INFSCI 1022
Database Management Systems

Today’s Evil Plan

- Database design / E-R Models
- Relationships / Cardinalities
- Hierarchies
- Introduction to Normalization

Database design / E-R Models

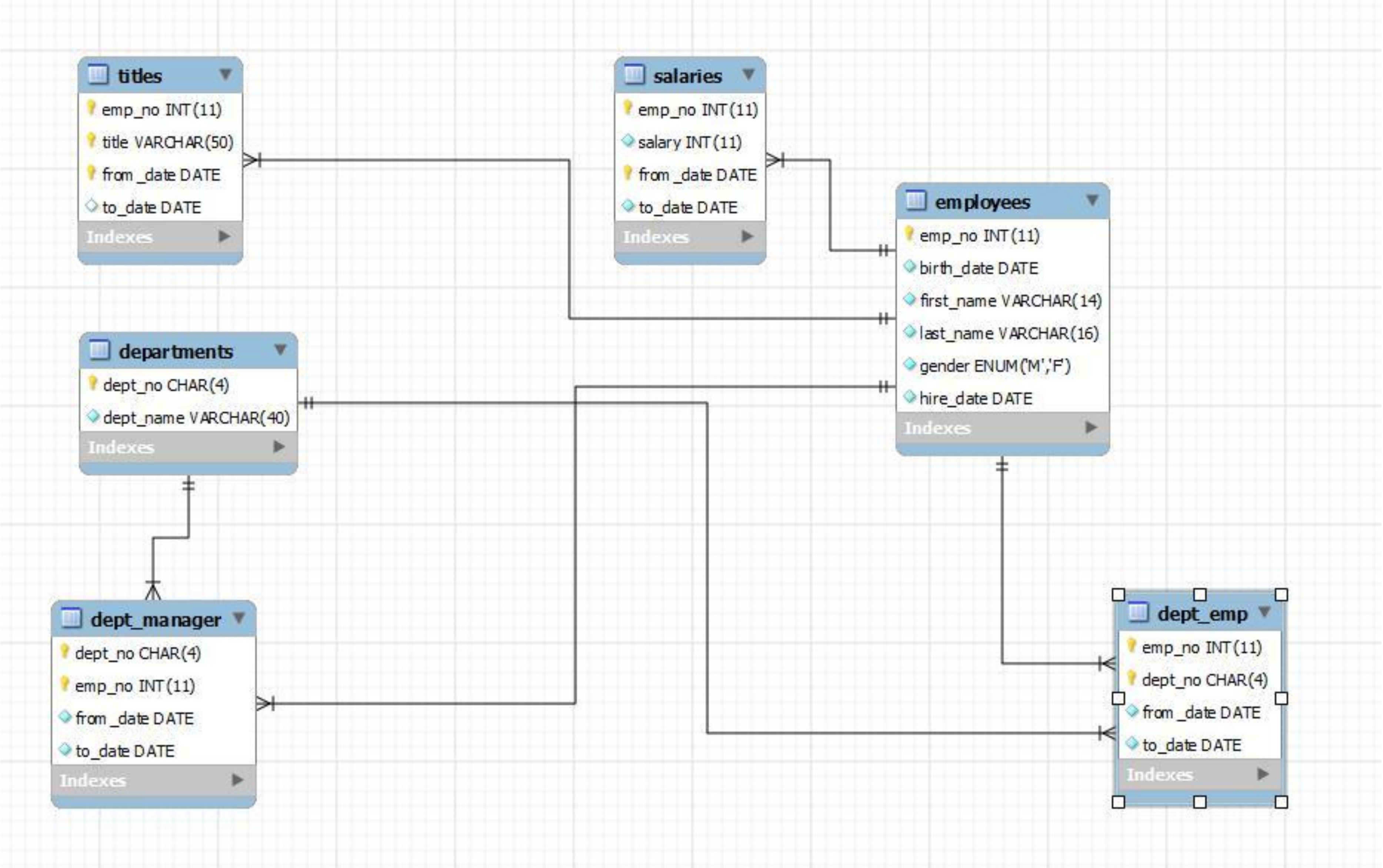
Relationships / Cardinalities

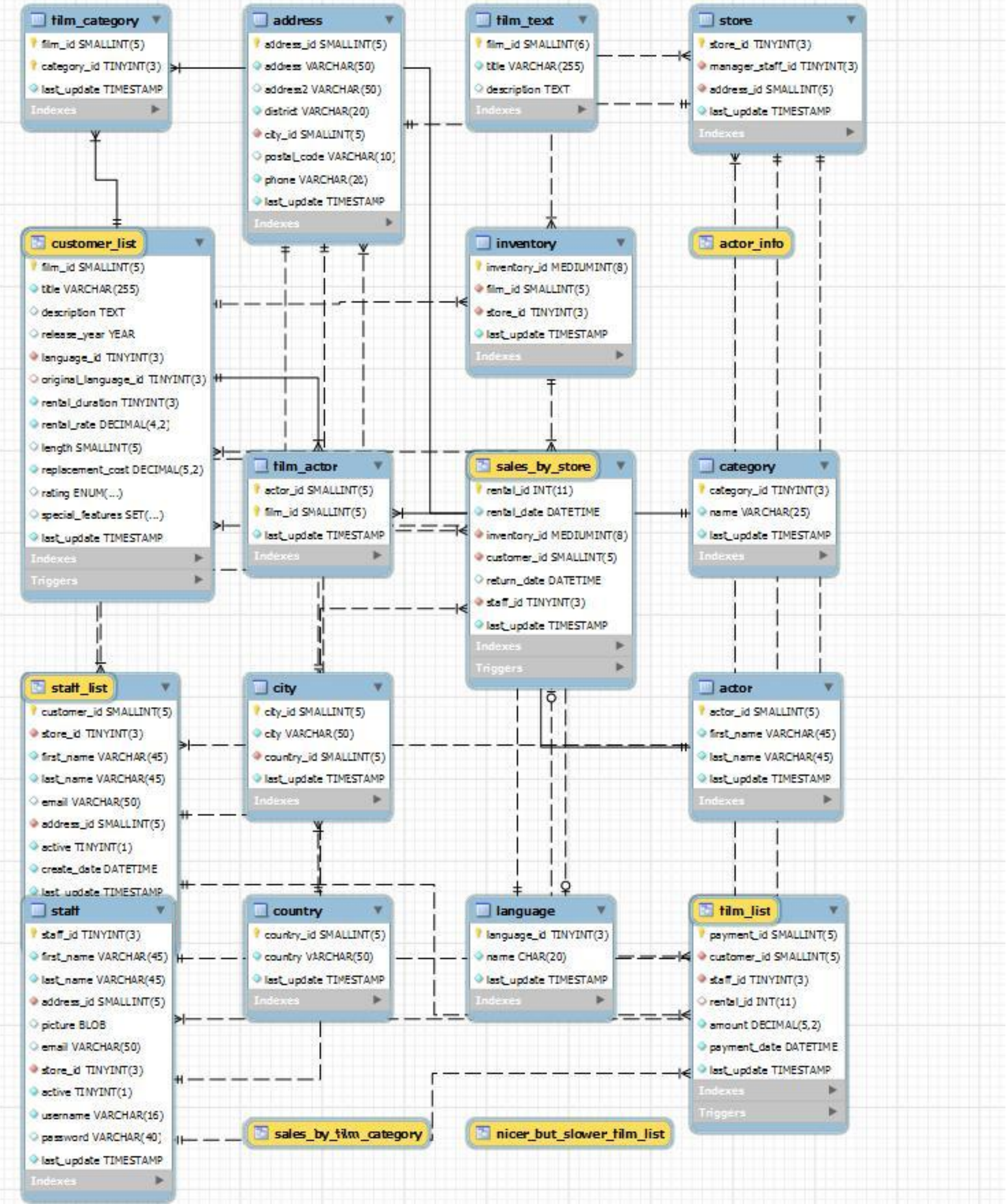
Hierarchies

Introduction to Normalization

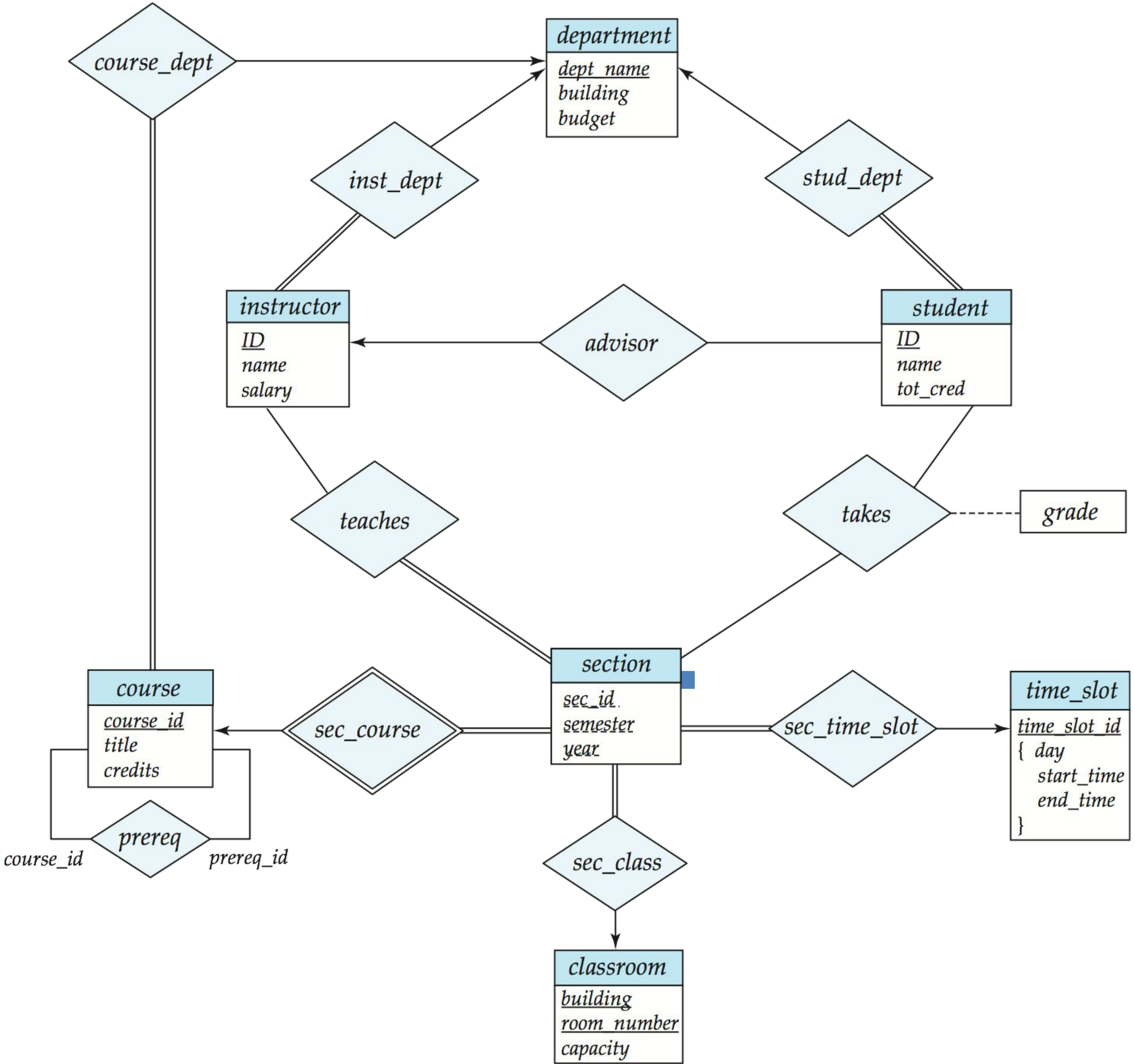
Modeling

- A *database* can be modeled as:
  - a collection of entities,
  - relationship among entities.
- An **entity** is an object that exists and is distinguishable from other objects.
  - Example: specific person, company, event, plant
- Entities have *attributes*
  - Example: people have *names* and *addresses*
- An **entity set** is a set of entities of the same type that share the same properties.
  - Example: set of all persons, companies, trees, holidays

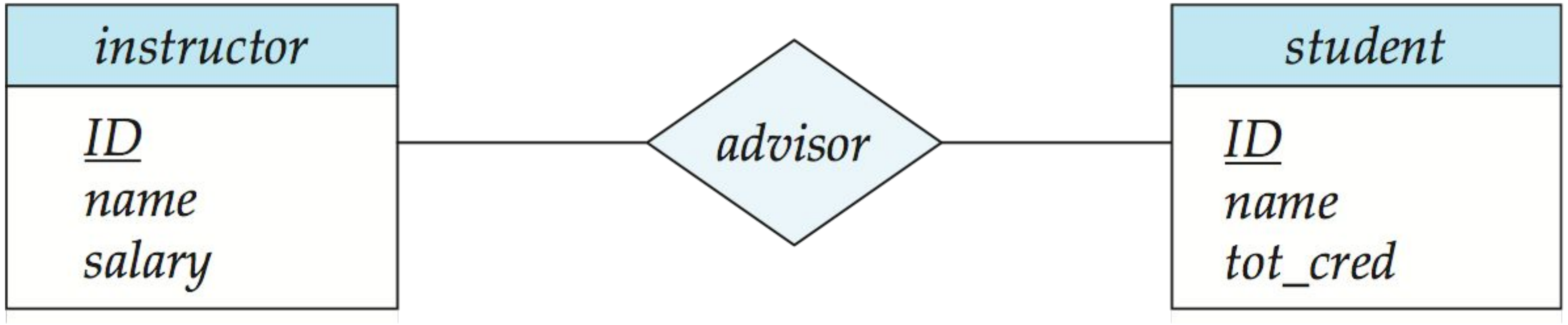

titles
emp_no INT(11)
title VARCHAR(50)
from_date DATE
to_date DATE
Indexes
salaries
emp_no INT(11)
salary INT(11)
from_date DATE
to_date DATE
Indexes
employees
emp_no INT(11)
birth_date DATE
first_name VARCHAR(14)
last_name VARCHAR(16)
gender ENUM('M','F')
hire_date DATE
Indexes
departments
dept_no CHAR(4)
dept_name VARCHAR(40)
Indexes
dept_manager
dept_no CHAR(4)
emp_no INT(11)
from_date DATE
to_date DATE
Indexes
dept_emp
emp_no INT(11)
dept_no CHAR(4)
from_date DATE
to_date DATE
Indexes


film_category
film_id SMALLINT(5)
category_id TINYINT(3)
last_update TIMESTAMP
Indexes
address
address_id SMALLINT(5)
address VARCHAR(50)
address2 VARCHAR(50)
district VARCHAR(20)
city_id SMALLINT(5)
postal_code VARCHAR(10)
phone VARCHAR(20)
last_update TIMESTAMP
Indexes
film_text
film_id SMALLINT(6)
title VARCHAR(255)
description TEXT
Indexes
store
store_id TINYINT(3)
manager_staff_id TINYINT(3)
address_id SMALLINT(5)
last_update TIMESTAMP
Indexes
customer_list
film_id SMALLINT(5)
title VARCHAR(255)
description TEXT
release_year YEAR
language_id TINYINT(3)
original_language_id TINYINT(3)
rental_duration TINYINT(3)
rental_rate DECIMAL(4,2)
length SMALLINT(5)
replacement_cost DECIMAL(5,2)
rating ENUM(...)
special_features SET(...)
last_update TIMESTAMP
Indexes
Triggers
inventory
inventory_id MEDIUMINT(8)
film_id SMALLINT(5)
store_id TINYINT(3)
last_update TIMESTAMP
Indexes
actor_info
film_actor
actor_id SMALLINT(5)
film_id SMALLINT(5)
last_update TIMESTAMP
Indexes
sales_by_store
rental_id INT(11)
rental_date DATETIME
inventory_id MEDIUMINT(8)
customer_id SMALLINT(5)
return_date DATETIME
staff_id TINYINT(3)
last_update TIMESTAMP
Indexes
Triggers
category
category_id TINYINT(3)
name VARCHAR(25)
last_update TIMESTAMP
Indexes
staff_list
customer_id SMALLINT(5)
store_id TINYINT(3)
first_name VARCHAR(45)
last_name VARCHAR(45)
email VARCHAR(50)
address_id SMALLINT(5)
active TINYINT(1)
create_date DATETIME
last_update TIMESTAMP
city
city_id SMALLINT(5)
city VARCHAR(50)
country_id SMALLINT(5)
last_update TIMESTAMP
Indexes
actor
actor_id SMALLINT(5)
first_name VARCHAR(45)
last_name VARCHAR(45)
last_update TIMESTAMP
Indexes
staff
staff_id TINYINT(3)
first_name VARCHAR(45)
last_name VARCHAR(45)
address_id SMALLINT(5)
picture BLOB
email VARCHAR(50)
store_id TINYINT(3)
active TINYINT(1)
username VARCHAR(16)
password VARCHAR(40)
last_update TIMESTAMP
Indexes
country
country_id SMALLINT(5)
country VARCHAR(50)
last_update TIMESTAMP
Indexes
language
language_id TINYINT(3)
name CHAR(20)
last_update TIMESTAMP
Indexes
film_list
payment_id SMALLINT(5)
customer_id SMALLINT(5)
staff_id TINYINT(3)
rental_id INT(11)
amount DECIMAL(5,2)
payment_date DATETIME
last_update TIMESTAMP
Indexes
Triggers
sales_by_film_category
nicer_but_slower_film_list

E-R Diagram for a University Enterprise



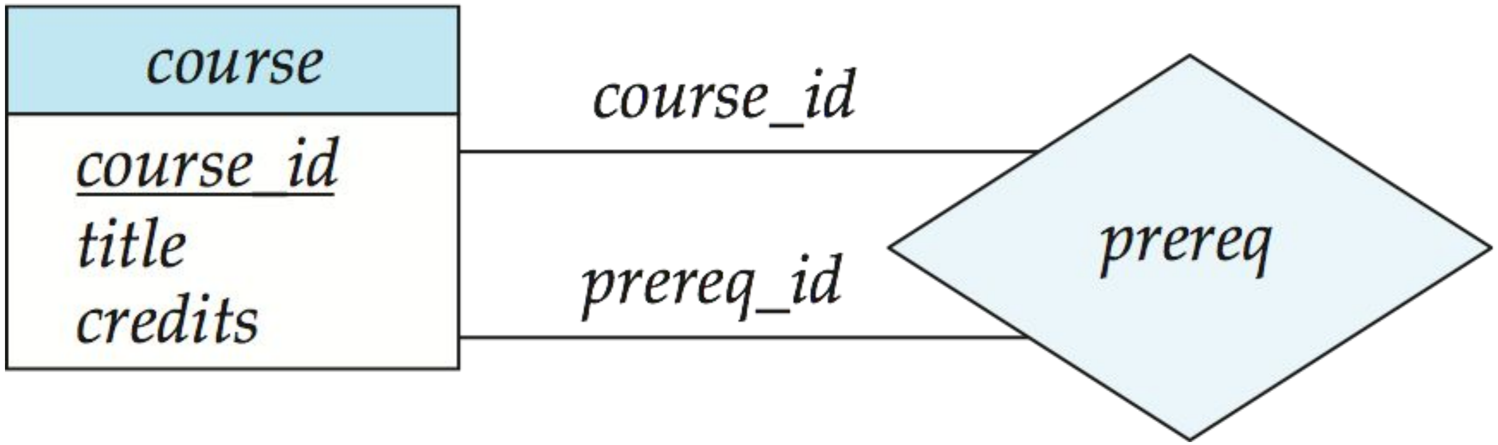
E-R Diagrams



- Rectangles represent entity sets.
- Diamonds represent relationship sets.
- Attributes listed inside entity rectangle
- Underline indicates primary key attributes

Roles

- Entity sets of a relationship need not be distinct
  – Each occurrence of an entity set plays a "role" in the relationship
- The labels "*course_id*" and "*prereq_id*" are called **roles**.



Database design / E-R Models

Relationships / Cardinalities

Hierarchies

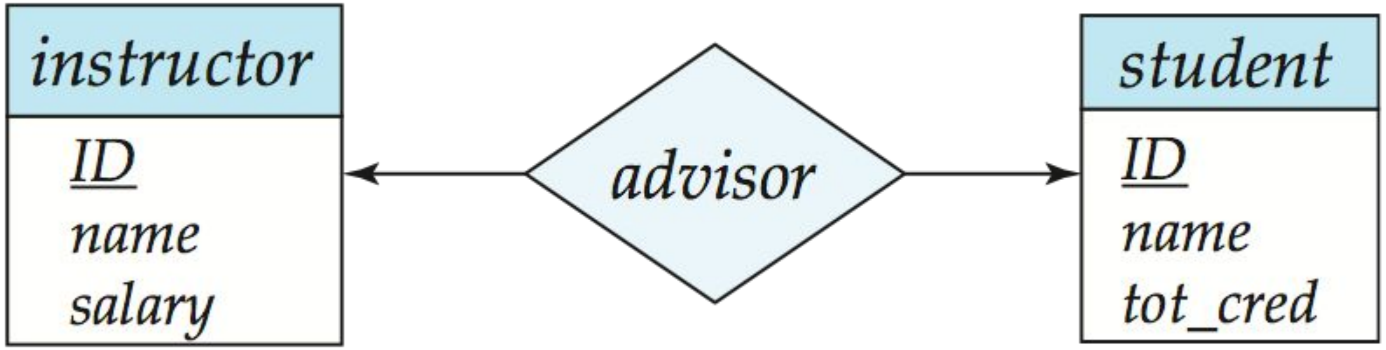
Introduction to Normalization

Cardinality Constraints

- We express cardinality constraints by drawing either a directed line (→), signifying “one,” or an undirected line (—), signifying “many,” between the relationship set and the entity set.

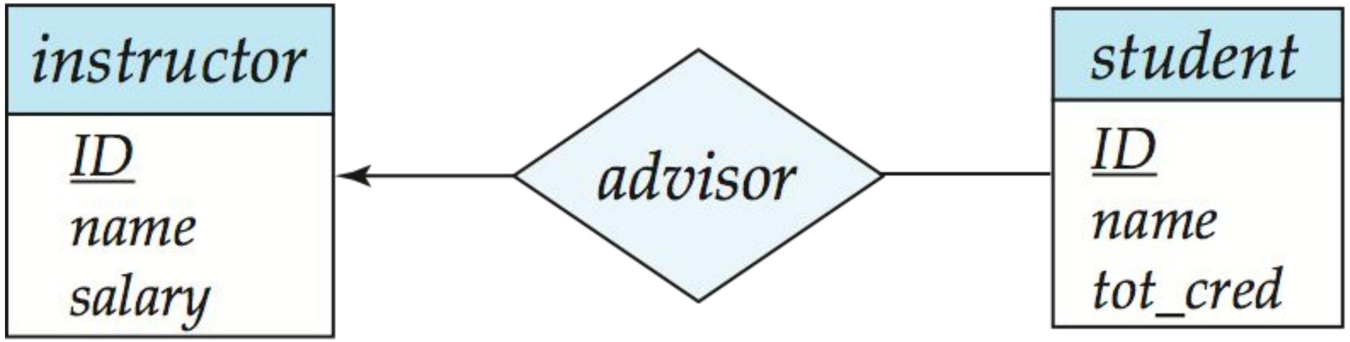
One-to-One Relationship

- one-to-one relationship between an *instructor* and a *student*
  - an instructor is associated with at most one student via *advisor*
  - and a student is associated with at most one instructor via *advisor*



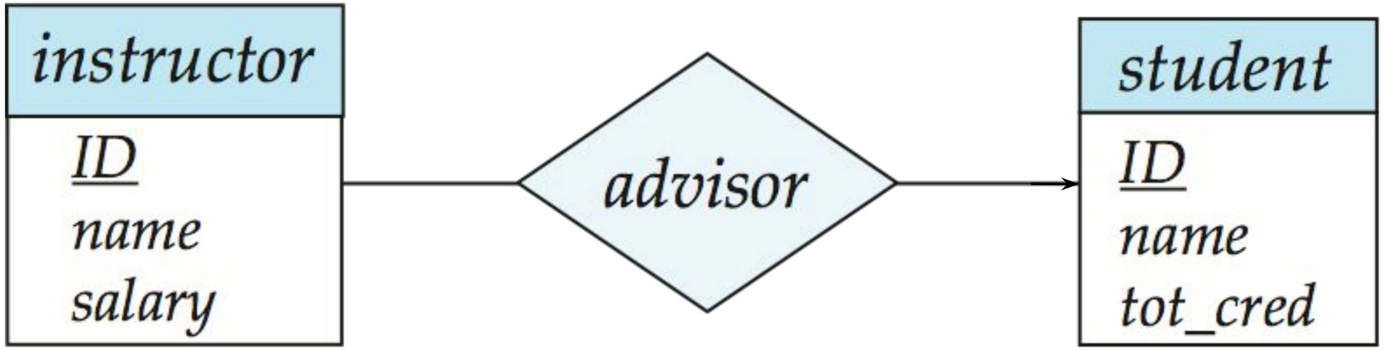
One-to-Many Relationship

- one-to-many relationship between an *instructor* and a *student*
  - an instructor is associated with several (including 0) students via *advisor*
  - a student is associated with at most one instructor via advisor,



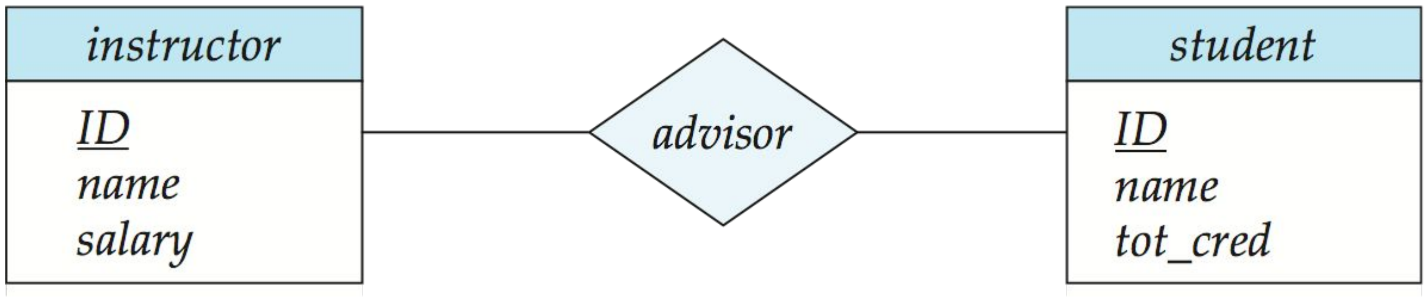
Many-to-One Relationships

- In a many-to-many relationship between an *instructor* and a *student,*
  - an instructor is associated with at most one student via *advisor*,
  - and a student is associated with several (including 0) instructors via *advisor*



Many-to-Many Relationship

- An instructor is associated with several (possibly 0) students via *advisor*
- A student is associated with several (possibly 0) instructors via *advisor*

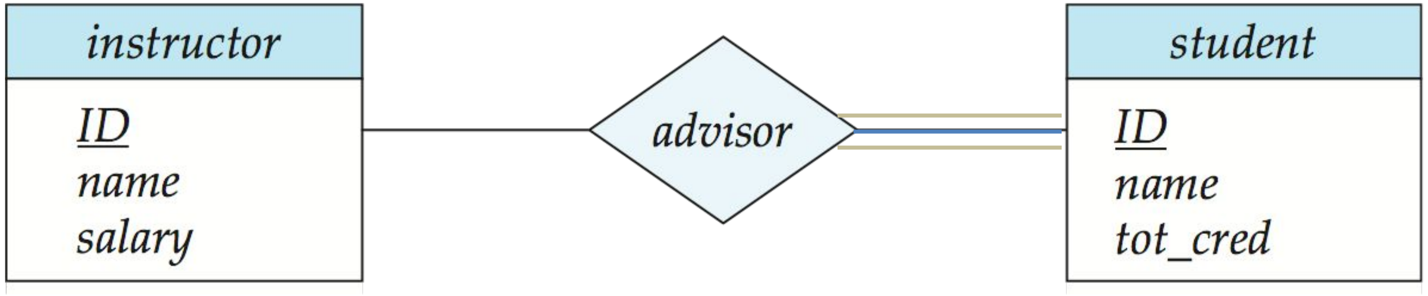


Participation of an Entity Set in a Relationship Set

- Total participation (indicated by double line): every entity in the entity set participates in at least one relationship in the relationship set
  - E.g., participation of *student* in *advisor* is total

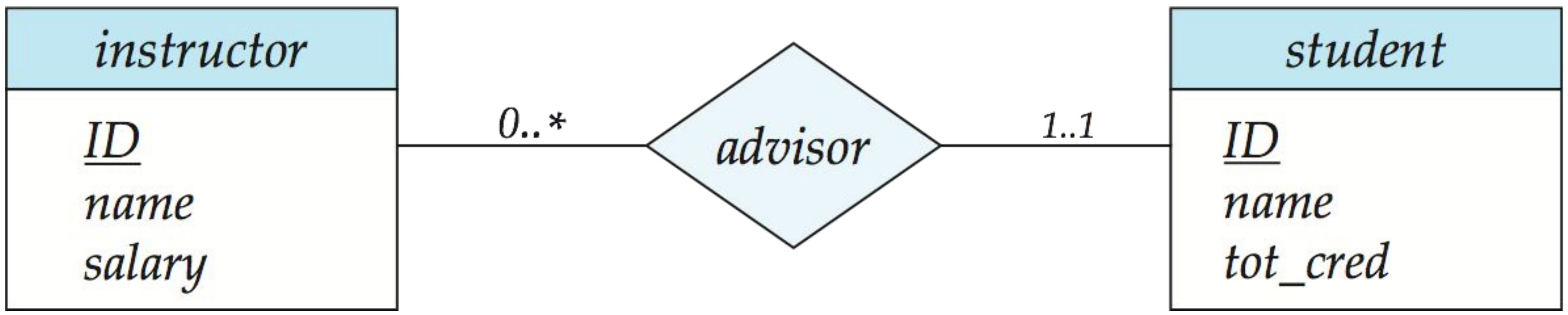
    every *student* must have an associated advisor
- Partial participation: some entities may not participate in any relationship in the relationship set
  - Example: participation of *instructor* in *advisor* is partial

    not every *instructor* must have an associated advisee

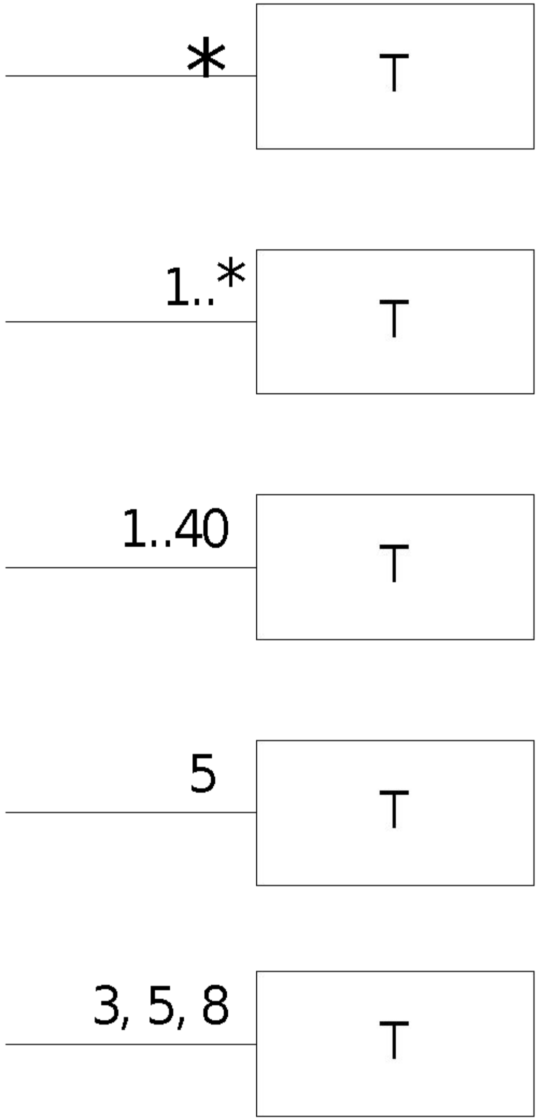


Alternative Notation for Cardinality Limits (a.k.a Multiplicities)

- Cardinality limits can also express participation constraints



Cardinality Limits



zero or more;
"many"

one or more

one to 40

exactly 5

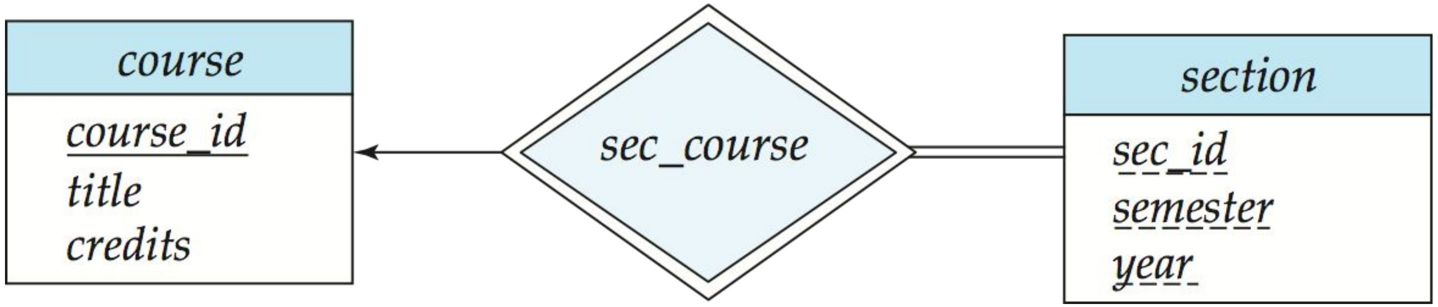
exactly 3, 5, or 8

Weak Entity Sets

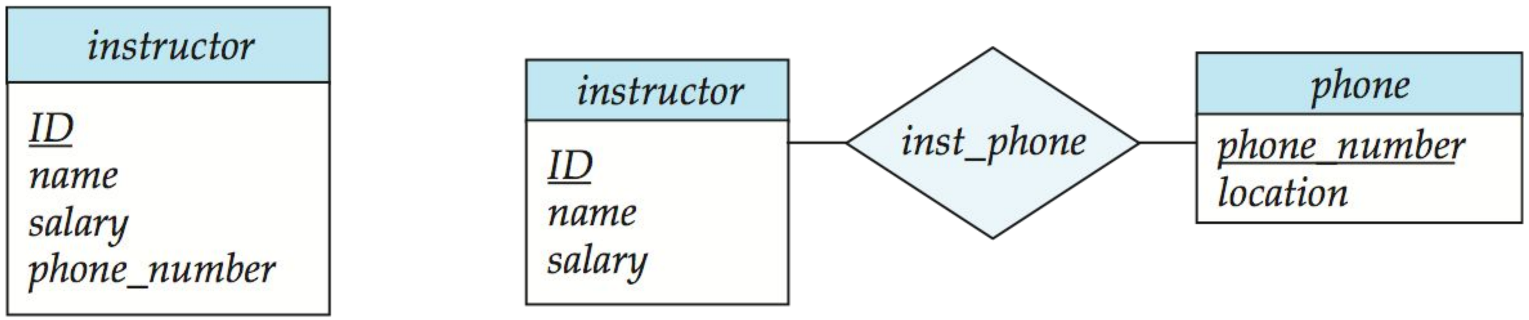
- An entity set that does not have a primary key is referred to as a **weak entity set.**
- The existence of a weak entity set depends on the existence of a **identifying entity set**
  - It must relate to the identifying entity set via a total, one-to-many relationship set from the identifying to the weak entity set
  - **Identifying relationship** depicted using a double diamond
- The **discriminator** (*or partial key)* of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set.
- The primary key of a weak entity set is formed by the primary key of the strong entity set on which the weak entity set is existence dependent, plus the weak entity set's discriminator.

Weak Entity Sets (Cont.)

- We underline the discriminator of a weak entity set with a dashed line.
- We put the identifying relationship of a weak entity in a double diamond.
- Primary key for *section* – (*course_id, sec_id, semester, year*)



Entity Sets vs. Attributes

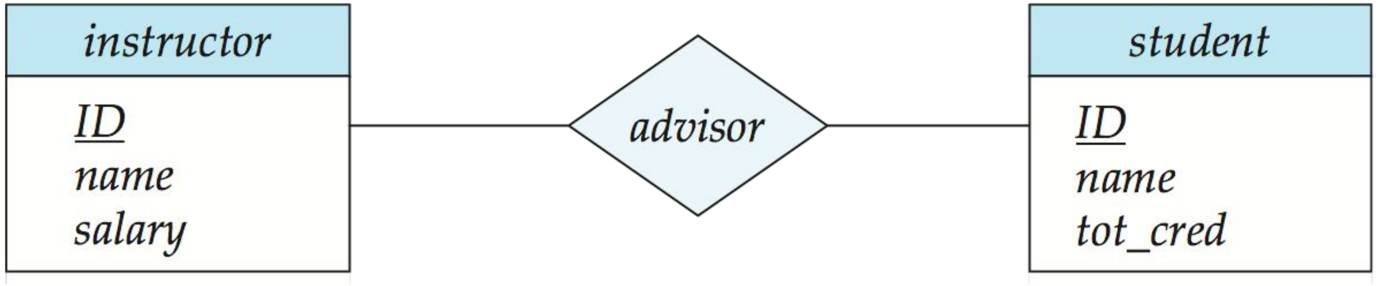


- Use of phone as an entity allows extra information about phone numbers (plus multiple phone numbers)

Representing Relationship Sets

- A many-to-many relationship set is represented as a schema with attributes for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.
- Example: schema for relationship set *advisor*

  advisor = (*<u>s_id, i_id</u>*)



Database design / E-R Models

Relationships / Cardinalities

Hierarchies

Introduction to Normalization

Inheritance

- Many real-life objects are related in a hierarchical fashion
- Lower-levels of hierarchy inherit characteristics of the upper levels

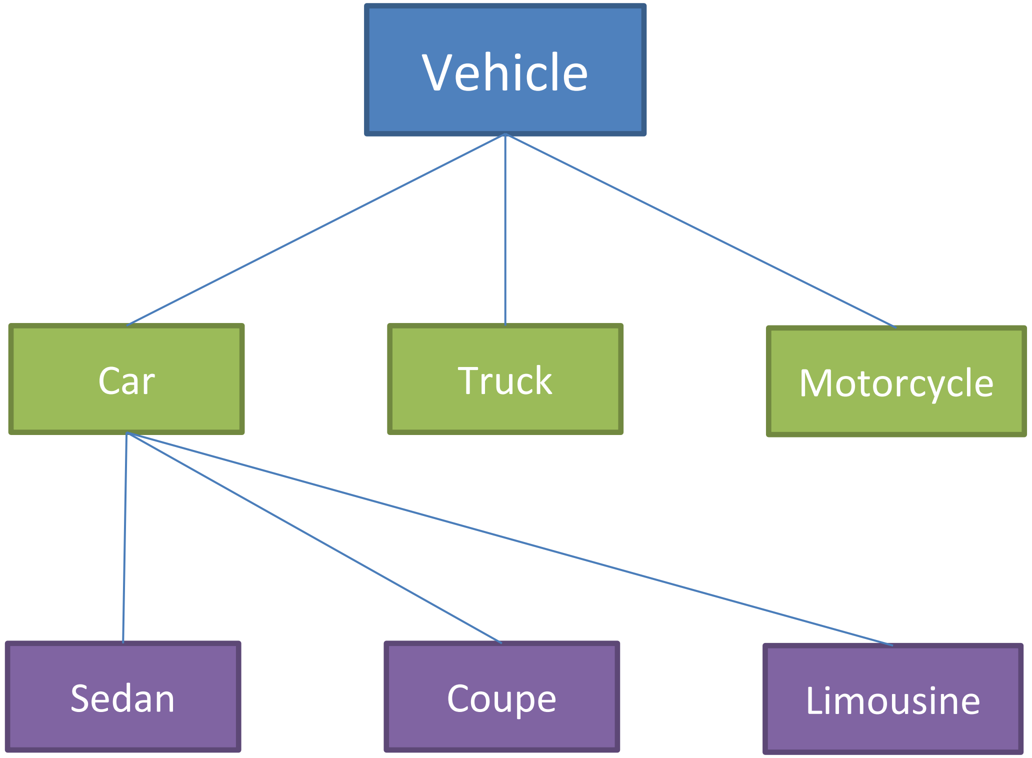
Inheritance Examples

- Mammal →Primate →Human
- Vehicle → Car → Honda → Honda Accord
- Person → Employee → Faculty member

Note: These types of hierarchies/relationships may be called **IS-A** (a primate **is-a** mammal, a car **is-a** vehicle)

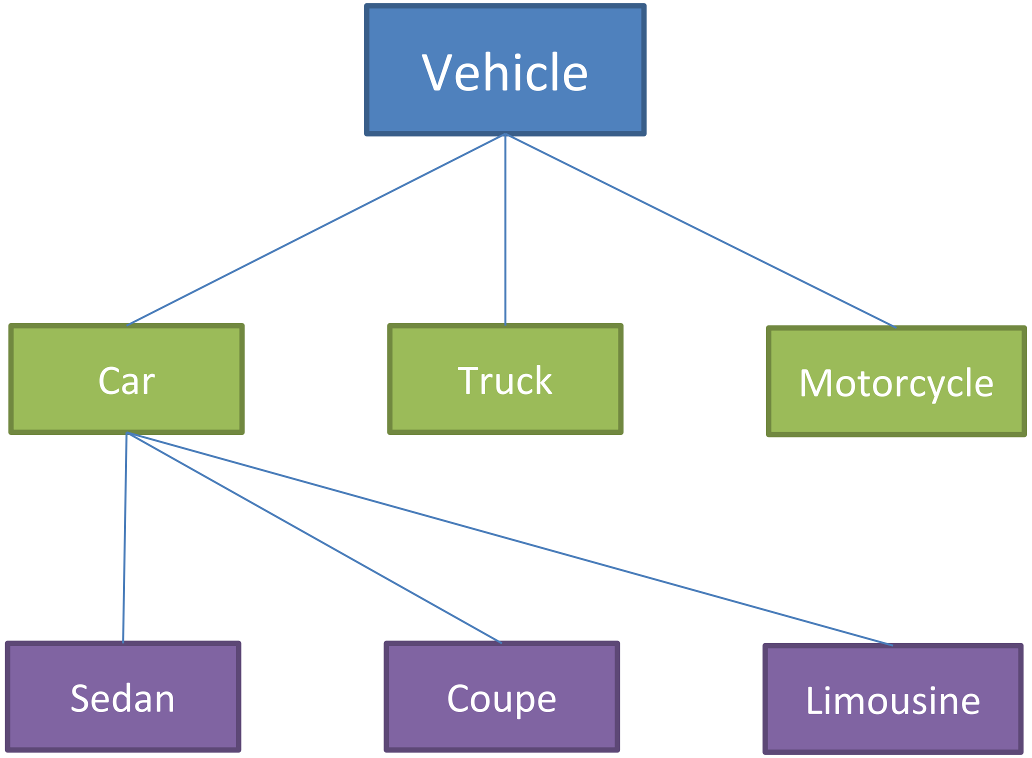
Inheritance Hierarchy

- The inheritance hierarchy is usually drawn as an inverted (upside-down) tree
- The tree can have any number of levels
- The tree is acyclic



Inheritance Hierarchy

- The entity at the top (base) of the tree is called the **root entity**

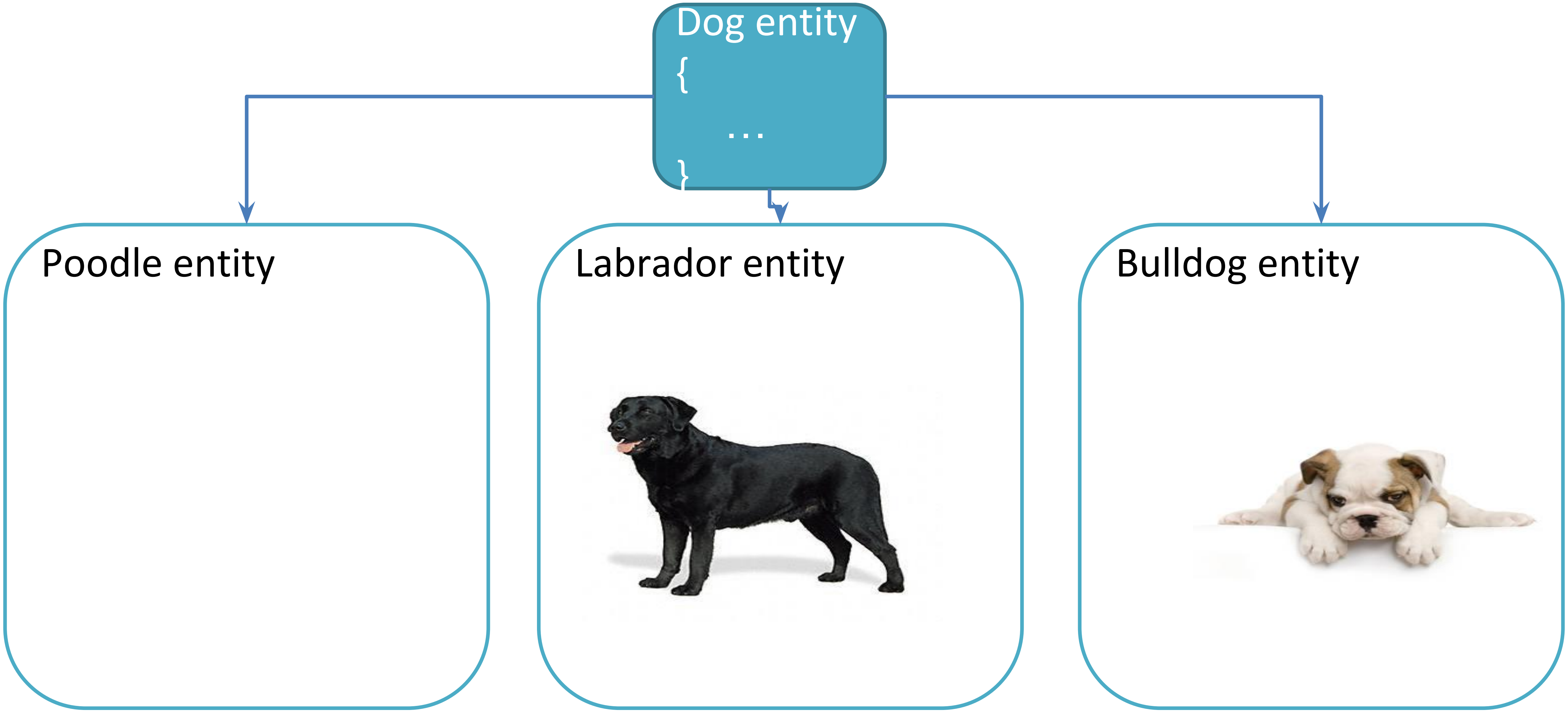


Inheritance Terminology

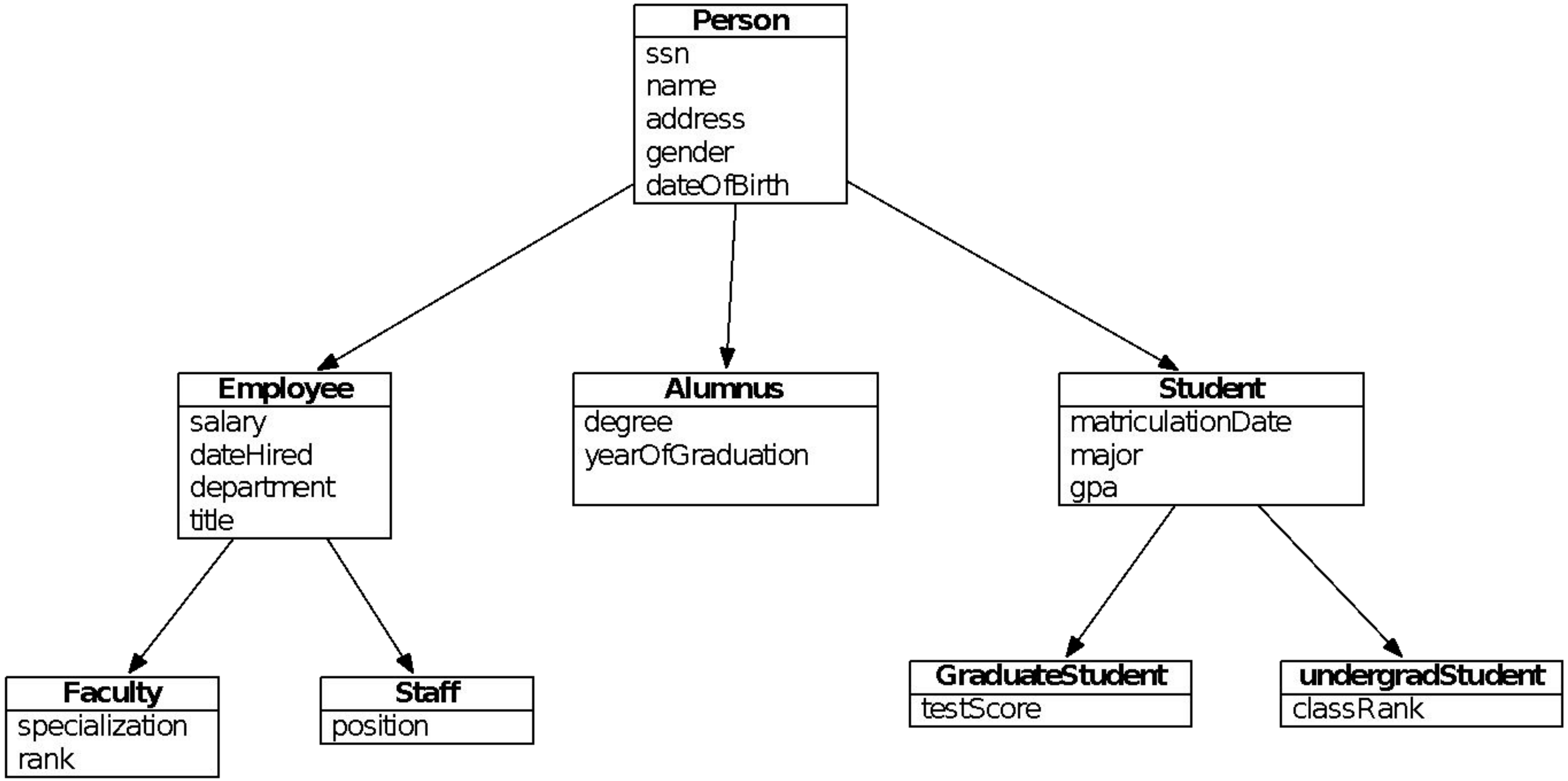
- **Root entity** – entity at the top of the hierarchy tree.
- **Derived (child) entity** – an entity that inherits characteristics of another entity.
- **Base (parent) entity** – an entity from which characteristics and behaviors are inherited by other entities

Subtypes/Supertypes

- A **subtype** is a subgrouping of the entities in an entity type.
- A **supertype** is a generic entity type that has a relationship with one or more subtypes.
- Supertypes and subtypes have parent/child relationships


Dog entity
{
...
}
Poodle entity
Labrador entity
Bulldog entity

Subtypes/Supertypes



Syntax Recap

- Rectangle: entity
- Ellipse: attribute **(do not USE)**
- Diamond: relationship
- Directed arrow (→): “one”
- Undirected line (—): “many”

Best (Badly Designed) Summary

- http://jcsites.juniata.edu/faculty/rhodes/dbms/ermodel.htm

Database design / E-R Models

Relationships / Cardinalities

Hierarchies

Introduction to Normalization

Excel != Database

Format Painter | Formatting | Table | Styles | Clear | Filter | Select

Clipboard | Font | Alignment | Number | Styles | Cells | Editing

D94 | Congenital anomalies

	A	B	C	D	E	F	G	H
1	FieldType	DataType	Data	DataField	FieldOption	Where from	What we broke by re-desgning the delive	
2	textfield	Text	Alpha and Freetext:	Place of birth	Hospital			
3				Place of birth	Birth Center			
4				Place of birth	clinic/office			
5				Place of birth	home birth planned			
6				Place of birth	home birth, unplanned			
7				Place of birth	other			
8			Date:	Date of first prenatal visit		Epic		
9			Date:	date of last prenatal visit		Epic		
10			Numeric:	number of prenatal visits		Epic		
11			Date:	LMP		PCM EDD Control		
12			Numeric:	living children		PCM Pregnancy History cont	x	
13			Date:	date of last live birth		PCM Pregnancy History control		
14			Numeric:	live births now deceased		PCM Pregnancy History control		
15			Numeric:	total number of other pregnancy outcomes		PCM Pregnancy History control		
16			Date:	date of last Other pregnancy outcome		PCM Pregnancy History control		
17			Numeric:	mother's weight at delivery	LBs NOT kg	admission assessment form		
18				pregnancy risk factors	GDM	dta	x	
19					Pre-pregnancy Diabetes	dta	x	
20					G HTN	dta	x	
21					HTN prior to pregnancy	dta	x	
22					previous preterm delivery	PCM Pregnancy History cont	x	
23					previous poor pregnancy outcome	PCM Pregnancy History cont	x	
24					pregnancy resulted from infertility treatm	DTA	x	
25					vaginal bleeding prior to onset of labor	DTA	x	
26					prev CS	PCM Pregnancy History cont	x	
27					# of prev CS	PCM Pregnancy History control		
28				Infections	Herpes	dta	x	
29					Chlamydia	dta	x	
30					Gonorrhae	dta	x	

Sheet1 | Birth cert | del form | operative del | shoulder | MD assessment | Anesth | powern ...

Format Painter | Clipboard | Font | Alignment | Number | Formatting | Table | Styles | Cells | Clear | Filter | Select | Editing

C56 | spontaneous labor augmented

	A	B	C	D	E
1		DTA type			
2					
3					
4					
5					
6					
7					
8					
9			chorio		659.33 ... antepartum condition or complication
10			prolonged ROM	SROM	658.21 ... delivered, with or without mention of antepartum condition
11				AROM	658.31 ... delivered, with or without mention of antepartum condition
12			did this fetus require intrauterine resuscitive measures durnig labor?	y/n	
13					
14					
15			if GBS pos	Treated< 4 hours prior to delivery	
16				Treated > 4 hours prior to delivery	
17				untreated	
18			Past pregnancy complications	None	
19				3-4th degree lac	
20				Blood transfusion	
21				Eclampsia	
22				CS	
23				PPH	
24				GDM	
25				Fetal anomaly	
26				Pre-eclampsia	
27				Pre-gest DM	
28				Pre-term labor	
29				PPROM	

Sheet1 | Birth cert | del form | operative del | shoulder | MD assessment | Anesth | powern(...

Format Painter | Clipboard | Font | Alignment | Merge & Center | Number | Formatting | Table | Styles | Cells | Clear | Filter | Select | Editing

G158 | x

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	DTA Type		Section Name	DTA Display Name	Nomenclature	ICD-9 Code	L+D ST	del note	Anesth	PP ST	Antepartum	Pass to peds	Pu fo thi
2	DTA type	source					Demographic						
3	Calculation:		Essential elements	Age			Gyn	x	x	x	x	x	x
4	Calculation:			G			Infectious	x	x			x	x
5	Calculation:			P			Hematological	x	x				x
6	Calculation:			EDC			Medical	x	x	x	x	x	x
7	Alpha:			multiple Gestation	Singleton		Social	x	x	x	x	x	x
8					twins	651.03 ... antepartum condition or complication	OB	x	x	x	x	x	x
9					triplets	651.43 ... antepartum condition or complication		x	x	x	x	x	x
10					quads	651.23 ... antepartum condition or complication		x	x	x	x	x	x
11	Numeric:		for each fetus (per multiples above)	EFW (Estimated Fetal Weight)				x	x		x		x
12	Alpha and Freetext:		for each fetus (per multiples above)	Position	vertex/breech/etc.			x	x	x	x	x	x
13	Alpha:			Cx Exam	dilation/effacement /station		x		x		x		
14	Numeric:			BP			x	x	x	x	x		
15	Alpha and Freetext:			pre-E sympt	HA		x	x	x	x	x		
16					Vis changes		x	x	x				
17					N/V		x	x	x				
18					Abd pain								
19													

Pitfalls in Relational Database Design

- Relational database design requires that we find a "good" collection of relation schemas (tables).
- A bad design may lead to
  - Repetition of Information.
  - Inability to represent certain information.

Design Goals

- Avoid redundant data
- Ensure that relationships among **entities** are represented
- Ensure that relationships among **attributes** are represented
- Facilitate the checking of updates for violation of database integrity constraints.

Example

- Consider the relation schema:

Lending-schema = (branch-name, branch-city, assets, customer-name, loan-number, amount)

branch-name	branch-city	assets	customer-name	loan-number	amount
Downtown	Brooklyn	9000000000	Jones	L-17	1000
Redwood	Palo Alto	2100000000	Smith	L-23	2000
Perryridge	Horseneck	1700000000	Hayes	L-15	1500
Downtown	Brooklyn	9000000000	Jackson	L-14	1500

What's Wrong?

branch-name	branch-city	assets	customer-name	loan-number	amount
Downtown	**Brooklyn**	**9000000000**	Jones	L-17	1000
Redwood	Palo Alto	2100000000	Smith	L-23	2000
Perryridge	Horseneck	1700000000	Hayes	L-15	1500
Downtown	**Brooklyn**	**9000000000**	Jackson	L-14	1500

- Redundancy:
  - Data for *branch-name, branch-city,* assets are repeated for each loan that a branch makes
  - Wastes space
  - Complicates updating, introducing possibility of inconsistency of *assets* value

What's Wrong?

branch-name	branch-city	assets	customer-name	loan-number	amount
Downtown	Brooklyn	9000000000	Jones	L-17	1000
Redwood	Palo Alto	2100000000	Smith	L-23	2000
Perryridge	Horseneck	1700000000	Hayes	L-15	1500
Downtown	Brooklyn	9000000000	Jackson	L-14	1500
Midtown	**Pittsburgh**	**2349000000**	**NULL**	**NULL**	**NULL**

- Null values
  - Cannot store information about a branch if no loans exist
  - Can use null values, but they are difficult to handle.

Decomposition

- Decompose the relation schema *Lending-schema* into:
  - *Branch-schema = (branch_name, branch_city, assets)*
  - *Loan-info-schema = (customer_name, loan_number, branch_name, amount)*
- All attributes of an original schema (*R)* must appear in the decomposition (R_1, R_2): $\boldsymbol{R = R_1 \cup R_2}$

Decompositon

branch-name	branch-city	assets	customer-name	loan-number	amount
Downtown	Brooklyn	9000000000	Jones	L-17	1000
Redwood	Palo Alto	2100000000	Smith	L-23	2000
Perryridge	Horseneck	1700000000	Hayes	L-15	1500
Downtown	Brooklyn	9000000000	Jackson	L-14	1500

branch-name	branch-city	assets
Downtown	Brooklyn	9000000000
Redwood	Palo Alto	2100000000
Perryridge	Horseneck	1700000000

branch-name	customer-name	loan-number	amount
Downtown	Jones	L-17	1000
Redwood	Smith	L-23	2000
Perryridge	Hayes	L-15	1500
Downtown	Jackson	L-14	1500

Dependency

- A dependency occurs in a database when information stored in the same database table uniquely determines other information stored in the same table.
- You can also describe this as a relationship where knowing the value of one attribute (or a set of attributes) is enough to tell you the value of another attribute (or set of attributes) in the same table.

Normalization

- Normalization is the process of efficiently organizing data in a database.
- Goals of the normalization process:
  - Eliminating redundant data
  - Ensuring that data dependencies make sense (good relationships)

Normal Forms

- A series of guidelines for ensuring that databases are normalized.
- Numbered from one (the lowest form of normalization, referred to as first normal form or 1NF) through five (fifth normal form or 5NF).
- In practical applications, you'll often see 1NF, 2NF, and 3NF, with only the occasional 4NF.

Normal Forms

- Fourth normal form is rarely seen.
- Fifth normal form is almost never seen.

Caveat…

- Normal forms are **guidelines** and guidelines only.
- Sometimes, it becomes necessary to stray from them to meet practical business requirements.
- When variations take place, it is still extremely important to evaluate any possible ramifications they could have on your system and account for possible inconsistencies.

First Normal Form (1NF)

- Eliminate duplicate columns from the same table.
- Create separate tables for each group of related data and identify each row with a unique column or set of columns (the primary key).

Eliminate duplicate columns from the same table.

- Referred to as the atomicity of a table.
- Tables that comply with this rule are said to be atomic.
- Let's explore this principle with a classic example - a table within a human resources database that stores the manager-subordinate relationship.
- For the purposes of our example, we'll impose the business rule that each manager may have one or more subordinates while each subordinate may have only one manager.

Example – Step 1

Assume that that each manager may have one or more subordinates while each subordinate may have only one manager.

manager	subordinate_1	subordinate_2	subordinate_3	subordinate_4
John	Mary	Josh	David	Jane

Example – Step 2

manager	subordinates
John	Mary, Josh, David, Jane

Example – Step 2

manager	subordinates
John	Mary, Josh, David, Jane

- The subordinates column is still duplicative and non-atomic.
- What happens when we need to add or remove a subordinate?

Example – Step 3

- Here's a table that satisfies the first rule of 1NF:

manager	subordinates
John	Mary
John	Josh
John	David
John	Jane

- In this case, each subordinate has a single entry, but managers may have multiple entries.

Example – Step 4

- Remember the second rule of 1NF: identify each row with a unique column or set of columns (the primary key)?

manager_id	manager	subordinate_id	subordinate
5	John	1	Mary
5	John	2	Josh
5	John	3	David
5	John	4	Jane

Second Normal Form (2NF)

- Meet all the requirements of the first normal form.
- Remove subsets of data that apply to multiple rows of a table and place them in separate tables.
- Create relationships between these new tables and their predecessors through the use of foreign keys.

2NF Example

2NF attempts to reduce the amount of redundant data in a table by extracting it, placing it in new table(s) and creating relationships between those tables.

employees	
employee_id	**name**
1	Mary
2	Josh
3	David
4	Jane
5	John

manager_subordinates	
manager_id	**subordinate_id**
5	1
5	2
5	3
5	4

2NF Example

You were hired to develop a POS system. Your client gave you a spreadsheet that contains customers' information. How do you convert it to database entities/tables that comply with 2NF?

Customers						
CustNum	FirstName	LastName	Address	City	State	Zip
1	Mary	Doe	743 Evergreen St.	Pittsburgh	PA	15217
2	Josh	Smith	134 Phillips Avenue	Pittsburgh	PA	15217
3	David	Burke	456 Hobart Street	Pittsburgh	PA	15217
4	Jane	Brown	7645 Liberty Ave.	Pittsburgh	PA	15222
5	John	Black	134 Phillips Avenue	Pittsburgh	PA	15217

2NF Example

What’s wrong with just leaving the table as-is?

Customers						
CustNum	FirstName	LastName	Address	City	State	Zip
1	Mary	Doe	743 Evergreen St.	Pittsburgh	PA	15217
2	Josh	Smith	**134 Phillips Avenue**	**Pittsburgh**	**PA**	**15217**
3	David	Burke	456 Hobart Street	Pittsburgh	PA	15217
4	Jane	Brown	7645 Liberty Ave.	Pittsburgh	PA	15222
5	John	Black	**134 Phillips Avenue**	**Pittsburgh**	**PA**	**15217**

2NF Example

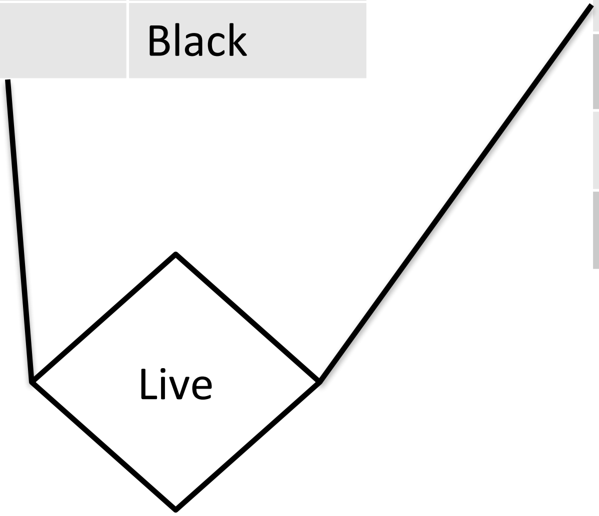
- In a 2NF-compliant database structure, this redundant information is extracted and stored in a separate table.
- We'll create two tables – ***Customers*** and ***Addresses***

Customers						
CustNum	FirstName	LastName	Address	City	State	Zip
1	Mary	Doe	743 Evergreen St.	Pittsburgh	PA	15217
2	Josh	Smith	**134 Phillips Avenue**	**Pittsburgh**	**PA**	**15217**
3	David	Burke	456 Hobart Street	Pittsburgh	PA	15217
4	Jane	Brown	7645 Liberty Ave.	Pittsburgh	PA	15222
5	John	Black	**134 Phillips Avenue**	**Pittsburgh**	**PA**	**15217**

2NF Example

Customers		
CustNum	FirstName	LastName
1	Mary	Doe
2	Josh	Smith
3	David	Burke
4	Jane	Brown
5	John	Black

Addresses				
AddressID	Address	City	State	Zip
1	743 Evergreen St.	Pittsburgh	PA	15217
2	134 Phillips Avenue	Pittsburgh	PA	15217
3	456 Hobart Street	Pittsburgh	PA	15217
4	7645 Liberty Ave.	Pittsburgh	PA	15222



But wait… There is more!

Customers

CustNum	FirstName	LastName
1	Mary	Doe
2	Josh	Smith
3	David	Burke
4	Jane	Brown
5	John	Black

Live

Addresses

AddressID	Address
1	743 Evergreen St.
2	134 Phillips Avenue
3	456 Hobart Street
4	7645 Liberty Ave.

ZipCodes

ZipID	City	State	Zip
1	Pittsburgh	PA	15217
3	Pittsburgh	PA	15222

Located in

Third Normal Form (3NF)

- Meet all the requirements of the second normal form.
- Remove columns that are not dependent upon the primary key.

Example – 3NF

- Consider the following table that's part of our POS system.
- This table contains information about customer orders.

order_num	cust_num	unit_price	quantity	total
X43565	1	2.75	4	11.00
Y43525	2	1000.00	28	28000.00
U43746	3	53.07	6	318.42
L86549	4	100.00	154	15400.00

Example – 3NF

Requirements of 1NF.

- Are there any duplicative columns?
- Do we have a primary key?

Requirements of 2NF.

- Are there any subsets of data that apply to multiple rows?

order_num	cust_num	unit_price	quantity	total
X43565	1	2.75	4	11.00
Y43525	2	1000.00	28	28000.00
U43746	3	53.07	6	318.42
L86549	4	100.00	154	15400.00

Example – 3NF

Are all of the columns fully dependent upon the primary key?

order_num	cust_num	unit_price	quantity	total
X43565	1	2.75	4	11.00
Y43525	2	1000.00	28	28000.00
U43746	3	53.07	6	318.42
L86549	4	100.00	154	15400.00

Example – 3NF

The **customer number** varies with the order number and it doesn't appear to depend upon any of the other fields.

order_num	cust_num	unit_price	quantity	total
X43565	**1**	2.75	4	11.00
Y43525	**2**	1000.00	28	28000.00
U43746	**3**	53.07	6	318.42
L86549	**4**	100.00	154	15400.00

Example – 3NF

- **Unit price** could be dependent upon the customer number if we charged each customer a set price.
- However, we could sometimes charge the same customer different prices.
- Therefore, the **unit price** is fully dependent upon the order number.

order_num	cust_num	unit_price	quantity	total
X43565	1	**2.75**	4	11.00
Y43525	2	**1000.00**	28	28000.00
U43746	3	**53.07**	6	318.42
L86549	4	**100.00**	154	15400.00

Example – 3NF

- The **quantity** of items also varies from order to order, so it is dependent of order_num

order_num	cust_num	unit_price	quantity	total
X43565	1	2.75	**4**	11.00
Y43525	2	1000.00	**28**	28000.00
U43746	3	53.07	**6**	318.42
L86549	4	100.00	**154**	15400.00

Example – 3NF

The **total** can be derived by multiplying the unit price by the quantity, therefore it's **NOT** fully dependent upon the primary key.

order_num	cust_num	unit_price	quantity	total
X43565	1	2.75	4	**11.00**
Y43525	2	1000.00	28	**28000.00**
U43746	3	53.07	6	**318.42**
L86549	4	100.00	154	**15400.00**

Example – 3NF

- We must remove **total** from the table to comply with the third normal form.

order_num	cust_num	unit_price	quantity
X43565	1	2.75	4
Y43525	2	1000.00	28
U43746	3	53.07	6
L86549	4	100.00	154

Example – 3NF

Before 3NF normalization:

- SELECT order_num, **total** FROM orders;

After 3NF normalization:

- SELECT order_num, **unit_price * quantity AS total** FROM orders;

order_num	cust_num	unit_price	quantity
X43565	1	2.75	4
Y43525	2	1000.00	28
U43746	3	53.07	6
L86549	4	100.00	154