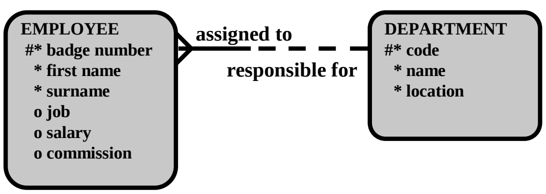
- **Modeling Recursive Relations**

Topic Objectives

This topic discusses Recursive Relationships. At the end of the topic, you will be able to;

- *Identify recursive relationships*
- *Explain some of the special problems in recursive relationships*
- *Model recursive relationships in an E-R Diagram*

Until this point, we have been dealing only with relationships between different entities, such as between EMPLOYEE and DEPARTMENT.



There are also many examples of entities which are related to themselves. Or, to put it more correctly, there can be relationships between some instances of an entity and other instances of the same entity. A relationship between an entity and itself is a recursive relationship.

How can we expand our E-R Model of EMPLOYEE and DEPARTMENT to represent the relationship between EMPLOYEE and his manager?

"Each EMPLOYEE may be managed by one and only one MANAGER."
"Each MANAGER may be the manager of one or more EMPLOYEEs."

Some sample instances of this relationship might be:

EMPLOYEE	MANAGER
John Brown	Mary Smith
Bob Phillips	Mary Smith
Mary Smith	Jim Jones
Jim Jones	

Every manager is also an employee. So manager is not a new entity, but just a subset of the instances of the entity EMPLOYEE.

Modeling Recursive Relations

Let us rewrite the relationshipsentences;

"Each EMPLOYEE may be managed by one and only one EMPLOYEE."
"Each EMPLOYEE may be the manager of one or more EMPLOYEEs."

This is a many-to-one relationship which is optional in both directions.

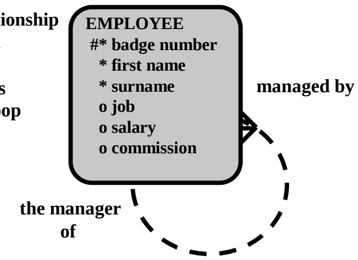
It is same as the relationship between two distinct entities.

Modeling Recursive Relations

Diagram a Recursive Relationship

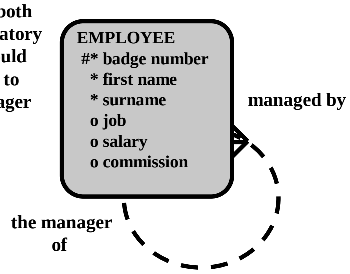
We can diagram the relationship with the same techniques.

A recursive relationship is always modelled with a loop or a bear's ear.



Modeling Recursive Relations

Note that both relationship sentences say "may be," so the relationship is optional in both directions. If it were mandatory in either direction, that would imly that an employee had to have a manager, or a manager had to have an employee.



Modeling Recursive Relations

Let's consider other types of data where these recursive relationships might occur. Entities which can be decomposed any number of times are often recursive.

Modeling Recursive Relations

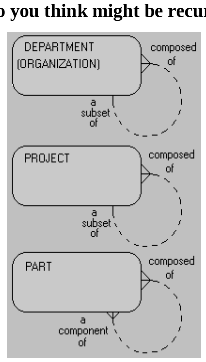
Which of the following entities do you think might be recursive?

A. PART

B. PROJECT

C. DEPARTMENT

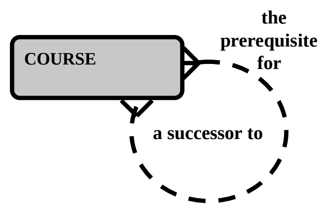
D. All of the above



Modeling Recursive Relations

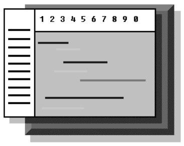
Another common type of recursive relationship appears when there is a chronological dependency. For example,

"Each COURSE may be the prerequisite for one or more COURSEs."
"Each COURSE may be a successor to one or more COURSEs."



Modeling Recursive Relations

Phases of a project, and any events which precede and follow one another usually show the same pattern.



All of them may e documented in your E-R Diagram in the same way as any other relationship.

Modeling Recursive Relations

Topic Summary

- ***A relationship between instances of the same entity is called a recursive relationship***
- ***Recursive relationships are modelled with a "loop" or "bear's ear"***
- ***Recursive relationships are usually optional in both directions.***

- ***Modeling Exclusive Relations***
- ***Identifying Sub and Supertypes (Entities)***
- ***Time Dependent Relations***

- **Exclusive Relations**
- **Arcs**

- ***Modeling Exclusive Relations***

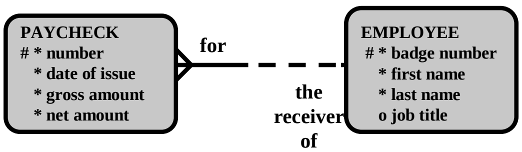
Modeling Exclusive Relations

Topic Objectives

This topic discusses Exclusive Relationships and arcs. At the end of the topic, you will be able to:

- **Identify exclusive relationships**
- **Model exclusive relationships using an arc**

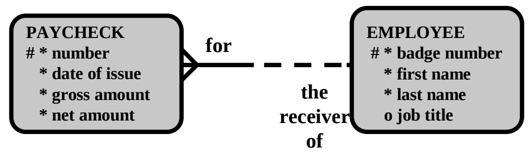
We have learned to model relationships that are mandatory (must be) and optional (may be).



We used Relationship Sentences to describe these relationships:

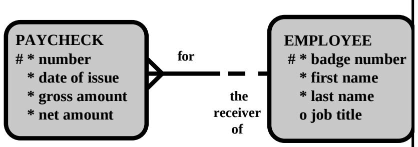
"Each PAYCHECK must be for one and only one EMPLOYEE."
"Each EMPLOYEE may be the receiver of one or more PAYCHECKs."

Occasionally, more complex data relationships are encountered in real-life.



Exclusive Relationships

Consider the situation in which a business utilizes both employees and contractors and and pays both employee's and contractor's paychecks. The business wants to track information about contractors that is different from the attributes of EMPLOYEE. So we will create a new entity called CONTRACTOR.



The entity PAYCHECK has an exclusive relationship with the entities EMPLOYEE and CONTRACTOR. Some instances of PAYCHECK will be for an EMPLOYEE. Other instances of PAYCHECK will be for a CONTRACTOR. A single instance of PAYCHECK must be for either an EMPLOYEE or a CONTRACTOR, never for both.

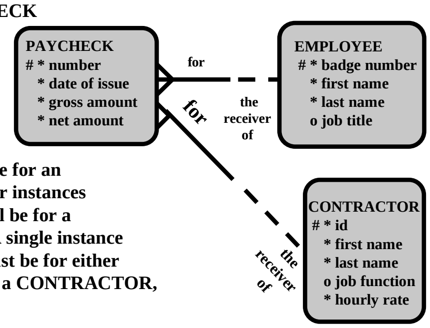
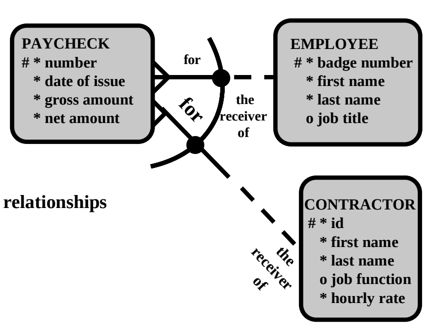


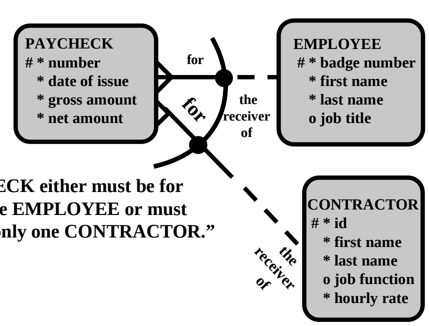
Diagram Arcs

We will use an arc to diagram an exclusive relationship.



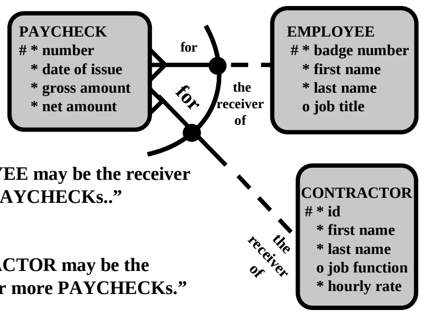
The arc spans the two relationships which are exclusive.

We read this exclusive relationship as:



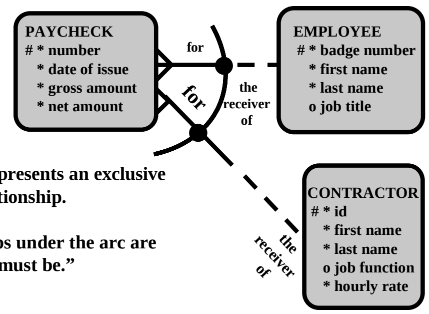
"Each PAYCHECK either must be for one and only one EMPLOYEE or must be for one and only one CONTRACTOR."

We read this exclusive relationship as:



“Each EMPLOYEE may be the receiver of one or more PAYCHECKs..”

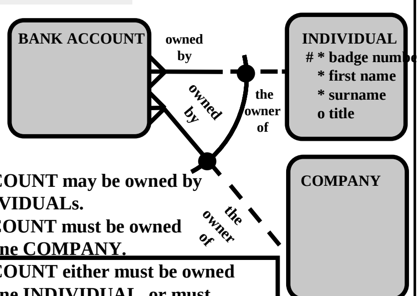
“Each CONTRACTOR may be the receiver of one or more PAYCHECKs.”



The diagram represents an exclusive mandatory relationship.

The relationships under the arc are mandatory or “must be.”

Which of the following Relationship Sentences describes this diagram.



A. Each BANK ACCOUNT may be owned by one or more INDIVIDUALs.

B. Each BANK ACCOUNT must be owned by one and only one COMPANY.

C. Each BANK ACCOUNT either must be owned by one and only one INDIVIDUAL, or must be owned by one and only one COMPANY.

D. Each BANK ACCOUNT either may be owned by one and only one INDIVIDUAL, or must be owned by one and only one COMPANY.

Topic Summary

- **An entity may have relationships with more than one other entity.**
- **A set of relationships from an entity may be mutually exclusive; for a single instance of that entity, only one relationship in that set can exist.**
- **An arc is used to denote relationships which are exclusive.**

Diagram Subtypes

Attributes of Subtypes

Relationships of Subtypes

Identify Subtypes and Supertypes

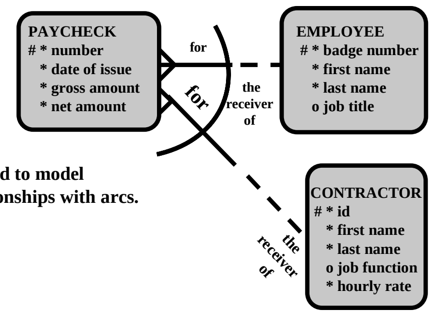
Identify Subtypes and Supertypes

Topic Objectives

This topic discusses Subtypes and Supertypes. At the end of the topic, you will be able to:

- **identify situations where a single entity may be subdivided into "subtypes"**
- **identify when two or more similar entities may be combined into one "supertype"**
- **model subtypes and supertypes in an E-R Diagram**

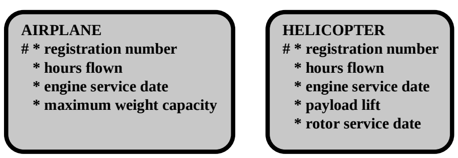
Identify Subtypes and Supertypes



We have learned to model exclusive relationships with arcs.

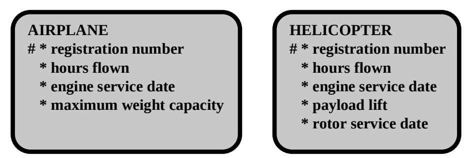
Identify Subtypes and Supertypes

Often we will encounter entities which are <u>mutually exclusive</u>, but have common attributes. For example, an AIRPLANE entity and a HELICOPTER entity are mutually exclusive-- an airplane cannot also be helicopter, and a helicopter cannot be an airplane.



Identify Subtypes and Supertypes

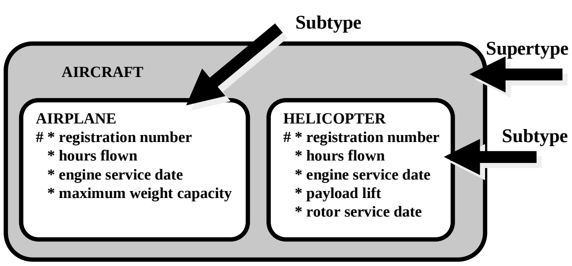
However, an AIRPLANE entity and HELICOPTER entity do share attributes in common- such as registration number, hours flown, and engine service date.



Identify Subtypes and Supertypes

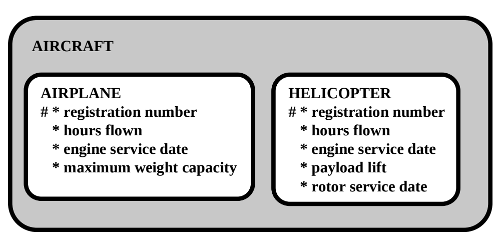
Diagram Subtypes

Exclusive entities can be modelled using a supertype and two or more subtypes. For example, we could define a supertype AIRCRAFT with subtypes of AIRPLANE and HELICOPTER.



Identify Subtypes and Supertypes

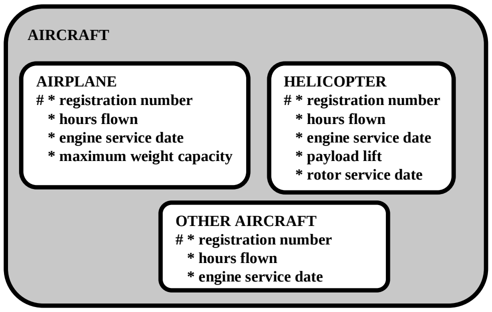
All instances of a supertype must belong to one and only one of the subtype entities. Can an AIRCRAFT be both an AIRPLANE and a HELICOPTER?



What about gliders, ultralights, and military ones? These are instances of AIRCRAFT which are not AIRPLANEs or HELICOPTERs.

Identify Subtypes and Supertypes

To accommodate these other AIRCRAFT, add a third subtype called OTHER AIRCRAFT

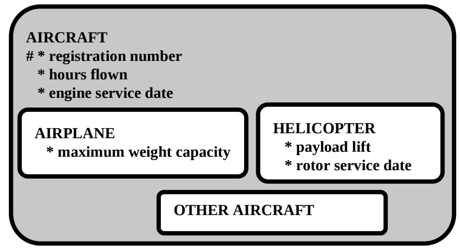


Now the subtypes of AIRCRAFT form a complete, non-overlapping set.

Identify Subtypes and Supertypes

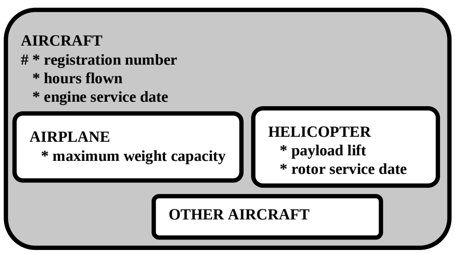
Attributes of Subtypes

A supertype can have attributes of its own. Each subtype will inherit the attributes of its supertype. Attributes shared by the subtypes should be defined at the supertype level. We can reassign the attributes shared by the subtypes HELICOPTER, AIRPLANE, and OTHER AIRCRAFT, to their supertype AIRCRAFT.



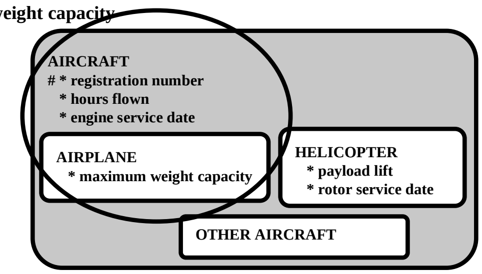
Identify Subtypes and Supertypes

For each instance of a subtype, the attributes of the supertype and the attributes specific to that subtype must be defined.



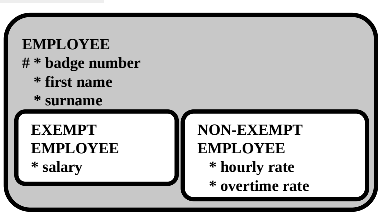
Identify Subtypes and Supertypes

For example, each instance of an AIRPLANE has the following attributes:
#* registration number
*** hours flown**
*** service date**
*** maximum weight capacity**



Identify Subtypes and Supertypes

What attributes exist for each instance of an EXEMPT EMPLOYEE?



A. * salary

B. # * badge number
*** salary**

C. # * badge number
*** first name**
*** surname**
*** salary**

D. # * badge number
*** first name**
*** surname**
*** hourly rate**
*** overtime rate**

Identify Subtypes and Supertypes

Relationships of Subtypes

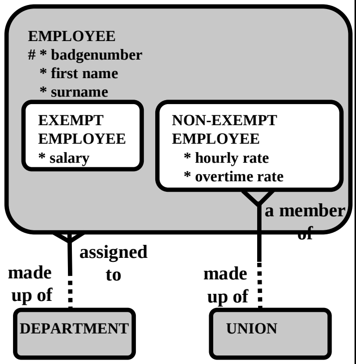
Both the supertype and its subtypes may have relationships to other entities.

Identify Subtypes and Supertypes

The EMPLOYEE supertype has a relationship with the entity DEPARTMENT.

"Each EMPLOYEE must be assigned to one and only one DEPARTMENT."

The relationship to DEPARTMENT applies to all instances of the supertype: all EXEMPT EMPLOYEEs and all NON-EXEMPT EMPLOYEEs. Subtypes inherit the relationship of their supertypes.

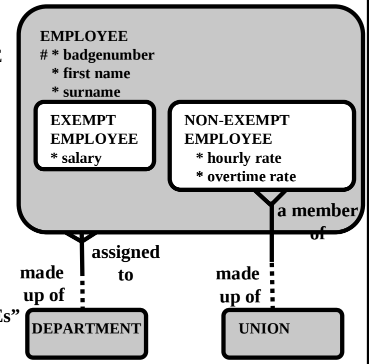


Identify Subtypes and Supertypes

A subtype may alsohave relationships of its own. Only non-exempt employees belong to the unions. So only the NON-EXEMPT EMPLOYEE subtype has a relationship with the UNION entity:

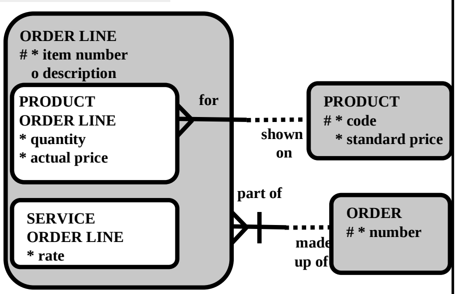
"Each NON-EXEMPT EMPLOYEE must be a member of one and only one UNION."

"Each UNION may be made up of one or more NON-EXEMPT EMPLOYEEs"



Identify Subtypes and Supertypes

What entities does PRODUCT ORDER LINE have relationships with?



A. ORDER **B. PRODUCT ORDER** **C. ORDER LINE SERVICE ORDER LINE** **D. ORDER LINE PRODUCT**

Identify Subtypes and Supertypes

Topic Summary

- **exclusive entities can be modelled by using a supertype and two or more subtype**
- **all instances of a supertype must belong to one and only one of the subtype entities**
- **a supertype can have attributes; each subtype will inherit the attributes of its supertype**
- **each subtype can also have attributes of its own**
- **a supertype and/or its subtypes may have relationships to other entities**

Time Dependent Relationships

Expand a Relationship

Add an Entity

- **Modeling Time Dependent Relations**

Topic Objectives

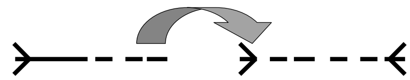
This topic discusses Time Dependent Relationships. At the end of the topic, you will be able to:

- **identify the changes in relationships by storing data over time**
- **modify an E-R Diagram to accommodate these changes**

Introduction

As you may have noticed, time does make a difference when we are constructing a data model.

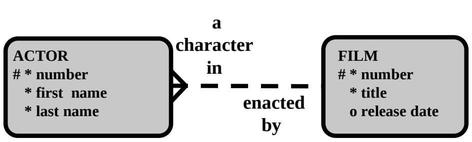
To accommodate data over time, we may have to change a relationship from "one and only one" to "one or more" and perhaps add an intersection entity to resolve a resultant many-to-many relationship. In other situations, we may have to add new entities and relationships.



Expand a Relationship

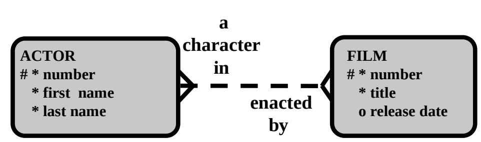
For example, if you are trying to decide how ACTOR and FILM are related, you might initially construct a Relationship Sentence like this:

"Each ACTOR may be a character in one and only one FILM."

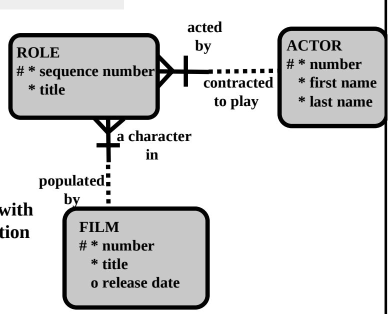


If you are thinking of the situation at one instant in time, then this may be true; however, a really busy actor could be working on more than one film at the same time. And if we add the explicit phrase, "over time", then clearly the sentence should be reformulated:

(Over time,) each ACTOR may be a character in one or more FILMs ."



By adding "over time," we have created a many-to-many relationship which should be resolved with the addition of an intersection entity. In an earlier workshop lesson, we saw this M:M resolved with the addition of the intersection entity ROLE.



See how addition of "over time" can lead to a more complex E-R Diagram?

Add an Entity

Let's take another example to see how the innocent little phrase "over time" can complicate things and force us to alter our E-R Diagram.

Look at the FILM entity again:

Modeling Time Dependent Relations

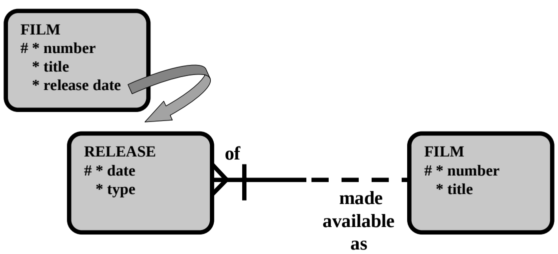
According to the first rule of normalization, each of the attributes must occur one and only one time. In other words, we can store only a single release date. As long as we just want to know the original release date, this design works fine.

Modeling Time Dependent Relations

But in validating our data model with the business people, we may find that there is a requirement to store not only the original release date, but also re-release dates for the same film. There may be requirements to store release dates for theaters, video stores, cable movie channels, second-run theaters, VCD releases, DVD releases, and independent TV stations.

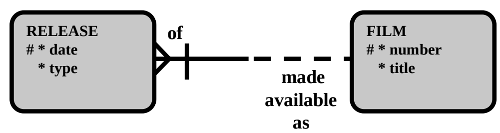
Modeling Time Dependent Relations

Since release date can not be repeated in the FILM entity (due to first rule of Normalization), we need an additional entity to store the multiple occurrences of release date, plus the type of release:



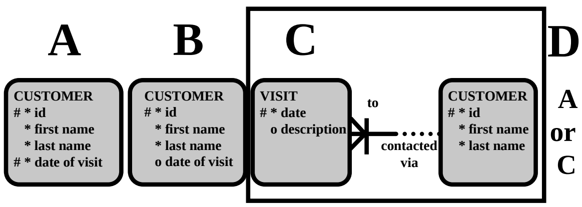
Modeling Time Dependent Relations

There is a many-to-one relationship between the new entity RELEASE and the old entity FILM. The UID (Unique Identitifer) of RELEASE will be the related FILM and the attribute *release date*.

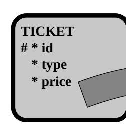


Modeling Time Dependent Relations

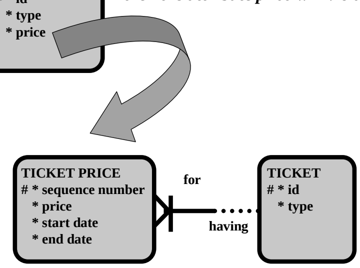
Initially, the date of visit attribute was assigned to the entity CUSTOMER in order to track the last customer visit. Now it is necessary to track each date a customer was visited and a description of the visit. How can the E-R Diagram be changed to accommodate this new requirement?



Modeling Time Dependent Relations

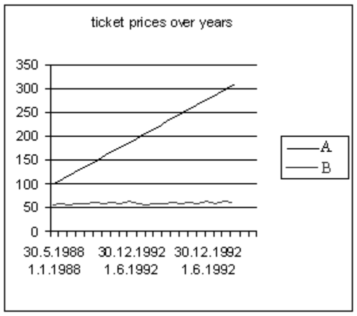


If you want to track all price changes for ticket over time, then the attribute *price* will violate the first-normal form.



Modeling Time Dependent Relations

Infrastructure for a Data Warehouse can be constructed through "Over time" inclusions.



Modeling Time Dependent Relations

Topic Summary

In general, the introduction of the time element into a data model will force the creation of new entities to store the chronology. Here are some typical causes;

- **we need to keep an audit trail**
- **values (e.g.price) may change over time**
- **relationships can change over time**

These "historical" entities will often include such attributes as "start date" and "end date."

Data Scale

Data Scale

There are 4 different data scales for attributes.
(Data scales and data types are different things)

- **1. Nominal Data (Atomic data). Data on which we cannot make statistical processes**
- **2. Numerical Data (We can make computations on this kind of data)**
- **3. Ordinal Data (Grouped or classified data. They can be used for statistical purposes)**
- **4. Boolean Data (Yes-No or True-False types of data)**

Data Scale

Data Scale

1. Nominal Data (Atomic data). Data on which we cannot make statistical processes

Nominal data can include data types of text, graphics, multimedia (blob) and even numerical ones. (for example; *Employee Badge Number*. We use numerical data type for badge number but it is not a numerical data as data scale, because it hasn't got any statistical meaning)

Nobody will ask you about the average, or mean value of employee badge numbers.

Data Scale

Data Scale

2. Numerical Data (We can make computations on this kind of data)

This data scale includes many numerical data types such as integer, float etc.

Date might be also used as numerical scale (for example; to calculate age)

Data Scale

Data Scale

3. Ordinal Data (Grouped or classified data. They can be used for statistical purposes)

Includes grouped or classified data. It might use many data types such as text, integer .

Examples; Bload Group, Document type, type of driving license, marital status etc.

Ordinal data may have statistical meaning.

Data Scale

Data Scale

4. Boolean Data (Yes-No or True-False types of data)

For example;
Gender, IsSomething (Is_married) types of data.
They might be used for statistical purposes.

Data Scale

Data Scale

CONCLUSION

During Conceptual Data Modeling We talk about Data Scale not Data Types.

Data Modeling (Conceptual)	Database Design
Data Scales →	Data Types
Independent of the Database Management System.	Dependent on the Database Management System
We have only 4 scales.	We have many data types.

Database Design from Data Models

- **Transform the Data Model into Database Design**

Main Topics

- **Understanding Relational Databases**
- **Transform Data Model into Design**
- **Understand Further Database Design**

- **Database Design**
- **Relational Tables**
- **Primary Keys**
- **Foreign Keys**
- **SQL-Structured Query Language**

Understanding Relational Databases

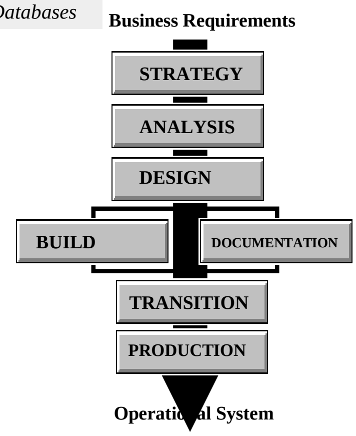
Topic Objectives

This topic introduces Database Design and Relational Database concepts. At the end of this topic, wou will be able to;

- **explain how Database Design fits into the Database Development Process**
- **identify the structures used in relational databases: tables and columns, primary and foreign keys**

Database Design

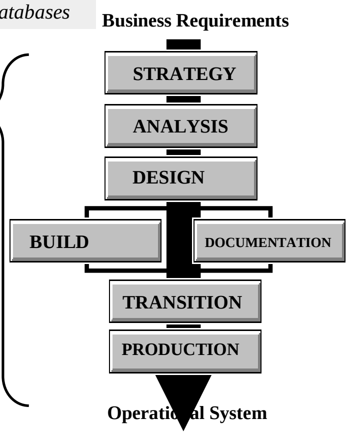
Conceptual Data Modeling is the first step of the top-down Database Design Process. Once the E-R Model is complete, it is ready to be translated into a database design. Database Design is performed during the Design stage of the System Development Life Cycle.



Software Development Life Cycle (Waterfall Model)

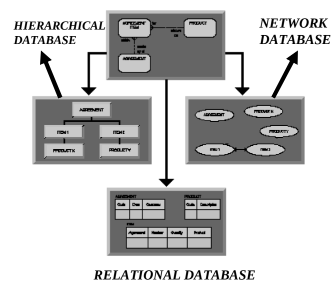
This type of Software Development Life Cycle is called Waterfall Model. Since it is difficult to swim up to the waterfall stream, it is costly to go back to the previous stages in life cycle.

Therefore, it is essential to finish a good data model before starting database design.



One of the goals of Conceptual Data Modeling is to build a model which is independent of any specific Database Management System (DBMS). A good data model can be mapped to a database design for any type of DBMS: hierarchical, network, relational, or any other type which exists or may be devised.

ENTITY-RELATIONSHIP MODEL



But today, the majority of new databases being designed will probably be implemented in a Relational DBMS.

In this lesson, we will discuss converting the generic data model into a physical database design, using a Relational DBMS.

Relational Tables

The name "Relational" DBMS comes from the term "Relation," which in mathematics means "Table". In a Relational DBMS, we, the users of the data, perceive the data as a collection of tables.

The data is actually stored in some type of file, which differs from one relational DBMS to another. Each RDBMS, however, manages to present the data to us so that it appears in the form of a simple table, or a collection of tables.

Tables consists of one or more columns, and any number of rows. A table may contain zero rows, but of course tables will usually consist of at least one row, and usually many hundreds, thousands, millions, or even billions.

Here is an example of a relational table called ACTOR:

ACTORS

TABLE

ROW

ACTOR NUMBER	SURNAME	FIRST NAME	AGENCY NUMBER
1251	WAYNE	JOHN	10
339	HURT	WILLIAM	15
2500	TURNER	KATHLEEN	10

COLUMN

One of the main principles of relational DBMSs is that all programs which retrieve data from the table must use column names: no applications can reference "the third column."

In relational DBMSs, the physical order of the rows does not give them any meaning. A row means the same thing whether it is first, last, or anywhere else in the table, and whether it precedes or follows some other row.

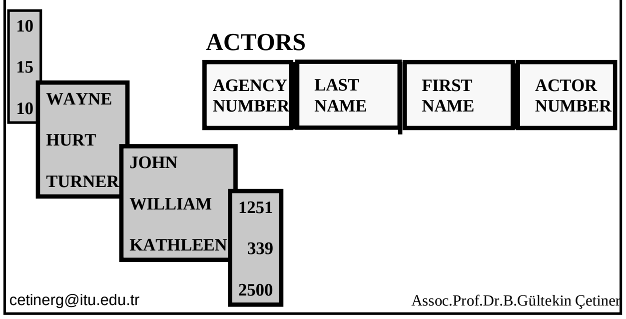
Suppose that the columns "ACTOR_NUMBER" and "AGENCY NUMBER" have been changed by redefining table structure, <u>the programs and queries will not be affected at all.</u>

ACTORS

AGENCY NUMBER	SURNAME	FIRST NAME	ACTOR NUMBER
10	WAYNE	JOHN	1251
15	HURT	WILLIAM	339
10	TURNER	KATHLEEN	2500

No meaning should be lost when the order of columns is changed.

(Select ACTOR_NUMBER, FIRST_NAME FROM ACTORS)



Each DBMS imposes differing limits on the kinds of names which can be used for columns, and on the types of columns which may be defined. But almost all relational DBMSs provide a variety of numeric types, character types, dates and times, and possibly graphic types.

GENERAL DATA TYPES

NUMERIC	DATE/TIME	TEXT	GRAPHIC/ DOCUMENT/ MEMO etc.

GENERAL DATA TYPES

DBMS	BLOB DATA TYPE	DATE/TIME DATA TYPE	NUMERICAL DATA TYPE	STRING DATA TYPE
MS SQL SERVER	BINARY BINARY() IMAGE VARBINARY VARBINARY()	DATETIME SMALLDATETIME TIMESTAMP	INT MONEY NUMERIC NUMERIC() NUMERIC(,) REAL SMALLINT SMALLMONEY TINYINT	CHAR CHAR() TEXT VARCHAR VARCHAR()
IBM DB2	GRAPHIC GRAPHIC() LONG VARGRAPHIC VARGRAPHIC()	DATE TIME TIMESTAMP	DECIMAL DECIMAL(,) FLOAT FLOAT() INTEGER REAL SMALLINT	CHAR CHAR() CHARACTER CHARACTER() LONG VARCHAR VARCHAR()

GENERAL DATA TYPES

DBMS	BLOB DATA TYPE	DATE/TIME DATA TYPE	NUMERICAL DATA TYPE	STRING DATA TYPE
ORACLE	LONG LONG RAW MLSLABEL RAW MLSLABEL RAW()	DATE	DECIMAL() DECIMAL(,) FLOAT INTEGER NUMBER NUMBER(*) NUMBER(,) SMALLINT	CHAR() CHARACTER() LONG VARCHAR VARCHAR2()
INTERBASE	BLOB	DATE	DECIMAL DECIMAL() DECIMAL(,) DOUBLE PRECISION FLOAT INTEGER NUMERIC NUMERIC() NUMERIC(,) SMALLINT	CHAR VARCHAR()

GENERAL DATA TYPES

DBMS	BLOB DATA TYPE	DATE/TIME DATA TYPE	NUMERICAL DATA TYPE	STRING DATA TYPE
INGRES	BYTE VARYING LONG BYTE	DATE	BYTE DECIMAL FLOAT FLOAT() FLOAT4 FLOAT8 INTEGER INTEGER1 INTEGER2 INTEGER4 MONEY SMALLINT	C CHAR() LONG VARCHAR TEXT() VARCHAR()
MS ACCESS	OLE OBJECT	DATE/TIME	AUTONUMBER BYTE CURRENCY DOUBLE INTEGER LONG INTEGER REPLICATION ID SINGLE YES/NO	MEMO TEXT()

GENERAL DATA TYPES

DBMS	BLOB DATA TYPE	DATE/TIME DATA TYPE	NUMERICAL DATA TYPE	STRING DATA TYPE
PARADOX	BINARY BINARY() GRAPHIC GRAPHIC() OLE OLE()	DATE TIME TIMESTAMP	AUTOINCREMENT BCD BCD() BYTES() LOGICAL LONG INTEGER MONEY NUMBER SHORT	ALPHA() FORMATTED MEMO FORMATTED MEMO() MEMO()
SYBASE	BINARY() IMAGE VARBINARY()	DATE/TIME SMALLDATETIME TIMESTAMP	BIT DECIMAL DECIMAL() DECIMAL(,) FLOAT INT MONEY NUMERIC NUMERIC() NUMERIC(,) REAL SMALLINT SMALLMONEY TINYINT	CHAR() TEXT VARCHAR()

GENERAL DATA TYPES

DBMS	BLOB DATA TYPE	DATE/TIME DATA TYPE	NUMERICAL DATA TYPE	STRING DATA TYPE
INFORMIX	BYTE	DATE	DEC DEC(,) DECIMAL DECIMAL(,) DOUBLE PRECISION DOUBLE PRECISION() FLOAT FLOAT() INT INTEGER MONEY MONEY(,) NUMERIC NUMERIC(,) REAL SERIAL SERIAL() SERIAL(1) SMALLFLOAT SMALLINT	CHAR CHAR() CHARACTER CHARACTER() NCHAR NCHAR() NVARCHAR() TEXT VARCHAR()

Understanding Relational Databases

GENERAL DATA TYPES

DBMS	BLOB DATA TYPE	DATE/TIME DATA TYPE	NUMERICAL DATA TYPE	STRING DATA TYPE
PROGRESS	X	DATE	DECIMAL DECIMAL() DECIMAL(,) FLOAT FLOAT() INTEGER LOGICAL NUMERIC NUMERIC() NUMERIC(,) REAL SMALLINT	CHAR() CHARACTER()
FOXPRO ve DBASE IV	MEMO	DATE	FLOAT(,) LOGICAL NUMERIC(,)	CHARACTER()

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

Understanding Relational Databases

Another principle of relational DBMSs is that each row of a table contains the information about one and only one instance of the entity. Therefore, each row has the same "weight" or importance as every row in the same table. In our example, each row is about one and only one actor or actress.

ACTORS

ACTOR NUMBER	SURNAME	FIRST NAME	AGENCY NUMBER
1251	WAYNE	JOHN	10
2500	TURNER	KATHLEE	10
339	HURT	WILLIAM	15

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

Understanding Relational Databases

Therefore, each entity has to have a unique identifier.

This is called consistency of the Entity.

ACTORS

ACTOR NUMBER	SURNAME	FIRST NAME	AGENCY NUMBER
1251	WAYNE	JOHN	10
2500	TURNER	KATHLEE	10
339	HURT	WILLIAM	15
339	QUEEN	ANTONY	15

Not allowed since actor number is unique identifier.

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

Understanding Relational Databases

Another rule:
No meaning should be lost when the order of rows is changed.

(Select ACTOR_NUMBER, FIRST_NAME FROM ACTORS where ACTOR_NUMBER=1251)

ACTORS

ACTOR NUMBER	SURNAME	FIRST NAME	AGENCY NUMBER
2500	TURNER	KATHLEE	10
1251	WAYNE	JOHN	10
339	HURT	WILLIAM	15

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

Understanding Relational Databases

In some older file designs (traditional approach), there were "header records" and "detail records." Also, perhaps there were "type 1,2,3, and 7" records. Many times in these kinds of files, the actual physical order of the records gave them their meaning. If a certain detail transaction record happened to follow the wrong header record, it was incorrectly associated with the wrong account. Or, in our example, if rows for films and rows for actors were intermixed in the same table, the actor could appear to be "in" the wrong film if the rows were in the wrong order.

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

In older file designs, physical order of data (for example; random access files) was important. When someone changed the header structure all the programs and queries had to be changed.

In short,

all the meaning that data has in a relational DBMS comes from the values in the row, not from any left-to-right or top-to-bottom position within the table.

In a relational table, the physical order of the rows and columns is immaterial. Which of the following tables are equivalent?

Table 1

EMPNO	FNAME	LNAME	DEPTNO
7369	MARY	SMITH	20
7902	HENRY	FORD	50
7521	SUE	WARD	30
7698	BOB	BLAKE	30

Table 2

EMPNO	DEPTNO	FNAME	LNAME
7369	20	MARY	SMITH
7902	50	HENRY	FORD
7521	30	SUE	WARD
7698	30	BOB	BLAKE

Table 3

EMPNO	FNAME	LNAME	DEPTNO
7698	BOB	BLAKE	30
7521	SUE	WARD	30
7902	HENRY	FORD	50
7369	MARY	SMITH	20

A. 1

B. 1,2

C. 1,3

D. 1,2,3

Primary Keys

A Primary Key (PK) is a column or set of columns that uniquely identifies each row in a table. Each table must have a primary key. The primary key of the ACTORS table is ACTOR_NUMBER.

ACTORS

ACTOR_NUMBER	LAST_NAME	FIRST_NAME	AGENT_NUMBER
1251	WAYNE	JOHN	5
339	HURT	WILLIAM	5
2500	TURNER	KATHLEEN	7

FOREIGN KEYS

A Foreign Key (FK) is a column or combination of columns in one table that refers to a primary key in the same or another table. AGENT_NUMBER is a foreign key in the ACTORS Table and refers to AGENT_NUMBER column in the AGENTS Table.

FOREIGN KEY

ACTORS

ACTOR NUMBER	LAST NAME	FIRST NAME	AGENT NUMBER
1251	WAYNE	JOHN	5
339	HURT	WILLIAM	5
2500	TURNER	KATHLEEN	7

PRIMARY KEY

AGENTS

AGENT NUMBER	AGENT NAME
5	MARY TAYLOR
7	PAUL JOHNSON

Structured Query Language-SQL

Structured Query Language (SQL) is the internationally accepted standard language for querying and manipulating data in relational databases. The following SQL statement retrieves the values of ACTOR_NUMBER, LAST_NAME, FIRST_NAME, and AGENT_NUMBER from the ACTORS Table for Actor Number 350.

```
SELECT ACTOR_NUMBER,
LAST_NAME,FIRST_NAME,AGENT_NUMBER
  FROM ACTORS
    WHERE ACTOR_NUMBER=350
```

A primary key must uniquely identify each row of a table. What column or combination of columns could serve as the primary key of the CATALOG_ITEMS Table?

CATALOG_ITEMS

PRODUCT NUMBER	VENDOR NUMBER	PACKAGE QUANTITY	ITEM PRICE
99	5	5	$25,50
99	6	10	$15,35
99	5	20	$23,00
102	5	5	$25,00
103	5	3	$5,00

A. **PRODUCT_NUMBER**

B. **PRODUCT_NUMBER, ITEM_PRICE**

C. **ITEM_PRICE**

D. **PRODUCT_NUMBER, VENDOR_NUMBER, PACKAGE_QUANTITY**

Topic Summary

- **Database Design comes after data modeling; it takes place during the design stage of the system development life cycle.**
- **During Database Design, an E-R model is mapped (tansformed) to a relational database design.**
- **One of the goals of conceptual data modeling is to build a model which is independent of any specific DBMS.**
- **A relational database is made of one or more relations or "tables" which contain data.**
- **Tables consist of one or more columns and any number of rows.**

Topic Summary

- **Columns are identified by names and may consist of different data types with different keywords for each DBMS.**
- **Each row of data in a table represents information for only one instance of entity (Consistency of Entity).**
- **Each table must have a primary key; a primary key is a column or set of columns that uniquely identifies each row in the table . Consistency of Entity is obtained via the use of unique identifier.**
- **Each table may also have a foreign key; a foreign key is a column or set of columns that refers to a primary key in the same or another table.**
- **SQL (Structured Query Language) is the internationally accepted language for querying and manipulating data in relational databases.**

- **Mapping (Transforming) Data Model into Database Design**

Mapping Entities

Mapping Attributes

Mapping Unique Attributes

Mapping Relationships

Mapping Data Scales

- **Mapping (Transforming) Data Model into Database Design**

Topic Objective:

This topic discusses the steps for mapping an E-R Diagram to a relational database design. At the end of this topic, you will be able to;

- **Translate an E-R Diagram into a relational database design**

Mapping (Transforming) Data Model into Database Design

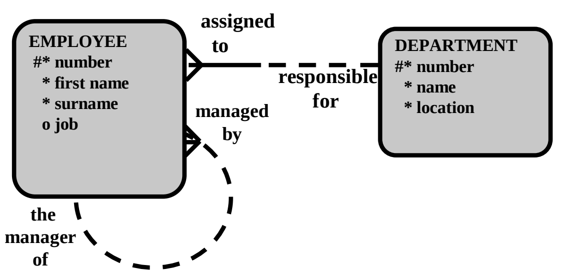
Mapping an E-R Diagram to a relational database physical design is fairly straightforward. Each part of the E-R Diagram translates directly into a part of a relational database design, as follows:

E-R MODEL		RELATIONAL DATABASE DESIGN
Entity	→	Table
Attribute	→	Column
Unique Identifier	→	Primary Key
Relationship	→	Foreign Key

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

Mapping (Transforming) Data Model into Database Design

Let's map the following E-R Diagram to a relational database design.



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

Mapping (Transforming) Data Model into Database Design

Map Entities

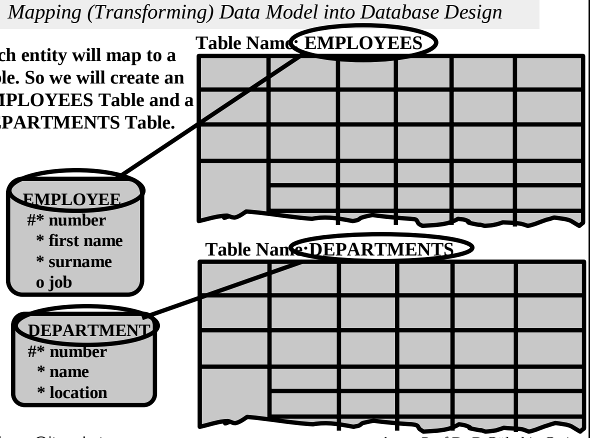
First we will map the entities to tables.

E-R MODEL		RELATIONAL DATABASE DESIGN
Entity	→	Table
Attribute	→	Column
Unique Identifier	→	Primary Key
Relationship	→	Foreign Key

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

Mapping (Transforming) Data Model into Database Design

Each entity will map to a table. So we will create an EMPLOYEES Table and a DEPARTMENTS Table.



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

Mapping (Transforming) Data Model into Database Design

Map Attributes

Each Attribute will map to a column

E-R MODEL		RELATIONAL DATABASE DESIGN
Entity	→	Table
Attribute	→	Column
Unique Identifier	→	Primary Key
Relationship	→	Foreign Key

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

For example, the attribute EMPLOYEE number will map the column EMPNO in the EMPLOYEES Table.

Table Name: EMPLOYEES

Column Name	EMPNO	FNAME	LNAME	JOB	
Key Type					
Nulls/ Unique					
Sample Data					

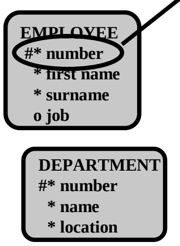


Table Name:DEPARTMENTS

Column Name	DEPTNO	DNAME	LOC
Key Type			
Nulls/ Unique			
Sample Data			

At this point in the database design, we will add sample data to the table to provide a visual check of the table's contents.

Table Name: EMPLOYEES

Column Name	EMPNO	FNAME	LNAME	JOB	
Key Type					
Nulls/ Unique	NN	NN	NN		
Sample Data	7369	MARY	SMITH		
	7902	HENRY	FORD	ANALYST	

Those attributes labeled "*" for mandatory will be marked "NN" for NOT NULL in the table design. The RDBMS will not allow a column marked NOT NULL to contain a missing or undefined value.

Table Name:DEPARTMENTS

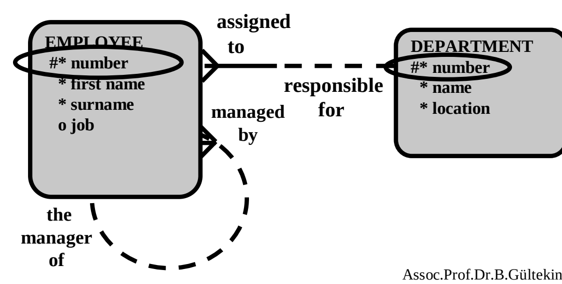
Column Name	DEPTNO	DNAME	LOC
Key Type			
Nulls/ Unique	NN	NN	NN
Sample Data	10	ACCOUNTING	NEWYORK
	20	RESEARCH	DALLAS

Map Unique Identifiers

Next we will map each entity's Unique Identifier (UID) to the corresponding table's primary key.

E-R MODEL	RELATIONAL DATABASE DESIGN
Entity	Table
Attribute	Column
Unique Identifier	Primary Key
Relationship	Foreign Key

The UID of the EMPLOYEE entity is the attribute EMPLOYEE number. So, the primary key of the EMPLOYEES Table will be the column EMPNO. We will label the EMPNO column PK for Primary Key and U for Unique. Likewise for the DEPARTMENT entity, the UID attribute DEPARTMENT number will map to the primary key DEPTNO.



For example;

Table Name: EMPLOYEES

Column Name	EMPNO	FNAME	LNAME	JOB	
Key Type	PK				
Nulls/ Unique	NN/U	NN	NN		
Sample Data	7369	MARY	SMITH		
	7902	HENRY	FORD	ANALYST	

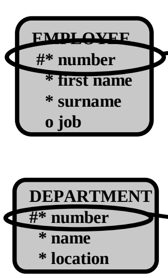


Table Name:DEPARTMENTS

Column Name	DEPTNO	DNAME	LOC
Key Type	PK		
Nulls/ Unique	NN/U	NN	NN
Sample Data	10	ACCOUNTING	NEWYORK
	20	RESEARCH	DALLAS

Map Relationships

Finally, we will map each relationship to a foreign key (FK) column.

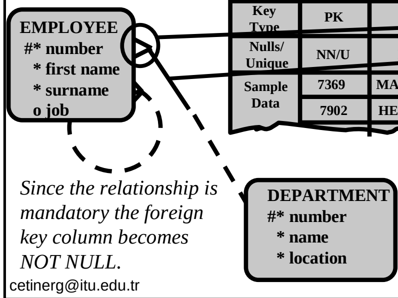
E-R MODEL	RELATIONAL DATABASE DESIGN
Entity	Table
Attribute	Column
Unique Identifier	Primary Key
Relationship	Foreign Key

Mapping (Transforming) Data Model into Database Design

For the relationship between EMPLOYEES and DEPARTMENTS, we will add the foreign key column DEPTNO to the EMPLOYEES Table, and label it FK1. This is the first foreign key column in the EMPLOYEES Table.

Table Name: EMPLOYEES

Column Name	EMPNO	FNAME	LNAME	JOB	DEPTNO
Key Type	PK				FK1
Nulls/ Unique	NN/U	NN	NN		NN
Sample Data	7369	MARY	SMITH		
	7902	HENRY	FORD	ANALYST	



Since the relationship is mandatory the foreign key column becomes NOT NULL.

Foreign Key column is added to the table of entity having crow-foot.

Mapping (Transforming) Data Model into Database Design

The foreign key DEPTNO column will allow the DEPARTMENT data for each Employee to be accessed.

For example, Mary Smith is assigned to DEPTNO=20 which is the RESEARCH department in Dallas.

Table Name: EMPLOYEES

Column Name	EMPNO	FNAME	LNAME	JOB	DEPTNO
Key Type	PK				FK1
Nulls/ Unique	NN/U	NN	NN		NN
Sample Data	7369	MARY	SMITH		20
	7902	HENRY	FORD	ANALYST	50

Table Name:DEPARTMENTS

Column Name	DEPTNO	DNAME	LOC
Key Type			
Nulls/ Unique	NN/U	NN	NN
Sample Data	10	ACCOUNTING	NEWYORK
	20	RESEARCH	DALLAS

Mapping (Transforming) Data Model into Database Design

The recursive relationship of the EMPLOYEE entity will be mapped to a second foreign key column in the EMPLOYEES Table. We will call this column MGR and it will contain the employee number for the employee's manager.

Table Name: EMPLOYEES

Column Name	EMPNO	FNAME	LNAME	JOB	DEPTNO	MGR
Key Type	PK				FK1	FK2
Nulls/ Unique	NN/U	NN	NN		NN	
Sample Data	7369	MARY	SMITH		20	7902
	7902	HENRY	FORD	ANALYST	50	7566

Mapping (Transforming) Data Model into Database Design

For example, Mary Smith's manager is employee number 7902-Henry Ford.

Table Name: EMPLOYEES

Column Name	EMPNO	FNAME	LNAME	JOB	DEPTNO	MGR
Key Type	PK				FK1	FK2
Nulls/ Unique	NN/U	NN	NN		NN	
Sample Data	7369	MARY	SMITH		20	7902
	7902	HENRY	FORD	ANALYST	50	7566

Mapping (Transforming) Data Model into Database Design

Now we have a complete relational database design for the EMPLOYEES and DEPARTMENTS Tables.

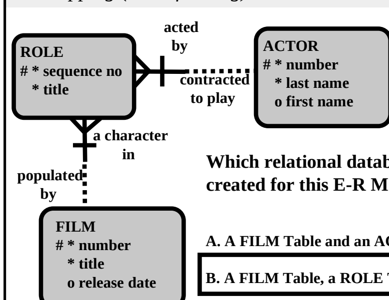
Table Name:DEPARTMENTS

Column Name	DEPTNO	DNAME	LOC
Key Type			
Nulls/ Unique	NN/U	NN	NN
Sample Data	10	ACCOUNTING	NEWYORK
	20	RESEARCH	DALLAS

Table Name: EMPLOYEES

Column Name	EMPNO	FNAME	LNAME	JOB	DEPTNO	MGR
Key Type	PK				FK1	FK2
Nulls/ Unique	NN/U	NN	NN		NN	
Sample Data	7369	MARY	SMITH		20	7902
	7902	HENRY	FORD	ANALYST	50	7566

Mapping (Transforming) Data Model into Database Design



Which relational database tables will be created for this E-R Model?

A. A FILM Table and an ACTOR Table

B. A FILM Table, a ROLE Table and an ACTOR Table

C. A FILM Table, an ACTOR and a CHARACTER Table

D. A FILM/ROLE Table

Mapping (Transforming) Data Model into Database Design

Which column(s) of the ACTOR Table will be the primary key?

ACTOR
 # * number
*** last name**
o first name

ACTOR

ACTOR_NUM	LAST_NAME	FIRST_NAME
1557	WAYNE	JOHN
499	HURT	WILLIAM
2407	TURNER	KATHLEEN

A. FIRST_NAME, LAST_NAME
B. FIRST_NAME
C. ACTOR_NUM
D. ACTOR_NUM,LAST_NAME

Mapping (Transforming) Data Model into Database Design

Transforming Data Scales

Data Scales are transformed into suitable data types by the analyst by asking the users, considering the necessary reports.

Mapping (Transforming) Data Model into Database Design

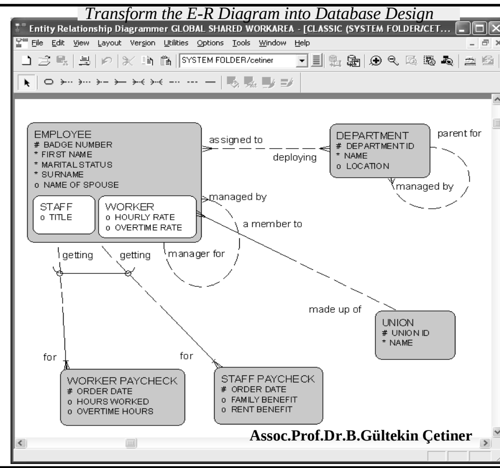
Topic Summary

- **Each part of an E-R Diagram translates directly into a part of a relational database design:**

entity	→	table
attribute	→	column
unique identifier	→	primary key
relationship	→	foreign key

- **Attributes labeled "*" for mandatory will be marked "NN" for NOT NULL in the table design.**
- **The UID is translated to the primary key column and marked "PK" for Primary Key and "U" for Unique**
- **Relationships are translated to foreign key columns and marked "FK" for Foreign Key.**

Transform the E-R Diagram into Database Design



Assoc.Prof.Dr.B.Gültekin Çetiner

Understand Further Database Design

Further on Database Design

Topic Objectives

This topic discusses what happens to the E R Diagram after it has been turned into a physical database design. When you finish this topic, you will be able to:

- **explain the need for Referential Constraints**
- **list the indexes required in a physical database**
- **define views in a physical database**
- **explain database denormalization**

Further on Database Design

After mapping an E-R Diagram into a basic relational database design, the next step is to implement the design.

Table Name: EMPLOYEES

Column Name	EMPNO	FNAME	LNAME	JOB	DEPTNO	MGR
Key Type	PK				FK1	FK2
Nulls/ Unique	NN/U	NN	NN		NN	
Sample Data	7369	MARY	SMITH		20	7902
	7902	HENRY	FORD	ANALYST	50	7566

Further on Database Design

Before you actually implement your design, you should consider a few additional aspects of relational database design:

- **Referential Integrity**
- **Indexing**
- **Views**
- **Denormalization**

Table Name: EMPLOYEES

Column Name	EMPNO	FNAME	LNAME	JOB	DEPTNO	MGR
Key Type	PK				FK1	FK2
Nulls/ Unique	NN/U	NN	NN		NN	
Sample Data	7369	MARY	SMITH		20	7902
	7902	HENRY	FORD	ANALYST	50	7566

Further on Database Design

Referential Integrity

Referential integrity addresses database “consistency” ensuring that the values in the various tables of the database are consistent. Referential integrity deals specifically with the consistency of foreign keys.

Further on Database Design

The DEPTNO column in the EMPLOYEE Table is a foreign key column and references the primary key of the DEPARTMENT Table. The business rule states:

“Each EMPLOYEE must be assigned to one and only one DEPARTMENT”.

A DEPTNO value is valid only if it references a valid DEPTNO in the DEPARTMENT Table.

Further on Database Design

You cannot enter a DEPTNO value in EMPLOYEES Table if that is not available in DEPARTMENTS Table.

Table Name: DEPARTMENTS

Reference Column	DEPTNO	DNAME	LOC
	NN/U	NN	NN
	10	ACCOUNTING	NEWYORK
	20	RESEARCH	DALLAS

Table Name: EMPLOYEES

Column Name	EMPNO	FNAME	LNAME	JOB	DEPTNO	MGR
Key Type	PK				FK1	FK2
Nulls/ Unique	NN/U	NN	NN		NN	
Sample Data	7369	MARY	SMITH		20	7902
	7902	HENRY	FORD	ANALYST	50	7566

Further on Database Design

Validated from the column DEPTNO in DEPARTMENTS Table

Validated from the column EMPNO in EMPLOYEES Table

Table Name: EMPLOYEES

Column Name	EMPNO	FNAME	LNAME	JOB	DEPTNO	MGR
Key Type	PK				FK1	FK2
Nulls/ Unique	NN/U	NN	NN		NN	
Sample Data	7369	MARY	SMITH		20	7902
	7902	HENRY	FORD	ANALYST	50	7566

MGR is foreign key and comes from the recursive relation within the EMPLOYEE entity.

So what happens if a DEPTNO for which employees work is deleted from the DEPARTMENT Table?

For example, what would happen if the row for DEPTNO=20 is deleted from the DEPARTMENT Table?

Table Name:DEPARTMENTS

Column Name	DEPTNO	DNAME	LOC
Key Type			
Nulls/ Unique	NN/U	NN	NN
Sample Data	10	ACCOUNTING	NEWYORK

?

Table Name: EMPLOYEES

Column Name	EMPNO	FNAME	LNAME	JOB	DEPTNO	MGR
Key Type	PK				FK1	FK2
Nulls/ Unique	NN/U	NN	NN		NN	
Sample Data	7369	MARY	SMITH		20	7902
	7902	HENRY	FORD	ANALYST	50	7566

What happens depends on what referential integrity rule was specified for the FK DEPTNO in the EMPLOYEES Table. The database designer or DBA should specify a referential integrity rule for every foreign key in the database.

Table Name: EMPLOYEES

Column Name	EMPNO	FNAME	LNAME	JOB	DEPTNO	MGR
Key Type	PK				FK1	FK2
Nulls/ Unique	NN/U	NN	NN		NN	
Sample Data	7369	MARY	SMITH		20	7902
	7902	HENRY	FORD	ANALYST	50	7566

The database designer can specify one of three options:

Delete restricted, which restricts the deletion of certain rows in the table.

Delete cascade, which deletes the corresponding rows of the associated table.

Delete nullify, which places null values in the corresponding rows of the associated table.

If the FK DEPTNO was defined as "Delete Restricted" then the RDBMS would restrict the deletion of DEPARTMENTS to only those rows which have no EMPLOYEES.

Table Name:DEPARTMENTS

Column Name	DEPTNO	DNAME	LOC
Key Type			
Nulls/ Unique	NN/U	NN	NN
Sample Data	10	ACCOUNTING	NEWYORK
	20	RESEARCH	DALLAS

For example, department 20 could not be deleted because an EMPLOYEE record is assigned to department 20.

Table Name: EMPLOYEES

Column Name	EMPNO	FNAME	LNAME	JOB	DEPTNO
Key Type	PK				FK1
Nulls/ Unique	NN/U	NN	NN		NN
Sample Data	7369	MARY	SMITH		20
	7902	HENRY	FORD	ANALYST	50

Delete Restricted

If the FK DEPTNO was defined as "Delete Cascade" then the RDBMS would cascade the deletion of a DEPARTMENT to the EMPLOYEE Table and would delete all EMPLOYEEs assigned to that DEPARTMENT.

Table Name:DEPARTMENTS

Column Name	DEPTNO	DNAME	LOC
Key Type			
Nulls/ Unique	NN/U	NN	NN
Sample Data	10	ACCOUNTING	NEWYORK

For example, if the Research Department 20 was deleted, Mary Smith and other EMPLOYEEs who work in the same DEPARTMENT would also be deleted.

Table Name: EMPLOYEES

Column Name	EMPNO	FNAME	LNAME	JOB	DEPTNO
Key Type	PK				FK1
Nulls/ Unique	NN/U	NN	NN		NN
Sample Data					
	7902	HENRY	FORD	ANALYST	50

Delete Cascade

If employee assignment to a DEPARTMENT was optional and the FK DEPTNO was defined as "Delete Nullify" the RDBMS would nullify any references to a department when that department was deleted.

Table Name:DEPARTMENTS

Column Name	DEPTNO	DNAME	LOC
Key Type			
Nulls/ Unique	NN/U	NN	NN
Sample Data	10	ACCOUNTING	NEWYORK

Delete Nullify

Table Name: EMPLOYEES

Column Name	EMPNO	FNAME	LNAME	JOB	DEPTNO
Key Type	PK				FK1
Nulls/ Unique	NN/U	NN	NN		NN
Sample Data	7369	MARY	SMITH		
	7902	HENRY	FORD	ANALYST	50

Referential integrity rules should be assigned to all foreign keys in a database

You can assign one of the three rules for each foreign key.

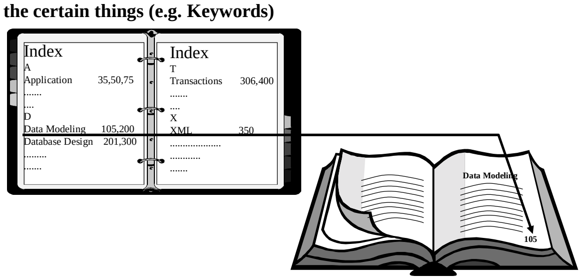
Indexing

Once the referential integrity constraints have been identified, it is time to decide on the indexing of the tables.

Indexes provide the DBMS with a quick way of looking up the location of rows, rather than sequentially scanning the table to satisfy every request.

Indexing

Indexing is like using indexes on a book to find certain items by looking at the certain things (e.g. Keywords)



Most DBMSs require that primary key be indexed, in order to enforce the requirement that a primary key must be unique. Most DBMSs do not require that foreign keys be indexed, but all of them normally would be indexed.

For example, for the EMPLOYEES Table, a unique index might be created on the PK column EMPNO and a non-unique index might be created on the FK column DEPTNO.

EMPLOYEE TABLE

ROWID	EMPNO	FNAME	LNAME	JOB	HIREDATE	MGR	DEPTNO
1011	7369	MARY	SMITH	CLERK	17-DEC-80	7902	20
1012	7902	HENRY	FORD	ANALYST	03-DEC-81	7566	50
1013	7521	SUE	WARD	SALESMAN	22-FEB-81	7698	30
1014	7698	BOB	BLAKE	MANAGER	01-MAY-81	7839	30
1015	7839	BOB	KING	PRESIDENT	17-NOV-81		10

EMP_INDEX_PRIME

EMPNO	ROWID
7369	1011
7521	1013
7698	1014
7839	1015
7902	1012

EMP_INDEX_FK

DEPTNO	ROWID
10	1015
20	1011
30	1013
30	1014
50	1012

In addition, we will index any column or combination that is frequently used as a search key (i.e., in an SQL WHERE clause), or as a sort key (i.e., an SQL ORDER BY, GROUP BY, or similar clause).

A table will usually have two to four indexes to support a typical mixture of transaction processing, batch jobs, and ad hoc query.

The greater the proportion of read-only processing, the more beneficial indexes will be.

The greater the proportion of update processing, the less beneficial indexes will be. For example, when a row is inserted into a table, the proper keys must be inserted into each index. If a table had 12 indexes, the insertion of a new row would be the equivalent of updating 13 files.

Views

A view is like a “window” onto a table- a window which can reveal only certain columns and/or rows, or can change the appearance of the data. A view of the EMPLOYEES Table could be used to restrict users from seeing employees’ salaries.

Base Table (EMPLOYEES)

EMPNO	LNAME	MGR	SAL
101	JONES	50	55010
50	SMITH		[illegible]
210	BROWN	50	[illegible]
443	GONZALES	101	25250
501	JOHNSON	210	35250

VIEW

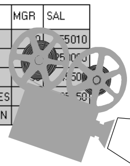
EMPNO	LNAME	MGR
101	JONES	50
50	SMITH	
210	BROWN	50
443	GONZALES	101
501	JOHNSON	210

Views

A view is also like a projector which reflects the base table onto a surface with a required form.

Base Table (EMPLOYEES)

EMPNO	LNAME	MGR	SAL
101	JONES	[illegible]	[illegible]
50	SMITH	[illegible]	[illegible]
210	BROWN	[illegible]	[illegible]
443	GONZALES	[illegible]	[illegible]
501	JOHNSON	[illegible]	[illegible]



VIEW

EMPNO	LNAME	MGR
101	JONES	50
50	SMITH	
210	BROWN	50
443	GONZALES	101
501	JOHNSON	210

A view can also be used to present normalized data in a denormalized form. For example, following the rules of normalization, the EMPLOYEE and DEPARTMENT Tables are separate.

Table Name: ***EMPLOYEES***

EMPNO	FNAME	LNAME	JOB	HIREDATE	MGR	DEPTNO
7369	MARY	SMITH	CLERK	17-DEC-80	7902	20
7902	HENRY	FORD	ANALYST	03-DEC-81	7566	50
7521	SUE	WARD	SALESMAN	22-FEB-81	7698	30
7698	BOB	BLAKE	MANAGER	01-MAY-81	7839	30
7839	BOB	KING	PRESIDENT	17-NOV-81		10

Table Name: ***DEPARTMENTS***

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	DEVELOPMENT	ATLANTA

A view defined accross both tables could be used to prejoin the tables so the user would only see a single "logical tale".

VIEW_EMPLOYEES

EMPNO	FNAME	LNAME	JOB	HIREDATE	MGR	DEPTNO	DNAME	LOC
7369	MARY	SMITH	CLERK	17-DEC-80	7902	20	RESEARCH	DALLAS
7902	HENRY	FORD	ANALYST	03-DEC-81	7566	50	DEVELOPMENT	ATLANTA
7521	SUE	WARD	SALESMAN	22-FEB-81	7698	30	SALES	CHICAGO
7698	BOB	BLAKE	MANAGER	01-MAY-81	7839	30	SALES	CHICAGO
7839	BOB	KING	PRESIDENT	17-NOV-81		10	ACCOUNTING	NEW YORK

The view is simply a SQL SELECT statement stored in the DBMS's catalog. Any SELECT statement, with a few exceptions imposed by the various DBMSs, can serve as the definition of a view.

A table may have any number of views associated with it, or none. Some DBAs would prefer to use views sparingly; views take time to define, take space in the catalog, and impose some catalog-processing overhead when SQL statements which reference them are being syntax-checked.

Other DBAs are much more willing to use views, but may want to examine each proposed new view to be sure that it will actually save time or money. Be prepared to explain to the DBA why you or the user feel that a view would be beneficial.

The most common benefits of using views include:

- **Enhancing Security: can be obtained by "hiding" sensitive columns from users; for example, the salary column of an employee table could be hidden from users who only need to process address changes.**
- **Storing complex SQL statements; in the catalog, rather than forcing each user to prepare the SQL; for example, summarization, complex multiple-table joins, subqueries, etc.**

The most common benefits of using views include:

- **Providing different column names:**
  Providing different column names to the user than those actually defined in the table; for example, the installation naming standards might dictate that a column name be eight characters or less, producing "empname" while the users might be happier with "employee_name".
- **Permitting a DBA to make certain kind of changes:**
  Without affecting application programs and queries, a DBA may make changes to the table layouts. A DBA might decide to combine an ORDER and ORDER ITEM table; if the programs have been accessing the ORDER table via a view, the DBA may be able to change the definition of vew such that when the table layout is changed the programs are unaware the changes.

DeNormalization

Data normalization minimizes data redundancy. Unnormalized data is redundant and prone to the data integrity problems.

An E-R Model normalized to Third Normal Form will automatically map to a relational database design in Third Normal Form.

DeNormalization

Denormalizing tables can sometimes improve database performance for large databases with demanding on-line transactions. Always start with tables in Third Normal Form,

and
be extremely reluctant to denormalize.

Consider all other performance tuning options before deciding to denormalize. Adding and changing indexes usually has the biggest impact on database performance.

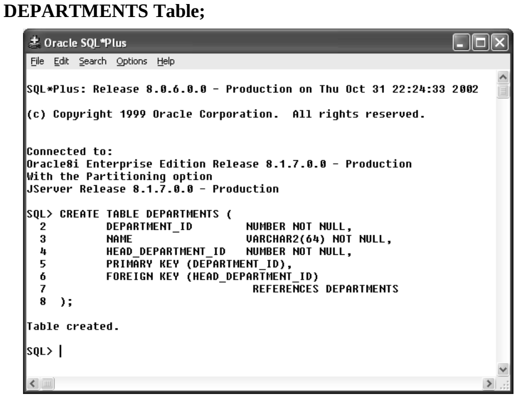
Denormalization can quickly lead to inconsistency problems in the redundant data. A denormalized database design will not support the business rules defined in the E-R Model.

Before Denormalization try all alternatives such as indexing.

Database Build

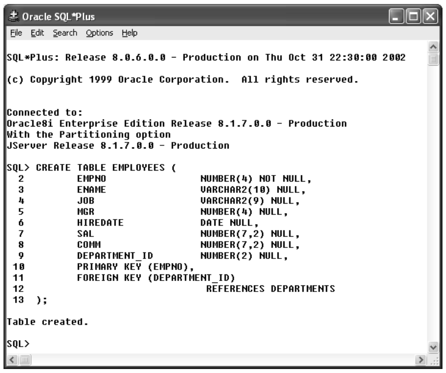
After addressing referential integrity, designing indexes, and creating views, you are now ready to create real RDBMS tables using the SQL CREATE statement. For example, to create the physical tables for the DEPARTMENTS and EMPLOYEES tables, you must use the following SQL statements.

For DEPARTMENTS Table;



Further on Database Design

For EMPLOYEES Table;



```
Oracle SQL*Plus
File  Edit  Search  Options  Help

SQL*Plus: Release 8.0.6.0.0 - Production on Thu Oct 31 22:30:00 2002

(c) Copyright 1999 Oracle Corporation.  All rights reserved.


Connected to:
Oracle8i Enterprise Edition Release 8.1.7.0.0 - Production
With the Partitioning option
JServer Release 8.1.7.0.0 - Production

SQL> CREATE TABLE EMPLOYEES (
  2         EMPNO                NUMBER(4) NOT NULL,
  3         ENAME                VARCHAR2(10) NULL,
  4         JOB                  VARCHAR2(9) NULL,
  5         MGR                  NUMBER(4) NULL,
  6         HIREDATE             DATE NULL,
  7         SAL                  NUMBER(7,2) NULL,
  8         COMM                 NUMBER(7,2) NULL,
  9         DEPARTMENT_ID        NUMBER(2) NULL,
 10         PRIMARY KEY (EMPNO),
 11         FOREIGN KEY (DEPARTMENT_ID)
 12                               REFERENCES DEPARTMENTS
 13  );

Table created.

SQL>
```


Further on Database Design

Topic Summary

- **Referential integrity ensures that the data values in the various tables of the database are accurate, specifically in respect with foreign key values**
- **There are three referential integrity options: delete restricted, delete cascade, delete nullify**
- **Indexes provide the DBMS wth a quick way of looking up the location of rows, rather than sequentially scanning the table to satisfy every request**
- **Most DBMSs require that any primary key be indexed, in order to enforce uniqueness**

Further on Database Design

Topic Summary

- **Most DBMSs do not require that foreign keys be indexed, but all of them normally would be indexed**
- **A view is like a window onto a table-It can reveal certain columns and/or rows with different appearance**
- **Data normalization minimizes data redundancy; unnormalized data is redundant and prone to data integrity problems**
- **Always start with tables in Third Normal Form, and be extremely reluctant to denormalize**