



Chapter 3: Relational Database Design

**Adama Science and Technology University
School of Electrical Engineering and Computing
Department of CSE
CSEg 2208: Database Systems
(2022)**

Outline

- ❖ Mapping conceptual schema to a relational schema
- ❖ Integrity constraints
- ❖ Relational algebra and relational calculus
- ❖ Functional dependency
- ❖ Normalization and Normal Forms

Relational Model: Logical Design

- ❖ What is the Relational Model:
 - ❖ Relations,
 - ❖ Domain Constraints.
- ❖ SQL
- ❖ Integrity Constraints
- ❖ Translating an ER diagram to the Relational Model and SQL

Relational Databases

- ❖ A relational database consists of a *collection of tables*:
 - ❖ Each table has a *unique name*,
 - ❖ Each row represents a uidgle entity.
- ❖ A table corresponds to the *mathematical concept* of a *relation*:
 - ❖ *Relations* corresponds to *tables*,
 - ❖ *Tuples* corresponds to *rows*.

Relations

- ❖ The key construct in the *relational model* is a relation:
 - ❖ A DB is a collection of *relations*.
- ❖ Relations consist of a *relation instance* and a *relation schema*:
 - ❖ A *relation instance* corresponds to an actual table with *a set of rows*.
 - ❖ A *relation schema* consists of the *column headings* of the table.

Relation Schema

- ❖ A *schema* describes:
 - ❖ The *relation's name*,
 - ❖ The *name of each column* in the table, also known as a *field*, or *attribute*,
 - ❖ The domain of *each field*.
 - ❖ Each domain has *a set of associated values* like a type.
- ❖ For example, the *Customer relation*:
 - ❖ Customer = (CID, fname, lname, age, income),
 - ❖ The schema should also specify the *domain of each attribute*,

Relation Schema

- ❖ A *domain of a field* is similar to the *type of a variable*.
- ❖ The *values* that can appear in a field are restricted *to the field's domain*.

Relation schema: Customer = (cid, fname, lname, age, salary)

CID	fname	lname	age	salary
111	Lemma	Abebe	23	43000.00
222	zeleke	Tesema	22	6764.87
333	Tesema	kemal	47	71098.65
444	kider	mume	17	4033.32

Relation instance

Each field (attribute) has a domain: char(11), char(20), char(20), integer, real

Relation Instance

- ❖ A relation instance is a *set of tuples*, or *records*.
- ❖ Each tuple has the *same number of fields* as the schema.
- ❖ No two rows (tuples) are *identical*:
 - ❖ As each relation is defined to be a set of unique tuples or rows.
 - ❖ A set is a collection of *distinct values*.
 - ❖ In practice DBMS allow tables to have *duplicate rows* in some circumstances.
- ❖ The *order of the rows* is not important.

More Terminology

- ❖ The *degree* (or arity) of a relation is the *number of fields* (or columns)
- ❖ The *degree* of the Customer relation is 5.
- ❖ The *cardinality* of a relation instance is the number of tuples in it
 - ❖ The *cardinality* of the Customer relation instance is 4.
- ❖ A *relational database* is a collection of relations with distinct *relation names*.
- ❖ A *relational database schema* is the collection of schemas for all relations in the database.

SQL DDL: Creating Tables in SQL

- ❖ SQL stands for *Structured Query Language*.
- ❖ SQL is divided into *two parts*:
 - ❖ Data Manipulation Language (DML) which allows users to *create*, *modify* and *query data*.
 - ❖ Data Definition Language (DDL) which is used to define *external* and *conceptual schemas*.
- ❖ The DDL supports the *creation*, *deletion* and *modification* of tables.
 - ❖ Including the *specification of domain constraints* and other constraints.

Creating Tables

- ❖ To create a table use the *CREATE TABLE* statement.
- ❖ Specify the *table name*, *field names* and *domains*.
- ❖ For example, to create the Customer table:

```
CREATE TABLE Customer (  
    cid          CHAR(11),  
    fname       CHAR(20),  
    lname       CHAR(20),  
    age         INTEGER,  
    income      REAL)
```

Inserting Records

- ❖ To insert a record into an existing table use the *INSERT statement*:
- ❖ The list of *column names* is optional.
- ❖ If omitted the values must be in the *same order as the columns*.

```
INSERT  
INTO Customer(cid, fname, lname, age, income)  
VALUES ('111', 'Sami', 'abebe', 23, 65234)
```

Deleting Records

- ❖ To delete a record use the *DELETE statement*:
 - ❖ The WHERE clause *specifies the record(s)* to be *deleted*.

```
DELETE  
FROM Customer  
WHERE cid = '111'
```

- ❖ Be careful, the following SQL query *deletes all the records* in a table.

```
DELETE  
FROM Customer
```


Modifying Records

- ❖ Use the *UPDATE statement* to modify a record, or records, in a table.
- ❖ Note that the *WHERE statement* is evaluated before the *SET statement*.
- ❖ An *UPDATE statement* may affect more than one record.

```
UPDATE Customer  
SET age = 37  
WHERE cid = '111'
```

Deleting Tables

- ❖ To delete a table use the *DROP TABLE statement*.
 - ❖ This not only deletes all of the records but also deletes *the table schema*.

```
DROP TABLE Customer
```

Modifying Tables

- ❖ Columns can be added or removed to tables using the *ALTER TABLE* statement:
 - ❖ *ADD* to add a column and,
 - ❖ *DROP* to remove a column.

```
ALTER TABLE Customer  
ADD height INTEGER
```

```
ALTER TABLE Customer  
DROP height
```

Integrity Constraints

- ❖ An *integrity constraint restricts* the data that can be stored in a DB.
 - ❖ To *prevent invalid data* being added to the DB.
 - ❖ E.g. two people with the same ID or
 - ❖ Someone with a negative age.
- ❖ When a *DB schema is defined*, the associated integrity constraints should also be specified.
- ❖ A DBMS checks *every update to ensure* that it does not violate any *integrity constraints*.

Types of Integrity Constraints

- ❖ Domain Constraints:

- ❖ Specified when tables are created by selecting the *type of the data*.
- ❖ E.g. age INTEGER

- ❖ Key Constraints:

- ❖ Identifies *primary keys* and other *candidate keys*.

- ❖ Foreign Key Constraints:

- ❖ References *primary keys* of other tables.

Key Constraints

- ❖ A key constraint states that a *minimal subset of the fields* in a relation is unique.
 - ❖ I.e. a candidate key
- ❖ SQL allows two sorts of *key constraints*.
- ❖ The *UNIQUE statement* identifies candidate keys.
 - ❖ Records may not have duplicate values for the set of attributes specified in the UNIQUE statement.
- ❖ A *PRIMARY KEY* identifies the primary key:
 - ❖ Records may not have *duplicate primary keys* and,
 - ❖ May not have *null values for primary key* attributes.

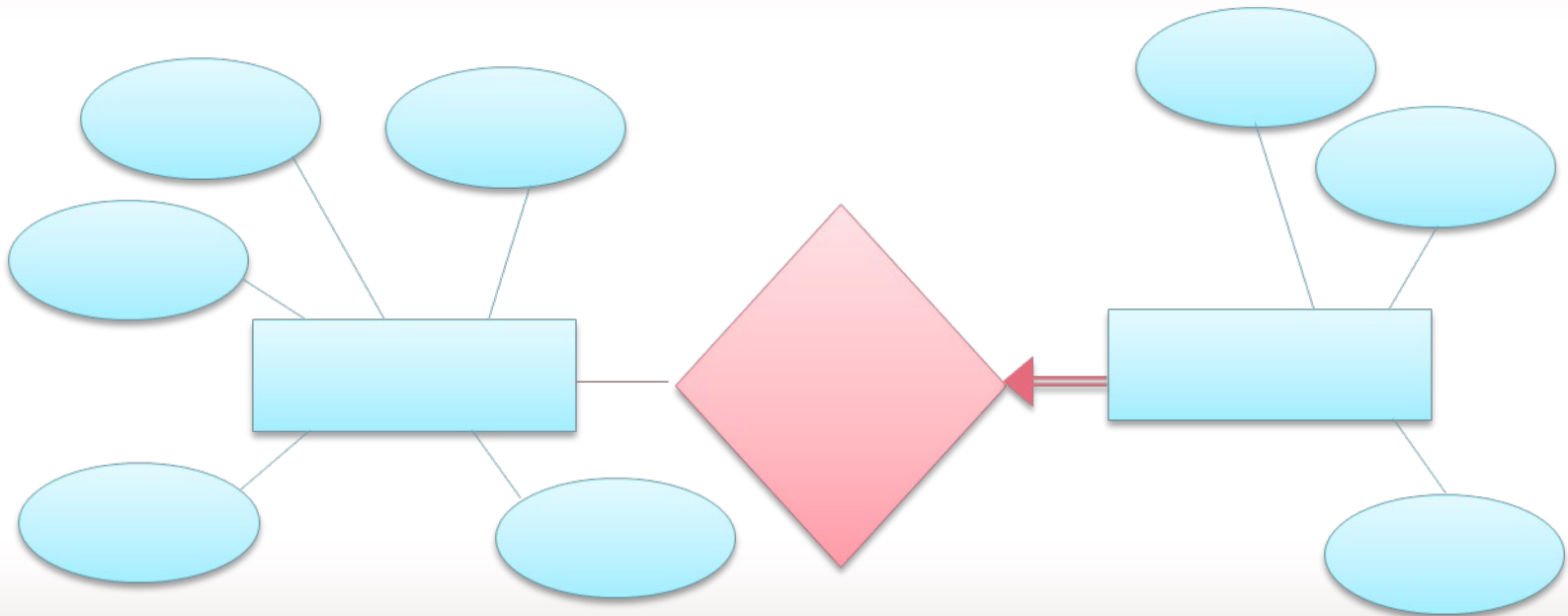
Identifying Key Constraints

- ❖ Assume that a patient can be *uniquely identified* by either PID.
- ❖ Pid is chosen as the *primary key*.

```
CREATE TABLE Patient (  
    PID          CHAR(11),  
    fName       CHAR(20),  
    lName       CHAR(20),  
    age         INTEGER,  
    PRIMARY KEY (cid) )
```

Converting ERDs to a DB

- ❖ Assume that a patient can be *uniquely identified* by either PID.
- ❖ Pid is chosen as the *primary key*.



Logical Database Design

- ❖ The ER model is an initial *high level design* which can be translated into a *relational schema*.
 - ❖ And therefore into SQL.
- ❖ Each entity and relationship set can be represented by a *unique relation*:
 - ❖ Although some relationship sets do not need to be represented by separate relations.
 - ❖ SQL constraints should be included in DB tables to represent constraints identified in the ERD.

Three basic rules to convert ER into tables or relations:

- ❖ Rule 1: Entity Names will automatically be table names.
- ❖ Rule 2: Mapping of attributes: attributes will be columns of the respective tables.
 - ❖ *Atomic or single-valued* or derived or stored attributes will be columns
 - ❖ *Composite attributes*: the parent attribute will be ignored and the decomposed attributes (child attributes) will be columns of the table.
 - ❖ *Multi-valued attributes*: will be mapped to a *new table* where the primary key of the main table will be posted for cross referencing.

Three basic rules to convert ER into tables or relations:

- ❖ Rule 3: Relationships: relationship will be mapped by using a foreign key attribute. Foreign key is a primary or candidate key of one relation used to create association between tables.
- ❖ One-to-One Cardinality: post the primary or candidate key of one of the table into the other as a foreign key.
- ❖ One-to-Many Cardinality: Post the primary key or candidate key from the “one” side as a foreign key attribute to the “many” side.
 - ❖ E.g.: For a relationship called “Belongs To” between Employee (Many) and Department (One) the primary or candidate key of the one side which is Department should be posted to the many side which is Employee table.

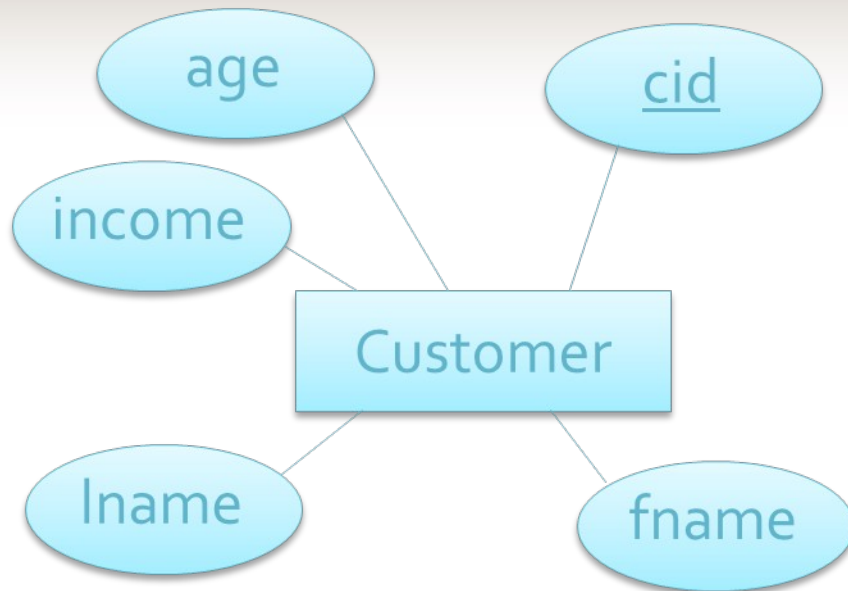
Three basic rules to convert ER into tables or relations:

- ❖ Many-to-Many Cardinality: for relationships having many to many cardinality, one has to create a new table (which is the associative entity) and post primary key or candidate key from the participant entities as foreign key attributes in the new table along with some additional attributes (if applicable).

Translating Entity Sets.....

- ❖ A table that represents a (strong) entity set should have the following characteristics.
 - ❖ One column for each attribute,
 - ❖ Each row represents a unique entity,
 - ❖ The domain of each attribute should be known and specified in the table,
 - ❖ The primary key should be specified in the table,
 - ❖ Other constraints that have been identified outside the ER model should also be created where possible,

Entity Sets



```
CREATE TABLE Customer (  
  cid      CHAR(11),  
  fname    CHAR(20),  
  lname    CHAR(20),  
  age      INTEGER,  
  income   REAL,  
  PRIMARY KEY (cid) )
```

Relationship Sets

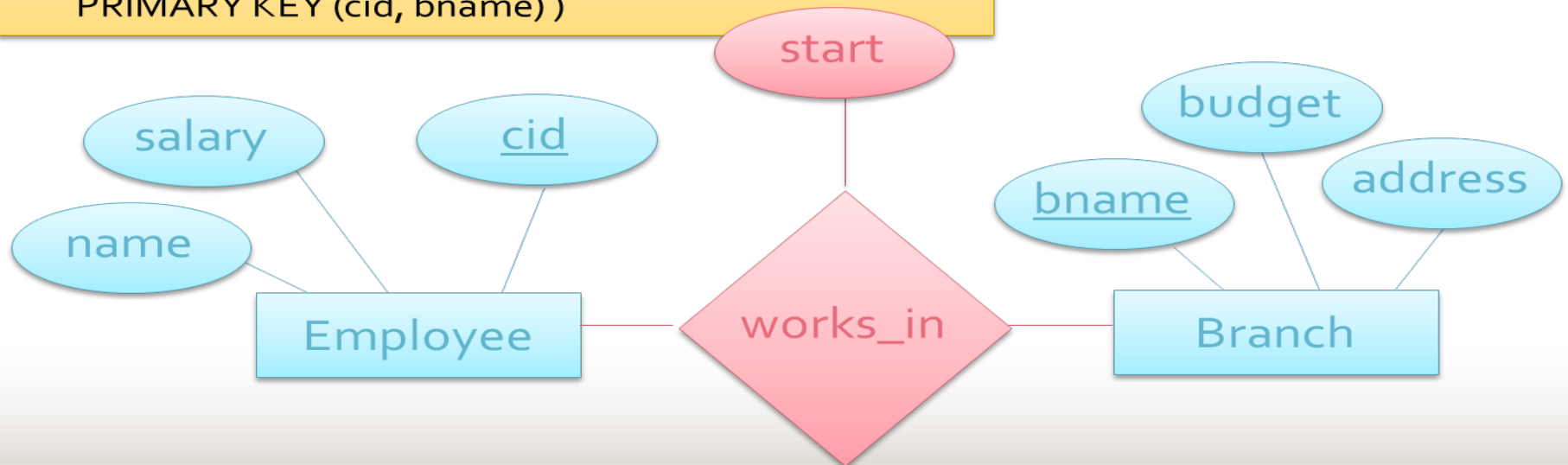
- ❖ The attributes of a relationship set are:
 - ❖ The primary keys of the participating entity sets, and
 - ❖ Any descriptive attributes of the relationship set.
- ❖ The mapping cardinalities of the entities involved in a relationship determine:
 - ❖ The primary key of the relationship set and,
 - ❖ Whether or not the relationship set needs to be represented as a separate table.
- ❖ Foreign keys should be created for the attributes derived from the participating entity sets.

Relationship Sets With No Cardinality Constraints

- ❖ A separate table is required to represent a relationship set with no cardinality constraints.
 - ❖ I.e. relationships that are many to many.
- ❖ The primary key of the relationship set is the union of the attributes derived from its entity sets.
 - ❖ A compound key made up of the primary keys of the participating entity sets.
- ❖ Attributes derived from its entity sets should be declared as foreign keys.

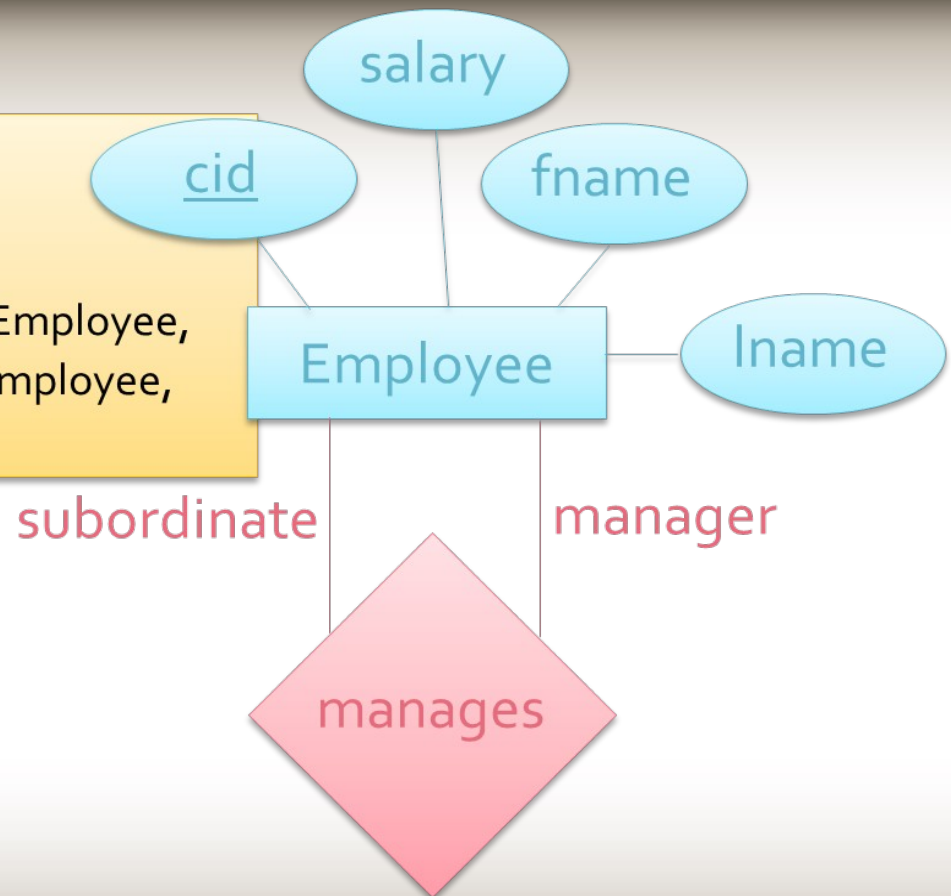
Relationship Set to Table

```
CREATE TABLE Workcid (  
  cid          CHAR(11),  
  bname        CHAR(30),  
  start        DATETIME,  
  FOREIGN KEY (cid) REFERENCES Employee,  
  FOREIGN KEY (bname) REFERENCES Branch,  
  PRIMARY KEY (cid, bname) )
```



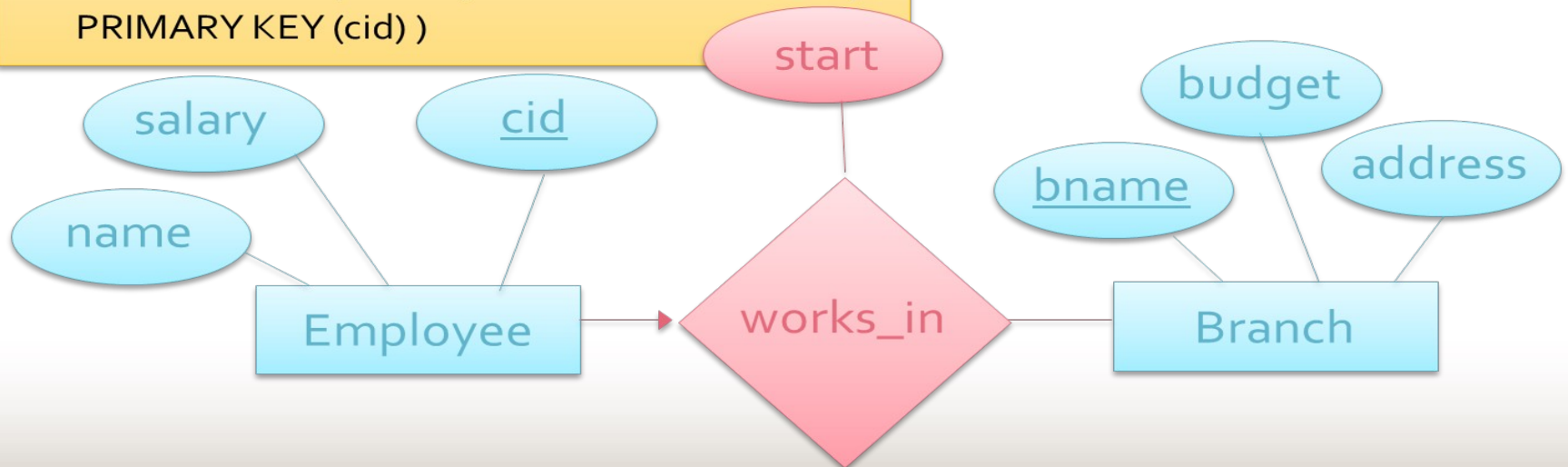
Relationship Set to Table ...

```
CREATE TABLE Manages (  
  mancid      CHAR(11),  
  subcid      CHAR(11),  
  FOREIGN KEY (mancid) REFERENCES Employee,  
  FOREIGN KEY (subcid) REFERENCES Employee,  
  PRIMARY KEY (mancid, subcid) )
```

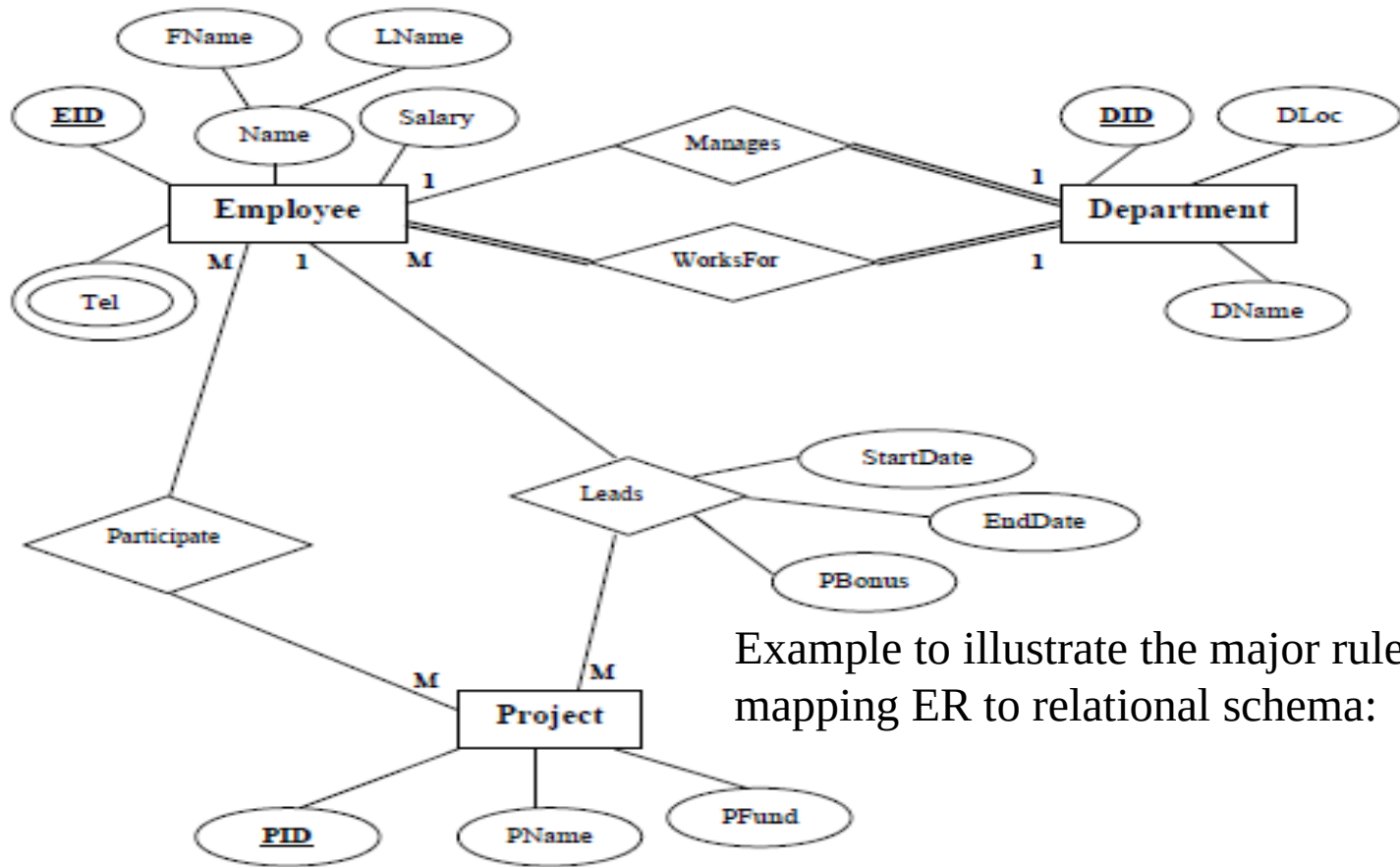


Many-to-One Relationship

```
CREATE TABLE Employee (  
  cid          CHAR(11),  
  name         CHAR(40),  
  salary       REAL,  
  bname        CHAR(30),  
  start        DATETIME,  
  FOREIGN KEY (bname) REFERENCES Branch,  
  PRIMARY KEY (cid) )
```



Example 2: Map ER Diagram to Tables



Example to illustrate the major rules in mapping ER to relational schema:

Example 2: Map ER Diagram to Tables

- ✓ Mapping EMPLOYEE Entity:

There will be *Employee* table with *EID*, *Salary*, *FName* and *LName* being the columns. The composite attribute *Name* will be ignored as its decomposed attributes (*FName* and *LName*) are columns in the *Employee* Table. The *Tel* attribute will be a new table as it is multi-valued.

Employee

<u>EID</u>	FName	LName	Salary
------------	-------	-------	--------

Telephone

<u>EID</u>	Tel
------------	-----

- ✓ Mapping DEPARTMENT Entity:

There will be *Department* table with *DID*, *DName*, and *DLoc* being the columns.

Department

<u>DID</u>	DName	DLoc
------------	-------	------

- ✓ Mapping PROJECT Entity:

There will be *Project* table with *PID*, *PName*, and *PFund* being the columns.

Project

<u>PID</u>	PName	PFund
------------	-------	-------

- ✓ Mapping the MANAGES Relationship:

As the relationship is having one-to-one cardinality, the PK or CK of one of the table can be posted into the other. But based on the recommendation, the Pk or CK of the partial participant (*Employee*) should be posted to the total participants (*Department*). This will require adding the PK of *Employee* (*EID*) in the *Department* Table as a foreign key. We can give the foreign key another name which is *MEID* to mean "managers employee id". this will affect the degree of the *Department* table.

Department

<u>DID</u>	DName	DLoc	MEID
------------	-------	------	------

- ✓ Mapping the WORKSFOR Relationship:

As the relationship is having one-to-many cardinality, the PK or CK of the "One" side (PK or CK of *Department* table) should be posted to the many side (*Employee* table). This will require adding the PK of *Department* (*DID*) in the *Employee* Table as a foreign key. We can give the foreign key another name which is *EDID* to mean "Employee's Department id". this will affect the degree of the *Employee* table.

Employee

<u>EID</u>	FName	LName	Salary	EDID
------------	-------	-------	--------	------

Example 2: Map ER Diagram to Tables

- ✓ Mapping EMPLOYEE Entity:

There will be *Employee* table with *EID*, *Salary*, *FName* and *LName* being the columns. The composite attribute Name will be ignored as its decomposed attributes (*FName* and *LName*) are columns in the *Employee* Table. The *Tel* attribute will be a new table as it is multi-valued.

Employee

<u>EID</u>	FName	LName	Salary
------------	-------	-------	--------

Telephone

<u>EID</u>	Tel
------------	-----

- ✓ Mapping DEPARTMENT Entity:

There will be *Department* table with *DID*, *DName*, and *DLoc* being the columns.

Department

<u>DID</u>	DName	DLoc
------------	-------	------

- ✓ Mapping PROJECT Entity:

There will be *Project* table with *PID*, *PName*, and *PFund* being the columns.

Project

<u>PID</u>	PName	PFund
------------	-------	-------

- ✓ Mapping the MANAGES Relationship:

As the relationship is having one-to-one cardinality, the PK or CK of one of the table can be posted into the other. But based on the recommendation, the PK or CK of the partial participant (*Employee*) should be posted to the total participants (*Department*). This will require adding the PK of *Employee* (*EID*) in the *Department* Table as a foreign key. We can give the foreign key another name which is *MEID* to mean "managers employee id". this will affect the degree of the *Department* table.

Department

<u>DID</u>	DName	DLoc	MEID
------------	-------	------	------

- ✓ Mapping the WORKSFOR Relationship:

As the relationship is having one-to-many cardinality, the PK or CK of the "One" side (PK or CK of *Department* table) should be posted to the many side (*Employee* table). This will require adding the PK of *Department* (*DID*) in the *Employee* Table as a foreign key. We can give the foreign key another name which is *EDID* to mean "Employee's Department id". this will affect the degree of the *Employee* table.

Employee

<u>EID</u>	FName	LName	Salary	EDID
------------	-------	-------	--------	------

Example 2: Map ER Diagram to Tables

✓ Mapping the PARTICIPATES Relationship:

As the relationship is having many-to-many cardinality, we need to create a new table and post the PK or CK of the Employee and Project table into the new table. We can give a descriptive new name for the new table like Emp_Partc_Project to mean "Employee participate in a project".

Emp_Partc_Project

<u>EID</u>	<u>PID</u>
------------	------------

✓ Mapping the LEADS Relationship:

As the relationship is associative entity, we are supposed to create a table for the associative entity where the PK of Employee and Project tables will be posted in the new table as a foreign key. The new table will have the attributes of the associative entity as columns. We can give a descriptive new name for the new table like Emp_Lead_Project to mean "Employee leads a project".

Emp_Lead_Project

<u>EID</u>	<u>PID</u>	PBonus	StartDate	EndDate
------------	------------	--------	-----------	---------

At the end of the mapping we will have the following relational schema (tables) for the logical database design phase.

Department

<u>DID</u>	DName	DLoc	MEID
------------	-------	------	------

Project

<u>PID</u>	PName	PFund
------------	-------	-------

Telephone

<u>EID</u>	Tel
------------	-----

Employee

<u>EID</u>	FName	LName	Salary	EDID
------------	-------	-------	--------	------

Emp_Partc_Project

<u>EID</u>	<u>PID</u>
------------	------------

Emp_Lead_Project

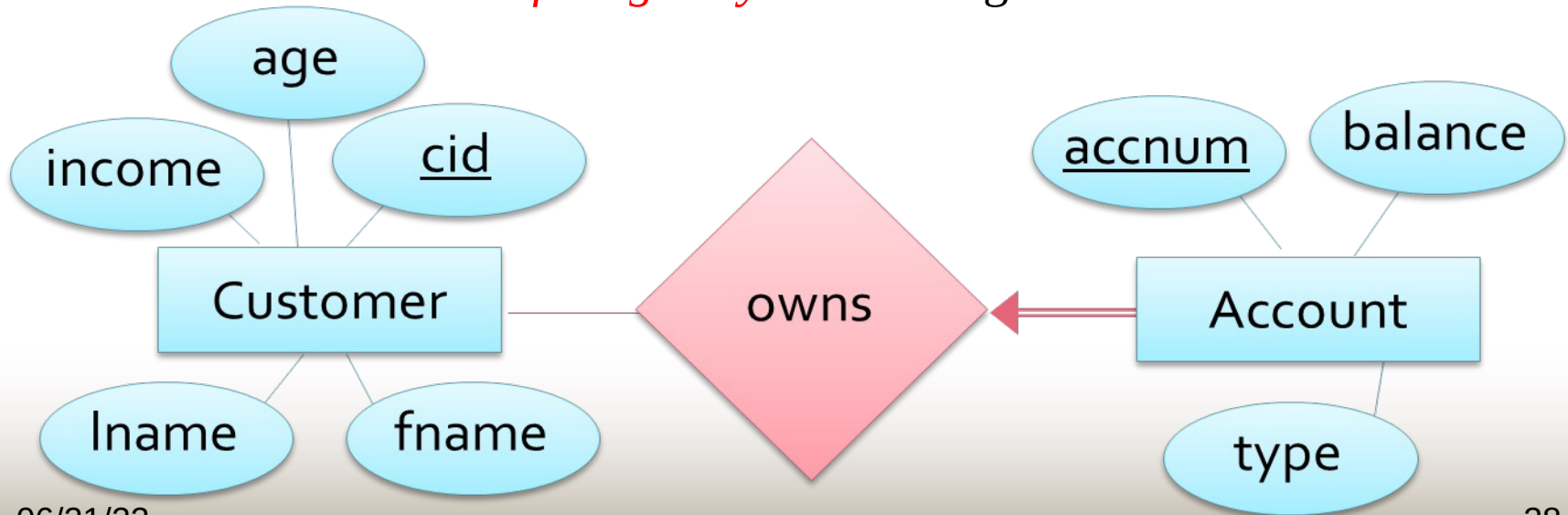
<u>EID</u>	<u>PID</u>	PBonus	StartDate	EndDate
------------	------------	--------	-----------	---------

Enforcing Integrity Constraints

- ❖ Whenever a table with a constraint is modified the constraint must be checked.
 - ❖ A modification can be a deletion, a change (update) or an insertion of a record.
 - ❖ If a transaction violates a constraint what happens?
- ❖ Primary key constraints:
 - ❖ A primary key constraint can be violated in two ways:
 - ❖ A record can be changed or inserted so that it *duplicates* the primary key of another record in the table or
 - ❖ A record can be changed or inserted so that (one of) the primary key attribute(s) is *null*.
 - ❖ In either case the transaction is rejected.

Enforcing Foreign Keys

- ❖ Consider these *schema*:
 - ❖ Customer = (*cid*, fname, lname, age, income)
 - ❖ Account = (accnum, balance, type, *cid*)
 - ❖ cid in Account is a *foreign key* referencing Customer.



Foreign Keys – Insertions in the Referencing Table

accnum	balance	type	cid
761	904.33	Loan	111
856	1011.45	Loan	333
903	12.05	Loan	222
1042	10000.00	SAV	333

Inserting {409, 0, Loan, 555} into *Account* violates the foreign key on *cid* as there is no *cid* of 555 in *Customer*

cid	fname	lname	age	salary
111	belay	tesema	23	43000.00
222	fatuma	kedir	22	6764.87
333	roman	tolosa	47	71098.65
444	Dawed	teshome	17	4033.32

The insertion is rejected; before it is processed a *Customer* with a *cid* of 555 must be inserted into the *Customer* table

Foreign Keys – Updates to the Referencing Table

accnum	balance	type	cid
761	904.33	CHQ	111
856	1011.45	CHQ	333
903	12.05	CHQ	222
1042	10000.00	SAV	333

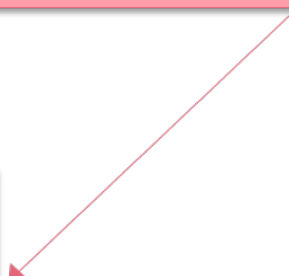
Changing this record's *cid* to 555 also violates the foreign key, again leading to the transaction being rejected

cid	fname	lname	age	salary
111	belay	tesema	23	43000.00
222	fatuma	kedir	22	6764.87
333	roman	tolosa	47	71098.65
444	Dawed	teshome	17	4033.32

Foreign Keys – Deletions in the Referenced Table

accnum	balance	type	cid
761	904.33	CHQ	111
856	1011.45	CHQ	333
903	12.05	CHQ	222
1042	10000.00	SAV	333

Deleting this record will violate the foreign key, because a record with that *cid* exists in the *Account* table

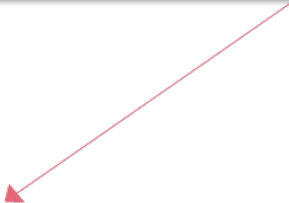


cid	fname	lname	age	salary
111	belay	tesema	23	43000.00
222	fatuma	keder	22	6764.87
333	roman	tolosa	47	71098.65
444	Dawed	teshome	17	4033.32

Foreign Keys – Updates to the Referenced Table

accnum	balance	type	cid
761	904.33	Loan	111
856	1011.45	Loan	333
903	12.05	Loan	222
1042	10000.00	SAV	333

Updating this record so that the *cid* = 666 will violate the foreign key, because a record with the original *cid* exists in the *Account* table



cid	fname	lname	age	salary
111	belay	tesema	23	43000.00
222	fatuma	keder	22	6764.87
333	roman	tolosa	47	71098.65
444	Dawed	teshome	17	4033.32

Foreign Key Violations

- ❖ A deletion or update transaction in the referenced table may violate a foreign key.
- ❖ Different responses can be specified in SQL:
 - ❖ Reject the transaction (the default) – *NO ACTION*
 - ❖ Delete or update the referencing record – *CASCADE*
 - ❖ Set the referencing record's foreign key attribute(s) to null (only on deletion) – *SET NULL*
 - ❖ Set the referencing record's foreign key attribute(s) to a default value (only on deletion) – *SET DEFAULT*
 - ❖ The default value must be specified in the foreign key.

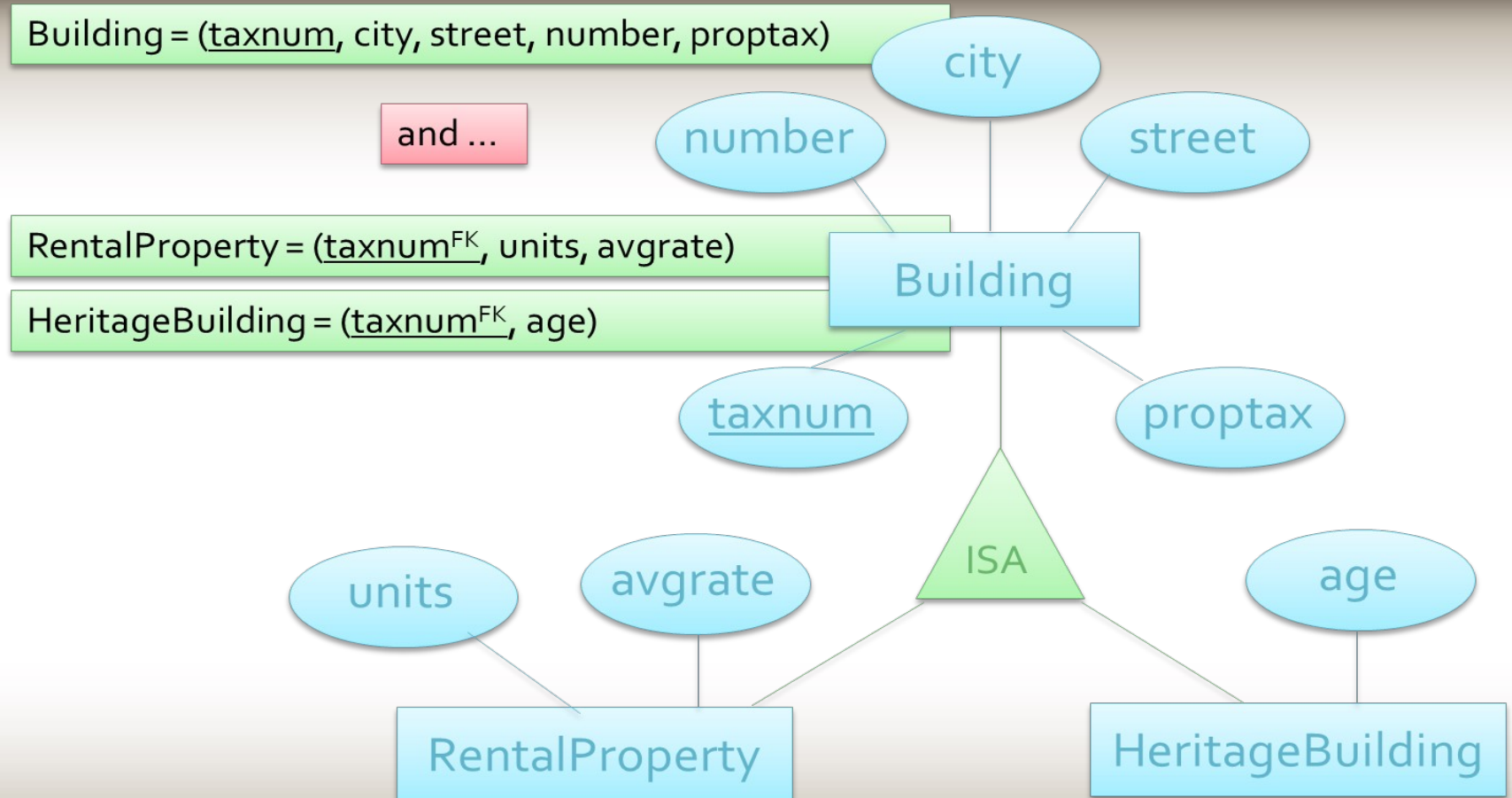
Class Hierarchies

- ❖ There are two basic approaches to translating class hierarchies to tables.
- ❖ Create separate tables for the superclass and for each subclass:
 - ❖ The superclass table only contains attributes of the superclass entity.
 - ❖ The subclass tables contain their own attributes, and the primary key attributes of the superclass.
 - ❖ Superclass attributes are declared as both the primary key and a foreign key in the subclasses.
 - ❖ Cascade deletion of superclass records to subclasses.
- ❖ Or ...

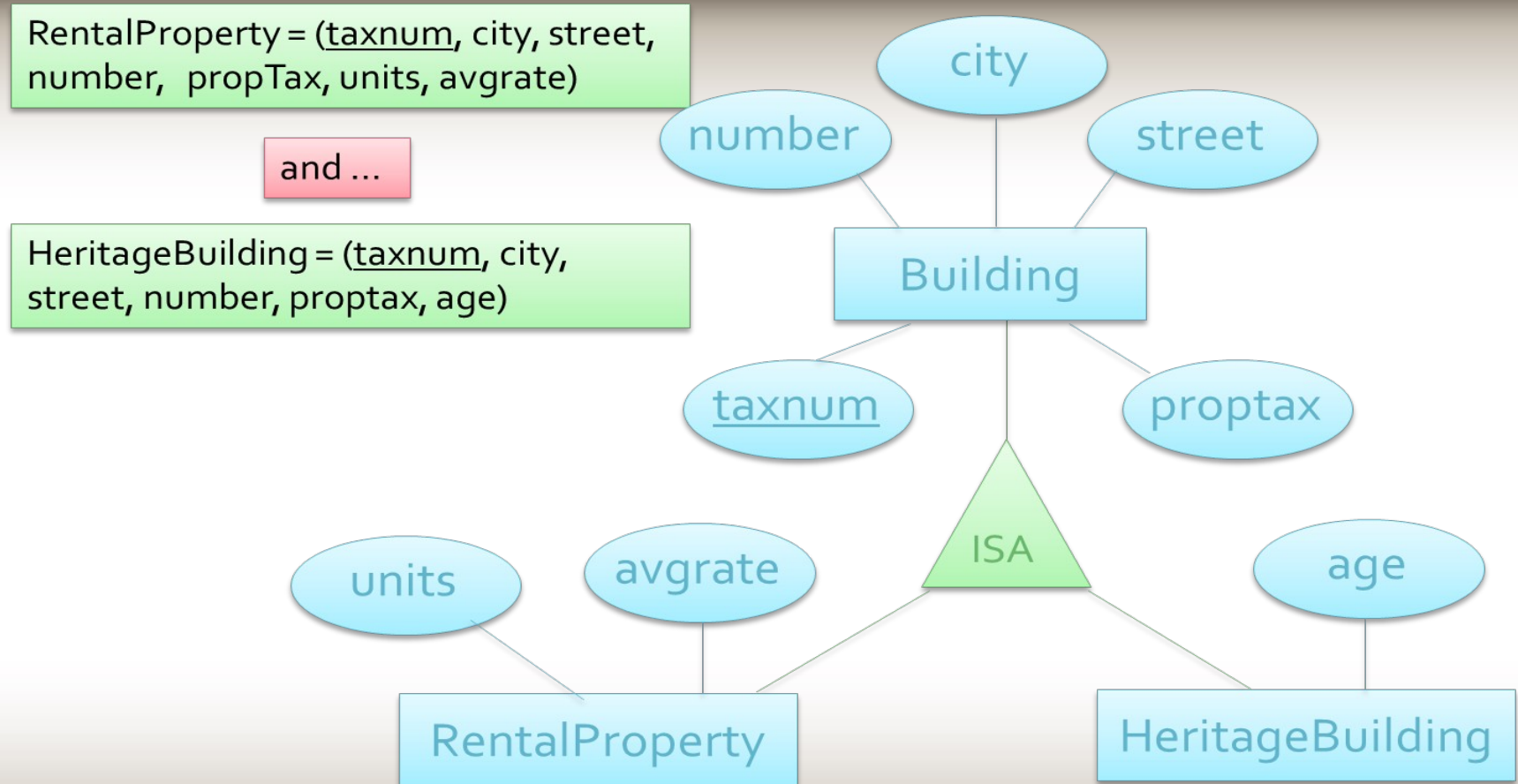
Class Hierarchies ...

- ❖ Create tables for the *subclasses only*:
 - ❖ The subclass tables contain their attributes, and all of the attributes of the superclass.
 - ❖ The primary key of the subclass is the primary key of the superclass.
- ❖ This assumes that there are no entities in the superclass that are not entities in a subclass:
 - ❖ i.e. That a coverage constraint exists.

Subclasses ... Either



Subclasses ... Or



Functional Dependencies and Normalization

❖ Functional Dependencies (FDs)

- ❖ Definition of FD
- ❖ Inference Rules for FDs
- ❖ Equivalence of Sets of FDs
- ❖ Minimal Sets of FDs

❖ Normal Forms

- ❖ First Normal Form
- ❖ Second Normal Form
- ❖ Third Normal Form
- ❖ Boyce-Codd Normal Form

Functional Dependencies and Normalization

- ❖ Simplified version of the COMPANY relational databases schema

EMPLOYEE

ENAME	<u>SSN</u>	BDATE	ADDRESS	DNUMBER f.k.
-------	------------	-------	---------	--------------

p.k.

DEPARTMENT

DNAME	<u>DNUMBER</u>	DMGRSSN f.k.
-------	----------------	--------------

p.k.

DEPT_LOCATIONS

<u>DNUMBER</u>	<u>DLOCATION</u>
----------------	------------------

f.k.

p.k.

PROJECT

PNAME	<u>PNUMBER</u>	PLOCATION	DNUM f.k.
-------	----------------	-----------	-----------

p.k.

WORKS_ON

<u>SSN</u>	<u>PNUMBER</u>	HOURS
------------	----------------	-------

f.k.

f.k.

p.k.

Functional Dependencies and Normalization

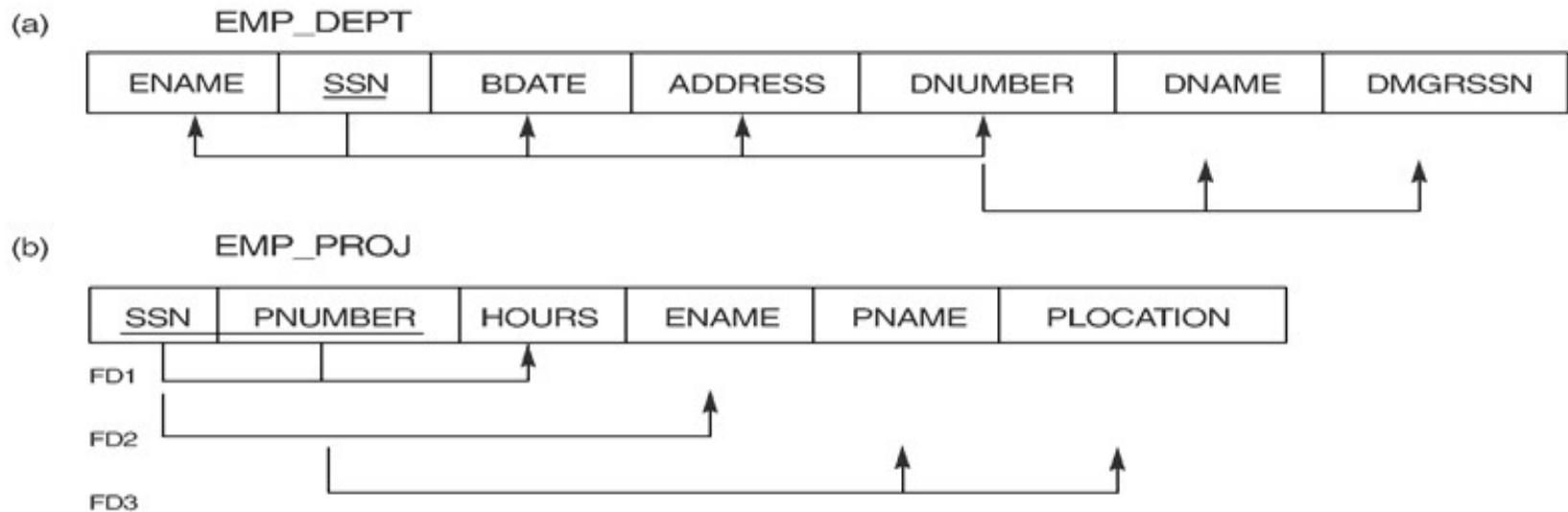
- ❖ Redundant Information in rows and *Update Anomalies*:
 - ❖ Mixing *attributes of multiple entities* may cause problems.
 - ❖ Information is stored redundantly *wasting storage*.
 - ❖ Problems with update anomalies: *Insertion*, *Deletion* and *Modification* anomalies.
- ❖ UPDATE ANOMALY - Consider the relation:
 - ❖ EMP_PROJ (Emp#, Proj#, Ename, Pname, No_hours)
- ❖ *Update Anomaly*: Changing the name of project number P_1 from “*Billing*” to “*Customer-Accounting*” may cause this update to be made for all 100 employees working on project P_1 .

Functional Dependencies and Normalization

- ❖ *Insert Anomaly*: Cannot insert a project unless an employee is assigned to
- ❖ Inversely - Cannot insert an employee unless an he/she is assigned to a project.
- ❖ *Delete Anomaly*: When a project is deleted, it will result in deleting all the employees who work on that project. Alternately, if an employee is the sole employee on a project, deleting that employee would result in deleting the corresponding project.

Functional Dependencies and Normalization

Figure 14.3 Two relation schemas and their functional dependencies. Both suffer from update anomalies. (a) The EMP_DEPT relation schema. (b) The EMP_PROJ relation schema.



Functional Dependencies and Normalization

Figure 14.4 Example relations for the schemas in Figure 14.3 that result from applying NATURAL JOIN to the relations in Figure 14.2. These may be stored as base relations for performance reasons.

EMP DEPT

ENAME	SSN	BDATE	ADDRESS	DNUMBER	DNAME	DMGRSSN
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5	Research	333445555
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	1	Administration	987654321
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4	Administration	987654321
Narayan, Ramesh K.	666884444	1962-09-15	975 Fire Oak, Humble, TX	5	Research	333445555
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5	Research	333445555
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4	Administration	987654321
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1	Headquarters	888665555

EMP PROJ

SSN	PNUMBER	HOURS	ENAME	PNAME	PLOCATION
123456789	1	32.5	Smith, John B.	ProductX	Bellaire
123456789	2	7.5	Smith, John B.	ProductY	Sugarland
666884444	3	40.0	Narayan, Ramesh K.	ProductZ	Houston
453453453	1	20.0	English, Joyce A.	ProductX	Bellaire
453453453	2	20.0	English, Joyce A.	ProductY	Sugarland
333445555	2	10.0	Wong, Franklin T.	ProductY	Sugarland
333445555	3	10.0	Wong, Franklin T.	ProductZ	Houston
333445555	10	10.0	Wong, Franklin T.	Computerization	Stafford
333445555	20	10.0	Wong, Franklin T.	Reorganization	Houston
999887777	30	30.0	Zelaya, Alicia J.	Newbenefits	Stafford
999887777	10	10.0	Zelaya, Alicia J.	Computerization	Stafford
987987987	10	35.0	Jabbar, Ahmad V.	Computerization	Stafford
987987987	30	5.0	Jabbar, Ahmad V.	Newbenefits	Stafford
987654321	30	20.0	Wallace, Jennifer S.	Newbenefits	Stafford
987654321	20	15.0	Wallace, Jennifer S.	Reorganization	Houston
888665555	20	null	Borg, James E.	Reorganization	Houston

© Addison Wesley Longman, Inc. 2000, Elmasri/Navathe, Fundamentals of Database Systems, Third Edition

Functional Dependencies and Normalization

- ❖ Guideline to Redundant Information in Tuples.
- ❖ ***GUIDELINE 1:*** Design a schema that does not suffer from the insertion, deletion and update anomalies. If there are any present, then note them so that applications can be made to take them into account.
- ❖ ***GUIDELINE 2:*** Relations should be designed such that their tuples will have as few NULL values as possible.
 - ❖ Attributes that are NULL frequently could be placed in separate relations (with the primary key).
 - ❖ Reasons for nulls:
 - ❖ Attribute not applicable or invalid,
 - ❖ Attribute value unknown (may exist),
 - ❖ Value known to exist, but unavailable.



Functional Dependencies

- ❖ Functional dependencies (FDs) are used to specify formal measures of the "goodness" of relational designs.
- ❖ FDs are constraints that are derived from the meaning and interrelationships of the data attributes.
- ❖ A set of attributes X functionally determines a set of attributes Y if the value of X determines a unique value for Y .
- ❖ $X \rightarrow Y$ holds if whenever two tuples have the same value for X , they must have the same value for Y
- ❖ For any two tuples $t1$ and $t2$ in any relation instance $r(R)$: If $t1[X]=t2[X]$, then $t1[Y]=t2[Y]$
- ❖ $X \rightarrow Y$ in R specifies a constraint on all relation instances $r(R)$
- ❖ Written as $X \rightarrow Y$; can be displayed graphically on a relation schema which is denoted by the arrow:
- ❖ FDs are derived from the real-world constraints on the attributes.

Functional Dependencies: Examples

- ❖ Social security number determines employee name $SSN \rightarrow ENAME$.
- ❖ Project number determines project name and location $PNUMBER \rightarrow \{PNAME, PLOCATION\}$.
- ❖ Employee ssn and project number determines the hours per week that the employee works on the project $\{SSN, PNUMBER\} \rightarrow HOURS$.
- ❖ If K is a key of R , then K functionally determines all attributes in R (since we never have two distinct tuples with $t1[K]=t2[K]$).

Normal Forms

- ❖ Normalization of Relations
- ❖ *Normalization*: The process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations.
- ❖ *Normal form*: Condition using keys and FDs of a relation to certify whether a relation schema is in a particular normal form.
 - ❖ 1NF, 2NF, 3NF, BCNF based on keys and FDs of a relation schema.
- ❖ *Denormalization*: the process of storing the join of higher normal form relations as a base relation—which is in a lower normal form.

Keys and Attributes Participating

- ❖ A super-key of a relation schema $R = \{A_1, A_2, \dots, A_n\}$ is a set of attributes S subset-of R with the property that no two tuples t_1 and t_2 in any legal relation state r of R will have $t_1[S] = t_2[S]$
- ❖ A key K is a super-key with the additional property that removal of any attribute from K will cause K not to be a superkey any more.
- ❖ If a relation schema has more than one key, each is called a candidate key. One of the candidate keys is arbitrarily designated to be the primary key, and the others are called secondary keys.
- ❖ A Prime attribute must be a member of some candidate key

First Normal Form

- ❖ Disallows composite attributes and multivalued attributes, whose values for an individual tuple are non-atomic.

Figure 14.8 Normalization into 1NF. (a) Relation schema that is not in 1NF. (b) Example relation instance. (c) 1NF relation with redundancy.

(a)

DEPARTMENT			
DNAME	<u>DNUMBER</u>	DMGRSSN	DLOCATIONS

(b)

DEPARTMENT			
DNAME	<u>DNUMBER</u>	DMGRSSN	DLOCATIONS
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

(c)

DEPARTMENT			
DNAME	<u>DNUMBER</u>	DMGRSSN	<u>DLOCATION</u>
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

© Addison Wesley Longman, Inc. 2000, Elmasri/Navathe, Fundamentals of Database Systems, Third Edition

First Normal Form

Figure 14.9 Normalizing nested relations into 1NF. (a) Schema of the EMP_PROJ relation with a “nested relation” PROJS. (b) Example extension of the EMP_PROJ relation showing nested relations within each tuple. (c) Decomposing EMP_PROJ into 1NF relations EMP_PROJ1 and EMP_PROJ2 by propagating the primary key.

(a)

EMP_PROJ

SSN	ENAME	PROJS	
		PNUMBER	HOURS

(b)

EMP_PROJ

SSN	ENAME	PNUMBER	HOURS
123456789	Smith, John B.	1	32.5
		2	7.5
666884444	Narayan, Ramesh K.	3	40.0
453453453	English, Joyce A.	1	20.0
		2	20.0
333445555	Wong, Franklin T.	2	10.0
		3	10.0
		10	10.0
		20	10.0
999887777	Zelaya, Alicia J.	30	30.0
987987987	Jabbar, Ahmad V.	10	10.0
		30	35.0
987654321	Wallace, Jennifer S.	30	5.0
		20	20.0
888665555	Borg, James E.	20	15.0
		20	null

(c)

EMP_PROJ1

<u>SSN</u>	ENAME
------------	-------

EMP_PROJ2

<u>SSN</u>	<u>PNUMBER</u>	HOURS
------------	----------------	-------

Disadvantages of INF

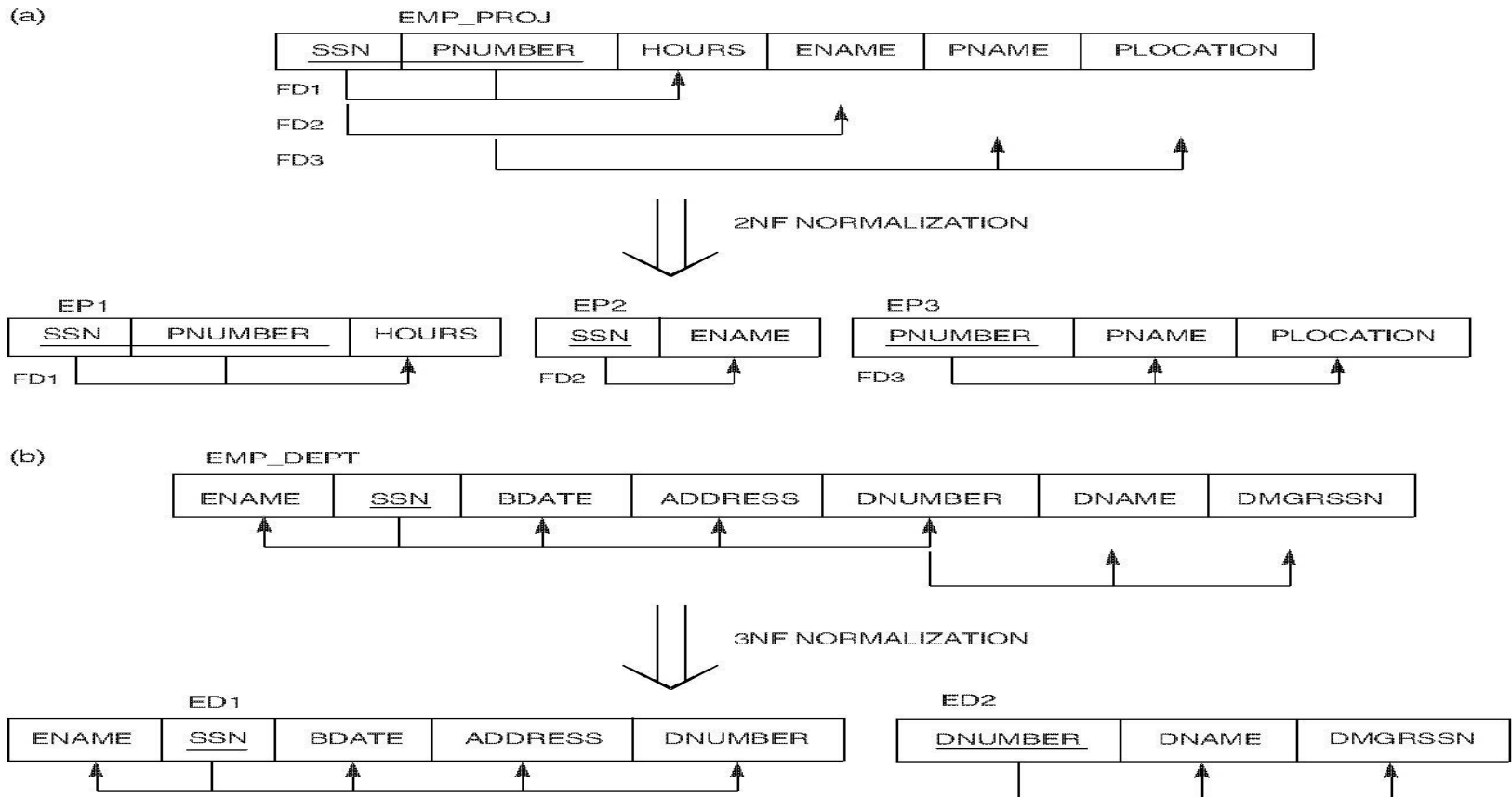
- ❖ Introduces redundancy
 - ❖ Much of the data needs to be repeated.
- ❖ Insert anomalies
 - ❖ A Project cannot be inserted if it doesn't have at least one employee
- ❖ Delete anomalies
 - ❖ Deleting the last employee of a project also deletes the project.
- ❖ Update anomalies
 - ❖ Many records may have to be changed to change the value of one attribute.

Second Normal Form

- ❖ *Prime attribute* - attribute that is member of the primary key K.
- ❖ *Full functional dependency* - a FD $Y \rightarrow Z$ where removal of any attribute from Y means the FD does not hold any more.
- ❖ *Examples:*
 - $\{SSN, PNUMBER\} \rightarrow HOURS$ is a full FD since neither $SSN \rightarrow HOURS$ nor $PNUMBER \rightarrow HOURS$ hold.
 - ❖ - $\{SSN, PNUMBER\} \rightarrow ENAME$ is not a full FD (it is called a partial dependency) since $SSN \rightarrow ENAME$ also holds
- ❖ A relation schema R is in *second normal form (2NF)* if every non-prime attribute A in R is fully functionally dependent on the primary key.

Second Normal Form

Figure 14.10 The normalization process. (a) Normalizing EMP_PROJ into 2NF relations. (b) Normalizing EMP_DEPT into 3NF relations.



Disadvantages of 2NF

- ❖ Second Normal Form only considers partial key dependencies.
 - ❖ And ignores any non-key dependencies.
- ❖ Therefore 2NF may still result in the same problems observed with 1NF, that is:
 - ❖ Redundancy,
 - ❖ Insert, delete and update anomalies.

Third Normal Form

- ❖ Transitive functional dependency - a FD $X \rightarrow Z$ that can be derived from two FDs $X \rightarrow Y$ and $Y \rightarrow Z$.
- ❖ Examples:
 - ❖ $SSN \rightarrow DMGRSSN$ is a transitive FD since $SSN \rightarrow DNUMBER$ and $DNUMBER \rightarrow DMGRSSN$ hold.
 - ❖ $SSN \rightarrow ENAME$ is non-transitive since there is no set of attributes X where $SSN \rightarrow X$ and $X \rightarrow ENAME$.
- ❖ A relation schema R is in third normal form (3NF) if it is in 2NF and no non-prime attribute A in R is transitively dependent on the primary key.

Question & Answer



Thanks !!!