College of computer technology
Information network department
Introduction to database
lecture 9

==========

THE ENTITY RELATIONSHIP MODEL(ERM)ENTITIES

The entity is an object of interest to the end user. entity correspond to the table – not to a row- in the relational environment.

ATTRIBUTES

Attributes are characteristics of entities .The STUDENT entity includes, among many others, the attributes NAME, FNAME, and INITIAL.
ER can be represented by:

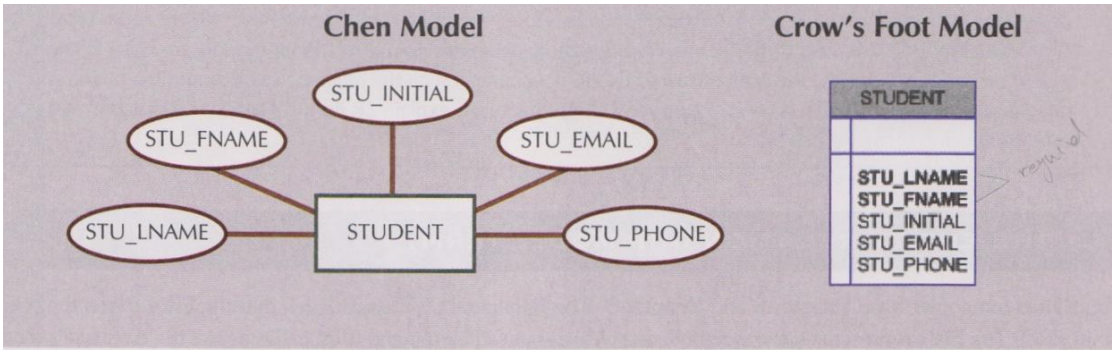
- The Chen notation favors conceptual modeling.
- The Crow's Foot notation favors a more implementation-oriented approach.
- The UML notation can be used for both conceptual and implementation modeling.

REQUIRED ATTRIBUTES

Is an attribute that must have a value; in other words it cannot be left empty. This attributes represented by a boldface in the Crow's Foot notation.

OPTIONAL ATTRIBUTES

Is an attribute that does not require a value; therefore , it can be left empty and those attributes are not presented in boldface in the entity in the Crow's Foot notation as following figure.



DOMAINS

Domain is the set of possible values for a given attribute For example, the domain for grade point average 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 instance, a student address share the same domain of all possible addresses.

College of computer technology
Information network department
Introduction to database
lecture 9

==========

Identifiers (Primary Keys)

Identifies, that is, one or more attributes that uniquely identify each entity instance. Such identifiers are mapped to primary keys (PKs) in tables. Identifiers are underlined in the ERD.
For example, a CAR entity may be represented by:
CAR(**CAR-VIN**,MOD_CODE,CAR_YEAR,CAR_COLOR)

Composite Identifiers

That is, a primary key composed of more than one attribute, for instance.
CLASS(**CRS-CODE**,**CLASS-SECTION**,CLASS_TIME,ROOM_CODE, PROF_NUM).

Composite Attributes

Is an attribute that can be further subdivided to yield additional attributes. for example the attribute ADDRESS can be subdivided into street, city, state.

Simple Attribute

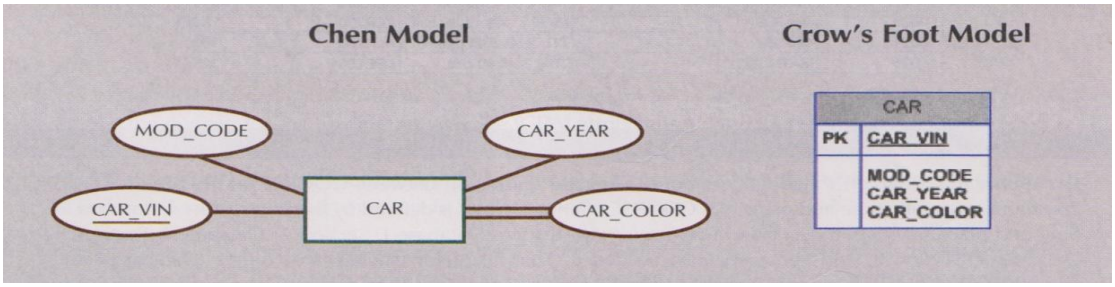
Is an attribute that cannot be subdivided. For example, age, sex can be classified as simple attributes.

Single-valued Attributes

Is an attribute that can have only a single value . For example a person can have only one Social Security 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 a single-valued but it is a composite attribute because it can be subdivided into the region .

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. a car's color may be subdivided into many colors the following figure show the ERD for car's color.

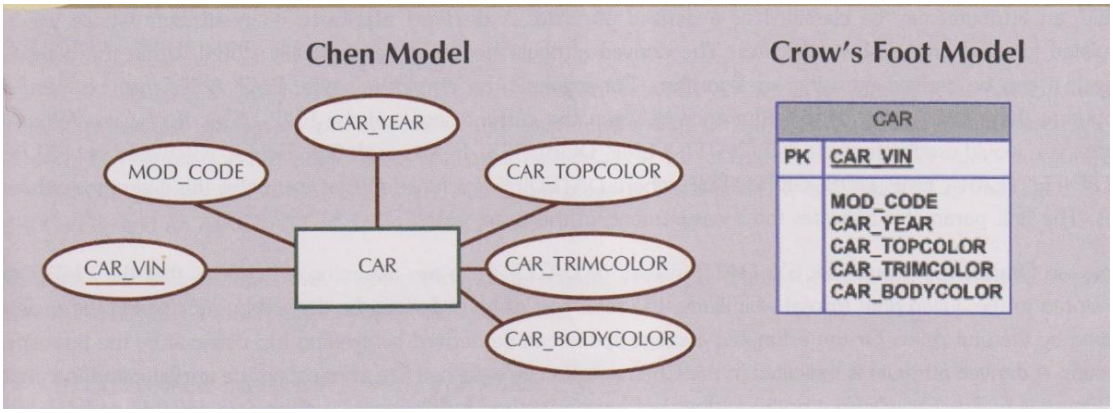


College of computer technology
Information network department
Introduction to database
lecture 9
==========

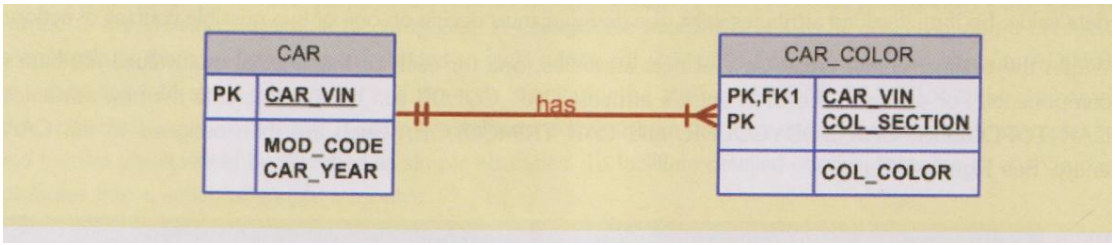
Implementing Multivalued Attributes

If mulivalued 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. as figure below. 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 logo color- are added for some cars, the table structure must be modified to accommodate the new color section.



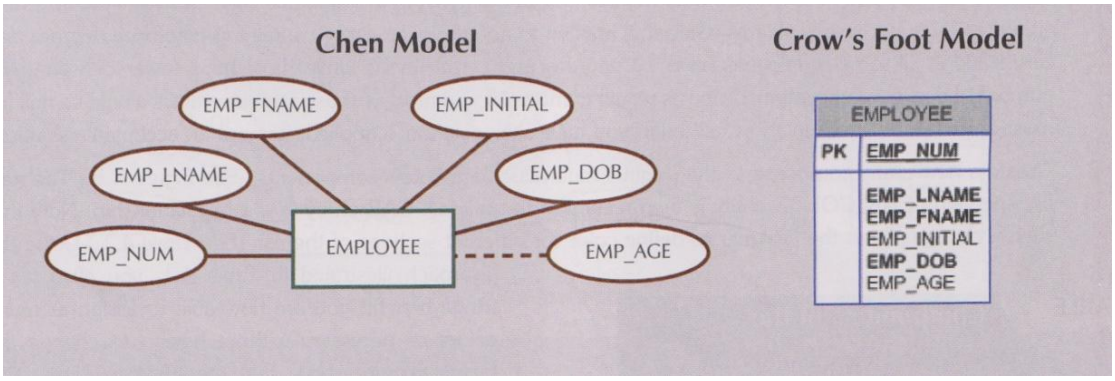
2. Create a new entity composed of the original multivalued attribute's component as in figure bellow .The new (independent)CAR_COLOR entity is then related to the original CAR entity in a 1:M relationship. This way yields several benefits: it's a more flexible, expandable solution, and it is compatible with the relational model.



Derived Attributes

An attribute may be classified as **derived attribute** is an attribute whose value is calculated(derived) from other attributes. The derived attribute need be physically stored within the database ;instead, it can be derived by using 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. A derived attribute is

indicated in the Chen notation by dashed line connecting the attribute and the entity, as follows figure



Derived attributes are sometimes referred to as **computed attributes**
The following table shows the advantages and disadvantages of storing(or not storing) derived attributes in the database.

	stored	**Not stored**
advantage	Saves CPU processing cycles Save data access time Data value is readily available Can be used to keep track of historical data	Saves strong 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

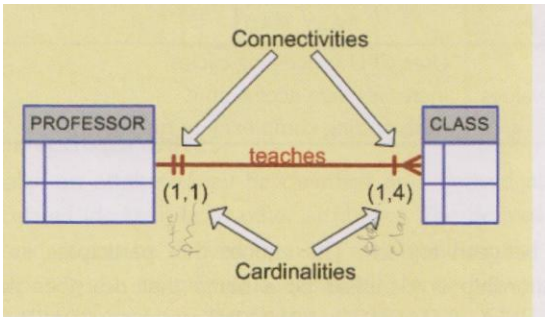
A relationship is an association between entities. The relationship name is an active or passive verb; for example, STUDENT takes a CLASS,PROFESSOR teaches a CLASS.
Relationships between entities always operate in both directions.

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 as in following figure

College of computer technology
Information network department
Introduction to database
lecture 9

==========



EXISTANCE 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 a part from one or more related entities, it is said to be **existence_ independent** .(sometimes designers refer to such an entity as a strong or regular entity) for example, suppose that the XYZ corporation uses parts to produce its products. farther, 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