may have several different phones, each with its own number.

- Similarly, a car’s color may be subdivided into many colors (that is, colors for the roof, body, and trim).

- In the Chen ERM, the multivalued attributes are shown by a double line connecting the attribute to the entity.

- The Crow’s Foot notation does not identify multivalued attributes. The ERD in Figure 4.4 contains all of the components introduced thus far. Note that CAR_VIN is the primary key, and CAR_COLOR is a multivalued attribute of the CAR entity.
CHEN MODEL

CROW’S FOOT MODEL

Fig. 4.4 A multivalued attribute in an entity
Implementing multivalued attributes

- Although the conceptual model can handle M:N relationships and multivalued attributes, *you should not implement them in the RDBMS.*

- Remember from Module 3 that in the relational table, each column/row intersection represents a single data value.

- So if multivalued attributes exist, the designer must decide on one of two possible courses of action:
1. Within the original entity, create several new attributes, one for each of the original multivalued attribute’s components.

For example, the CAR entity’s attribute CAR_COLOR can be split to create the new attributes:

CAR_TOPCOLOR, CAR_BODYCOLOR, and CAR_TRIMCOLOR,
which are then assigned to the CAR entity.

- Although this solution seems to work, its adoption can lead to major structural problems in the table.

- For example, if additional color components—such as a logo color—are added for some cars, the table structure must be modified to accommodate the new color section.

- In short, although you have seen solution 1 applied, it is not an acceptable solution.
Fig. 4.5 Splitting the multivalued attribute into new attributes
2. Create a new entity composed of the original multivalued attribute’s components.

- This new entity allows the designer to define color for different sections of the car.
- Then, this new CAR_COLOR entity is related to the original CAR entity in a 1:M relationship.

Table 4.1 Components of the multivalued attribute

<table>
<thead>
<tr>
<th>SECTION</th>
<th>COLOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top</td>
<td>White</td>
</tr>
<tr>
<td>Body</td>
<td>Blue</td>
</tr>
<tr>
<td>Trim</td>
<td>Gold</td>
</tr>
<tr>
<td>Interior</td>
<td>Blue</td>
</tr>
</tbody>
</table>
• Using the approach illustrated in Table 4.1, you even get a fringe benefit:

you are now able to assign as many colors as necessary without having to change the table structure.

Fig. 4.6 A new entity set composed of a multivalued attribute’s components
• This is the preferred way to deal with multivalued attributes.

• Creating a new entity in a 1:M relationship with the original entity yields several benefits:

  It is a more flexible, expandable solution, and it is compatible with the relational model!
Derived Attributes

• Finally, an attribute may be classified as a derived attribute.

• A **derived attribute** is an attribute whose value is calculated (derived) from other attributes.

• The derived attribute need not be physically stored within the database; instead, it can be derived by using an algorithm.
For example, an employee’s age, EMP_AGE, may be found by computing the integer value of the difference between the current date and the EMP_DOB.

If you use Microsoft Access, you would use the formula INT((DATE() – EMP_DOB)/365).

In Microsoft SQL Server, you would use SELECT DATEDIFF("YEAR", EMP_DOB, GETDATE()), where DATEDIFF is a function that computes the difference between dates. The first parameter indicates the measurement, in this case, years.
• A derived attribute is indicated in the Chen notation by a **dashed line** connecting the attribute and the entity.

• The Crow’s Foot notation does not have a method for distinguishing the derived attribute from other attributes.

• Derived attributes are sometimes referred to as *computed attributes*.

• A derived attribute computation can be as simple as adding two attribute values located on the same row, or it can be the result of aggregating the sum of values located on many table rows (from the same table or from a different table).

• The decision to store derived attributes in database tables depends on the processing requirements and the constraints placed on a particular application.

• The designer should be able to balance the design in accordance with such constraints.
CHEN MODEL

CROW’S FOOT MODEL

Fig. 4.6 Depiction of a derived attribute
Table 4.2 Advantages and Disadvantages of Storing Derived Attributes

<table>
<thead>
<tr>
<th>DERIVED ATTRIBUTE</th>
<th>STORED</th>
<th>NOT STORED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advantage</td>
<td>Saves CPU processing cycles</td>
<td>Saves storage space</td>
</tr>
<tr>
<td></td>
<td>Saves data access time</td>
<td>Computation always yields current value</td>
</tr>
<tr>
<td></td>
<td>Data value is readily available</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Can be used to keep track of historical data</td>
<td></td>
</tr>
<tr>
<td>Disadvantage</td>
<td>Requires constant maintenance to ensure derived value is current, especially if any values used in the calculation change</td>
<td>Uses CPU processing cycles</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Increases data access time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Adds coding complexity to queries</td>
</tr>
</tbody>
</table>
Relationships

• A **relationship** is an association between entities.

• The entities that participate in a relationship are also known as **participants**, and each relationship is identified by a name that describes the relationship.

• The relationship name is an active or passive verb;

• For example, a STUDENT **takes** a CLASS, a PROFESSOR **teaches** a CLASS, a DEPARTMENT **employs** a PROFESSOR.
• Relationships between entities always operate in both directions.

• To define the relationship between the entities named CUSTOMER and INVOICE, you would specify that:
  ▫ A CUSTOMER may generate many INVOICEs.
  ▫ Each INVOICE is generated by one CUSTOMER.

• Because you know both directions of the relationship between CUSTOMER and INVOICE, it is easy to see that this relationship can be classified as 1:M.
The relationship classification is difficult to establish if you know only one side of the relationship.

For example, if you specify that:
- A DIVISION is managed by one EMPLOYEE.

You don’t know if the relationship is 1:1 or 1:M. Therefore, you should ask the question “Can an employee manage more than one division?”

If the answer is yes, the relationship is 1:M, and the second part of the relationship is then written as:
- An EMPLOYEE may manage many DIVISIONs.

If an employee cannot manage more than one division, the relationship is 1:1, and the second part of the relationship is then written as:
- An EMPLOYEE may manage only one DIVISION.
Connectivity and Cardinality

- The term **connectivity** is used to describe the relationship classification.

- **Cardinality** expresses the minimum and maximum number of entity occurrences associated with one occurrence of the related entity.

- In the ERD, cardinality is indicated by placing the appropriate numbers beside the entities, using the format (x,y).

- The first value represents the minimum number of associated entities, while the second value represents the maximum number of associated entities.
• Knowing the minimum and maximum number of entity occurrences is very useful at the application software level.

• For example, Tiny College might want to ensure that a class is not taught unless it has at least 10 students enrolled.

• Similarly, if the classroom can hold only 30 students, the application software should use that cardinality to limit enrollment in the class.
• Cardinalities represent the number of occurrences in the related entity.

• For example, the cardinality (1,4) written next to the CLASS entity in the “PROFESSOR teaches CLASS” relationship indicates that each professor teaches up to four classes, which means that the PROFESSOR table’s primary key value occurs at least once and no more than four times as foreign key values in the CLASS table.

• If the cardinality had been written as (1,N), there would be no upper limit to the number of classes a professor might teach.

• Similarly, the cardinality (1,1) written next to the PROFESSOR entity indicates that each class is taught by one and only one professor.

• That is, each CLASS entity occurrence is associated with one and only one entity occurrence in PROFESSOR.
• Connectivities and cardinalities are established by very concise statements known as business rules, which were introduced in Chapter 2.

• Such rules, derived from a precise and detailed description of an organization’s data environment, also establish the ERM’s entities, attributes, relationships, connectivities, cardinalities, and constraints.

• Because business rules define the ERM’s components, making sure that all appropriate business rules are identified is a very important part of a database designer’s job.
• The placement of the cardinalities in the ER diagram is a matter of convention.

• The Chen notation places the cardinalities on the side of the related entity.

• The Crow’s Foot and UML diagrams place the cardinalities next to the entity to which the cardinalities apply.
Existence Dependence

• An entity is said to be existence-dependent if it can exist in the database only when it is associated with another related entity occurrence.

• In implementation terms, an entity is existence-dependent if it has a mandatory foreign key—that is, a foreign key attribute that cannot be null.

• For example, if an employee wants to claim one or more dependents for tax-withholding purposes, the relationship “EMPLOYEE claims DEPENDENT” would be appropriate.

• In that case, the DEPENDENT entity is clearly existence-dependent on the EMPLOYEE entity because it is impossible for the dependent to exist apart from the EMPLOYEE in the database.
• If an entity can exist apart from all of its related entities (it is **existence-independent**), then that entity is referred to as a **strong entity** or **regular entity**.

• For example, suppose that the XYZ Corporation uses parts to produce its products.

• Furthermore, suppose that some of those parts are produced in-house and other parts are bought from vendors.

• In that scenario, it is quite possible for a PART to exist independently from a VENDOR in the relationship “PART is supplied by VENDOR,” because at least some of the parts are not supplied by a vendor.

• Therefore, PART is existence-independent from VENDOR.
Relationship Strength

• The concept of relationship strength is based on how the primary key of a related entity is defined.

• To implement a relationship, the primary key of one entity appears as a foreign key in the related entity.
Weak (Non-identifying relationships)

- A **weak relationship**, also known as a non-identifying relationship, exists if the PK of the related entity does not contain a PK component of the parent entity.

- By default, relationships are established by having the PK of the parent entity appear as an FK on the related entity.
For example, suppose that the COURSE and CLASS entities are defined as:

COURSE(CRS_CODE, DEPT_CODE, CRS_DESCRIPTION, CRS_CREDIT)
CLASS(CLASS_CODE, CRS_CODE, CLASS_SECTION, CLASS_TIME, ROOM_CODE, PROF_NUM)

In this case, a weak relationship exists between COURSE and CLASS because the CLASS_CODE is the CLASS entity’s PK, while the CRS_CODE in CLASS is only an FK.

In this example, the CLASS PK did not inherit the PK component from the COURSE entity.

Figure 4.8 shows how the Crow’s Foot notation depicts a weak relationship by placing a dashed relationship line between the entities. The tables shown below the ERD illustrate how such a relationship is implemented.
Table name: COURSE

<table>
<thead>
<tr>
<th>CRS_CODE</th>
<th>DEPT_CODE</th>
<th>CRS_DESCRIPTION</th>
<th>CRS_CREDIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCT-211</td>
<td>ACCT</td>
<td>Accounting I</td>
<td>3</td>
</tr>
<tr>
<td>ACCT-212</td>
<td>ACCT</td>
<td>Accounting II</td>
<td>3</td>
</tr>
<tr>
<td>CIS-220</td>
<td>CIS</td>
<td>Intro. to Microcomputing</td>
<td>3</td>
</tr>
<tr>
<td>CIS-420</td>
<td>CIS</td>
<td>Database Design and Implementation</td>
<td>4</td>
</tr>
<tr>
<td>MATH-243</td>
<td>MATH</td>
<td>Mathematics for Managers</td>
<td>3</td>
</tr>
<tr>
<td>QM-261</td>
<td>CIS</td>
<td>Intro. to Statistics</td>
<td>3</td>
</tr>
<tr>
<td>QM-362</td>
<td>CIS</td>
<td>Statistical Applications</td>
<td>4</td>
</tr>
</tbody>
</table>

Table name: CLASS

<table>
<thead>
<tr>
<th>CLASS_CODE</th>
<th>CRS_CODE</th>
<th>CLASS_SECTION</th>
<th>CLASS_TIME</th>
<th>ROOM_CODE</th>
<th>PROF_NUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>10012</td>
<td>ACCT-211</td>
<td>1</td>
<td>MWF 8:00-8:50 a.m.</td>
<td>BUS311</td>
<td>105</td>
</tr>
<tr>
<td>10013</td>
<td>ACCT-211</td>
<td>2</td>
<td>MWF 9:00-9:50 a.m.</td>
<td>BUS200</td>
<td>105</td>
</tr>
<tr>
<td>10014</td>
<td>ACCT-211</td>
<td>3</td>
<td>TTh 2:30-3:45 p.m.</td>
<td>BUS252</td>
<td>342</td>
</tr>
<tr>
<td>10015</td>
<td>ACCT-212</td>
<td>1</td>
<td>MWF 10:00-10:50 a.m.</td>
<td>BUS311</td>
<td>301</td>
</tr>
<tr>
<td>10016</td>
<td>ACCT-212</td>
<td>2</td>
<td>Th 6:00-8:40 p.m.</td>
<td>BUS252</td>
<td>301</td>
</tr>
<tr>
<td>10017</td>
<td>CIS-220</td>
<td>1</td>
<td>MWF 9:00-9:50 a.m.</td>
<td>KLR209</td>
<td>228</td>
</tr>
<tr>
<td>10018</td>
<td>CIS-220</td>
<td>2</td>
<td>MWF 9:00-9:50 a.m.</td>
<td>KLR211</td>
<td>114</td>
</tr>
<tr>
<td>10019</td>
<td>CIS-220</td>
<td>3</td>
<td>MWF 10:00-10:50 a.m.</td>
<td>KLR209</td>
<td>228</td>
</tr>
<tr>
<td>10020</td>
<td>QM-261</td>
<td>1</td>
<td>VV 6:00-8:40 p.m.</td>
<td>KLR209</td>
<td>162</td>
</tr>
<tr>
<td>10021</td>
<td>QM-261</td>
<td>2</td>
<td>MWF 8:00-8:50 a.m.</td>
<td>KLR200</td>
<td>114</td>
</tr>
<tr>
<td>10022</td>
<td>QM-261</td>
<td>2</td>
<td>TTh 1:00-2:15 p.m.</td>
<td>KLR200</td>
<td>114</td>
</tr>
<tr>
<td>10023</td>
<td>QM-362</td>
<td>1</td>
<td>MWF 11:00-11:50 a.m.</td>
<td>KLR200</td>
<td>162</td>
</tr>
<tr>
<td>10024</td>
<td>QM-362</td>
<td>2</td>
<td>TTh 2:30-3:45 p.m.</td>
<td>KLR200</td>
<td>162</td>
</tr>
<tr>
<td>10025</td>
<td>MATH-243</td>
<td>1</td>
<td>Th 6:00-8:40 p.m.</td>
<td>DRE155</td>
<td>325</td>
</tr>
</tbody>
</table>
Strong (Identifying) Relationship

- A **strong relationship**, also known as an identifying relationship, exists when the PK of the related entity contains a PK component of the parent entity.

- For example, the definitions of the COURSE and CLASS entities

  \[
  \text{COURSE(} \text{CRS\_CODE, DEPT\_CODE, CRS\_DESCRIPTION, CRS\_CREDIT)}
  \]

  \[
  \text{CLASS(} \text{CRS\_CODE, CLASS\_SECTION, CLASS\_TIME, ROOM\_CODE, PROF\_NUM)}
  \]

  indicate that a strong relationship exists between COURSE and CLASS, because the CLASS entity’s composite PK is composed of CRS\_CODE + CLASS\_SECTION. (Note that the CRS\_CODE in CLASS is also the FK to the COURSE entity.)
Fig. 4.9 A strong (identifying) relationship between COURSE and CLASS

Table name: COURSE

<table>
<thead>
<tr>
<th>CRS_CODE</th>
<th>DEPT_CODE</th>
<th>CRS_DESCRIPTION</th>
<th>CRS_CREDIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCT-211</td>
<td>ACCT</td>
<td>Accounting I</td>
<td>3</td>
</tr>
<tr>
<td>ACCT-212</td>
<td>ACCT</td>
<td>Accounting II</td>
<td>3</td>
</tr>
<tr>
<td>CIS-220</td>
<td>CIS</td>
<td>Intro. to Microcomputing</td>
<td>3</td>
</tr>
<tr>
<td>CIS-420</td>
<td>CIS</td>
<td>Database Design and Implementation</td>
<td>4</td>
</tr>
<tr>
<td>MATH-243</td>
<td>MATH</td>
<td>Mathematics for Managers</td>
<td>3</td>
</tr>
<tr>
<td>QM-261</td>
<td>CIS</td>
<td>Intro. to Statistics</td>
<td>3</td>
</tr>
<tr>
<td>QM-362</td>
<td>CIS</td>
<td>Statistical Applications</td>
<td>4</td>
</tr>
</tbody>
</table>

Table name: CLASS

<table>
<thead>
<tr>
<th>CRS_CODE</th>
<th>CLASS_SECTION</th>
<th>CLASS_TIME</th>
<th>ROOM_CODE</th>
<th>PROF_NUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCT-211</td>
<td>1</td>
<td>MWF 8:00-8:50 a.m.</td>
<td>BUS311</td>
<td>105</td>
</tr>
<tr>
<td>ACCT-211</td>
<td>2</td>
<td>MWF 9:00-9:50 a.m.</td>
<td>BUS200</td>
<td>105</td>
</tr>
<tr>
<td>ACCT-211</td>
<td>3</td>
<td>TTh 2:30-3:45 p.m.</td>
<td>BUS252</td>
<td>342</td>
</tr>
<tr>
<td>ACCT-212</td>
<td>1</td>
<td>MWF 10:00-10:50 a.m.</td>
<td>BUS311</td>
<td>301</td>
</tr>
<tr>
<td>ACCT-212</td>
<td>2</td>
<td>Th 6:00-8:40 p.m.</td>
<td>BUS252</td>
<td>301</td>
</tr>
<tr>
<td>CIS-220</td>
<td>1</td>
<td>MWF 9:00-9:50 a.m.</td>
<td>KLR209</td>
<td>228</td>
</tr>
<tr>
<td>CIS-220</td>
<td>2</td>
<td>MWF 9:00-9:50 a.m.</td>
<td>KLR211</td>
<td>114</td>
</tr>
<tr>
<td>CIS-220</td>
<td>3</td>
<td>MWF 10:00-10:50 a.m.</td>
<td>KLR209</td>
<td>228</td>
</tr>
<tr>
<td>CIS-420</td>
<td>1</td>
<td>VV 6:00-8:40 p.m.</td>
<td>KLR209</td>
<td>162</td>
</tr>
<tr>
<td>MATH-243</td>
<td>1</td>
<td>Th 6:00-8:40 p.m.</td>
<td>DRE155</td>
<td>325</td>
</tr>
<tr>
<td>QM-261</td>
<td>1</td>
<td>MWF 8:00-8:50 a.m.</td>
<td>KLR200</td>
<td>114</td>
</tr>
<tr>
<td>QM-261</td>
<td>2</td>
<td>TTh 1:00-2:15 p.m.</td>
<td>KLR200</td>
<td>114</td>
</tr>
<tr>
<td>QM-362</td>
<td>1</td>
<td>MWF 11:00-11:50 a.m.</td>
<td>KLR200</td>
<td>162</td>
</tr>
<tr>
<td>QM-362</td>
<td>2</td>
<td>TTh 2:30-3:45 p.m.</td>
<td>KLR200</td>
<td>162</td>
</tr>
</tbody>
</table>
• The Crow’s Foot notation depicts the strong (identifying) relationship with a solid line between the entities, shown in Figure 4.9.

• Whether the relationship between COURSE and CLASS is strong or weak depends on how the CLASS entity’s primary key is defined.

• Keep in mind that the order in which the tables are created and loaded is very important.

• For example, in the “COURSE generates CLASS” relationship, the COURSE table must be created before the CLASS table.
• After all, it would not be acceptable to have the CLASS table’s foreign key reference a COURSE table that did not yet exist.

• You must load the data of the “1” side first in a 1:M relationship to avoid the possibility of referential integrity errors, regardless of whether the relationships are weak or strong.

• Remember that the nature of the relationship is often determined by the database designer, who must use professional judgment to determine which relationship type and strength best suit the database transaction, efficiency, and information requirements.
Weak Entities

• In contrast to the strong or regular entity, a weak entity is one that meets two conditions:

1. The entity is existence-dependent; that is, it cannot exist without the entity with which it has a relationship.

2. The entity has a primary key that is partially or totally derived from the parent entity in the relationship.
For example, a company insurance policy insures an employee and his/her dependents.

For the purpose of describing an insurance policy, an EMPLOYEE might or might not have a DEPENDENT, but the DEPENDENT must be associated with an EMPLOYEE.

Moreover, the DEPENDENT cannot exist without the EMPLOYEE; that is, a person cannot get insurance coverage as a dependent unless s(he) happens to be a dependent of an employee.

DEPENDENT is the weak entity in the relationship “EMPLOYEE has DEPENDENT.”
FIG 4.10 A Weak Entity in an ERD

CHEN MODEL

EMPLOYEE 1 has M DEPENDENT

EMP_NUM
EMP_LNAME
EMP_FNAME
EMP_INITIAL
EMP_DOB
EMP_HIREDATE

EMP_NUM
DEP_NUM
DEP_FNAME
DEP_DOB

CROW FOOT’S MODEL
• Figure 4.11 illustrates the implementation of the relationship between the weak entity (DEPENDENT) and its parent or strong counterpart (EMPLOYEE).

• Note that DEPENDENT’s primary key is composed of two attributes, EMP_NUM and DEP_NUM, and that EMP_NUM was inherited from EMPLOYEE.
**Fig. 4.11 A Weak Entity in a Strong Relationship**

**Table Name: EMPLOYEE**

<table>
<thead>
<tr>
<th>EMP_NUM</th>
<th>EMP_LNAME</th>
<th>EMP_FNAME</th>
<th>EMP_INITIAL</th>
<th>EMP_DOB</th>
<th>EMP_HIREDATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>Callifante</td>
<td>Jeanine</td>
<td>J</td>
<td>12-Mar-64</td>
<td>25-May-97</td>
</tr>
<tr>
<td>1002</td>
<td>Smithson</td>
<td>William</td>
<td>K</td>
<td>23-Nov-70</td>
<td>28-May-97</td>
</tr>
<tr>
<td>1003</td>
<td>Washington</td>
<td>Herman</td>
<td>H</td>
<td>15-Aug-68</td>
<td>28-May-97</td>
</tr>
<tr>
<td>1004</td>
<td>Chen</td>
<td>Lydia</td>
<td>B</td>
<td>23-Mar-74</td>
<td>15-Oct-98</td>
</tr>
<tr>
<td>1005</td>
<td>Johnson</td>
<td>Melanie</td>
<td></td>
<td>28-Sep-66</td>
<td>20-Dec-98</td>
</tr>
<tr>
<td>1006</td>
<td>Ortega</td>
<td>Jorge</td>
<td>G</td>
<td>12-Jul-79</td>
<td>05-Jan-02</td>
</tr>
<tr>
<td>1007</td>
<td>O'Donnell</td>
<td>Peter</td>
<td>D</td>
<td>10-Jun-71</td>
<td>23-Jun-02</td>
</tr>
<tr>
<td>1008</td>
<td>Brzenski</td>
<td>Barbara</td>
<td>A</td>
<td>12-Feb-70</td>
<td>01-Nov-03</td>
</tr>
</tbody>
</table>

**Table Name: DEPENDENT**

<table>
<thead>
<tr>
<th>EMP_NUM</th>
<th>DEP_NUM</th>
<th>DEP_FNAME</th>
<th>DEP_DOB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>1</td>
<td>Annelise</td>
<td>05-Dec-97</td>
</tr>
<tr>
<td>1001</td>
<td>2</td>
<td>Jorge</td>
<td>30-Sep-02</td>
</tr>
<tr>
<td>1003</td>
<td>1</td>
<td>Suzanne</td>
<td>25-Jan-04</td>
</tr>
<tr>
<td>1006</td>
<td>1</td>
<td>Carlos</td>
<td>25-May-01</td>
</tr>
<tr>
<td>1008</td>
<td>1</td>
<td>Michael</td>
<td>19-Feb-95</td>
</tr>
<tr>
<td>1008</td>
<td>2</td>
<td>George</td>
<td>27-Jun-98</td>
</tr>
<tr>
<td>1008</td>
<td>3</td>
<td>Katherine</td>
<td>18-Aug-03</td>
</tr>
</tbody>
</table>
• Given this scenario, and with the help of this relationship, you can determine that:

• Jeanine J. Callifante claims two dependents, Annelise and Jorge.

• Keep in mind that the database designer usually determines whether an entity can be described as weak based on the business rules.
Relationship Participation

- Participation in an entity relationship is either **optional** or **mandatory**.

- Recall that relationships are bidirectional; that is, they operate in both directions. If COURSE is related to CLASS, then by definition, CLASS is related to COURSE.

- Because of the bidirectional nature of relationships, it is necessary to determine the connectivity of the relationship from COURSE to CLASS and the connectivity of the relationship from CLASS to COURSE.

- Similarly, the specific maximum and minimum cardinalities must be determined in each direction for the relationship.

- Once again, you must consider the bidirectional nature of the relationship when determining participation.
Optional participation means that one entity occurrence does not require a corresponding entity occurrence in a particular relationship.

For example, in the “COURSE generates CLASS” relationship, you noted that at least some courses do not generate a class. In other words, an entity occurrence (row) in the COURSE table does not necessarily require the existence of a corresponding entity occurrence in the CLASS table. (Remember that each entity is implemented as a table.)

Therefore, the CLASS entity is considered to be optional to the COURSE entity. In Crow’s Foot notation, an optional relationship between entities is shown by drawing a small circle (O) on the side of the optional entity.

The existence of an optional entity indicates that the minimum cardinality is 0 for the optional entity. (The term optionality is used to label any condition in which one or more optional relationships exist.)

Remember that the burden of establishing the relationship is always placed on the entity that contains the foreign key. In most cases, that will be the entity on the “many” side of the relationship.
• **Mandatory participation** means that one entity occurrence requires a corresponding entity occurrence in a particular relationship.

• If no **optionality symbol** is depicted with the entity, the entity is assumed to exist in a **mandatory relationship** with the related entity.

• If the **mandatory participation** is depicted graphically, it is typically shown as a *small hash* mark across the relationship line, similar to the Crow’s Foot depiction of a connectivity of 1.

• The existence of a mandatory relationship indicates that the **minimum cardinality is at least 1** for the mandatory entity.
### Table 4.3 Crow’s Foot Symbols

<table>
<thead>
<tr>
<th>CROW’S FOOT SYMBOL</th>
<th>CARDINALITY</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>O⊆</td>
<td>(0,N)</td>
<td>Zero or many. Many side is optional.</td>
</tr>
<tr>
<td>⋮</td>
<td>(1,N)</td>
<td>One or many. Many side is mandatory.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O-</td>
<td></td>
<td>(0,1)</td>
</tr>
</tbody>
</table>
Crow’s Foot Example

• If you examine the “PROFESSOR teaches CLASS” relationship, it is quite possible for a PROFESSOR not to teach a CLASS. Therefore, CLASS is optional to PROFESSOR. On the other hand, a CLASS must be taught by a PROFESSOR. Therefore, PROFESSOR is mandatory to CLASS.

• Note that the ERD model in shows the cardinality next to CLASS to be (0,3), thus indicating that a professor may teach no classes at all or as many as three classes. And each CLASS table row will reference one and only one PROFESSOR row—assuming each class is taught by one and only one professor—represented by the (1,1) cardinality next to the PROFESSOR table.
• Failure to understand the distinction between mandatory and optional participation in relationships might yield designs in which awkward (and unnecessary) temporary rows (entity instances) must be created just to accommodate the creation of required entities.

• Therefore, it is important that you clearly understand the concepts of mandatory and optional participation.
Relationship Degree

• A **relationship degree** indicates the number of entities or participants associated with a relationship.

• A **unary relationship** exists when an association is maintained within a single entity.

• A **binary relationship** exists when two entities are associated.

• A **ternary relationship** exists when three entities are associated.

• Although higher degrees exist, they are rare and are not specifically named.
Fig. 4.12 Three Types of Relationship Degree
A **recursive relationship** is one in which a relationship can exist between occurrences of the same entity set. (Naturally, such a condition is found within a unary relationship.)

**Fig. 4.13 ER Representation of Recursive Relationships**
Developing an ER Diagram

• The process of database design is an iterative rather than a linear or sequential process. The verb *iterate* means “to do again or repeatedly.”

• An **iterative process** is, thus, one based on repetition of processes and procedures.

• Building an ERD usually involves the following activities:
  1. Create a detailed narrative of the organization’s description of operations.
  2. Identify the business rules based on the description of operations.
  3. Identify the main entities and relationships from the business rules.
  4. Develop the initial ERD.
  5. Identify the attributes and primary keys that adequately describe the entities.
  6. Revise and review the ERD.
During the review process, it is likely that additional objects, attributes, and relationships will be uncovered.

Therefore, the basic ERM will be modified to incorporate the newly discovered ER components.

Subsequently, another round of reviews might yield additional components or clarification of the existing diagram.

The process is repeated until the end users and designers agree that the ERD is a fair representation of the organization’s activities and functions.
• During the design process, the database designer does not depend simply on interviews to help define entities, attributes, and relationships.

• A surprising amount of information can be gathered by examining the business forms and reports that an organization uses in its daily operations.

• To illustrate the use of the iterative process that ultimately yields a workable ERD, let’s start with an initial interview with the Tiny College administrators. The interview process yields the following business rules:
1. Tiny College (TC) is divided into several schools: a school of business, a school of arts and sciences, a school of education, and a school of applied sciences. Each school is administered by a dean who is a professor. Each professor can be the dean of only one school, and a professor is not required to be the dean of any school.

Therefore, a 1:1 relationship exists between PROFESSOR and SCHOOL. Note that the cardinality can be expressed by writing (1,1) next to the entity PROFESSOR and (0,1) next to the entity SCHOOL.
2. Each school comprises several departments.

For example, the school of business has an accounting department, a management/marketing department, an economics/finance department, and a computer information systems department.

Note again the cardinality rules: The smallest number of departments operated by a school is one, and the largest number of departments is indeterminate (N). On the other hand, each department belongs to only a single school;

Thus, the cardinality is expressed by (1,1). That is, the minimum number of schools that a department belongs to is one, as is the maximum number.
First segment of Tiny College ERD
3. Each department may offer courses.

For example, the management/marketing department offers courses such as Introduction to Management, Principles of Marketing, and Production Management.

Note that this relationship is based on the way Tiny College operates.

If, for example, Tiny College had some departments that were classified as “research only,” those departments would not offer courses; therefore, the COURSE entity would be optional to the DEPARTMENT entity.
Second segment of Tiny College ERD

![Database ERD Diagram](image)

**DEPARTMENT**
- PK: DEPT_CODE
- DEPT_NAME

**COURSE**
- PK: CRS_CODE
- FK1: DEPT_CODE, CRS_TITLE, CRS_DESCRIPTION, CRS_CREDIT
4. A CLASS is a section of a COURSE. That is, a department may offer several sections (classes) of the same database course. Each of those classes is taught by a professor at a given time in a given place.

In short, a 1:M relationship exists between COURSE and CLASS. However, because a course may exist in Tiny College’s course catalog even when it is not offered as a class in a current class schedule, CLASS is optional to COURSE.
Third segment of Tiny College ERD
5. Each department should have one or more professors assigned to it. One and only one of those professors chairs the department, and no professor is required to accept the chair position.

Therefore, DEPARTMENT is optional to PROFESSOR in the “chairs” relationship.
6. Each professor may teach up to four classes; each class is a section of a course. A professor may also be on a research contract and teach no classes at all.

Fifth segment of Tiny College ERD
7. A student may enroll in several classes but takes each class only once during any given enrollment period.

For example, during the current enrollment period, a student may decide to take five classes—Statistics, Accounting, English, Database, and History—but that student would not be enrolled in the same Statistics class five times during the enrollment period!

Each student may enroll in up to six classes, and each class may have up to 35 students, thus creating an M:N relationship between STUDENT and CLASS. Because a CLASS can initially exist (at the start of the enrollment period) even though no students have enrolled in it, STUDENT is optional to CLASS in the M:N relationship. This M:N relationship must be divided into two 1:M relationships through the use of the ENROLL entity, but note that the optional symbol is shown next to ENROLL. If a class exists but has no students enrolled in it, that class doesn’t occur in the ENROLL table. Note also that the ENROLL entity is weak: it is existence-dependent, and its (composite) PK is composed of the PKs of the STUDENT and CLASS entities. You can add the cardinalities (0,6) and (0,35) next to the ENROLL entity to reflect the business rule constraints.
Sixth segment of Tiny College ERD
8. Each department has several (or many) students whose major is offered by that department. However, each student has only a single major and is, therefore, associated with a single department.

However, in the Tiny College environment, it is possible—at least for a while—for a student not to declare a major field of study. Such a student would not be associated with a department; therefore, DEPARTMENT is optional to STUDENT. It is worth repeating that the relationships between entities and the entities themselves reflect the organization’s operating environment. That is, the business rules define the ERD components.
Seventh segment of Tiny College ERD
9. Each student has an advisor in his or her department; each advisor counsels several students. An advisor is also a professor, but not all professors advise students. Therefore, STUDENT is optional to PROFESSOR in the “PROFESSOR advises STUDENT” relationship.
10. The CLASS entity contains a ROOM_CODE attribute. Given the naming conventions, it is clear that ROOM_CODE is an FK to another entity. Clearly, because a class is taught in a room, it is reasonable to assume that the ROOM_CODE in CLASS is the FK to an entity named ROOM. In turn, each room is located in a building. So the last Tiny College ERD is created by observing that a BUILDING can contain many ROOMs, but each ROOM is found in a single BUILDING. In this ERD segment, it is clear that some buildings do not contain (class) rooms. For example, a storage building might not contain any named rooms at all.
Ninth segment of Tiny College ERD
Using the preceding summary, you can identify the following entities:

- SCHOOL
- COURSE
- DEPARTMENT
- CLASS
- PROFESSOR
- STUDENT
- BUILDING
- ROOM
- ENROLL (the associative entity between STUDENT and CLASS)

Once you have discovered the relevant entities, you can define the initial set of relationships among them. Next, you describe the entity attributes. Identifying the attributes of the entities helps you to better understand the relationships among entities.

Table 4.4 summarizes the ERM’s components, and names the entities and their relations.
Table 4.4 Components of the ERM

<table>
<thead>
<tr>
<th>ENTITY</th>
<th>RELATIONSHIP</th>
<th>CONNECTIVITY</th>
<th>ENTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHOOL</td>
<td>Operates</td>
<td>1:M</td>
<td>DEPARTMENT</td>
</tr>
<tr>
<td>DEPARTMENT</td>
<td>Has</td>
<td>1:M</td>
<td>STUDENT</td>
</tr>
<tr>
<td>DEPARTMENT</td>
<td>Employs</td>
<td>1:M</td>
<td>PROFESSOR</td>
</tr>
<tr>
<td>DEPARTMENT</td>
<td>Offers</td>
<td>1:M</td>
<td>COURSE</td>
</tr>
<tr>
<td>COURSE</td>
<td>Generates</td>
<td>1:M</td>
<td>CLASS</td>
</tr>
<tr>
<td>PROFESSOR</td>
<td>Is dean of</td>
<td>1:1</td>
<td>SCHOOL</td>
</tr>
<tr>
<td>PROFESSOR</td>
<td>Chairs</td>
<td>1:1</td>
<td>DEPARTMENT</td>
</tr>
<tr>
<td>PROFESSOR</td>
<td>Teaches</td>
<td>1:M</td>
<td>CLASS</td>
</tr>
<tr>
<td>PROFESSOR</td>
<td>Advises</td>
<td>1:M</td>
<td>STUDENT</td>
</tr>
<tr>
<td>STUDENT</td>
<td>Enrolls in</td>
<td>M:N</td>
<td>CLASS</td>
</tr>
<tr>
<td>BUILDING</td>
<td>Contains</td>
<td>1:M</td>
<td>ROOM</td>
</tr>
<tr>
<td>ROOM</td>
<td>Is used for</td>
<td>1:M</td>
<td>CLASS</td>
</tr>
</tbody>
</table>

Note: ENROLL is the composite entity that implements the M:N relationship “STUDENT enrolls in CLASS”
• You must also define the connectivity and cardinality for the just-discovered relations based on the business rules.

• However, to avoid crowding the diagram, the cardinalities are not shown. Figure 4.35 shows the Crow’s Foot ERD for Tiny College.

• Note that this is an implementation-ready model. Therefore it shows the ENROLL composite entity.
The completed ERD for Tiny College
Conceptual UML Class diagram for Tiny College
Database Design Challenges: Conflicting Goals

- Database designers often must make design compromises that are triggered by conflicting goals, such as adherence to design standards (design elegance), processing speed, and information requirements.

Design standards. The database design must conform to design standards. Such standards have guided you in developing logical structures that minimize data redundancies, thereby minimizing the likelihood that destructive data anomalies will occur.
• **Processing speed.** In many organizations, particularly those generating large numbers of transactions, high processing speeds are often a top priority in database design. High processing speed means minimal access time, which may be achieved by minimizing the number and complexity of logically desirable relationships.

• **Information requirements.** The quest for timely information might be the focus of database design. Complex information requirements may dictate data transformations, and they may expand the number of entities and attributes within the design. Therefore, the database may have to sacrifice some of its “clean” design structures and/or some of its high transaction speed to ensure maximum information generation.
Module 4: Summary

- The ERM uses ERDs to represent the conceptual database as viewed by the end user.

- The ERM’s main components are entities, relationships, and attributes.

- The ERD also includes connectivity and cardinality notations.

- An ERD can also show relationship strength, relationship participation (optional or mandatory), and degree of relationship (unary, binary, ternary, etc.).
• Connectivity describes the relationship classification (1:1, 1:M, or M:N).

• Cardinality expresses the specific number of entity occurrences associated with an occurrence of a related entity.

• Connectivities and cardinalities are usually based on business rules.

• In the ERM, an M:N relationship is valid at the conceptual level.

• However, when implementing the ERM in a relational database, the M:N relationship must be mapped to a set of 1:M relationships through a composite entity.
ERDs may be based on many different ERMs. However, regardless of which model is selected, the modeling logic remains the same. Because no ERM can accurately portray all real-world data and action constraints, application software must be used to augment the implementation of at least some of the business rules.

Unified Modeling Language (UML) class diagrams are used to represent the static data structures in a data model. The symbols used in the UML class and ER diagrams are very similar. The UML class diagrams can be used to depict data models at the conceptual or implementation abstraction levels.
Database designers, no matter how well they are able to produce designs that conform to all applicable modeling conventions, are often forced to make design compromises.

Those compromises are required when end users have vital transaction-speed and/or information requirements that prevent the use of “perfect” modeling logic and adherence to all modeling conventions.

Therefore, database designers must use their professional judgment to determine how and to what extent the modeling conventions are subject to modification.

To ensure that their professional judgments are sound, database designers must have detailed and in-depth knowledge of data-modeling conventions.

It is also important to document the design process from beginning to end, which helps keep the design process on track and allows for easy modifications down the road.