Data Modeling Using Oracle (Barker Notations)

cetinerg@itu.edu.tr Assoc.Prof.Dr.B.Gültekin Çetiner

CONTENTS

DATA MODELING

- ***Entity Definition and Relations***
- ***Defining Attributes and Unique Identifiers***
- ***Normalizing Data Model***
- ***Understanding Advanced Relations***
- ***Transform Data Model to Database (Designing Database from Data Model)***

- *Defining Entities*
- *Understanding Relations*

•*Defining Entities*

- Objectives
  - Introduction of *Entities.* At the end of topic, you will be able to
    - Identify Entities
    - Assign a name for each entity
    - Give examples for each entity
    - Document entities and start entity-relation diagram.

- **Starting point in developing a good data model is a good set of data requirements.**

These requirements might be in forms of

**** interview notes,***
**** verbal conversations with the user and,***
**** formal requirements specification document (RSD).***

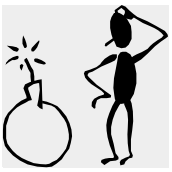
- These requirements may include information about just data, or about the data and the business functions * which use this data.

* Functions required to be implemented in a business without regarding the knowledge of how to do it.

Defining Entities

- **First step is to digest information and then to find the entities* about which the business needs to store data.**

*** Entity: A thing of significance, whether concrete or abstract, about which information needs to be known or held.**

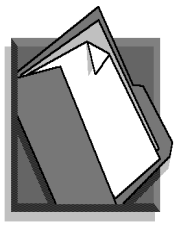


Defining Entities

- **What is an entity? Some definitions are as follows:**

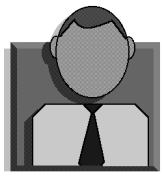
Defining Entities

- **An entity is a thing of significance about which information needs to be stored. In other words, an entity is something important enough to your organization so the organization is willing to spend money to keep records about it: CUSTOMERs, EMPLOYEEs etc.**



Defining Entities

- **An entity is a class or category of thing. A single employee is not an entity, but the general category EMPLOYEE is an entity.**



Defining Entities

- **An entity is a named thing. It may be a tangible thing, such as a TRUCK, or a concept, such as a COST CENTER.**

Defining Entities

From the following sentence, try to pick out the things which we would call "entities".

"Every time a customer places an order, we fill out a form. "

A. PLACE and FILL OUT
B. CUSTOMER, ORDER and FORM
C. CUSTOMER, PLACE, ORDER, FILL OUT and FORM
D. none

Naming Entities

Entities are "named things". As soon as you think you have identified an entity, you should try to name it.

An entity's name should be:

- **Unique in the entire organization; furthermore, try to avoid similar names for different entities, such as PROJECT and PROJECTION.**

Choose a non confusing name for both.

PROJECT

PROJECTION

An entity's name should be:

- **Unique in the entire organization; furthermore, try to avoid similar names for different entities, such as PROJECT and PROJECTION.**
- **Familiar to all; everyone should know instantly what an EMPLOYEE or ORDER is; but they might not know that an "EMPLOYEE DEPARTMENTAL MUTUALLY DETERMINED EFFORT" is actually a PROJECT.**

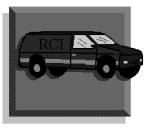
If different terms are used to define the same thing under different departments synonyms can be defined. For example, some departments may call it a CUSTOMER while others may call a CLIENT.

Finally one has to be chosen in database but when speaking to different people we must choose their terms.

Entity Examples

We have already seen some examples of entities,
Some more examples are as follows:

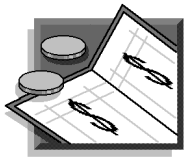
TRUCK



Entity Examples

We have already seen some examples of entities,
Some more examples are as follows:

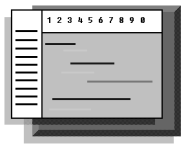
BANK ACCOUNT



Entity Examples

We have already seen some examples of entities,
Some more examples are as follows:

PROJECT



Entity Examples

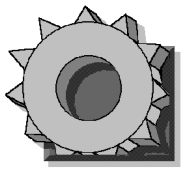
We have already seen some examples of entities,
Some more examples are as follows:

CONTRACTOR

Entity Examples

We have already seen some examples of entities,
Some more examples are as follows:

PART



Entity Examples

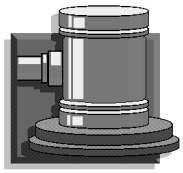
We have already seen some examples of entities,
Some more examples are as follows:

INSURANCE POLICY

Entity Examples

We have already seen some examples of entities,
Some more examples are as follows:

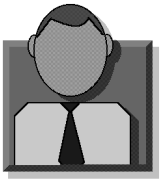
DEFENDANT



Entity Examples

We have already seen some examples of entities,
Some more examples are as follows:

CUSTOMER



Defining Entities

A set of conventions was used to name an entity. Which statement best reflects our conventions?

A. The entity name is singular, upper case
B. The entity name is mixed upper and lower case
C. The entity name is always a single word
D. The entity name is always one or more verbs

NOTE: These conventions may vary. You can define your own standards.

Defining Entities

Instances of entities

If you are not certain whether something is an entity or not, it may help to ask yourself the question,

"What are the instances (examples) of this entity?"

Defining Entities

Suppose you asked this question to a company with the case of CUSTOMER.

Can you give examples to the entity CUSTOMER?

They will immediately give some names of their customers.

So the entity CUSTOMER is OK.

Defining Entities

On the other hand, if you have defined an entity called CUSTOMER LIST,

What are some of the specific occurrences of customer lists? The alphabetical by name list, the geographical list, the top five list, and the inactive lists are a few of the possible occurrences.

Defining Entities

In this case, try to find the facts, or attributes* about each CUSTOMER LIST.

The paper type CUSTOMER LIST is printed on,
Weight or quality of printing paper,
Printing Date,
The person who printed it, and so on.

Are these facts or attributes important to the business?

*** Any detail that serves to qualify, identify, classify, quantify, or express the state of an entity.**

Defining Entities

Since we need to store data about CUSTOMER itself, the best name for the entity is CUSTOMER rather than CUSTOMER LIST.

Identifying instances helps to find the proper entity names.

Validate Entities

To make sure your list includes real entities,

Here are some other ways:

Is its name a noun? Not all nouns represent entities, but all entity names should be nouns.

Validate Entities

To make sure your list includes real entities,

Here are some other ways:

Does this entity have any attributes?

For example, if you think you have an entity named STUDENT, do you need to store facts about students?

Name, Student Number, Address, Birth Date ....

YES, IT IS AN ENTITY.

Validate Entities

To make sure your list includes real entities,

Here are some other ways:

If you think you have an entity called CUSTOMER NAME, ask the same question: Does it have any facts, attributes to store? The country of origin, number of wovels, meaning of the name ....

Are these really important facts to store in your database?

NOT! Then CUSTOMER NAME is not an entity.

Question - Which nouns in the following sentence represent entities?

"When I get an order, I look first to see if the name and address are filled in."

A. ORDER, NAME and ADDRESS
B. I, LOOK and IN
C. WHEN, GET and SEE
D. ORDER

Diagramming Entities

Entities are drawn in diagram as soon as they are identified.

Defining Entities

An E R Diagram (Entity Relationship Diagram) easily conveys your findings to other people:

Database Administrators, Programmers and non-technical business people.

Entities are drawn as boxes in a diagram. In the following lessons, relations and other elements will be added to diagrams.

Defining Entities

For now, we just draw boxes to represent entities. You can choose some conventions for drawing entities on E R Diagrams.

*** Entity names are singular, upper case, brief, unique nouns.**

For example;

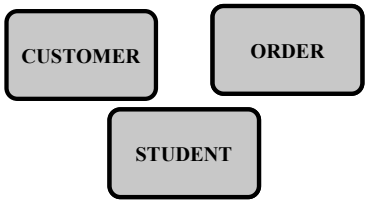
CUSTOMER
STUDENT
ORDER
RESERVATION
DOCUMENT etc.

Defining Entities

For now, we just draw boxes to represent entities. You can choose some conventions for drawing entities on E R Diagrams.

*** Boxes are soft, i.e. with rounded corners.**

For example;



Defining Entities

Topic Summary

* An entity is a thing of significance about which information needs to be stored; a class or category of thing; a named thing.

* When considering whether something is an entity or not, it helps to ask yourself the question, "What would some instances of this entity be?"

* An entity's name should be unique in the entire organization and familiar to all.

* Once you have identified an entity and named it, you should diagram it; the diagram pictorially represents entities, the vital business relationships between them and the attributes to describe them.

* An attribute is any detail that serves to qualify, identify, classify, quantify, or express the state of an entity.

Understanding Relations

Beginning
Identify Relationships
Relationship Sentences
Relationship Names
Optionality in Relationships
Relationship Degree
Diagramming Relationships
Relationship Types
Validating Relationships

Understanding Relations

Topic Objectives

This topic introduces relationships. At the end of this section, you will be able to:

*** Identify the important relationships between entities**

*** Name each relationship**

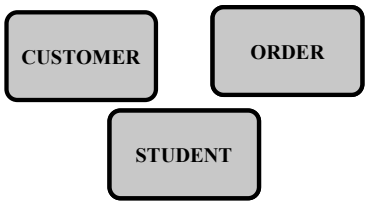
*** Explain the difference between optional and mandatory relationships**

*** Specify the degree, or cardinality, of each relationship: one or more, one and only one**

*** Develop Relationship Sentences to describe each relationship in an Entity Relationship Diagram.**

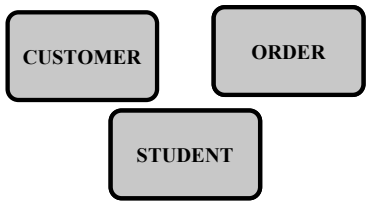
Understanding Relations

So far Entity-Relationship Diagram is not very informative. It consists of some boxes, but no connections between them, and no text to explain what the boxes mean.



Understanding Relations

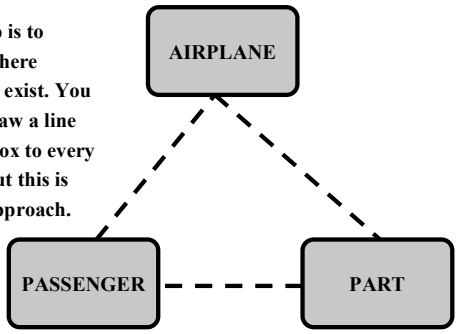
In this section, we will discuss how to define the relationships between entities, and how you make the E-R Diagram meaningful to people other than yourself.



Understanding Relations

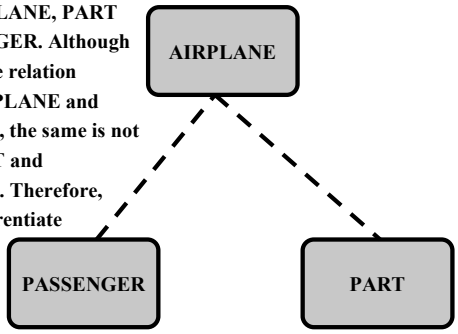
Identify Relationships

The first step is to Determine where relationships exist. You could just draw a line from every box to every other box. But this is not a good approach.



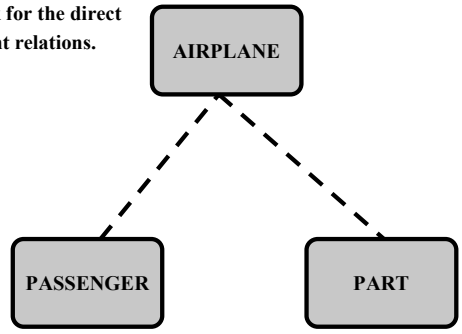
Understanding Relations

For example, consider entities AIRPLANE, PART and PASSENGER. Although there is a close relation between AIRPLANE and PASSENGER, the same is not true for PART and PASSENGER. Therefore, we must differentiate between these relations.



Understanding Relations

We must look for the direct and significant relations.



Understanding Relations

Relationship Sentences

Even though the E-R Diagram consists of boxes and lines just a few words, you as the analyst should be able to "read" it to either a technical Database Administrator or a non-technical business person.

E-R Diagrams are essential for communicating data requirements in business.

Understanding Relations

The key to make your ER Diagram "readable" is the Relationship Sentence. It is a complete sentence and can be constructed in any language.

Understanding Relations

Suppose you have two entities: CUSTOMER and ORDER. We can describe how customers and orders are related with two sentences:

"Each CUSTOMER may be the originator of one or more ORDERs."

"Each ORDER must be placed by one and only one CUSTOMER."

Understanding Relations

Let's divide one of these sentences.

"Each ORDER must be placed by one and only one CUSTOMER."

The formal syntax of the sentence is:

Each ENTITY1	{must be} or {may be}	name	{one or more} or {one and only one}	ENTITY2

So, ORDER and CUSTOMER are the entities, and "placed by" is the name of the relationship.

Understanding Relations

The statement "*must be*" or "*may be*" describes whether the relationship is mandatory or optional.

The statement "*one or more*" or "*one and only one*" describes the cardinality (degree) of relationship.

Each ENTITY1	{must be} or {may be}	name	{one or more} or {one and only one}	ENTITY2

We will look at each of these parts of the sentence in detail.

Understanding Relations

Relationship Names

Entity names are always nouns.

Relationship names may be derived from verbs, for example, "placed by."

Relationship names also represent a role, for example "the originator of."

Understanding Relations

Always try to find some phrase that ordinary business people would use to describe the relationship that binds two entities together.

We should use business oriented names so that we can explain the diagram to non-technical business people easily*.

*** In fact, the owner of the data is these people, i.e. USERS.**

We can literally read a relationship sentence to the users and ask them, "Is that the way it actually works?" They can help validate our data model during the early analysis phase, rather than waiting until the system is implemented and then saying, "Oh, this is not how it actually works."

Let's try a couple of quick questions to make sure you understand the concept of relationship sentences.

Fill in the blank

"Each EMPLOYEE must be assigned to ____ DEPARTMENT "

A. one and only one
B. one or more
C. at least one
D. exactly one

Fill in the blank

"Each PROJECT may be ____ one or more DEPARTMENTs"

A. sponsored by
B. staffed by
C. managed by
D. any of the above

Relationship Optionality

Naming the relationship is not enough. We must also describe whether a relationship must exist, or may exist.

In the Relationship Sentence, you have a choice of "must be" or "may be" to describe a relationship's optionality.

If you say that entity1 "must be" related to entity2 in a certain way, you are saying that, in that direction, the relationship is mandatory.

If you say "may be", then the relationship is optional.

Let's look at the following sentence:

"Each ORDER must be made by one and only one CUSTOMER"

The choice of term "must be" indicates that an order cannot exist without a customer to place it. This relationship is mandatory.

If there is ORDER then there must be CUSTOMER who made.

Understanding Relations

Let's look at the other half of the relationship sentence.

"Each CUSTOMER may be the originator of one or more ORDERs"

The choice of the term "may be" indicates that a customer may exist in our database without ever placing an order.

Think of a CUSTOMER who has been sent a CATALOG but never bought anything yet.

Understanding Relations

Is the relationship mandatory or optional?

"Each EMPLOYEE ____ assigned to one and only one DEPARTMENT"

A. must be
B. may be

Understanding Relations

Is the relationship mandatory or optional?
Choose "must be" or "may be"

"Each PROJECT _____ carried out by one or more EMPLOYEEs"

A. must be
B. may be

Understanding Relations

Is the relationship mandatory or optional?
Choose "must be" or "may be"

"Each RESERVATION _____ made by one and only one EMPLOYEE"

A. must be
B. may be.

Understanding Relations

Relationship Degree

Now let's look at the other end of the Relationship Sentences;

"Each ORDER must be made by <u>one and only one</u> CUSTOMER."

"Each CUSTOMER may be the originator of <u>one or more</u> ORDERs."

Understanding Relations

The degree of relationship is stated as either "<u>*one and only one*</u>" or "<u>*one or more*</u>". The "one and only one" is known as a "single valued" relationship.

"Each ORDER must be made by <u>one and only one</u> CUSTOMER."

"One or more" means "one, or any number". "One or more" usually used in optional relations. "One or more" allows the CUSTOMER to place one ORDER, a hundred or a thousand. The CUSTOMER may also be in database with zero orders.

This relation is called "many-valued relationship".

"Each CUSTOMER may be the originator of one or more ORDERs."

Choose the relationship degree using one of the options.

"Each ORDER must be made up of _____ ORDER LINE ITEMs"

A. "one and only one"
B. "between one and twelve"
C. "one or more"
D. "any number except zero"

Choose the relationship degree using one of the options.

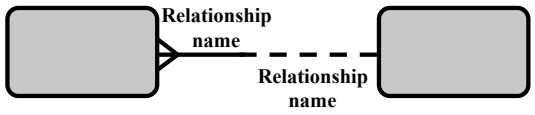
"Each ORDER LINE ITEM must be contained in _____ ORDER"

A. "one and only one"
B. "one or more"
C. "zero, or one or more"
D. "none of the above"

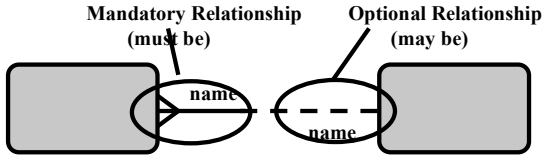
Diagramming Relationships

Now it is time to represent all these in an E-R Diagram.

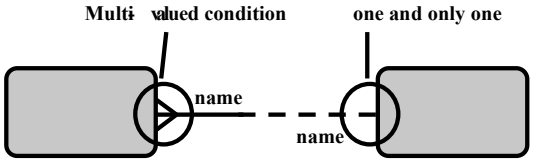
This diagram indicates how relationship names, optionality and degree are indicated.



Each half of the line is either dashed (- - -) to indicate an optional relationship, or solid (———) to indicate a mandatory one.



The "crow's foot" indicates "one or more" or "multi-valued" condition. If there is not a crow's foot then the relationship degree is "one and only one".

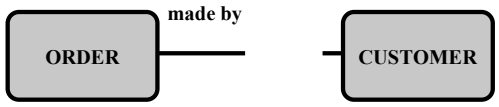


Understanding Relations

"Each ORDER must be made by one and only one CUSTOMER"

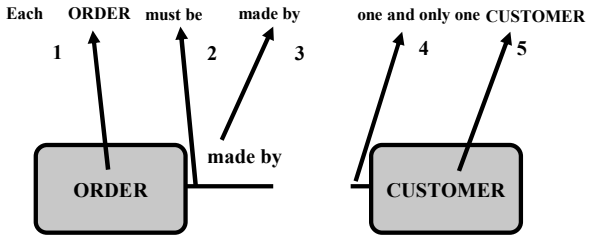
sentence is mandatory and single valued. Therefore, we draw a solid line from ORDER to CUSTOMER and write the relation name as *made by.*

Absence of crow foot shows single valued relation.



Understanding Relations

Read the sentence in the following order;

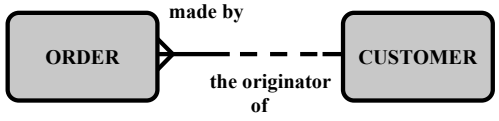


Understanding Relations

Other sentence,

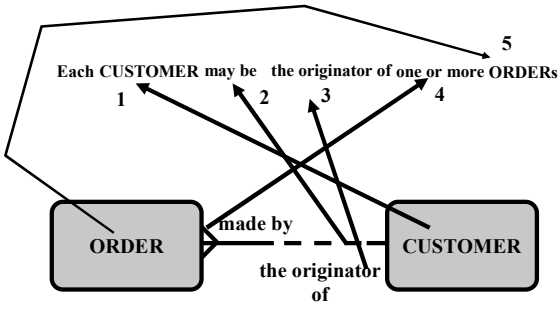
"Each CUSTOMER may be the originator of one or more ORDERs"

Optional and multi valued. Therefore, a dashed line is drawn from CUSTOMER to ORDER and the relation sentence '*originator of*' is written below this line. A crow foot is drawn next to ORDER entity to show the multi valued relation.



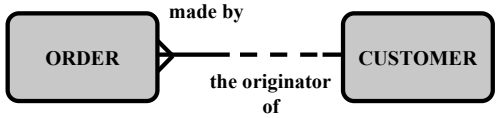
Understanding Relations

You can read this part in the following order;



Understanding Relations

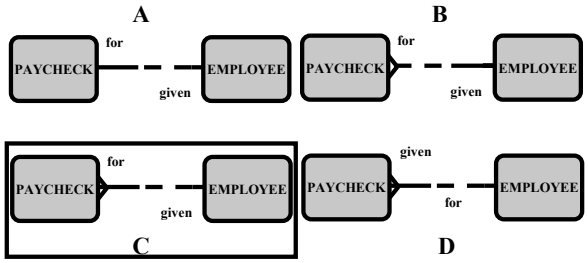
We construct 2 sentences for each pair of entities. The following type of relation is called one to many (or many to one).



Understanding Relations

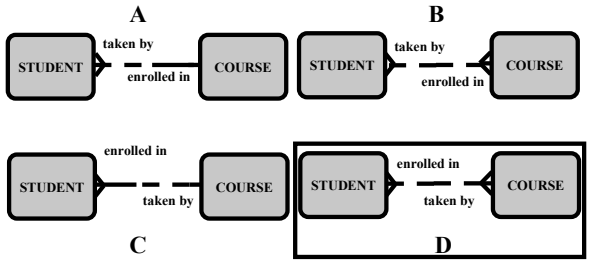
Find the correct E-R Diagram for the following sentences.

"Each PAYCHECK must be for one and only one EMPLOYEE."
"Each EMPLOYEE may be given one or more PAYCHECKs."



Show the diagram for STUDENT and COURSE entities according to following sentences.

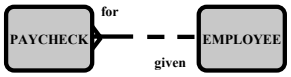
"Each STUDENT may be enrolled to one or more COURSEs."
"Each COURSE may be taken by one or more STUDENTs."



Types of relations

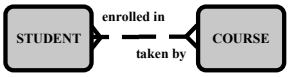
There are 3 types of relations.

The relation between PAYCHECK and EMPLOYEE is a one-to-many relation. It is also called many-to-one relation.

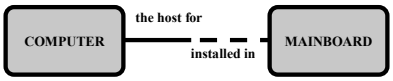


When finished ER Diagram usually consists of this type of relations.

The relation between STUDENT and COURSE is many-valued in both sides. This kind of relation is called many-to-many relation.



3rd type of relation is single-valued in both sides. For example, COMPUTER and MAINBOARD entities might have this type of relation.



"Each COMPUTER must be the host for one-and-only one MAINBOARD."

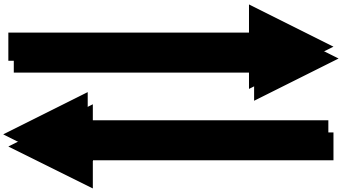
"Each MOTHERBOARD may be installed in one and only one COMPUTER."

This relationship is a fairly unusual relationship. They are not seen during analysis and design stages. But may appear in a finished database.

Understanding Relations

Validating the relations

To fully validate a relationships, you should read it first one way, And then the other.



Understanding Relations

Regardless of the order in which you read the relationships, your E-R Diagram should plainly and accurately reflect the business rules.

These business rules are gathered during the early analysis phases of the project.

You should be able to read it to technical people such as database administrator (DBA). He can tell you whether it sounds reasonable.

Understanding Relations

More importantly you should be able to read it to a business person or user. They don't need to know data modeling.

They can tell you whether the relations are reflecting the business rules accurately.

Do not forget: They are the owners of the data.

Understanding Relations

Do read the diagram aloud for yourself;

Compare the diagrams with the shapes of your previous diagrams.

Understanding Relations

Topic summary

Once you have determined the entities about which you need to store your data, determine the relationships by following these steps.

- **determine the existence of relationships**
- **assign a name to each direction of each relationship**
- **determine the optionality of each direction of each relationship**
- **determine the degree of each direction of each relationship**
- **diagram the relationships**
- **validate the E-R Diagram by reading it aloud to various people**

Identifying Attributes

Assigning Unique Identifiers

Diagram Attributes
Identify Attributes
Name Attributes
Elementary Attributes
Derived Data
Attribute Optionality
Distinguish Attributes and Entities

Diagram Attributes

An attribute is a fact about an entity, for example:

***name* and *address* are facts about EMPLOYEE**

EMPLOYEE
name
address

***altitude* and *mean January temperature* are facts about CITY**

CITY
altitude
mean January temperature

***name* and *first enrollment date* are facts about STUDENT**

STUDENT
name
first enrollment date

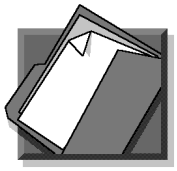
Identify Attributes

If you do not have a thorough set of business specifications,

Where do you find out about the data items, facts, or ATTRIBUTES?

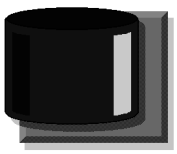
Some possible sources are:

Headings from existing printed reports



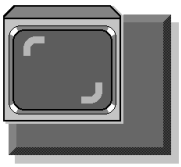
Some possible sources are:

Headings from existing printed reports
Fields stored in existing files and databases



Some possible sources are:

Headings from existing printed reports
Fields stored in existing files and databases
Captions from screens



Identify Attributes

Some possible sources are:

Headings from existing printed reports
Fields stored in existing files and databases
Captions from screens
Nouns that business people use in everyday conversations

Identify Attributes

Some possible sources are:

Headings from existing printed reports
Fields stored in existing files and databases
Captions from screens
Nouns that business people use in everyday conversations
Values used for sorting reports

Identify Attributes

Some possible sources are:

Headings from existing printed reports
Fields stored in existing files and databases
Captions from screens
Nouns that business people use in everyday conversations
Values used for sorting reports

When collecting requirements from existing procedures or software, BEWARE OF DERIVED DATA.

Derived data are not attributes

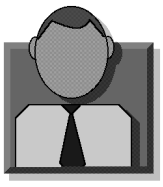
Identify Attributes

Name Attributes

Names of attributes should be the names that people in the business use every day when they discuss this item of information.

Identify Attributes

For example, if you think of an attribute called *Record number* for an entity called EMPLOYEE, but the people in business call it *Employee ID* then you should use *Employee ID* for the attribute name.



EMPLOYEE
Employee ID

Identify Attributes

Name Attributes: Uniqueness

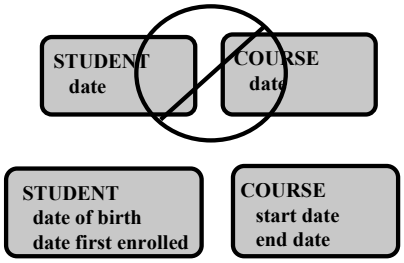
Attribute names must, at least, be unique within a single entity.

If possible, they should be unique across the entire E-R Diagram. That is hard to do, but it is a good goal to aim for.

Identify Attributes

Name Attributes: Uniqueness

Look at the following example.



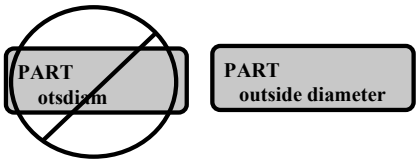
Identify Attributes

Name Attributes: Clarity

Attribute names should not be encoded. Spell them out, generally without abbreviations. Remember that the owner of data is business community.

Identify Attributes

For example, *outside diameter* is clearer than *otsdiam.*



If in doubt, ask the business people.

Identify Attributes

Most data processing installations have developed over the years fairly detailed naming conventions for data items, often based on the limitations of a specific data dictionary, database management system, or programming language.

Resist the temptation to use those cryptic names in your E-R Diagram.

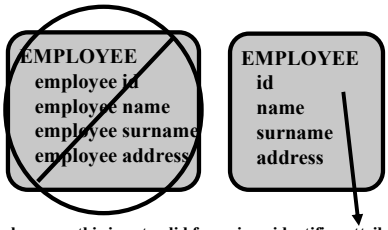
Do not forget: Business people are not FORTRAN COMPILERS.

Identify Attributes

Name Attributes: Qualification

Attribute names in an E-R Diagram should not include the entity name.

For example, in the EMPLOYEE entity, there is no reason to call each attribute "employee id", "employee name", "employee surname", "employee address", and so forth.

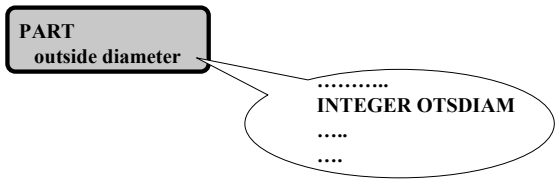


Note however this is not valid for unique identifier attribute.

Identify Attributes

Do not forget: Business people are not FORTRAN COMPILERS.

Later, when you define the database, or write the program code, you may want to introduce the entity name to qualify the attribute name.



Identify Attributes

If a CUSTOMER places an order, some of the attributes of ORDER might be:

A. date of order, and date filled

B. date1, and date2

C. date, and date

D. order, and filled

Identify Attributes

What are the possible attributes for a PART entity?

A. part name, and part description

B. name, and desc

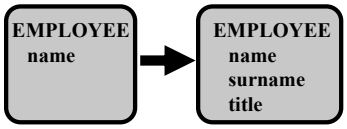
C. name, and description

D. Name, and Desc

Identify Attributes

Elementary Attributes

One goal of data modeling is to break down complex data items into their simplest components.

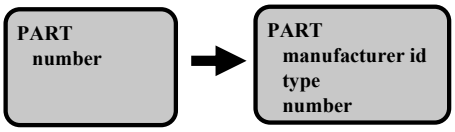


For example, you can break a person's whole name down into individual names and titles.

That way, you give the application developers the flexibility of using separate names (for example, "Dear Dr. Moore", "Moore, Dr. James Thomas")

Identify Attributes

This may be expressed as "one fact, one data item." In some businesses, part numbers may actually consist of multiple facts: manufacturer, type, and a unique number, for example.

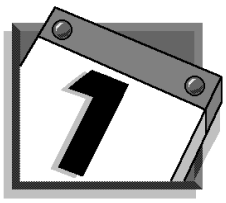


In this case, define three attributes, "manufacturer," "type," and "number."

Identify Attributes

The extent to which you "decompose" data items into their most elementary forms will depend on the business requirements.

For example, it could be argued that a date actually consists of three more elementary data items: "year," "month," and "day."



Identify Attributes

If your business is intensely interested in the month of a transaction, for seasonal reporting and forecasting, then you may want to break all dates down into three elements.

But conventionally, dates, times, postal codes, tax-id numbers, social security numbers and the like are not broken down. They are defined as one attribute of entity.

Identify Attributes

Derived Data

Try to avoid defining an attribute which is derived from one or more others.

For example, instead of storing the "age" of an employee, store the "date of birth." The date of birth probably never changes, and the age can always be derived from it. But the age changes every year, and if not properly changed, age could convey inaccurate information.

Identify Attributes

Also do not store numbers that can be calculated from the other data items. These include year-to-date totals, count of things, averages, etc. Calculated data is redundant, and leads to possible inconsistencies.

Identify Attributes

Some people argue that by storing totals and other summary data items, you can save a lot of computing time.

But that is a physical database design or implementation issue, not a data modeling issue. At this stage of the data model, there should be no derived or calculated attributes.

Identify Attributes

Attribute Optionality

Once each attribute is assigned to the correct entity, you can begin to decide whether it is optional or mandatory.

An optional attribute permits the value to be missing in any instance of entity. An optional attribute is not applicable for some instances of entity. For example, the "spouse name" of an EMPLOYEE would be missing for an unmarried employee.

Identify Attributes

A mandatory attribute cannot be missing: for example "number" of an ORDER or the "name" of an EMPLOYEE.

If you are familiar with relational databases, you may know this as the question of "null values". Technically, a null value in a database is the implementation of an optional attribute.

Identify Attributes

Oracle convention for indicating optional or required attributes is:

- **an asterisk (*) indicates a required attribute**
- **a circle (letter o) indicates an optional attribute**

Identify Attributes

For example;

EMPLOYEE
*** id**
*** name**
*** last name**
o title
*** sex**
o weight

The "asterisks" indicate that an EMPLOYEE must include values for *id, last name, first name* and *sex*.

The "circles" show that there may or may not be a *title* or *weight* known for each EMPLOYEE.

Identify Attributes

For the entity SHIP, the attributes *name, country of origin, registration number, length, width*, and *class* have been identified. *Width* and *class* are often unknown, but the other attributes are required. Which of the following diagrams correctly models SHIP and its attributes?

A

SHIP
o name
o country of origin
o registration number
*** length**
o width
*** class**

B

SHIP
*** name**
*** country of origin**
*** registration number**
*** length**
 # width
 # class

C

SHIP
*** name**
*** country of origin**
*** registration number**
*** length**
o width
o class

D

SHIP
*** name**
*** country of origin**
*** registration number**
o length
o width
o class

Identify Attributes

Distinguishing Attributes and Entities

Sometimes it is not exactly clear whether a piece of information is an attribute of entity, or an entity itself. A noun might be en entity or an attribute, depending on the business requirements.

Identify Attributes

If you are in doubt, ask this question about the thing:

"Do we need to store any facts about this thing?"

If the answer is YES, then it is an ENTITY.

If the answer is NO, then it is an ATTRIBUTE of an entity.

Identify Attributes

For example, in a university application let's say we have a thing called "grade."

What are some instances of grades? A, B, etc.

Are there any facts about As, Bs, etc?

Identify Attributes

If we must store a numeric equivalent for computing grade point average, e.g. A=4, B=3, etc.: or, if we must reduce tuition depending on grade, e.g. A=40%, B=25%, etc.; then "grade" becomes an entity, GRADE.

GRADE
*** grade id**
*** name**
o numerical equivalency
o reduction of percentage on fee

If there are no facts about grades, then they are just attributes of a student\s registration or an entity called ENROLLMENT.

ENROLLMENT
…..
…….
*** grade**
……

Identify Attributes

Topic Summary

Once you collect the possible attributes from your business specifications, or collect them from reports, screens, conversations, and so forth, the steps for putting them into your E-R Diagram are:

- **be sure they are current necessary facts**
- **be sure they are named clearly and uniquely**
- **be sure they are not derived from others**
- **decide whether each is optional or required**
- **include them in the E-R Diagram in their entity's softbox**

Unique Identifiers (UID) and Primary Keys

Identifying UIDs

Diagram UIDs in E-R Diagram

UIDs via Relationships

Composite UIDs

Artificial UIDs

Candidate UIDs

Assigning Unique Identifiers

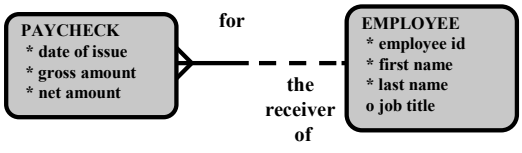
Topic Objectives

This topic discusses Unique Identifiers. At the end of this topic, you will be able to:

- **choose a Unique Identifier (UID) for an entity**
- **identify entities which are uniquely identified at least in part by relationships**
- **explain candidate UIDs, artificial keys, and compound (composite) UIDs**

Assigning Unique Identifiers

In previous lessons, we learned to identify entities, to model the relationships between the entities, and to assign attributes. Now it is time to decide which attribute or attributes uniquely identify each instance of an entity. A unique identifier or UID uniquely identifies each instance of an entity/



Assigning Unique Identifiers

You are probably an instance of TAX PAYER, CUSTOMER, EMPLOYEE, etc., in many databases, and therefore have many UIDs: a *"tax id number," "customer number," "employee number,"* etc. The UID for most entities is obvious, and it will take little or no effort to find in your data model.

For some entities, the UID will require a little more effort to identify. Each entity must have a UID which uniquely identifies each instance of entity.

Assigning Unique Identifiers

UIDs and Primary Keys

If you have worked with almost any kind of files or databases, you are probably familiar with Primary Keys. UIDs and Primary Keys are not exactly the same thing.

When our logical data model is converted into a physical database design, the UIDs will become the Primary Keys of the files, segments, or tables.

Unique Identifier is the term for Data Model
Primary Key is the term for Physical Database

Assigning Unique Identifiers

Identify UIDs

Usually, all you have to do to find a UID is to ask a business person,

"How do you uniquely identify/differentiate a customer/part/course/student/employee/truck?"

Assigning Unique Identifiers

The answer might be a single answer; *"customer number"* or *"part number."*

Also, the answer might include several data items: *"The manufacturer code, the part number, and a random one-digit suffix"* or *"last name, first name, middle initial, and last four digits of the phone number."*

Assigning Unique Identifiers

At this point of data modeling process, there is no limit to the number of attributes which you can use together to uniquely identify entities. In some files and databases, there may be a limit, but do not worry about that at this point.

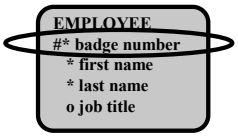
Assigning Unique Identifiers

Diagram UIDs

Now let's consider the EMPLOYEE entity. Each employee has a unique badge number. So the attribute "badge number" should be the UID for the EMPLOYEE entity.

Assigning Unique Identifiers

You indicate this with a # (pound sign) before the attribute name and optionality tag. Each instance of EMPLOYEE is uniquely identified by the attribute *"badge number."*



Assigning Unique Identifiers

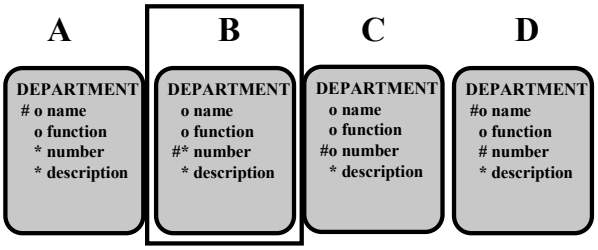
To be UID, an attribute must be required, not optional. Therefore, UID attributes will always have #* before the name. Since this is redundant, you may use the # by itself.

That is,

#* badge number and # badge number mean exactly the same thing.

Assigning Unique Identifiers

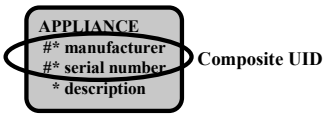
How do you model the UID for the entity DEPARTMENT.



Assigning Unique Identifiers

In another example, it takes two attributes, "manufacturer" and "serial number" to uniquely identify each instance of APPLIANCE. The # before each attribute name does not mean there are two UIDs, since by definition there cannot be.

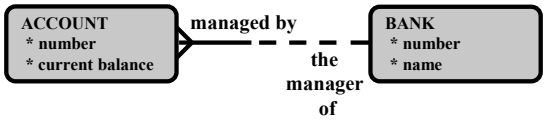
The two # indicate a compound, or icomposite UID.



Assigning Unique Identifiers

UIDs via Relationships

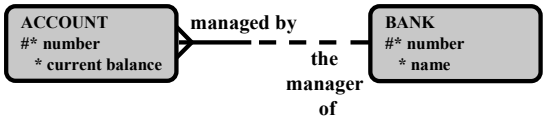
Suppose we have two related entities, BANK and ACCOUNT. Let's determine the UID for each entity. Banks have unique numbers assigned by some government body; let's call this attribute "number" and let it be the UID of BANK.



Assigning Unique Identifiers

So we tag the BANK's attribute *"number"* with a *"#."* Now what is the UID of ACCOUNT? The problem is that accounts do not necessarily have unique numbers across all banks in the world or even in the country. The account number is usually just a number that is unique within a single bank.

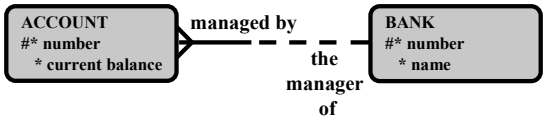
In other words, different BANKs may use the same ACCOUNT numbers. Therefore, "number" in ACCOUNT cannot uniquely identify all instances of ACCOUNT.



Assigning Unique Identifiers

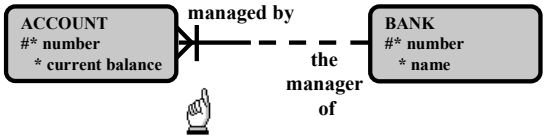
In data modeling terms, an account is uniquely identified by its number, plus its relationship to BANK.

The UID for ACCOUNT is composed of both the attribute "number" and the BANK that manages the ACCOUNT.



Assigning Unique Identifiers

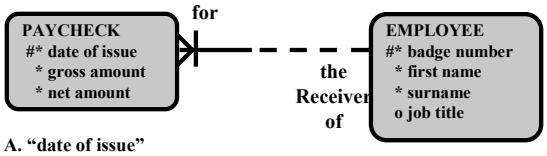
In an E-R Diagram, a relationship which is part of a UID is indicated by a bar across the relationship line, like this:



The bar across the relationship from ACCOUNT to BANK indicates that the relationship is part of the UID of ACCOUNT.

Assigning Unique Identifiers

What is the UID of the entity PAYCHECK?



A. "date of issue"

B. the EMPLOYEE that the PAYCHECK is for

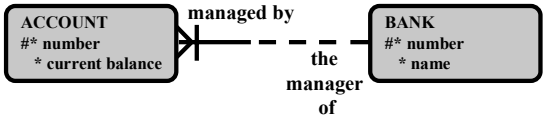
C. "badge number"

D. "date of issue" and the EMPLOYEE that the PAYCHECK is for

Assigning Unique Identifiers

Composite UIDs

In the previous example, a bank account is identified by its own number, plus the number of the bank the ACCOUNT is managed by.



Assigning Unique Identifiers

This illustrates a very common situation: the UID of an entity may consist of any number of attributes and/or relationships. These are called *"compound"* UIDs, or *"composite"* UIDs.

Assigning Unique Identifiers

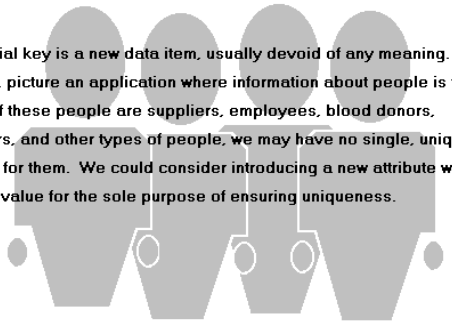
Artificial UIDs

If the UID of an entity consists of so many attributes that it becomes difficult to document the design, then you might want to consider introducing an "artificial key" to serve as the UID.

An artificial key may be helpful if none of the attributes of an entity can be guaranteed to be unique.

Assigning Unique Identifiers

An artificial key is a new data item, usually devoid of any meaning. For example, picture an application where information about people is to be stored. If these people are suppliers, employees, blood donors, customers, and other types of people, we may have no single, unique identifier for them. We could consider introducing a new attribute with an arbitrary value for the sole purpose of ensuring uniqueness.



Assigning Unique Identifiers

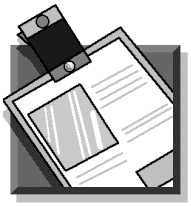
Examples of arbitrary, unique values used for this purpose are:

*** a system timestamp**
*** a system-assigned number**
*** a control number printed on a form**

Assigning Unique Identifiers

Candidate UIDs

Sometimes, an entity may have more than one potential UID. For example, for an EMPLOYEE entity, there may be both a company-assigned employee-id, and a government assigned number such as social security number.



Assigning Unique Identifiers

We must eventually select one of the unique identifiers to be the official UID: all Database Management Systems require that each object have one and only one UID, or *"Primary Key."* If we can guarantee the uniqueness of the company-assigned "employee id", but the uniqueness of the government-assigned number is questionable, then we should select the company "employee-id" to be the UID.

Assigning Unique Identifiers

Only the official UID will be tagged with a # on the E-R Model. Example;

EMPLOYEE
#* employee id
*** social security number**
*** first name**
*** last name**
o job title

The UID of EMPLOYEE is *employee-id*. Although an EMPLOYEE can be also uniquely identified by *social security number*, the attribute *social security number* is not tagged.

Assigning Unique Identifiers

Topic Summary

- **each entity must have a unique identifier (UID) which uniquely identifies each instance of that entity**
- **a UID may be composed of one or more attributes and/or one or more relationships**
- **the UID of an entity that consists of multiple attributes and/or relationships is a compound or composite UID.**
- **when a data model is converted to a physical database design, the UIDs become primary keys.**

Assigning Unique Identifiers

Topic Summary

- **UID attributes are notated with a "#" symbol when diagramming attributes**
- **UID relationships are notated with a bar on the relationship line**
- **artificial UIDs may be used to reduce complexity and ensure uniqueness of a UID**
- **The official UID of an entity is selected from the various candidate UIDs of an entity**

Normalizing Data Model

Resolving Many-to-many Relations

Modeling Recursive Relations

First Normal Form

Second Normal Form

Third Normal Form

Fourth Normal Form

Fifth Normal Form

- **Normalizing Data Model**

Topic Objectives

This topic discusses Normalization. At the end of the topic, you will be able to:

- **define Normalization**
- **identify an entity which is in its First Normal Form**
- **identify an entity which is in its Second Normal Form**
- **identify an entity which is in its Third Normal Form**
- **identify an entity which is in its Fourth Normal Form**
- **identify an entity which is in its Fifth Normal Form**

In the earlier topic on Attributes, we discussed some of the characteristics of attributes, or data elements:

- **each attribute should be elementary, not derived from others**
- **an attribute may be optional or mandatory**
- **the name should be unique and clear**

But we assumed that an attribute was in its correct place. We somehow assigned each attribute to the correct entity.

In this topic, you will see how to systematically assign each attribute to the proper entity.

Normalization

In Data Modeling, the term "Normalization" describes the step-by-step process of assigning attributes to entities.

Various authors have identified a number of rules, or principles, of Normalization. Some people claim there are five rules, others identify three or four rules.

Everyone aggrees on the first three, and for most business applications, applying the first three principles of Normalization produces a good data model.

The First Three Rules of Normalization

As you tentatively assign an attribute to an entity, you can ask three questions to verify that it (attribute) really belongs there:.

1) will it occur just once?

2) is it dependent on the entire Unique Identifier?

3) is it not dependent on another attribute?

If the answer to all three questions is "*Yes*" then the attribute is in the right place.

First Normal Form

Rule 1: will it occur just once?

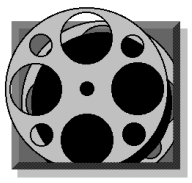
Here are some examples of attributes that probably meet this first rule:

CUSTOMER	FILM	PROJECT	PART
number	title	start date	name
first name	studio		
last name			

If we were to ask a business person, *"Can a customer have more than one customer number?"* or *"Can a part have more than one name?"* they would probably say *"No."*

Someone might argue that a film could be released under different titles, or could be re-released by a studio other than the one which released it the first time.

If this is true, then we must ask our business contact whether they would consider that to be the same film, or two different films.



Here are some examples of attributes which are misplaced, since they obviously violate the rule against repeating data items:

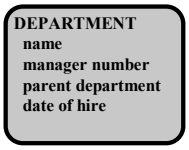
CUSTOMER	FILM	PROJECT	PART
payment amount	star	phone number	colour
	actor name	internal report	

The amount of a payment is not a fact that occurs once, for all time, per customer. It is misplaced; there must be another entity representing payments, where each row has a payment amount. The same is true for the attributes. They belong to some other entity.

If the proper entity already exists, then all you have to do is move the misplaced attribute.

If there is no proper entity, you have discovered a new one. You should go back to the beginning with that new entity, properly name it, and determine its relationships and Unique Identifier.

If you have an entity, DEPARTMENT, which of the following attributes clearly violates the rule against repeating data items?



A. name

B. manager number

C. parent department

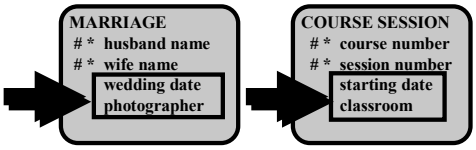
D. date of hire

The date of hire would occur many times for a DEPARTMENT.

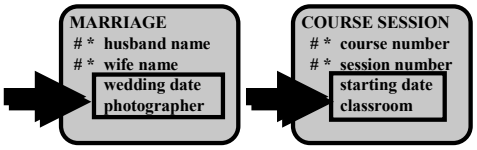
Second Normal Form

Rule 2: is it dependent on the entire Unique Identifier?

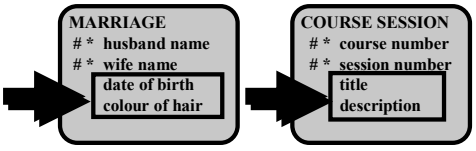
Here are some examples of attributes which depend on (are facts about) their entire UID, where # denotes the UID.



The wedding date and location, the person who officiated, the photographer, etc., are not facts about the husband alone, or the wife alone, but about the combination of them. Likewise, the time and location of a course session are not about the course alone, but about about a course and its specific session.

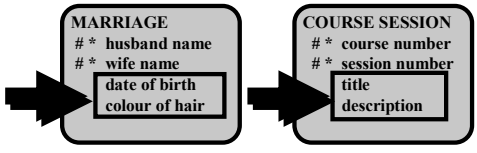


Here are some examples of attributes which are facts about only part of the entity's UID:



The date of birth should probably be two separate attributes, one for each spouse. Then it would be completely apparent that those attributes belonged to a different entity.

The title and description of a course are very likely the same for every session. In this case, these attributes belong to the COURSE entity, not COURSE SESSION.

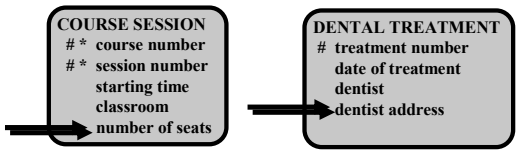


Third Normal Form

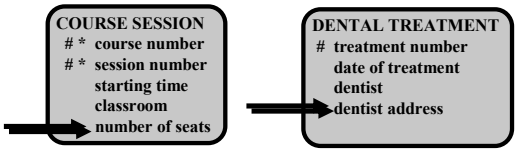
Rule 3: is it not dependent on another attribute?

A closely related question is, "*Does this attribute depend on something other than its entity's UID?*"

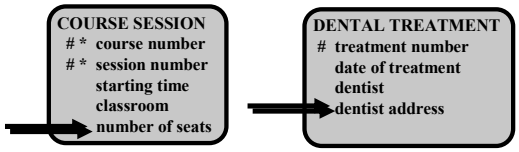
Here are examples of attributes which are facts about something other than the UID, and therefore fail the third rule of normalization:



The number of seats is a fact about the classroom, but not about the UID of the COURSE SESSION entity. If there is not another entity for CLASSROOM, then there should be, and "*number of seats*" should be one of its attributes.



In DENTAL TREATMENT, the dentist's address is not really a fact about the treatment number.



Fourth Normal Form

Rule 4: will it not occur several times?

This rule forbids the multi- valued relation between the UID and attribute.

Fourth Normal Form

Rule 4: will it not occur several times?

This rule forbids the multi- valued relation between the UID and attribute.

If the attribute will occur several times then it should belong to another entity.

Fourth Normal Form

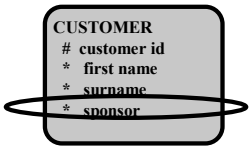
EXAMPLE;

Let us say that a BANK wants to keep record about sponsor under an entity called CUSTOMER.

Fourth Normal Form

EXAMPLE;

Let us say that a BANK wants to keep record about sponsor under an entity called CUSTOMER.

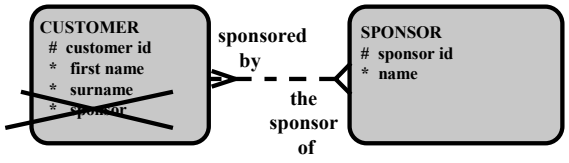


What if a CUSTOMER has more than one SPONSOR.

Fourth Normal Form

EXAMPLE;

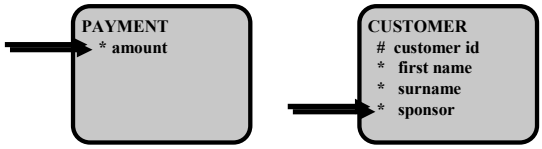
If a CUSTOMER has got more than one sponsor (even if only few times) it must belong to different entity.



Fourth Normal Form

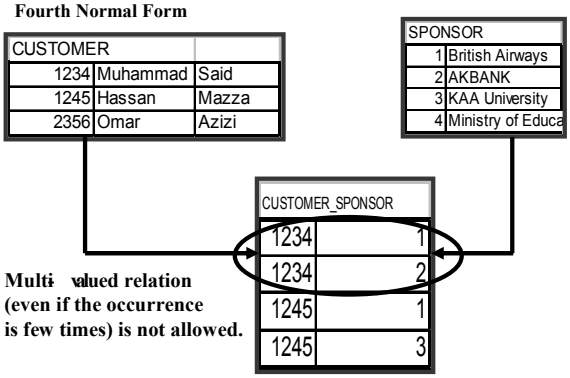
Difference of violation between 1st Normal Form and 4th Normal Form;

Repetition of attribute in 1st Normal Form is many times
Repetition of attribute in 4th Normal Form is only few times.



Violation of 1st Normal Form **Violation of 4th Normal Form**

Fourth Normal Form



Multi- valued relation (even if the occurrence is few times) is not allowed.

Fourth Normal Form

We rarely operate this rule. If applied in all cases, the number of tables in database increases tremendously.

Fourth Normal Form

Which of the following attributes violates 4th Normal Form?

DEFENDANT
 # defendant number
first name
last name
address

A. address

B. Surname

C. first name

D. None

Fifth Normal Form

In 5th Normal form;

If you have more than 3 candidate UIDs,
You should have a different entity for each UID and this should not cause redundancy in data.

Fifth Normal Form

You can ask a question like following for 5th Normal Form.

Has this entity got at least 3 candidate UIDs and can we construct different entities for each candidate UID without data redundancy (repetition).

Fifth Normal Form

In practice, this is very difficult.

Think of a global information system developed for central government.

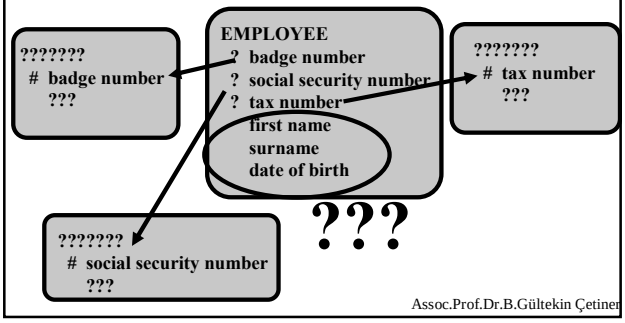
Fifth Normal Form

Following example has got 3 candidate UIDs. Try to define 3 entities corresponding to UIDs

EMPLOYEE
? badge number
? social security number
? tax number
first name
surname
date of birth

Fifth Normal Form

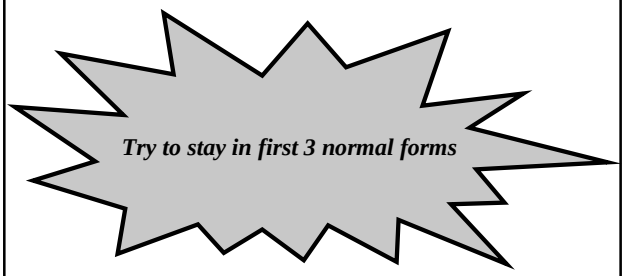
Following example has got 3 candidate UIDs. Try to define 3 entities corresponding to UIDs?



Summary of Normalization:

"Each fact (attribute) must be a fact about the entity, the whole entity, and nothing but the entity"

If each attribute of every entity passes this test, then the E-R Diagram is fully normalized.



CASE STUDIES

Data models reversed from Sample applications who clearly violate Normalization rules.

- **Student Registration System ***
- **Election Results ***

•Data models were obtained using *'ERWin Reverse Engineering Utility'*

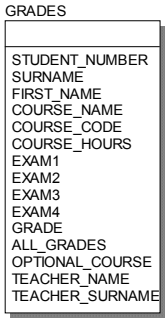
Öğrenci Not Takip Programı

CASE STUDY

Normalization Violations

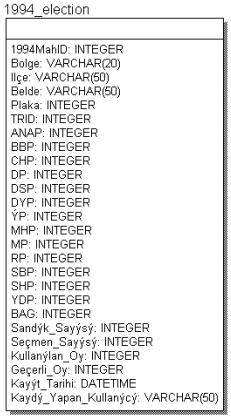
Find violations of 1st normal Form, 2nd Normal Form and 3rd Normal Form.

ONLY 15 ATTRIBUTES WERE SHOWN OUT OF 49 ATTRIBUTES



Election Results

Find Normalization Violations.



Normalizing Data Model **Normalization Violations**

Sample Data You notice the data repetition even without understanding.

Contents of 1994 Mahalli Seçimi Belde

Definition | Data | Enter SQL

1994MahID	Bolge	Ilçe	Belde	Plaka	TRID
1207	İSTANBUL	Büyükçekmece		34	41
1208	İSTANBUL	Büyükçekmece	Esenyurt	34	41
1209	İSTANBUL	Büyükçekmece	Gürpınar	34	41
1210	İSTANBUL	Büyükçekmece	Kavaklı	34	41
1211	İSTANBUL	Büyükçekmece	Kıraç	34	41
1212	İSTANBUL	Büyükçekmece	Kumburgaz	34	41
1213	İSTANBUL	Büyükçekmece	Mimarsinan	34	41
1214	İSTANBUL	Büyükçekmece	Tepecik	34	41
1215	İSTANBUL	Büyükçekmece	Yakuplu	34	41

Normalizing Data Model **Normalization Violations**

Extract the entity from the following table and discuss the problem.

TABLE NAME: CITIES

CITY1	CITY2	CITY3	CITY4	CITY5		CITY77	CITY78	CITY79
ADANA	ADIYAMAN	AFYON	AĞRI	AMASYA	.....	YALOVA	KARABÜK	KİLİS

Normalizing Data Model **Normalization Violations**

Extract the entity from the following table and discuss the problem.

TABLE NAME: CITIES

CITY1	CITY2	CITY3	CITY4	CITY5		CITY77	CITY78	CITY79
ADANA	ADIYAMAN	AFYON	AĞRI	AMASYA	.....	YALOVA	KARABÜK	KİLİS

CITY
City1
City2
City3
...
.....
....
City78
City79

Normalizing Data Model

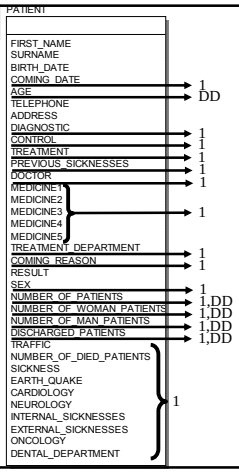
Normalization Violations

In

A Hospital Management System

1 1st Normal Form Violation

DD Derived Data



Identifying Many-to-many Relationships

Resolving Many-to-many Relations

Intersection Entities

Validation

Resolving Many-to-many Relations

Resolving Many-to-many Relations

Topic Objectives

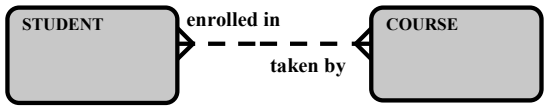
In this part, many-to-many relations are discussed. At the end of topic, You will be able to

- **explain why we need to solve the many-to-many relations**
- **identify intersection (association) entities**

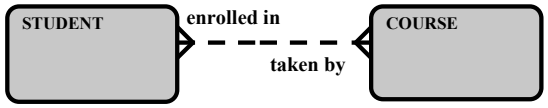
In a many-to-many relationship, relationship is multi-valued at both ends.

For example;

"Each STUDENT may be enrolled in one or more COURSEs"
"Each COURSE may be taken by one or more STUDENTs"

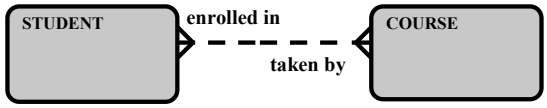


If there is a multi-valued relation at both ends of entities we cannot put attributes (facts) under any of entities (they will violate first normalization rule).



For example; attributes (or facts) such as enrollment date, exam date, and grade are hidden inside the relations. They are not the facts about STUDENT or COURSE.

They are all facts about the association between a STUDENT and a COURSE.

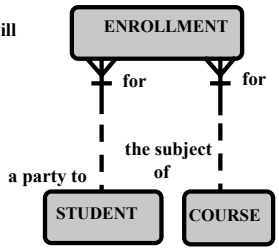


For example; try to put enrollment date under STUDENT and COURSE.

enrollment date violates 1st normal form both under STUDENT and COURSE.

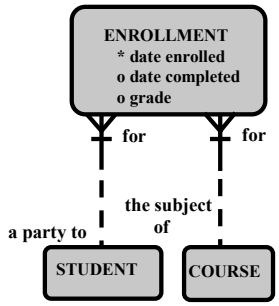
Resolving Many to many Relationships

To accommodate these facts about the relationship we "resolve" a many-to-many relationship by adding a third entity, ENROLLMENT, which will associate the two entities.



Resolving Many-to-many Relations

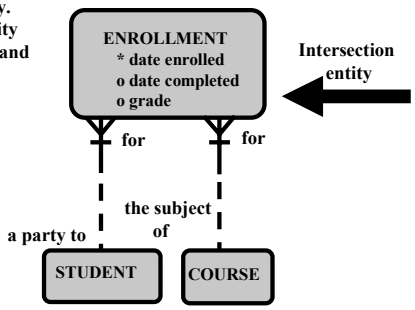
The facts about the relationship between STUDENT and COURSE now become attributes of the entity ENROLLMENT.



Resolving Many-to-many Relations

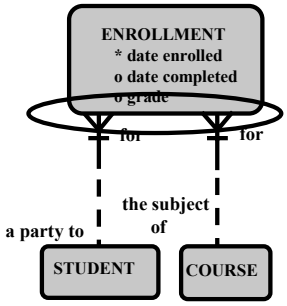
Intersection Entities

A new entity added to resolve a many-to-many relationship is called an intersection entity.
We also call this entity "association entity" and "junction entity".



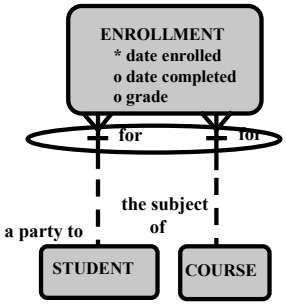
Resolving Many-to-many Relations

Note that any intersection entity will be on the many-valued end of the relationships to the two original entities.



Resolving Many-to-many Relations

And the Unique Identifier of an intersection entity will usually be its relationships to the two original entities, not any of its attributes.

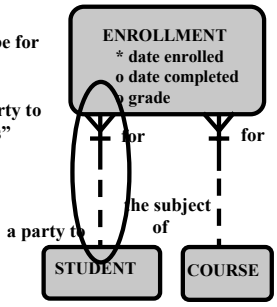


Resolving Many-to-many Relations

Validation
After drawing and naming the intersection entity, validate the new relationships by writing Relationship Sentences.

"Each ENROLLMENT must be for one and only one STUDENT"

"Each STUDENT may be a party to one or more ENROLLMENTs"

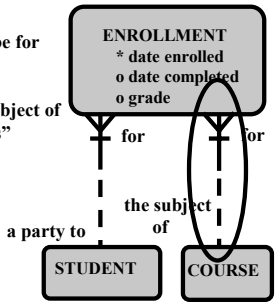


Resolving Many-to-many Relations

Validation
After drawing and naming the intersection entity, validate the new relationships by writing Relationship Sentences.

"Each ENROLLMENT must be for one and only one COURSE"

"Each COURSE may be the subject of one or more ENROLLMENTs"



Resolving Many-to-many Relations

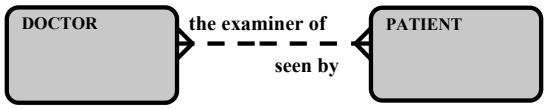
If we can read these sentences aloud to someone who knows the education business and that person agrees that it is the way the business operates, then we are OK.

"Each ENROLLMENT must be for one and only one COURSE"
"Each COURSE may be the subject of one or more ENROLLMENTs"

"Each ENROLLMENT must be for one and only one STUDENT"
"Each STUDENT may be a party to one or more ENROLLMENTs"

Resolving Many-to-many Relations

If one DOCTOR may be the examiner of many PATIENTs, and a PATIENT may e seen by many DOCTORs over time, what is the most likely intersection entity to connect them?



A. patient number

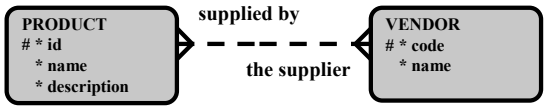
B. clinic address

C. VISIT

D. DOCTOR NUMBER

Resolving Many-to-many Relations

What is the likely intersection entity between PRODUCT and VENDOR if there is many-to-many relationship?



A. CATALOG

B. CUSTOMER

C. PRICE

D. DATE OF DELIVERY

Resolving Many-to-many Relations

Topic Objectives

- *Most business' data is full of many-to-many relationships-- students and teachers, doctors and patients, buildings and departments, trucks and drivers, and so on.*
- *You can always automatically draw a box for an intersection entity, and connect it to the others with many-to-one relationships.*
- *Be careful to evaluate the entity's relationships to be sure that they are truly many-to-one.*