Lecture 2: Link | Syllabus
*Class Week 2/15

Database Systems by Coronel & Morris

Chapter 4: Entity Relationship (ER) Modeling

* 4.1 *The Entity Relationship Model (ERM)*

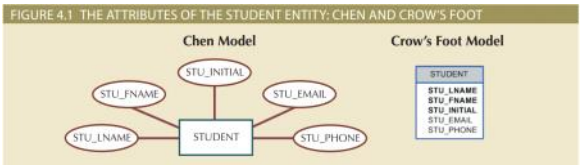
A conceptual models such as the ERM can be used to understand and design the data requirements of an organization. Therefore, the ERM is independent of the database type. Conceptual models are used in the conceptual design of databases, while relational models (ERD) are used in the logical design of databases

* 4.1-a *Entities*

At the ER modeling level, an entity actually refers to the entity set and not to a single entity occurrence. In other words, an 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.

* 4.1-b *Attributes*

Attributes are characteristics of entities. For example, the STUDENT entity includes the attributes STU_LNAME, STU_FNAME, and STU_INITIAL, among many others.

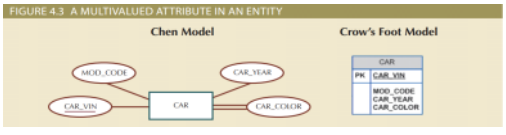


* A required attribute must have a value; in other words, it cannot be left empty. As shown in Figure 4.1, the two boldfaced attributes in the Crow's Foot notation indicate that data entry will be required. STU_LNAME and STU_FNAME require data entries because all students are assumed to have a last name and a first name

* 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. The domain for a gender attribute consists of only two possibilities: M or F (or some other equivalent code).

* Identifiers (Primary Keys) The ERM uses identifiers—one or more attributes that uniquely identify each entity instance. In the relational model, entities are mapped to tables, and the entity identifier is mapped as the table's primary key (PK). Identifiers are underlined in the ERD, this also applies to all identifiers in a composite identifier
* A composite identifier, a primary key composed of more than one attribute

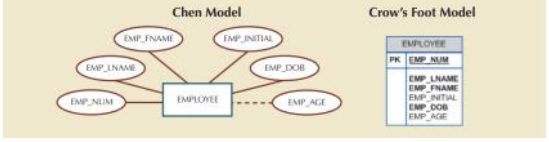
* A composite attribute is not the same as a composite identifier. It's an attribute that can be further subdivided to yield additional attributes. For example, a phone number such as 615-898- 2368 may be divided into an area code (615), an exchange number (898), and a four-digit code (2368)
* A simple 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
* 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
* A 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 for the roof, body, and trim. In the Chen ERM, multivalued attributes are shown by a double line connecting the attribute to the entity. The Crow's Foot notation does not identify multivalued attributes



* Although the conceptual model can handle M:N relationships and multivalued attributes, you should not implement them in the RDBMS. Remember from Chapter 3 that in the relational table, each column and row intersection represents a single data value

* 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

FIGURE 4.6 DEPICTION OF A DERIVED ATTRIBUTE



ADVANTAGES AND DISADVANTAGES OF STORING DERIVED ATTRIBUTES

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

*** 4.1-c** *Relationships*

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.

*** 4.1-d** *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. Many database designers who use Crow's Foot modeling notation do not depict the specific cardinalities on the ER diagram itself because the specific limits described by the cardinalities cannot be implemented directly through the database design

*** 4.1-e** *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. If an entity can exist apart from all of its related entities, then it is existence-independent, and it is referred to as a strong entity or regular entity

*** 4.1-f** *Relationship Strength*

Weak (Non-Identifying) Relationships is a weak relationship, also known as a non-identifying relationship, exists if the primary key of the related entity does not contain a primary key component of the parent entity. By default, relationships are established by having the primary key of the parent entity appear as a foreign key (FK) on the related entity (also known as the child entity).
Strong (Identifying) Relationships is a strong (identifying) relationship exists when the primary key of the related entity contains a primary key component of the parent entity

*** 4.1-e** *Relationship Participation*

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

CROW'S FOOT SYMBOLS

SYMBOL	CARDINALITY	COMMENT
O<	(0,N)	Zero or many; the "many" side is optional.
|<	(1,N)	One or many; the "many" side is mandatory.
||	(1,1)	One and only one; the "1" side is mandatory.
O|	(0,1)	Zero or one; the "1" side is optional.

*** 4.1-i** *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. Binary relationships exists when two entities are associated. A ternary relationship when three entities are associated. Although higher degrees exist, they are rare and are not specifically named.

*** 4.1-j** *Recursive Relationship*

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.) For example, a 1:M unary relationship can be expressed by "an EMPLOYEE may manage many EMPLOYEEs, and each EMPLOYEE is managed by one EMPLOYEE." Also, as long as polygamy is not legal, a 1:1 unary relationship may be expressed by "an EMPLOYEE may be married to one and only one other EMPLOYEE." Finally, the M:N

unary relationship may be expressed by "a COURSE may be a prerequisite to many other COURSEs, and each COURSE may have many other COURSEs as prerequisites."

FIGURE 4.17 AN ER REPRESENTATION OF RECURSIVE RELATIONSHIPS

*** 4.1-k** *Associative (Composite) Entities*

A M:N relationships are a valid construct at the conceptual level, and therefore are found frequently during the ER modeling process. However, implementing the M:N particularly in the relational model, requires the use of an additional entity. The ER model uses the associative entity to represent an M:N relationship between two or more entities. This associative entity, also called a composite or bridge entity, is in a 1:M relationship with the parent entities and is composed of the primary key attributes of each parent entity.

FIGURE 4.23 CONVERTING THE M:N RELATIONSHIP INTO TWO 1:M RELATIONSHIPS

Table name: STUDENT — Database name: Ch04_CollegeTry

STU_NUM	STU_LNAME
321452	Bowser
324257	Smithson

Table name: ENROLL

CLASS_CODE	STU_NUM	ENROLL_GRADE
10014	321452	C
10014	324257	B
10018	321452	A
10018	324257	B
10021	321452	C
10021	324257	C

Table name: CLASS

CLASS_CODE	CRS_CODE	CLASS_SECTION	CLASS_TIME	ROOM_CODE	PROF_NUM
10014	ACCT-211	3	TTh 2:30-3:45 p.m.	BUS252	342
10018	CIS-220	2	MWF 9:00-9:50 a.m.	KLR211	114
10021	QM-261	1	MWF 8:00-8:50 a.m.	KLR200	114

* Note that the composite ENROLL entity in Figure 4.23 is existence-dependent on the other two entities; the composition of the ENROLL entity is based on the primary keys of the entities that are connected by the composite entity. The composite entity may also contain additional attributes that play no role in the connective process. For example, although the entity must be composed of at least the STUDENT and CLASS primary keys, it may also include such additional attributes as grades, absences, and other data uniquely identified by the student's performance in a specific class

*** 4.2** *Developing an ER Diagram*

The process of database design is iterative rather than a linear or sequential process. The verb iterate means "to do again or repeatedly." Thus, an iterative process is based on repetition of processes and procedures.

Building an ERD usually involves the following activities:

- Create a detailed narrative of the organization's description of operations.
- Identify the business rules based on the description of operations
- Identify the main entities and relationships from the business rules.
- Develop the initial ERD.
- Identify the attributes and primary keys that adequately describe the entities.
- Revise and review the ERD

Chapter 5: Advance Data Modeling

*** 5.1** *Extended Entity Relationship Model*

The extended entity relationship model (EERM), sometimes referred to as the enhanced entity relationship model, is the result of adding more semantic constructs to the original entity relationship (ER) model. As you might expect, a diagram that uses the EERM is called an EER diagram (EERD).

*** 5.1-a** *Entity Supertypes and Subtypes*

Most employees possess a wide range of skills and special qualifications, data modelers must find a variety of ways to group employees based on their characteristics. For instance, a retail company could group employees as salaried and hourly, while a university could group employees as faculty, staff, and administrators.

The grouping of employees into various types provides two important benefits:

- It avoids unnecessary nulls in attributes when some employees have characteristics that are not shared by other employees.
- It enables a particular employee type to participate in relationships that are unique to that employee type

In modeling terms, an entity supertype is a generic entity type that is related to one or more entity subtypes. The entity supertype contains common characteristics, and the entity subtypes each contain their own unique characteristics

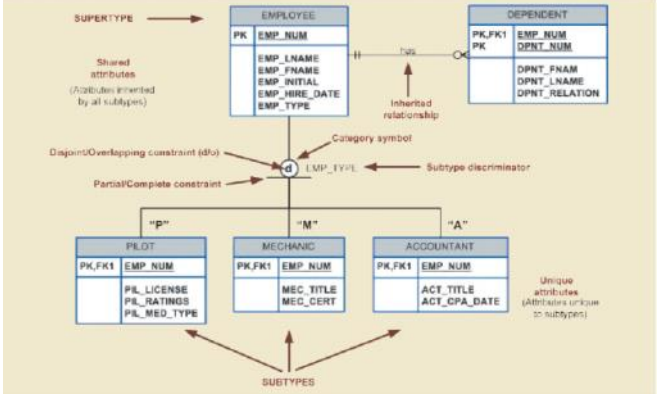
Two criteria help the designer determine when to use subtypes and supertypes :

- There must be different, identifiable kinds or types of the entity in the user's environment.
- The different kinds or types of instances should each have one or more attributes that are unique to that kind or type of instance.

*** 5.1-b** *Specialization Hierarchy*

Entity supertypes and subtypes are organized in a specialization hierarchy, which depicts the arrangement of higher-level entity supertypes (parent entities) and lower-level entity subtypes (child entities). Specialization hierarchies enable the data model to capture additional semantic content (meaning) into the ERD

FIGURE 5.2 A SPECIALIZATION HIERARCHY

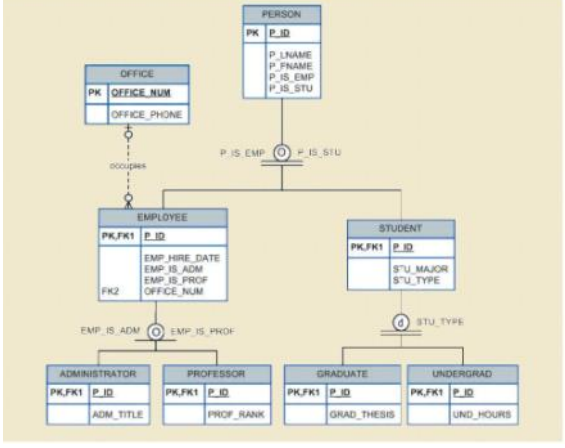


* A specialization hierarchy provides the means to:
- Support attribute inheritance
- Define a special supertype attribute known as the subtype discriminator
- Define disjoint/overlapping constraints and complete/partial constraints

* 5.1-c *Inheritance*

The property of inheritance enables an entity subtype to inherit the attributes and relationships of the supertype. As discussed earlier, a supertype contains attributes that are common to all of its subtypes. *One important inheritance characteristic is that all entity subtypes inherit their primary key attribute from their supertype.* At the implementation level, the supertype and its subtype(s) depicted in the specialization hierarchy maintain a 1:1 relationship.

FIGURE 5.4 SPECIALIZATION HIERARCHY WITH OVERLAPPING SUBTYPES



* Inheriting the relationships of their supertypes does not mean that subtypes cannot have relationships of their own. Figure 5.4 illustrates a 1:M relationship between EMPLOYEE, a subtype of PERSON, and OFFICE. Because only employees and no other type of person will ever have an office within this system, the relationship is modeled with the subtype directly

* 5.1-d *Subtype Discriminator*

A subtype discriminator is the attribute in the supertype entity that determines to which subtype the supertype occurrence is related. Note that the default comparison condition for the subtype discriminator attribute is the equality comparison. However, in some situations the subtype discriminator is not necessarily based on an equality comparison.

* 5.1-e *Disjoint and Overlapping Constraints*

An entity supertype can have disjoint or overlapping entity subtypes. Disjoint subtypes, also known as nonoverlapping subtypes, are subtypes that contain a unique subset of the supertype entity set; in other words, each entity instance of the supertype can appear in only one of the subtypes. For example, in Figure 5.2, an employee (supertype) who is a pilot (subtype) can appear only in the PILOT subtype, not in any of the other subtypes. In an ERD, such disjoint subtypes are indicated by the letter d inside the category shape. if the business rule specifies that employees can have multiple classifications, the EMPLOYEE supertype may contain overlapping job classification subtypes. Overlapping subtypes are subtypes that contain nonunique subsets of the supertype entity set; that is, each entity instance of the supertype may appear in more than one subtype.

SPECIALIZATION HIERARCHY CONSTRAINT SCENARIOS

TYPE	DISJOINT CONSTRAINT	OVERLAPPING CONSTRAINT
Partial 	Supertype has optional subtypes. Subtype discriminator can be null. Subtype sets are unique.	Supertype has optional subtypes. Subtype discriminators can be null. Subtype sets are not unique.
Total 	Every supertype occurrence is a member of only one subtype. Subtype discriminator cannot be null. Subtype sets are unique.	Every supertype occurrence is a member of at least one subtype. Subtype discriminators cannot be null. Subtype sets are not unique.

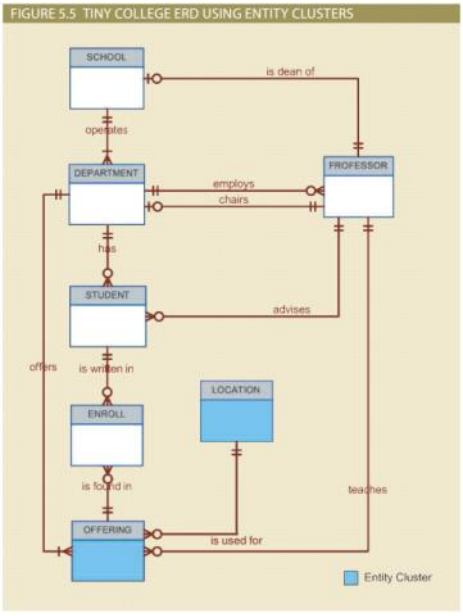
* **5.1-g** *Specialization and Generalization*

Specialization is the top-down process of identifying lower-level, more specific entity subtypes from a higher-level entity supertype. Specialization is based on grouping the unique characteristics and relationships of the subtypes. In the aviation example, you used specialization to identify multiple entity subtypes from the original employee supertype. Generalization is the bottom-up process of identifying a higher-level, more generic entity supertype from lower-level entity subtypes. Generalization is based on grouping the common characteristics and relationships of the subtypes.

* **5.2** *Entity Clustering*

An entity cluster is a "virtual" entity type used to represent multiple entities and relationships in the ERD. An entity cluster is formed by combining multiple interrelated entities into a single, abstract entity object. An entity cluster is considered "virtual" or "abstract" in the sense that it is not actually an entity in the final ERD. Instead, it is a temporary entity used to represent multiple entities and relationships, with the purpose of simplifying the ERD and thus enhancing its readability

FIGURE 5.5 TINY COLLEGE ERD USING ENTITY CLUSTERS



* Figure 5.5 illustrates the use of entity clusters based on the Tiny College exp.
  - OFFERING, which groups the SEMESTER, COURSE, and CLASS entities and relationships
  - LOCATION, which groups the ROOM and BUILDING entities and relationships

* Note also that the ERD in Figure 5.5 does not show attributes for the entities. When using entity clusters, the key attributes of the combined entities are no longer available.

Without the key attributes, primary key inheritance rules change. In turn, the change in the inheritance rules can have undesirable consequences, such as changes in relationships—from identifying to nonidentifying or vice versa—and the loss of foreign key attributes from some entities.

To eliminate those problems, the general rule is to *avoid the display of attributes when entity clusters are used*

* **5.3** *Entity Integrity: Selecting Primary Keys*

Arguably, the most important characteristic of an entity is its primary key (a single attribute or some combination of attributes), which uniquely identifies each entity instance. The primary key's function is to guarantee entity integrity. Furthermore, primary keys and foreign keys work together to implement relationships in the relational model. Therefore, the importance of properly selecting the primary key has a direct bearing on the efficiency and effectiveness of database implementation

* **5.3-a** *Natural Keys and Primary Keys*

A natural key or natural identifier is a real-world, generally accepted identifier used to distinguish—that is, uniquely identify—real-world objects. As its name implies, a natural key is familiar to end users and forms part of their day-today business vocabulary. Usually, if an entity has a natural identifier, a data modeler uses it as the primary key of the entity being modeled. Generally, most natural keys make acceptable primary key identifiers. The next section presents some basic guidelines for selecting primary keys.

* **5.3-b** *Primary Key Guidelines*

A primary key is the attribute or combination of attributes that uniquely identifies entity instances in an entity set.

1. First, you should understand the function of a primary key. Its main function is to uniquely identify an entity instance or row within a table. In particular, given a primary key value—that is, the determinant—the relational model can determine the value of all dependent attributes that "describe" the entity. *The function of the primary key is to guarantee entity integrity, not to "describe" the entity*
2. Second, primary keys and foreign keys are used to implement relationships among entities. However, the implementation of such relationships is done mostly behind the scenes, hidden from end users. In the real world, end users identify objects based on the characteristics they know about the objects

* **5.3-c** *When to Use Composite Primary Keys*

In the previous section, you learned about the desirable characteristics of primary keys. For example, you learned that the primary key should use the minimum number of attributes possible. However, that does not mean that composite primary keys are not permitted in a model.

In fact, composite primary keys are particularly useful in two cases :

- As identifiers of composite entities, in which each primary key combination is allowed only once in the M:N relationship
- As identifiers of weak entities, in which the weak entity has a strong identifying relationship with the parent entity

*** 5.3-d** *When to Use Surrogate Primary Keys*

In some instances a primary key doesn't exist in the real world or the existing natural key might not be a suitable primary key. In these cases, it is standard practice to create a surrogate key. A surrogate key is a primary key created by the database designer to simplify the identification of entity instances.

The surrogate key has no meaning in the user's environment—it exists only to distinguish one entity instance from another (just like any other primary key). One practical advantage of a surrogate key is that because it has no intrinsic meaning, values for it can be generated by the DBMS to ensure that unique values are always provided