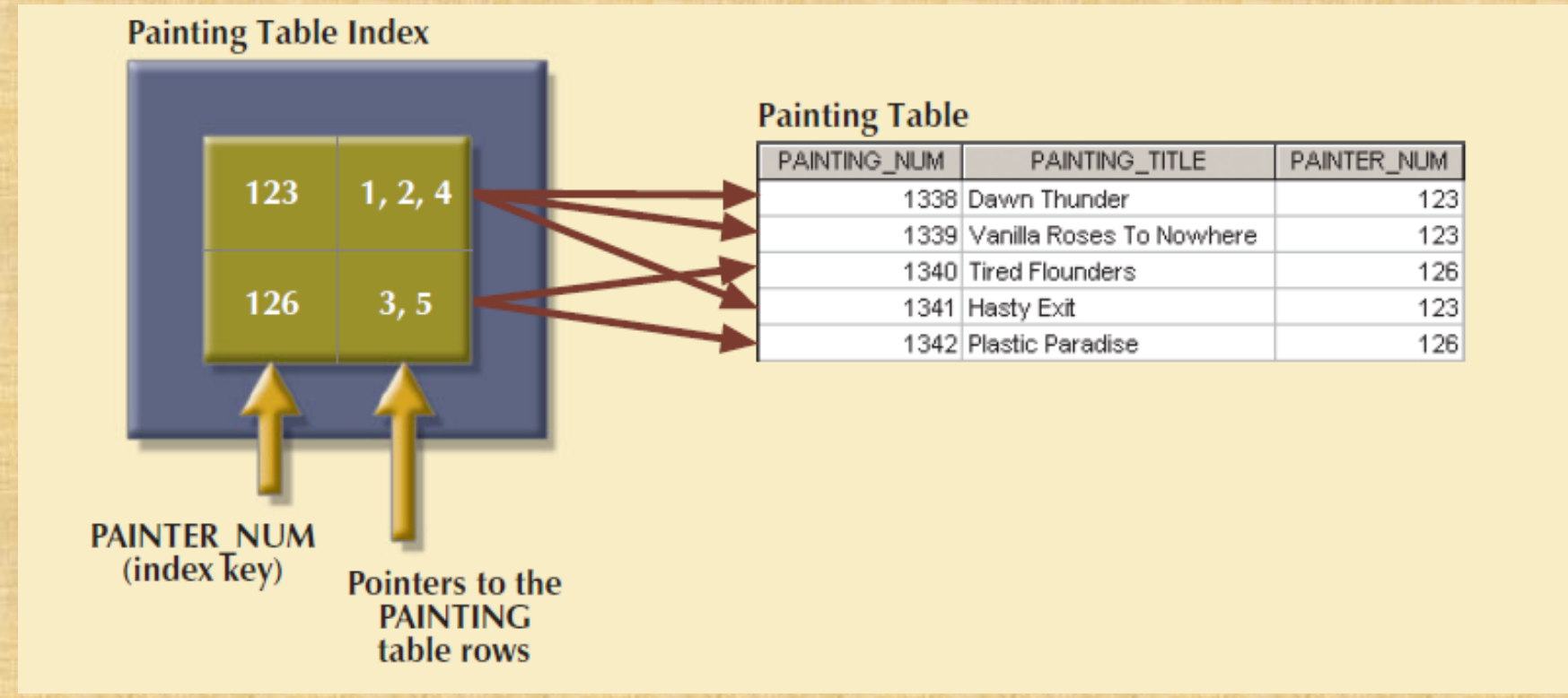
Indexes

- Suppose you want to locate a particular book in a library. Does it make sense to look through every book in the library until you find the one you want? Of course not; you use the library's catalog, which is indexed by title, topic, and author. The index (in either a manual or a computer system) points you to the book's location, thereby making retrieval of the book a quick and simple matter. An index is an orderly arrangement used to logically access rows in a table.
- From a conceptual point of view, an index is composed of an index key and a set of pointers. The index key is, in effect, the index's reference point. More formally, an index is an ordered arrangement of keys and pointers. Each key points to the location of the data identified by the key.

Indexes

- For Example: Paintings created by a given painter, Without an index, you must read each row in the PAINTING table and see if the PAINTER_NUM matches the requested painter. However, if you index the PAINTER table and use the index key PAINTER_NUM, you merely need to look up the appropriate PAINTER_NUM in the index and find the matching pointers.

Components of an index



Indexes

- **Note** that the first PAINTER_NUM index key value (123) is found in records 1, 2, and 4 of the PAINTING table in Figure
- The second PAINTER_NUM index key value (126) is found in records 3 and 5 of the PAINTING table in Figure
- DBMSs use indexes for many different purposes. You just learned that an index can be used to retrieve data more efficiently.
- But indexes can also be used by a DBMS to retrieve data ordered by a specific attribute or attributes. For example, creating an index on a customer's last name will allow you to retrieve the customer data alphabetically by the customer's last name. Also, an index key can be composed of one or more attributes. For example, in Figure, you can create an index on VEND_CODE and PROD_CODE to retrieve all rows in the PRODUCT table ordered by vendor, and within vendor, ordered by product.

Indexes

- A unique index, as its name implies, is an index in which the index key can have only one pointer value (row) associated with it. (The index in Figure is not a unique index because the PAINTER_NUM has multiple pointer values associated with it. For example, painter number 123 points to three rows—1, 2, and 4—in the PAINTING table.)

Codd's Relational Database Rules

TABLE 3.8 Dr. Codd's 12 Relational Database Rules

RULE	RULE NAME	DESCRIPTION
1	Information	All information in a relational database must be logically represented as column values in rows within tables.
2	Guaranteed Access	Every value in a table is guaranteed to be accessible through a combination of table name, primary key value, and column name.
3	Systematic Treatment of Nulls	Nulls must be represented and treated in a systematic way, independent of data type.
4	Dynamic On-Line Catalog Based on the Relational Model	The metadata must be stored and managed as ordinary data, that is, in tables within the database. Such data must be available to authorized users using the standard database relational language.
5	Comprehensive Data Sublanguage	The relational database may support many languages. However, it must support one well defined, declarative language with support for data definition, view definition, data manipulation (interactive and by program), integrity constraints, authorization, and transaction management (begin, commit, and rollback).
6	View Updating	Any view that is theoretically updatable must be updatable through the system.
7	High-Level Insert, Update and Delete	The database must support set-level inserts, updates, and deletes.
8	Physical Data Independence	Application programs and ad hoc facilities are logically unaffected when physical access methods or storage structures are changed.
9	Logical Data Independence	Application programs and ad hoc facilities are logically unaffected when changes are made to the table structures that preserve the original table values (changing order of column or inserting columns).
10	Integrity Independence	All relational integrity constraints must be definable in the relational language and stored in the system catalog, not at the application level.
11	Distribution Independence	The end users and application programs are unaware and unaffected by the data location (distributed vs. local databases).
12	Nonsubversion	If the system supports low-level access to the data, there must not be a way to bypass the integrity rules of the database.
	Rule Zero	All preceding rules are based on the notion that in order for a database to be considered relational, it must use its relational facilities exclusively to manage the database.

Chapter 4: Entity Relationship (ER) Modeling

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**

ERM is independence

- 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.
- The relational model is used extensively to explain ER constructs and the way they are used to develop database designs.

The Entity Relationship Model (ERM)

- The Relational Database Model, that 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.
- Because an entity represents a real-world object, the words entity and object are often used interchangeably.

The Entity Relationship Model (ERM)

- 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.

The Entity Relationship Model (ERM)

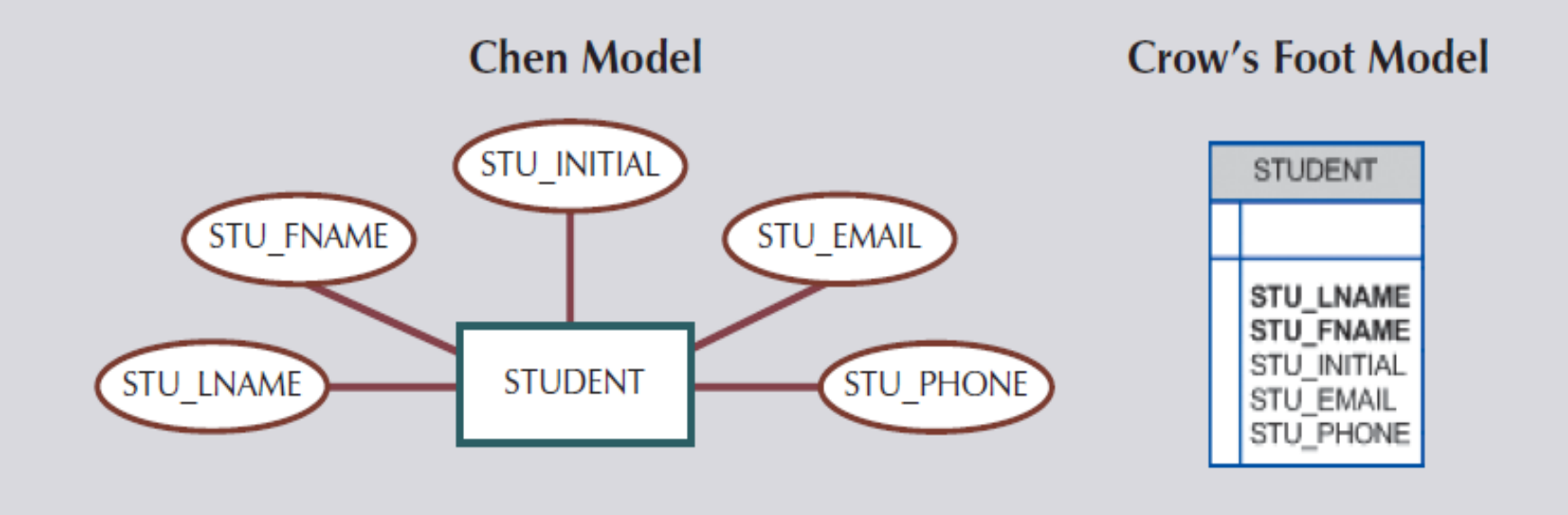
- Entities:
  - An entity actually refers to the entity set and not to a single entity occurrence. In other words, the word entity in the ERM corresponds to a table—not to a row—in the relational environment.
  - The ERM refers to a table row as an entity instance or entity occurrence. In both the Chen 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.

The Entity Relationship Model (ERM)

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

The Entity Relationship Model (ERM)

- Attributes:



Because the Chen representation is rather space consuming, software vendors have adopted the Crow's Foot style attribute display.

The Entity Relationship Model (ERM)

- Required and Optional Attributes:
  - A required attribute is an attribute that must have a value; in other words, it cannot be left empty. 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.

The Entity Relationship Model (ERM)

- 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 (0,4) because the lowest possible GPA value is 0 and the highest possible value is 4.
- Attributes may share a domain. For example, the PROFESSOR and STUDENT entities may each have an attribute named ADDRESS and could therefore share a domain.

The Entity Relationship Model (ERM)

- Identifiers (Primary Keys)
  - 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)

For example, a CAR entity may be represented by:

- CAR (CAR_VIN, MOD_CODE, CAR_YEAR, CAR_COLOR)

  (Each car is identified by a unique vehicle identification number, or CAR_VIN.)

Composite Identifiers

- Ideally, an entity identifier is composed of only a single attribute. For Example: single-attribute primary key named CLASS_CODE.
- However, it is possible to use a composite identifier, that is, a primary key composed of more than one attribute. For Example: composite primary key composed of the combination of CRS_CODE and CLASS_SECTION.
- Given the current structure of the CLASS table shown in Figure in the next slide, CLASS_CODE is the primary key and the combination of CRS_CODE and CLASS_SECTION is a proper candidate key. If the CLASS_CODE attribute is deleted from the CLASS entity, the candidate key (CRS_CODE and CLASS_SECTION) becomes an acceptable composite primary key.

Composite Identifiers

The CLASS table (entity) components and contents

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

CLASS (**CLASS_CODE**, CRS_CODE, CLASS_SECTION, CLASS_TIME, ROOM_CODE, PROF_NUM)

CLASS (**CRS_CODE**, **CLASS_SECTION**, CLASS_TIME, ROOM_CODE, PROF_NUM)

Composite and Simple Attributes

- 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. Another Example: PHONE_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

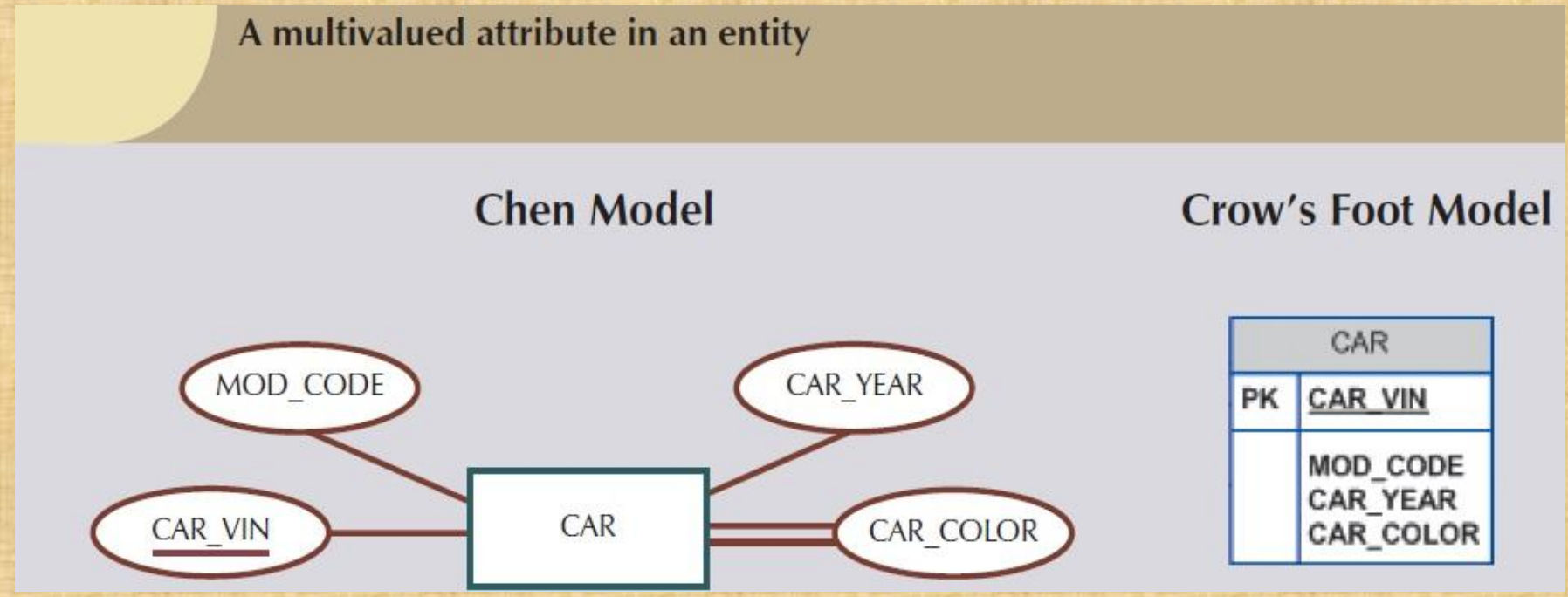
- A single-valued attribute is an attribute that can have only a single value. For example, a person can have only one Social Security number, and a manufactured part can have only one serial number.
- Keep in mind that a single-valued attribute is not necessarily a simple attribute. For instance, a 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 in the next slide, 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.

Multivalued Attributes

A multivalued attribute in an entity



Multivalued Attributes

NOTE

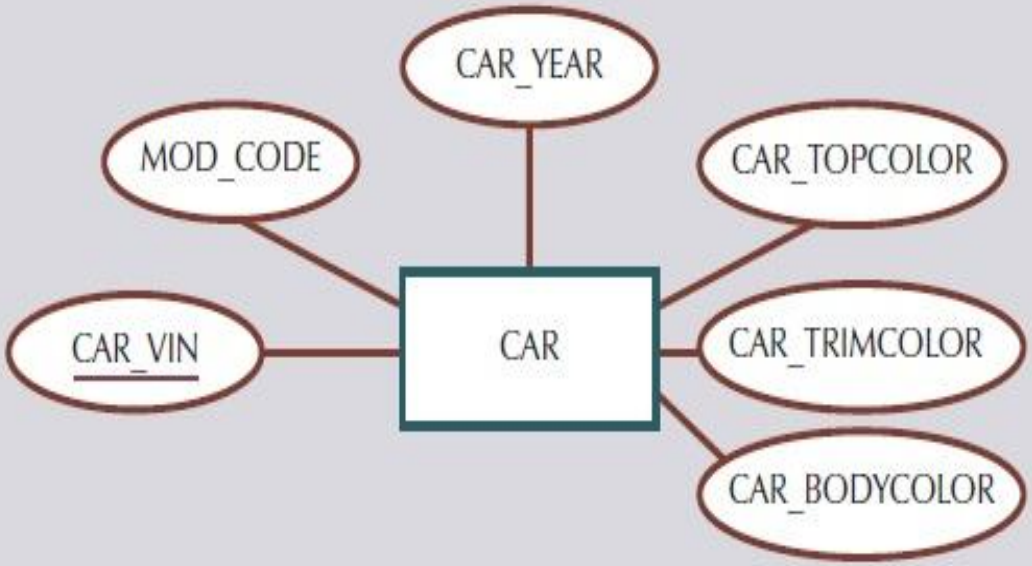
In the ERD models in Figure 4.3, the CAR entity's foreign key (FK) has been typed as MOD_CODE. This attribute was manually added to the entity. Actually, proper use of a database modeling software will automatically produce the FK when the relationship is defined. In addition, the software will label the FK appropriately and write the FKs implementation details in a data dictionary. Therefore, when you use database modeling software like Visio Professional, *never type the FK attribute yourself*; let the software handle that task when the relationship between the entities is defined. (You can see how that's done in **Appendix A, Designing Databases with Visio Professional: A Tutorial**, in the Student Online Companion).

Implementing Multivalued Attributes

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

Implementing Multivalued Attributes

Splitting the multivalued attribute into new attributes



Crow's Foot Model

CAR	
PK	CAR_VIN
	MOD_CODE CAR_YEAR CAR_TOPCOLOR CAR_TRIMCOLOR CAR_BODYCOLOR

Implementing Multivalued Attributes

- 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 that case, cars that do not have such color sections generate nulls for the non-existing components, or their color entries for those sections are entered as N/A to indicate "not applicable." (Imagine how the solution in Figure—splitting a multivalued attribute into new attributes—would cause problems when it is applied to an employee entity containing employee degrees an certifications.
- If some employees have 10 degrees and certifications while most have fewer or none, the number of degree/certification attributes would number 10 and most of those attribute values would be null for most of the employees.) In short, although you have seen solution 1 applied, it is not an acceptable solution.

Implementing Multivalued Attributes

- Solution 2: Create a new entity composed of the original multivalued attribute's components. The new (independent) CAR_COLOR entity is then related to the original CAR entity in a 1:M relationship. Note that such a change allows the designer to define color for different sections of the car. (See Table 4.1.) 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.
- Note that the ERM reflects the components listed in Table 4.1. 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's a more flexible, expandable solution, and it is compatible with the relational model

Implementing Multivalued Attributes

TABLE 4.1 Components of the Multivalued Attribute

SECTION	COLOR
Top	White
Body	Blue
Trim	Gold
Interior	Blue

A new entity set composed of a multivalued attribute's components

Derived Attributes

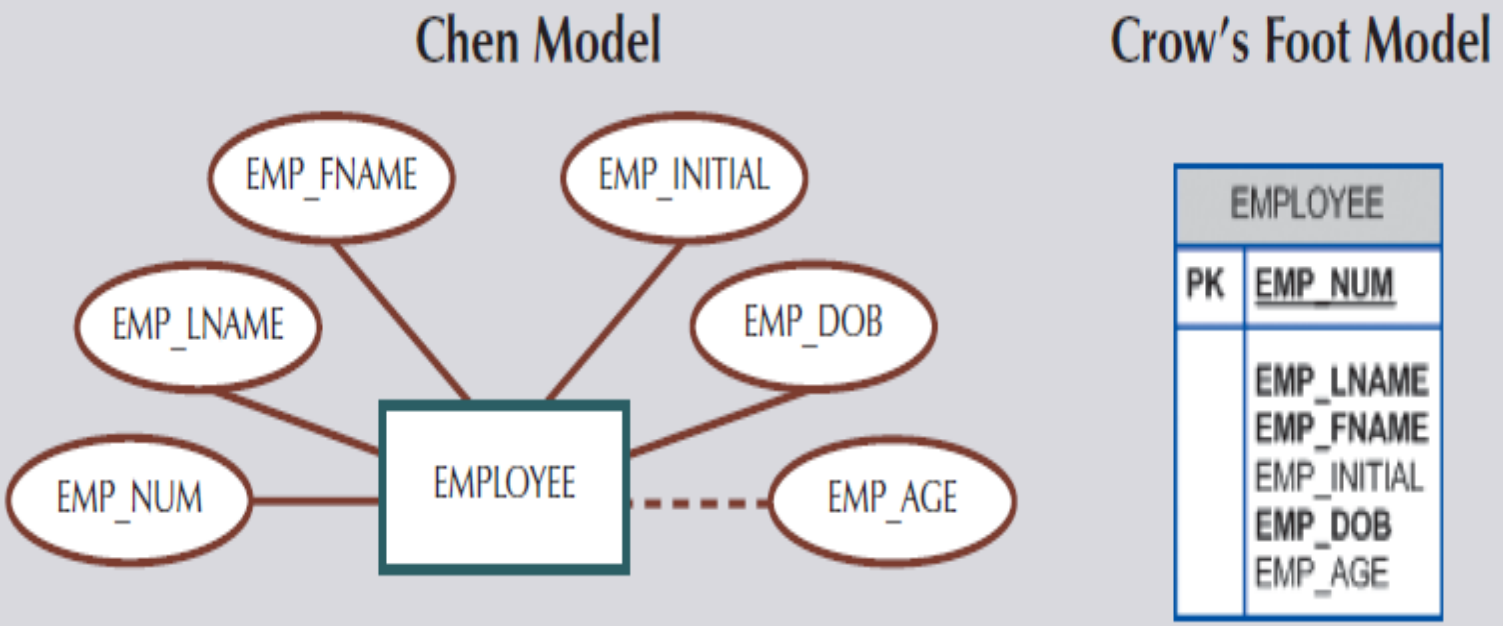
- Derived attribute is an attribute whose value is calculated (derived) from other attributes. The derived attribute need not be physically stored within the database; instead, it can be derived by using an algorithm.
- For example, an employee's age, EMP_AGE, may be found by computing the integer value 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.

Derived Attributes

- If you use Oracle, you would use SYSDATE instead of DATE(). assuming, of course, that the EMP_DOB was stored in the Julian date format.)
- Similarly, the total cost of an order can be derived by multiplying the quantity ordered by the unit price. Or the estimated average speed can be derived by dividing trip distance by the time spent en route. 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

FIGURE 4.6 Depiction of a derived attribute



Derived 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.

Derived Attributes

TABLE 4.2 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

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, a DIVISION is managed by an EMPLOYEE, and an AIRCRAFT is flown by a CREW.
- Relationships between entities always operate in both directions. That is, to define the relationship between the entities named CUSTOMER and INVOICE:
  - A CUSTOMER may generate many INVOICEs.
  - Each INVOICE is generated by one CUSTOMER.
  - Relationship can be classified as 1:M.

Relationships

Another Example:

- The relationship classification is difficult to establish if only one side of the relationship is known. For example, if :
  - A DIVISION is managed by one EMPLOYEE.
- Don’t know if the relationship is 1:1 or 1:M. Therefore, 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 the 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.

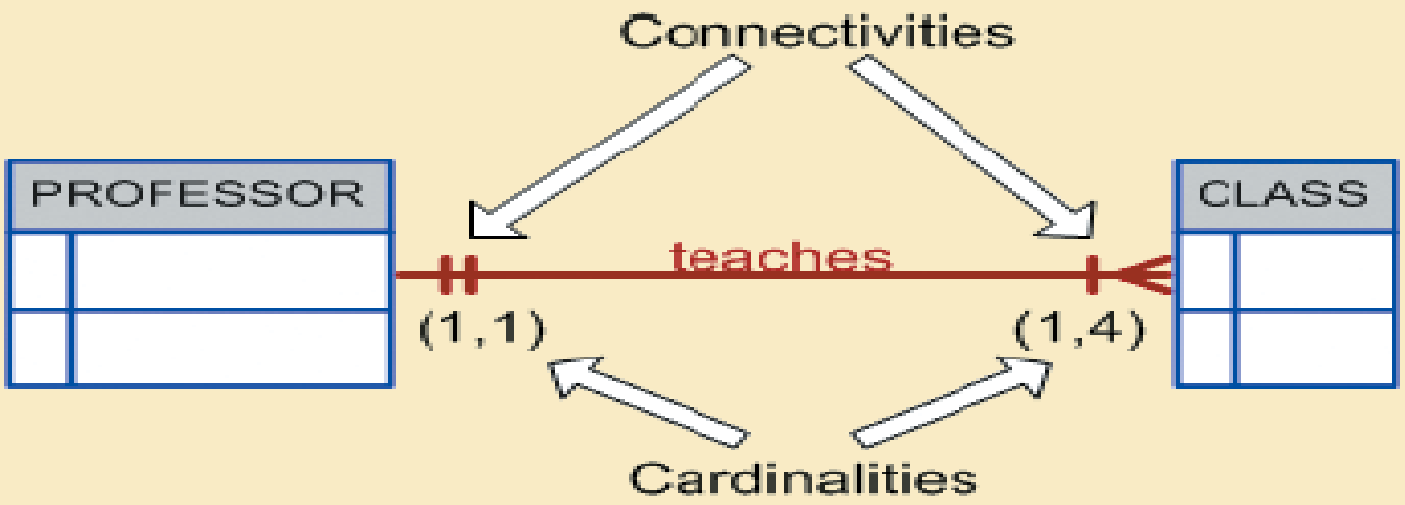
Connectivity and Cardinality

- Entity relationships may be classified as one-to-one, one-to-many, or many-to-many.
- 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. Some Crow's Foot ER modeling tools do not print the numeric cardinality range in the diagram; instead, you could add it as text. In Crow's Foot notation, cardinality is implied by the use of symbols in Figure in the next slide.

Connectivity and Cardinality

- The numeric cardinality range has been added using the Visio text drawing tool.

FIGURE 4.7 Connectivity and cardinality in an ERD



Connectivity and Cardinality

- 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. However, keep in mind that the DBMS cannot handle the implementation of the cardinalities at the table level—that capability is provided by the application software or by triggers.

Connectivity and Cardinality

- Examine the Crow's Foot diagram in Figure 4.7, keep in mind that the 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 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

Connectivity and Cardinality

- The numeric cardinality range has been added using the Visio text drawing tool.

FIGURE 4.7 Connectivity and cardinality in an ERD