BZAN 6354

Lecture 6

February 26, 2024

Dr. Mark Grimes, Ph.D.
gmgrimes@bauer.uh.edu

UNIVERSITY of HOUSTON
C. T. BAUER COLLEGE of BUSINESS
Department of Decision & Information Sciences

Agenda

- Quick Review
- Module 6.3 – Data Integrity Constraints (Continued…)
- Module 6.4 – Relational Algebra and sets
- Module 6.5 – Views and Materialized Views
- Break
- Module 6.7 – Logical Modeling
- Exam Discussion

Last time we talked about…

- Attribute Placement
  - Attributes mean different things depending on what entity they are part of, OR if they are an attribute of a relationship!
- Weak Entities / Partial Keys / Identifying relationships
  - Weak entities do not exist without the strong/base entity that “identifies” them
  - Weak entities have a partial key that is NOT unique by itself
- Decomposing M:N Relationships with the Gerund
  - Primary key from each participating relation become a foreign key in the gerund
- Data modeling errors
  - Syntactical Errors: Using the grammar incorrectly, relatively easy to spot
  - Semantic Errors: Not modeling the business case correctly, you must understand the case to spot these

Last time we talked about…

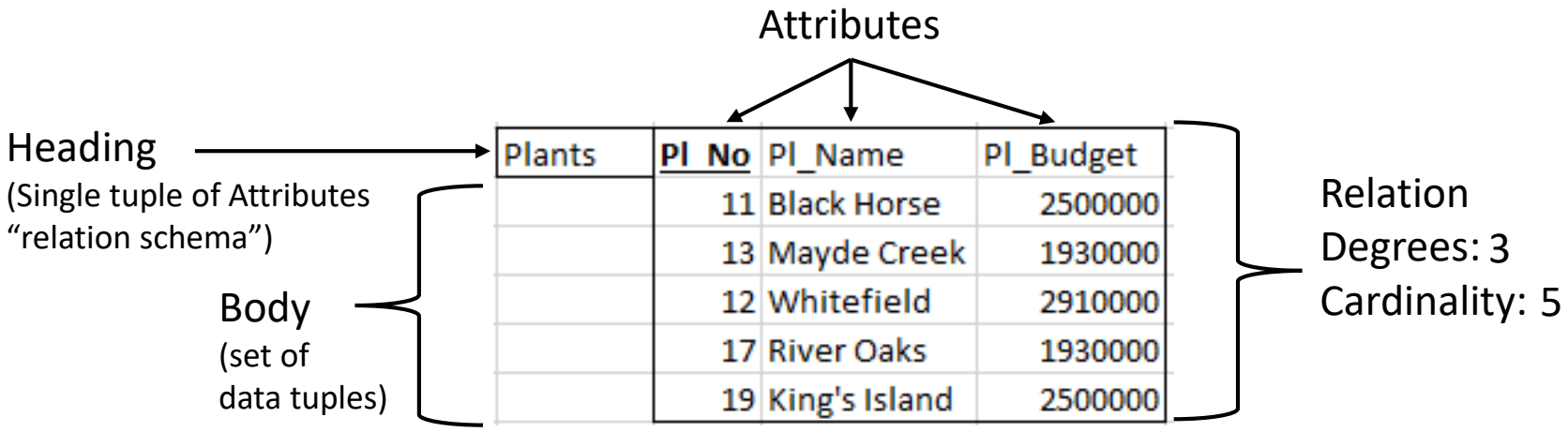
- Relational modeling
  - Degree of a relation: Number of attributes
  - Cardinality of a relation: Number of tuples
  - Characteristics of a relation

- The first part of “Data Integrity Constraints”
  - Super Keys: Unique
  - Candidate Keys: Unique and Irreducible
  - Primary Key: Unique, Irreducible, and cannot be NULL (Entity Integrity Constraint)
  - Alternate Keys: Candidate keys not selected as primary key
  - Key attribute: Proper subset of a candidate key
  - Non-key attribute: Not a subset of a candidate key

- …And we ran out of time before finishing integrity constraints and talking about foreign key placement!

Review: What is a relation?

- A mathematical terms approximated by a two dimensional table:
  - Heading – a single tuple listing the attributes (Relation Schema)
  - Body – collection of data tuples



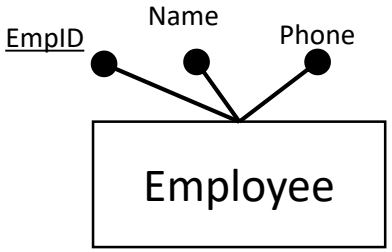
Plants	Pl_No	Pl_Name	Pl_Budget
	11	Black Horse	2500000
	12	Whitefield	2910000
	17	River Oaks	1930000
	19	King's Island	2500000

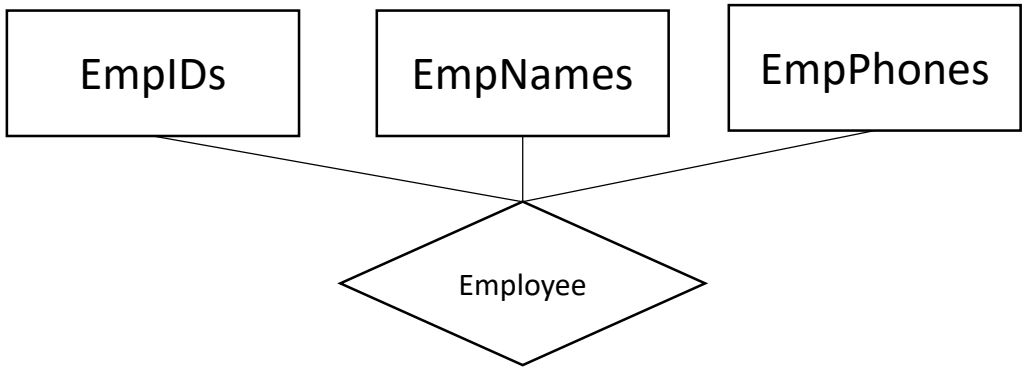
Relation
Degrees: 3
Cardinality: 4

Review: What is the degree of a relation?

- Degree is the number of attributes (columns) in a relation
  - But why?

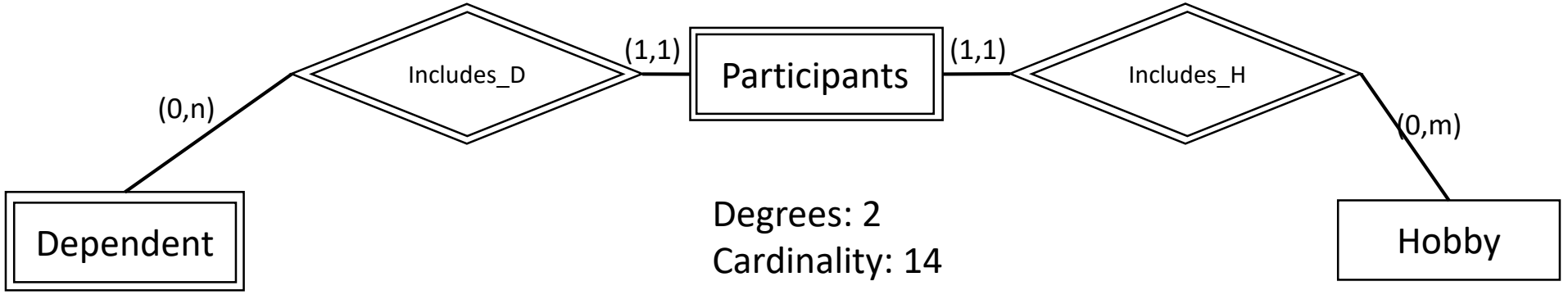
- Imagine this: each attribute of an entity can be considered an entity that is associated with the other attributes…





- We would never model it this way
  - …but this is how degree describes the number of attributes

Review: What is the cardinality of a relation?



Degrees: 2
Cardinality: 14

Degrees: 4
Cardinality: 5

Dependents

EmpID	Name	Birthdate	Gender
100	Mark	5/11/1975	M
101	Jill	5/4/1976	F
102	Norm	3/1/1984	M
103	Mike	1/31/1992	M
104	Sue	9/2/1990	F

Participants

EmpID	HobbyID
100	1
100	2
100	4
101	3
101	5
102	1
103	1
103	2
103	3
103	4
103	5
104	1
104	4
104	5

Degrees: 4
Cardinality: 5

Hobby

HobbyID	HobbyName	In_Act	Gr_Act
1	Soft Ball	O	G
2	Flag Football	O	G
3	Knitting	I	I
4	Cycling	O	I
5	Movies	I	G

Exam 1 – Next week

- March 4
  - In class on paper
  - 75 minutes
  - Closed Book / Individual Effort

- Will be approximately one third each of:
  - Multiple choice
  - Short answer / Matching / Fill in the Blank / etc.
  - Creating ERDs

Module 6.3
Data Integrity Constraints

- Be able to define: Super keys, candidate keys, key attributes, non-key attributes, primary keys, and alternate keys (Already Done)
- What is the entity integrity constraint? (Already done)
- What is the referential integrity constraint?
- What is the foreign key constraint?

Referential Integrity Constraint

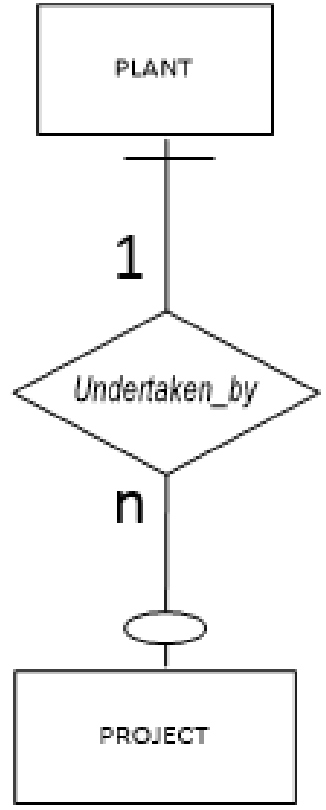
- Key constraints (superkey and candidate key) and entity integrity constraint (primary key) pertain to individual relation schemas
- Referential integrity constraints are specified between two relation schemas (i.e., R1 and R2)
- Specifically, a referential integrity constraint is specified between two relations in order to maintain consistency across tuples of the two relations
- Informal definition: A tuple in one relation that refers to another relation must refer to an existing tuple in that relation.

Foreign Key Constraint

- A special form of referential integrity constraint specification
- Establishes an explicit association between two relation schemas and maintains the integrity of such an association
- Foreign key: An attribute(s) set, A2, in a relation schema R2 that shares the same domain with a **candidate key** (A1) of another relation schema R1; A2 is said to reference or refer to the relation schema R1.
  - Note: R2 is known as the referencing relation and R1 is called the referenced relation. The attribute(s) doing the referencing (A2 in R2) is the foreign key, while the candidate key (**advisably, the primary key**) being referenced (A1 in R1) is the referenced attribute(s).
- Referred to as Inclusion Dependency, this constraint is algebraically expressed as: **R2.{A2} ⊆ R1.{A1}**
- Meaning: Child.foreignkey (R2.{A2}) is inclusion dependent on parent.primarykey (R1.{A1})

Example – Source Schema

- Bearcat Incorporated is a manufacturing company that has several plants in the northeastern part of the United States. These plants are responsible for leading different projects that the company might undertake, depending on a plants' function. A certain plant might even be associated with several projects but a project is always under the control of just one plant. Some plants do not undertake any projects at all.

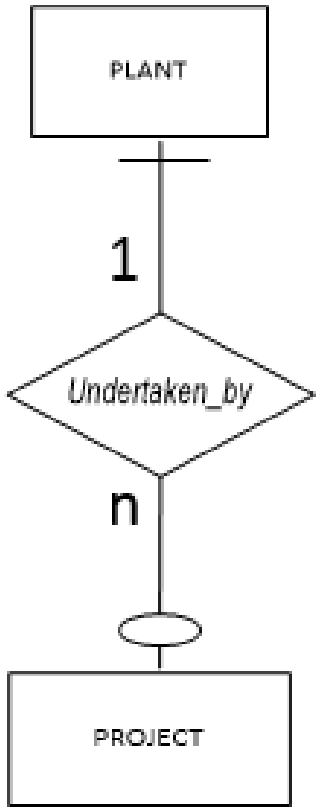


FK in the 1:n relationship

PLANT	Pl_name	Pl_p#	Pl_budget
	Black Horse	11	1230000
	Mayde Creek	13	1930000
	Whitefield	12	2910000
	River Oaks	17	1930000
	King's Island	19	2500000
	Ashton	15	2500000

Foreign Key

PROJECT	Prj_name	Prj_n#	*Prj_pl_p#*
	Solar Heating	41	11
	Lunar Cooling	17	17
	Synthetic Fuel	29	17
	Nitro-Cooling	23	12
	Robot Sweeping	31	11
	Robot Painting	37	19
	Ozone Control	13	19



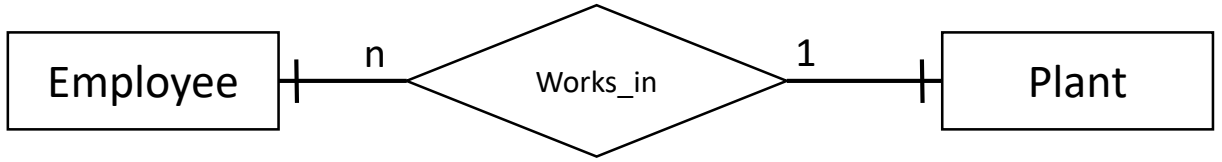
R2.{A2} ⊆ R1.{A1}

Project.Prj_pl_P# ⊆ Plant.Pl_p#

Note: PROJECT.**Prj_pl_p#** is the foreign key referencing PLANT.**Pl_p#**, the primary key of PLANT.

Note: Prj_n# would be a better primary key for the Project entity. We'll talk about why later, but this works for this example

Foreign key placement



- What is the cardinality of the relationship? 1:n
- Which entity is the parent? Plant
- In a 1:M relationship, Foreign key (FK) goes with the "child" side
  - The FK is an attribute in the child entity (employees in this case) that refers to the primary key of the entity on the other side of the relationship

Employees

EmpID_A	**EmpID_N**	Fname	Minit	Lname	NameTag	PL_No
E	12345	Adam	B	Christie		10
E	22233	Danny	E	Francisco		10
C	32112	Greg		Hernandez	1	10
E	43210	Ivana	J	Klink		11
E	52525	Greg		Hernandez	2	11

Plants

PL_No	PL_Name	Budget
10	Underwood	3000000
11	Garnett	3000000
12	Belmont	3500000
13	Vanderbilt	3500000

- Would this even make sense the other way?

Note this is dataset is truncated. In reality every entity in the plant table must have at least 100 employees (as specified in the business rule – the ERD above merely specifies that it must participate)

Note: Employees.PL_No is a FK that refers to Plants.PL_No. All values of Employees.PL_No must be found in the domain of values for Plants.PL_No (i.e., must be a valid plant)

In 1:m The FK must go with the child entity

- If we put the FK with the parent it is INCORRECT:
  - Lots of data redundancy in the Plants table if the FK is there

Employees

EmpID_A	EmpID_N	Fname	Minit	Lname	NameTag
E	12345	Adam	B	Christie	
E	22233	Danny	E	Francisco	
C	32112	Greg		Hernande	1
E	43210	Ivana	J	Klink	
E	52525	Greg		Hernande	2

Plants

PL_No	PL_Name	Budget	EmpID
10	Underwood	3000000	E12345
10	Underwood	3000000	E22233
10	Underwood	3000000	C32112
11	Garnett	3000000	E43210
11	Garnett	3000000	E52525
12	Belmont	3500000	**NULL**
13	Vanderbilt	3500000	**NULL**

Note: In this case, Plants.EmpID is a FK that refers to the composite attribute Employees.EmpID_A + Employees.EmpID_N.

There is now redundancy for the Plants.PL_Name and Plants.Budget attributes, **and** PL_No is no longer unique!

- No redundancy when FK is with Employees (CORRECT):

Employees

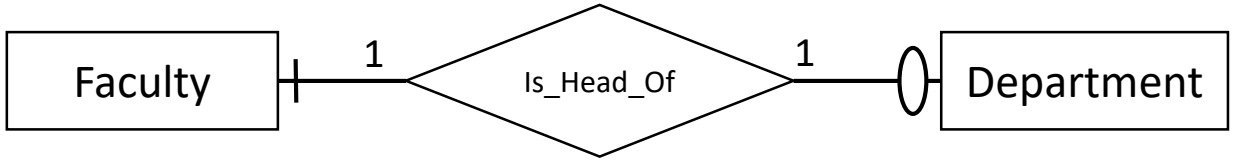
EmpID_A	EmpID_N	Fname	Minit	Lname	NameTag	PL_No
E	12345	Adam	B	Christie		10
E	22233	Danny	E	Francisco		10
C	32112	Greg		Hernandez	1	10
E	43210	Ivana	J	Klink		11
E	52525	Greg		Hernandez	2	11

Plants

PL_No	PL_Name	Budget
10	Underwood	3000000
11	Garnett	3000000
12	Belmont	3500000
13	Vanderbilt	3500000

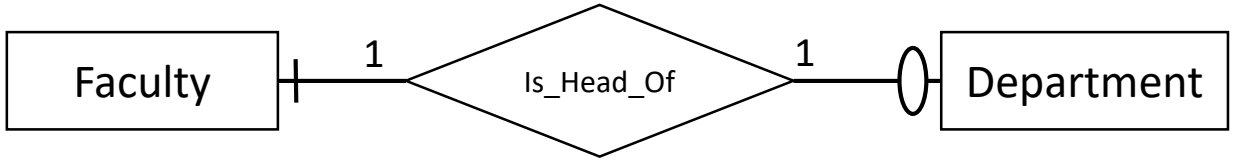
Where do we put the foreign key in a 1:1 relationship?

- There is no clear parent….



In 1:1 we place the FK based on participation

- FK goes with the entity that has total (mandatory) participation



Faculty		
EmpID	Name	Title
112	Chad Larson	Assistant Professor
108	Dusya Vera	Professor
109	Edward Blair	Professor
111	Gerald Lobo	Professor
106	Kris Jacobs	Professor
102	Mark Grimes	Assistant Professor
110	Melanie Rudd	Assistant Professor
103	Michael Parks	Associate Professor
101	Norm Johnson	Professor
104	Praveen Kumar	Professor
107	Steve Werner	Professor
105	Thomas George	Professor

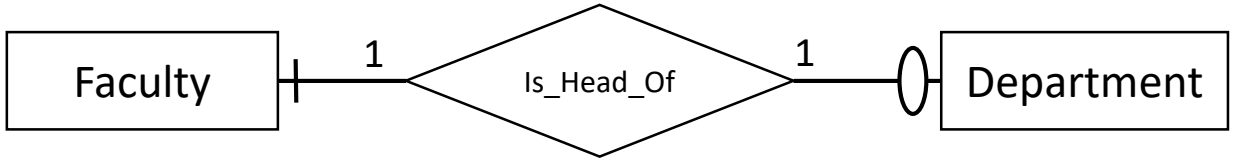
Departments		
DeptID	Deptname	Head
1	Accounting	111
2	DISC	101
3	Finanace	104
4	Management	107
5	Marketing	109

Note: In this case, Departments.Head is a FK that refers to Faculty.EmpID.

CORRECT WAY!

In 1:1 we place the FK based on participation

- Imagine if we did it the other way, it still works, but lots of NULL values!
  - NULL values are difficult to work with – avoid them as much as possible!



Faculty			
EmpID	Name	Title	Head of
112	Chad Larson	Assistant Professor	
108	Dusya Vera	Professor	
109	Edward Blair	Professor	5
111	Gerald Lobo	Professor	1
106	Kris Jacobs	Professor	
102	Mark Grimes	Assistant Professor	
110	Melanie Rudd	Assistant Professor	
103	Michael Parks	Associate Professor	
101	Norm Johnson	Professor	2
104	Praveen Kumar	Professor	3
107	Steve Werner	Professor	4
105	Thomas George	Professor	

Departments	
DeptID	Deptname
1	Accounting
2	DISC
3	Finanace
4	Management
5	Marketing

Note: In this case, Faculty.HeadOf is a FK that refers to Departments.DeptID.

No data redundancy, but lots of nasty NULL values

WRONG WAY!

In 1:1 we place the FK based on participation

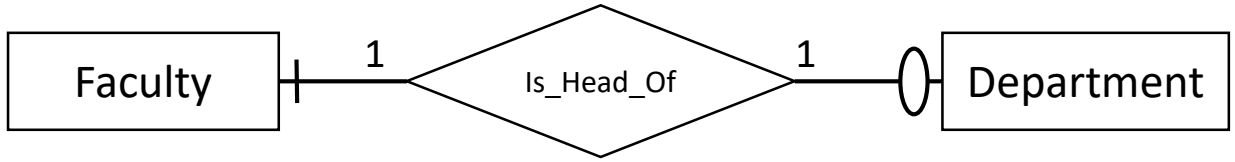
- There are some fringe cases…

- If both sides of a 1:1 have mandatory participation
  - It doesn't matter where you place the FK
  - This would be a pretty rare thing to find

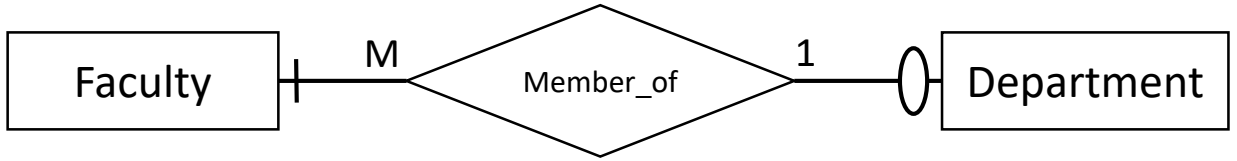
- If both sides of a 1:1 have optional participation
  - No hard and fast rule, but generally would be better to put it with the entity that will have fewer instances
  - Alternatively, you can use a gerund to ensure you will have no NULL values!

Watch out!

- The “is head of” relationships is different than the “member of” relationship:



- Where should the FK go for the “Member of” relationship? Faculty (The Child)



Note: In this case, Faculty.Department is a FK that refers to Departments.DeptID - this is the “Member_of relationship.

Departments.Head is a FK that refers to Faculty.EmpID - this is the “Is_Head_of” relationship.

We have no redundancy and no NULL values with this design

Faculty

EmpID	Name	Title	Department
112	Chad Larson	Assistant Professor	1
108	Dusya Vera	Professor	4
109	Edward Blair	Professor	5
111	Gerald Lobo	Professor	1
106	Kris Jacobs	Professor	3
102	Mark Grimes	Assistant Professor	2
110	Melanie Rudd	Assistant Professor	5
103	Michael Parks	Associate Professor	2
101	Norm Johnson	Professor	2
104	Praveen Kumar	Professor	3
107	Steve Werner	Professor	4
105	Thomas George	Professor	3

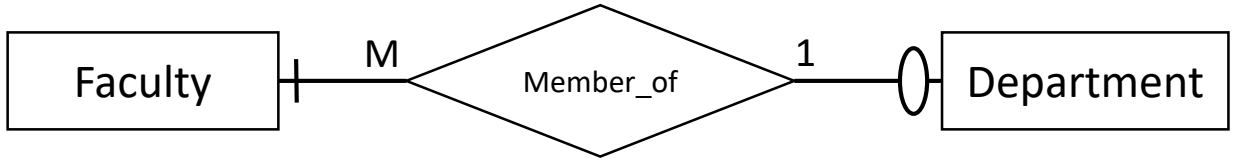
Departments

DeptID	Deptname	Head
1	Accounting	111
2	DISC	101
3	Finanace	104
4	Management	107
5	Marketing	109

CORRECT WAY!

Watch out!

- What if we did it the wrong way?



Faculty

EmpID	Name	Title
112	Chad Larson	Assistant Professor
108	Dusya Vera	Professor
109	Edward Blair	Professor
111	Gerald Lobo	Professor
106	Kris Jacobs	Professor
102	Mark Grimes	Assistant Professor
110	Melanie Rudd	Assistant Professor
103	Michael Parks	Associate Professor
101	Norm Johnson	Professor
104	Praveen Kumar	Professor
107	Steve Werner	Professor
105	Thomas George	Professor

Departments

DeptID	Deptname	Head	EmpID
1	Accounting	111	111
1	Accounting	111	112
2	DISC	101	101
2	DISC	101	102
2	DISC	101	103
3	Finanace	104	104
3	Finanace	104	105
3	Finanace	104	106
4	Management	107	107
4	Management	107	108
5	Marketing	109	109
5	Marketing	109	110

Redundant data for the Deptname and Head attributes **AND** DeptID is no longer unique!

WRONG WAY!

Module 6.3
Data Integrity Constraints

- Be able to define: Super keys, candidate keys, key attributes, non-key attributes, primary keys, and alternate keys
- What is the entity integrity constraint?
- What is the referential integrity constraint?
- What is the foreign key constraint?

Module 6.4
Relational Algebra

- What are the eight operations in relational algebra?
- Describe the difference in select and project
- Describe the difference in the Union, Intersection, and Difference operations
- What is a natural join?

Relational algebra primer

- Relational algebra just lets us abstractly talk about relations and data (like algebra lets us abstractly talk about numbers)

- We can substitute symbols to represent data and do abstract calculations just like in traditional algebra
  - Area of a room is L x W
    - L = 10 W=20 A = L x W = ? → A = 10 x 20 = 200

  - If we want a room that is 4 times as large, we can nest our equations
    - 4(A) → 4(L x W) → 4(10 x 20) → 4(200) → 800

- For databases
  - We consider a relation (R) as being made up of attributes A_1, A_2, A_3,…A_n
  - The relation schema is R(A_1, A_2, A_3,…A_n)

Relational algebra primer

- An attribute is defined as an ordered set (N, D)
  - N is the name of the attribute
  - D is the domain of the attribute

- A set of attributes can be represented as a vector (C)
  - C is the set $\{(N_1,D_1), (N_2,D_2), (N_3,D_3),\ldots(N_n,D_n)\}$

Relational algebra primer

- So…. A relation is a set (R,C) where R is the name of the relation schema and C is the list of attributes that make up R

- You database is made up of multiple relations (R,C)

- Each (R,C) is equivalent to…
  - $R(A_1, A_2, A_3,\ldots A_n)$ which is equivalent to …
  - $R(\{(N_1,D_1), (N_2,D_2), (N_3,D_3),\ldots(N_n,D_n)\})$

Relational algebra primer

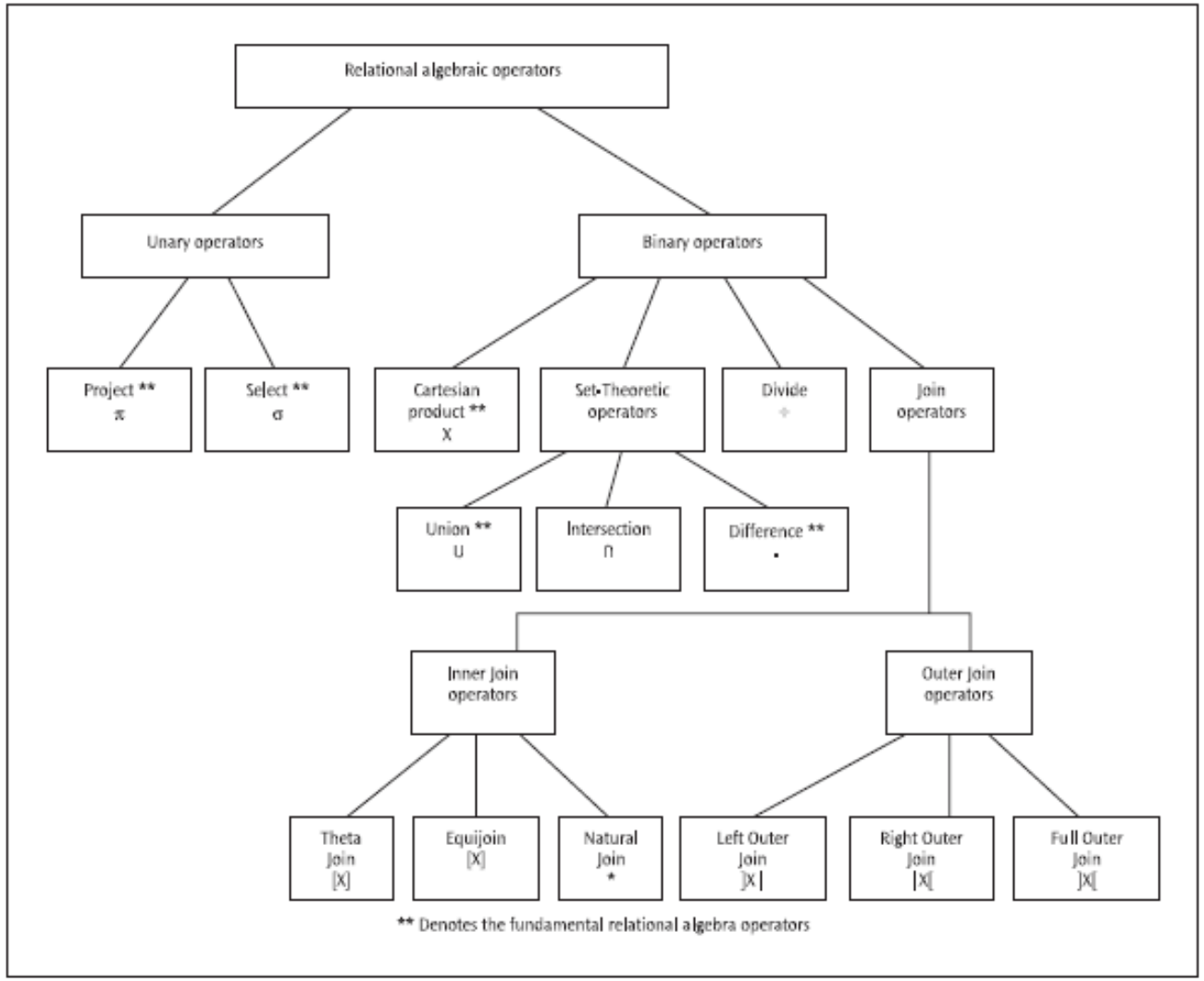


Figure 11.2 Classification of relational algebra operators

Eight basic functions of relational algebra

- Two unary operations
  - Selection (σ)
  - Projection (π)
- Six binary operators
  - Union (U)
  - Intersection ($\cap$)
  - Difference (-)
  - Join ($\bowtie$)
  - Cartesian product (X)
  - Division ($\div$)

(We'll talk about Cartesian product and division after spring break)

Imagine these three relations

- Award Winning Plants

AW_PLANT		
Aw_pl_No	Aw_pl_Name	Aw_pl_Budget
11	Black Horse	2500000
13	Mayde Creek	1930000
12	Whitefield	2910000
17	River Oaks	1930000
19	King's Island	2500000
15	Ashton	2500000

- Texas Plants

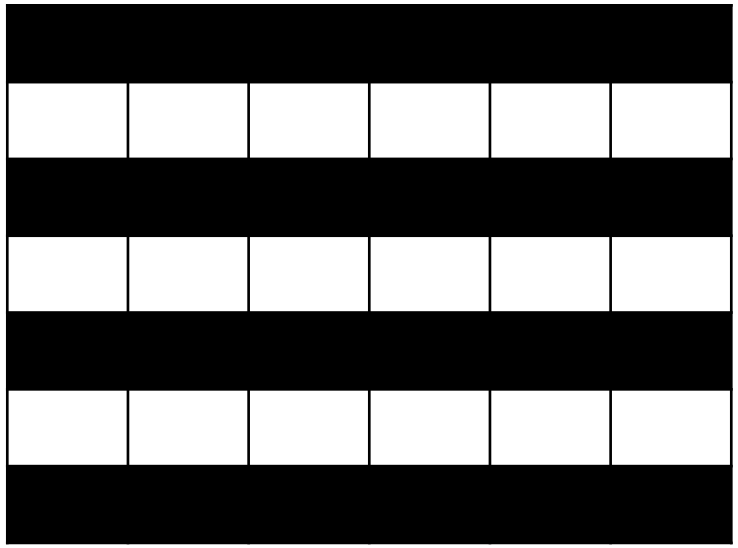
TX_PLANT		
Tx_pl_No	Tx_pl_Name	Tx_pl_Budget
16	Southern Oaks	1930000
17	River Oaks	1930000
18	Kingwood	1930000

- Projects

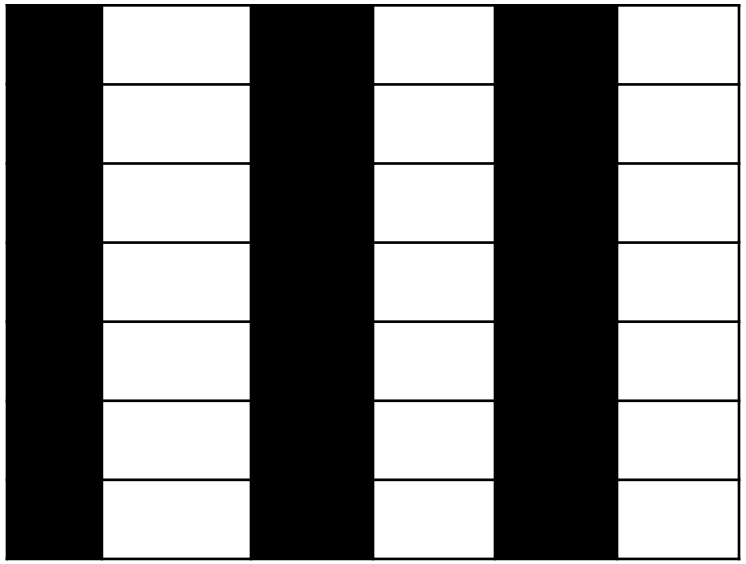
PROJECTS			
Prj_name	Prj_no	Prj_location	Prj_aw_pl_no
Solar Heating	41	Sealy	11
Lunar Cooling	17	Yoakum	17
Synthetic Fuel	29	Salem	17
Nitro-Cooling	23	Parthi	12
Robot Sweeping	31	Ponca City	11
Robot Painting	37	Poakum	19
Ozone Control	13	Parthi	19

Unary Operations

Selection (σ)



Projection (π)



Select Operator

- Selects a horizontal subset of tuples that satisfy a selection condition from the relation

$$\sigma_{\text{<selection condition>}}R$$

- Lower case sigma (σ) designates “select”

- <selection condition> is a Boolean expression specified on the attributes of relation R

Selection (Unary Operation)

- Creates a second relation by extracting a subset of tuples
- Question: Which award winning plants have a budget over $2,000,000
- $\sigma_{\text{Aw\_pl\_Budget} > 2000000}$ AW_Plant
- SELECT * FROM AW_Plant WHERE Aw_pl_Budget > 2000000

AW_PLANT		
Aw_pl_No	Aw_pl_Name	Aw_pl_Budget
11	Black Horse	2500000
13	Mayde Creek	1930000
12	Whitefield	2910000
17	River Oaks	1930000
19	King's Island	2500000
15	Ashton	2500000

AW_PLANT		
Aw_pl_No	Aw_pl_Name	Aw_pl_Budget
11	Black Horse	2500000
12	Whitefield	2910000
19	King's Island	2500000
15	Ashton	2500000

Projection Operator

- Selects a vertical subset of attributes from a relation

$$\pi_{\text{<attribute list>}}R$$

- Lower case pi (π) designates “project”
- <attribute list> is a subset of attributes of relation R

Projection (Unary Operation)

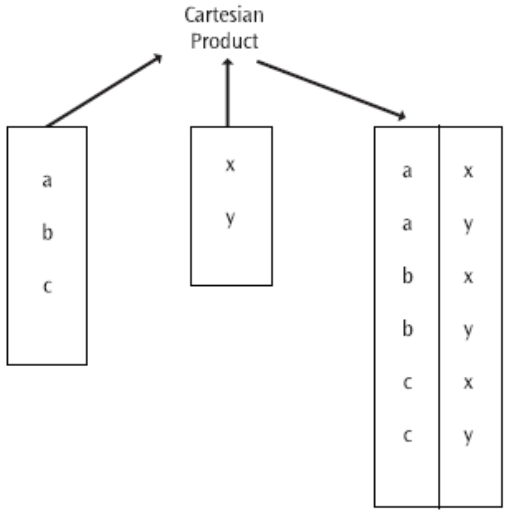
- Creates a second relation by extracting a subset of columns
- Question: What is the plant number and budget for each of the award winning plants?
- $\pi_{Aw\_pl\_No,\ Aw\_pl\_Budget}\ AW\_Plant$
- SELECT Aw_pl_No, Aw_pl_Budget FROM AW_Plant

AW_PLANT		
Aw_pl_No	Aw_pl_Name	Aw_pl_Budget
11	Black Horse	2500000
13	Mayde Creek	1930000
12	Whitefield	2910000
17	River Oaks	1930000
19	King's Island	2500000
15	Ashton	2500000

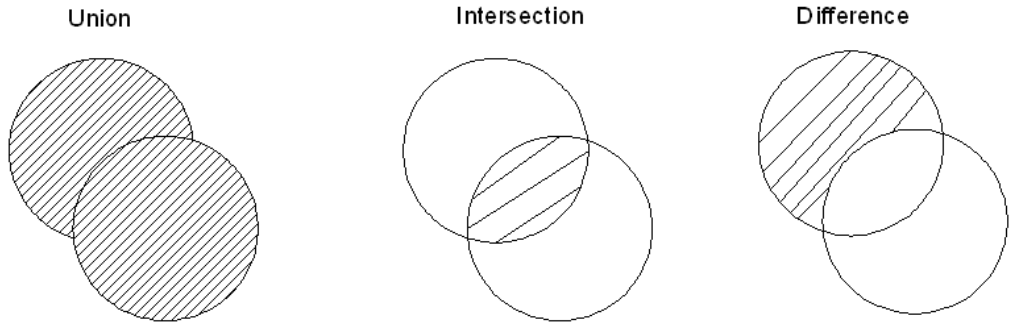
AW_PLANT	
Aw_pl_No	Aw_pl_Budget
11	2500000
13	1930000
12	2910000
17	1930000
19	2500000
15	2500000

Binary Operators

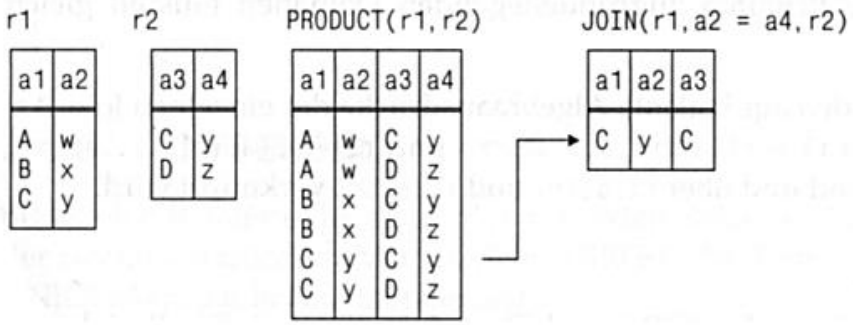
Cartesian product



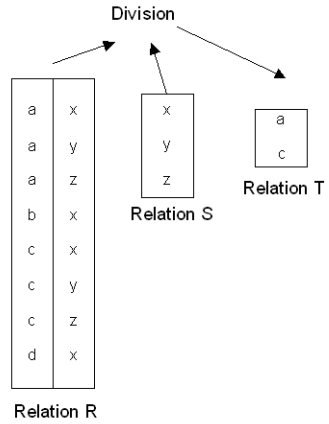
Union (∪), Intersection (∩), and Difference (-)



Join



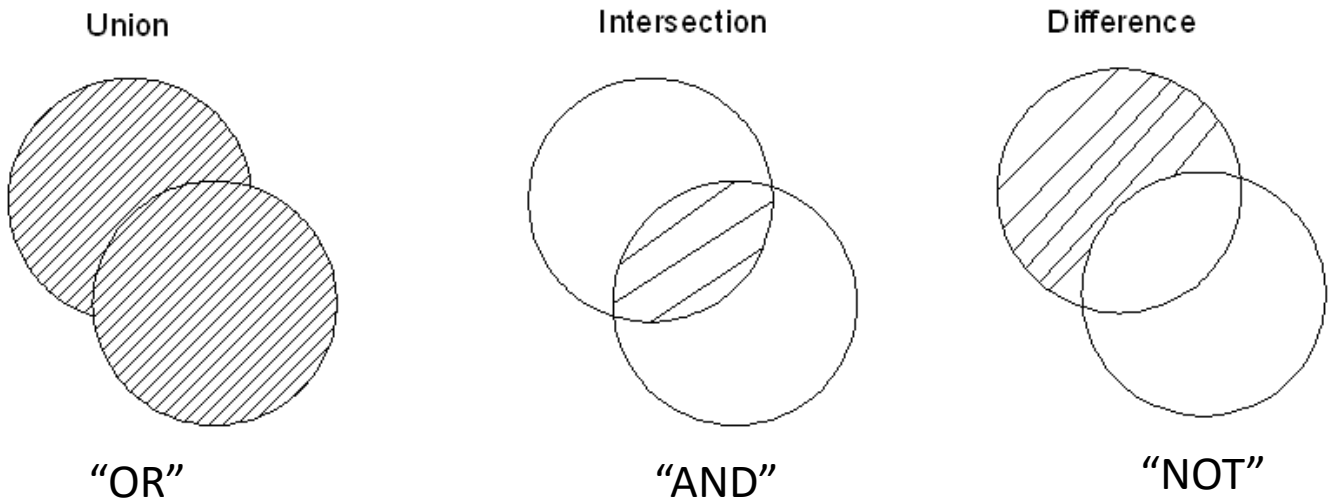
Division



Set Operations

- Two relations (R and S) are “union compatible” if they:
  - Have the same degree (number of attributes)
  - Pairs of attributes from R and S have the same domain

- Union compatibility is a requirement for all set operations (union, intersection, and difference)

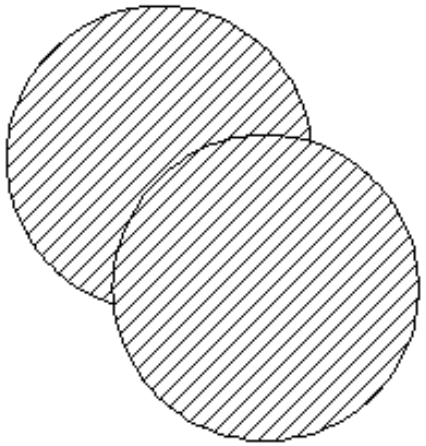


Set Theory and Relational Algebra

- Database theory is based on set theory
- Manipulations referred to as “Relational Algebra”

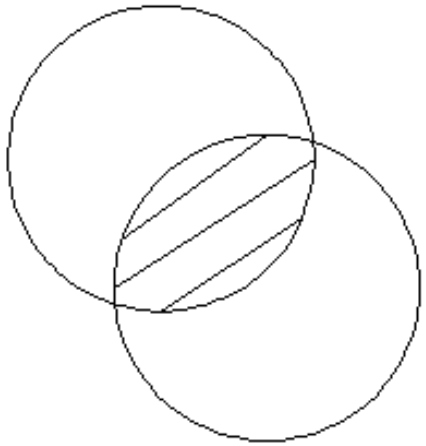
Set Theory Operators

Union



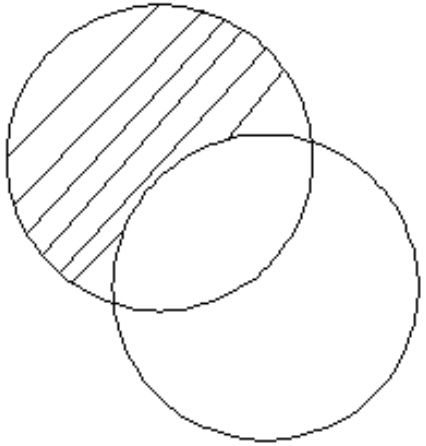
"OR"

Intersection



"AND"

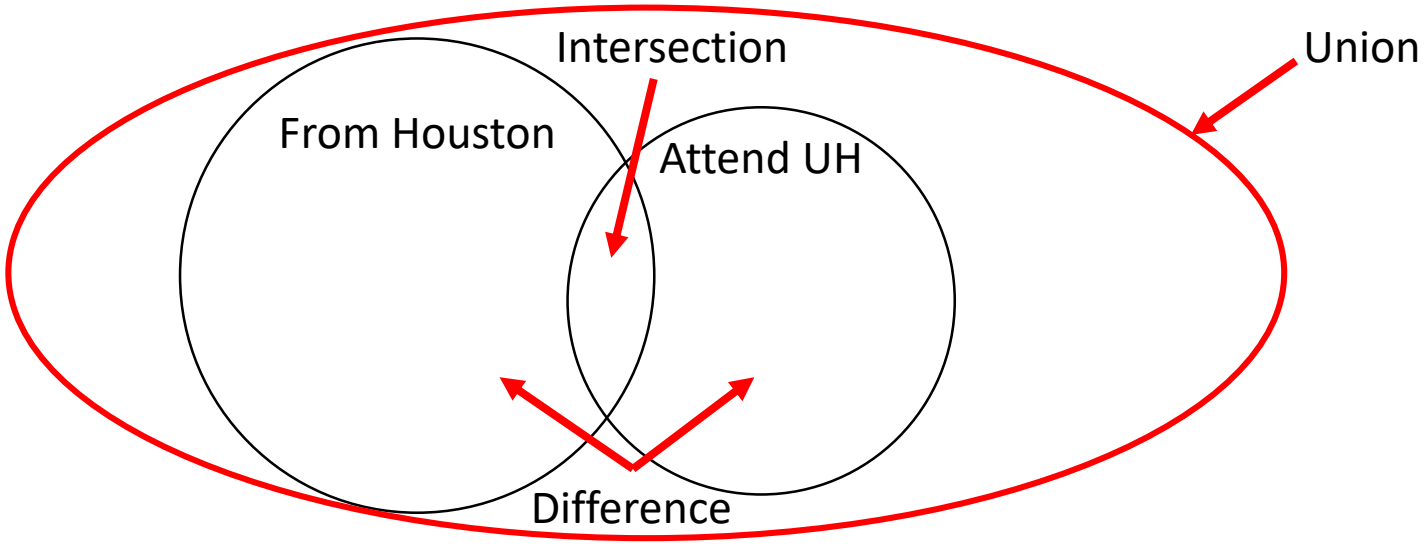
Difference



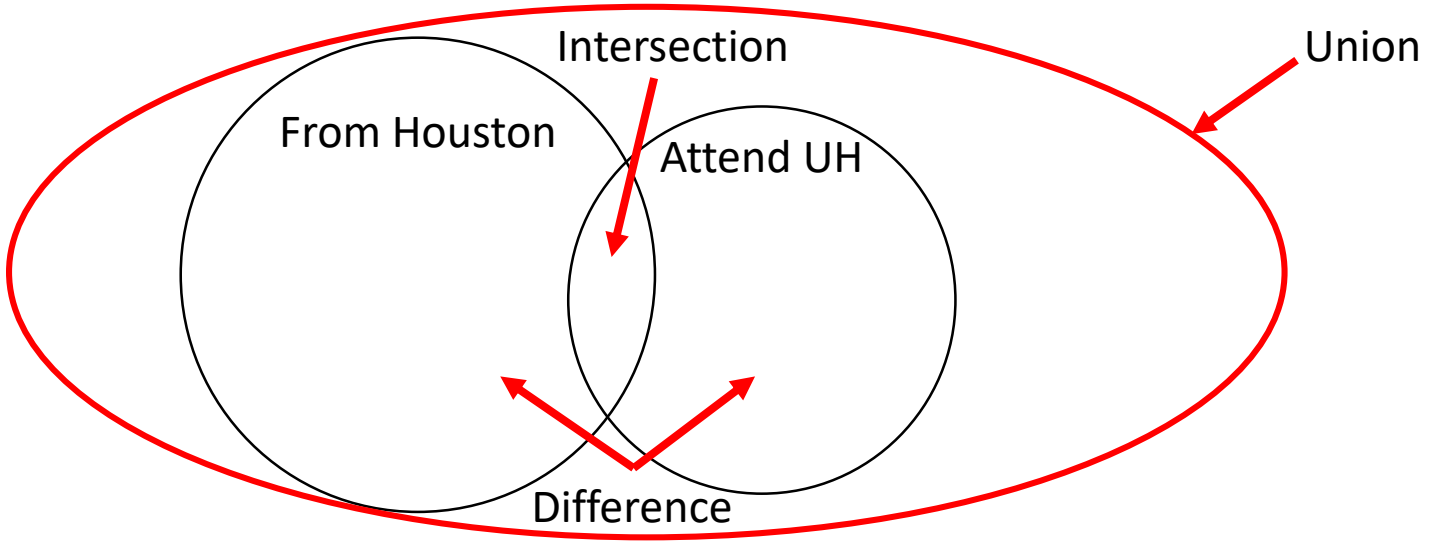
"NOT"

Set Theory Operators

- Some people are from Houston
- Some people attend UH
- Union: People that are from Houston OR go to UH
- Intersection: People that are from Houston AND go to UH
- Difference: People from Houston but do NOT go to UH
  People that go to UH but NOT from Houston

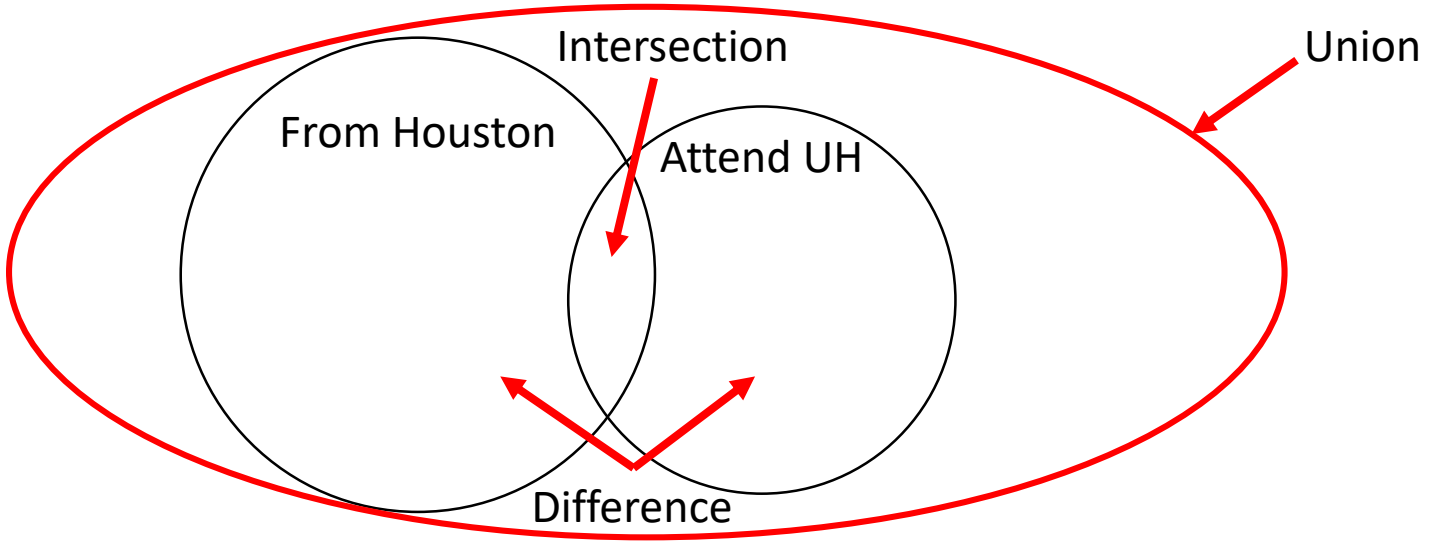


Union of Houston and UH



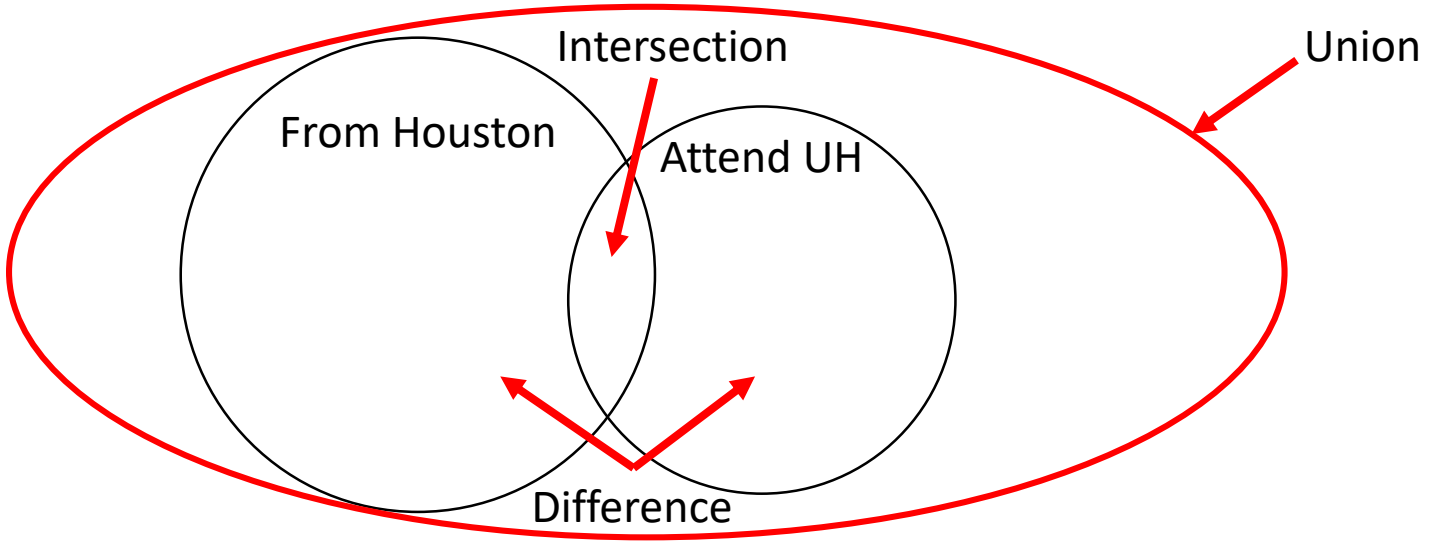
StuID	Name	Hometown	School
1111	Adam	Houston	UH
2222	Beth	Dallas	UH
3333	Chris	Houston	UH
4444	Dave	Austin	UH
5555	Eunice	Houston	Rice
6666	Frank	Houston	Rice

Intersection of Houston and UH



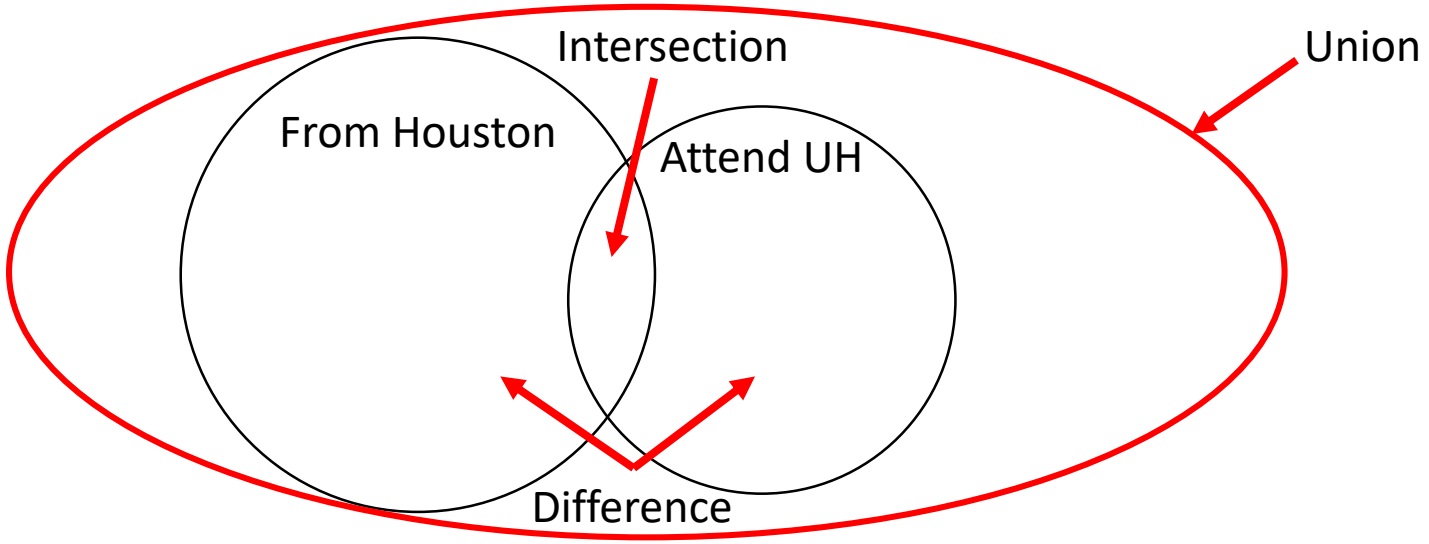
StuID	Name	Hometown	School
1111	Adam	Houston	UH
2222	Beth	Dallas	UH
3333	Chris	Houston	UH
4444	Dave	Austin	UH
5555	Eunice	Houston	Rice
6666	Frank	Houston	Rice

UH minus Houston



StuID	Name	Hometown	School
1111	Adam	Houston	UH
2222	Beth	Dallas	UH
3333	Chris	Houston	UH
4444	Dave	Austin	UH
5555	Eunice	Houston	Rice
6666	Frank	Houston	Rice

Houston minus UH



StuID	Name	Hometown	School
1111	Adam	Houston	UH
2222	Beth	Dallas	UH
3333	Chris	Houston	UH
4444	Dave	Austin	UH
5555	Eunice	Houston	Rice
6666	Frank	Houston	Rice

Union (Binary Operation)

- Creates a third relation containing tuples from either relation
- What plants are either in Texas OR are award winning plants?
- AW_Plant ∪ TX_Plant
- SELECT * FROM AW_Plant UNION SELECT * FROM TX_Plant

AW_PLANT		
Aw_pl_No	Aw_pl_Name	Aw_pl_Budget
11	Black Horse	2500000
13	Mayde Creek	1930000
12	Whitefield	2910000
17	River Oaks	1930000
19	King's Island	2500000
15	Ashton	2500000

OR

TX_PLANT		
Tx_pl_No	Tx_pl_Name	Tx_pl_Budget
16	Southern Oaks	1930000
17	River Oaks	1930000
18	Kingwood	1930000

R_AW_PLANT		
Aw_pl_No	Aw_pl_Name	Aw_pl_Budget
11	Black Horse	2500000
13	Mayde Creek	1930000
12	Whitefield	2910000
17	River Oaks	1930000
19	King's Island	2500000
15	Ashton	2500000
16	Southern Oaks	1930000
18	Kingwood	1930000

Intersection (Binary Operation)

- Creates a third relation containing tuples present in both relations
- What plants are in Texas AND are award winning plants?
- AW_Plant ∩ TX_Plant
- SELECT * FROM AW_Plant INTERSECT SELECT * FROM TX_Plant

AW_PLANT		
Aw_pl_No	Aw_pl_Name	Aw_pl_Budget
11	Black Horse	2500000
13	Mayde Creek	1930000
12	Whitefield	2910000
17	River Oaks	1930000
19	King's Island	2500000
15	Ashton	2500000

AND

TX_PLANT		
Tx_pl_No	Tx_pl_Name	Tx_pl_Budget
16	Southern Oaks	1930000
17	River Oaks	1930000
18	Kingwood	1930000

R_AW_PLANT		
Aw_pl_No	Aw_pl_Name	Aw_pl_Budget
17	River Oaks	1930000

Difference (Binary Operation)

- Creates a third relation containing tuples present in one relation but not the other relation
- What plants are in Texas, but are not award winning?
- TX_Plant – AW_Plant
- SELECT * FROM TX_Plant MINUS SELECT * FROM AW_Plant

TX_PLANT		
Tx_pl_No	Tx_pl_Name	Tx_pl_Budget
16	Southern Oaks	1930000
17	River Oaks	1930000
18	Kingwood	1930000

NOT

AW_PLANT		
Aw_pl_No	Aw_pl_Name	Aw_pl_Budget
11	Black Horse	2500000
13	Mayde Creek	1930000
12	Whitefield	2910000
17	River Oaks	1930000
19	King's Island	2500000
15	Ashton	2500000

TX_PLANT		
Tx_pl_No	Tx_pl_Name	Tx_pl_Budget
16	Southern Oaks	1930000
18	Kingwood	1930000

Difference (Binary Operation)

- Creates a third relation containing tuples present in one relation but not the other relation
- What plants are award winning but are not in Texas?
- AW_Plant – TX_Plant
- SELECT * FROM AW_Plant MINUS SELECT * FROM TX_Plant

AW_PLANT		
Aw_pl_No	Aw_pl_Name	Aw_pl_Budget
11	Black Horse	2500000
13	Mayde Creek	1930000
12	Whitefield	2910000
17	River Oaks	1930000
19	King's Island	2500000
15	Ashton	2500000

Not

TX_PLANT		
Tx_pl_No	Tx_pl_Name	Tx_pl_Budget
16	Southern Oaks	1930000
17	River Oaks	1930000
18	Kingwood	1930000

R_AW_PLANT		
Aw_pl_No	Aw_pl_Name	Aw_pl_Budget
11	Black Horse	2500000
13	Mayde Creek	1930000
12	Whitefield	2910000
19	King's Island	2500000
15	Ashton	2500000

Natural Join

- Combines two relations into a third by matching values that come from the same domain for attributes in the two relations.
- What award winning plant is each project assigned to?
- SELECT * FROM AW_Plant NATURAL JOIN Projects
- SELECT * FROM AW_Plant INNER JOIN Projects ON Aw_pl_No = Prj_aw_pl_no

AW_PLANT		
Aw_pl_No	Aw_pl_Name	Aw_pl_Budget
11	Black Horse	2500000
13	Mayde Creek	1930000
12	Whitefield	2910000
17	River Oaks	1930000
19	King's Island	2500000
15	Ashton	2500000

PROJECTS			
Prj_name	Prj_no	Prj_location	Prj_aw_pl_no
Solar Heating	41	Sealy	11
Lunar Cooling	17	Yoakum	17
Synthetic Fuel	29	Salem	17
Nitro-Cooling	23	Parthi	12
Robot Sweeping	31	Ponca City	11
Robot Painting	37	Poakum	19
Ozone Control	13	Parthi	19

PROJECTS					
Prj_name	Prj_no	Prj_location	Prj_aw_pl_no	Prj_pl_Name	Aw_pl_Budget
Solar Heating	41	Sealy	11	Black Horse	2500000
Lunar Cooling	17	Yoakum	17	River Oaks	1930000
Synthetic Fuel	29	Salem	17	River Oaks	1930000
Nitro-Cooling	23	Parthi	12	Whitefield	2910000
Robot Sweeping	31	Ponca City	11	Black Horse	2500000
Robot Painting	37	Poakum	19	King's Island	2500000
Ozone Control	13	Parthi	19	King's Island	2500000

Module 6.4
Relational Algebra

- What are the eight operations in relational algebra?
- Describe the difference in select and project
- Describe the difference in the Union, Intersection, and Difference operations
- What is a natural join?

Module 6.5
Views and Materialized Views

- What do views do and why are they useful?
- How do growth and restructuring impact views?
- What is the difference in a normal view and a materialized view?

Views

- Last time we talked about different types of joins and projections – these are "views" of the data

- Unlike a relation schema, a view does not contain data
  - Is a logical window to the attributes and tuples from one or a set of relations

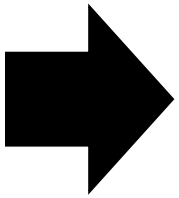
- Views provide many benefits:
  - Allows the same data to be seen in different ways by different users (or applications)
  - Provides security by restricting access to data
  - Hides complexity by making data from several relations appear as a single object

Horse view - Selection

- For a doctor that only needs to see female horses:
  - SELECT * FROM Horses WHERE Gender='F'

Relation

Horses				
Name	Color	Spots	Gender	Weight
Amy	Yellow	Yes	F	920
Dave	Grey	No	M	1800
Ed	Yellow	No	M	1100
Sally	Black	No	F	1200
Sarah	Grey	No	F	1700
Tom	Red	Yes	M	1500
Jane	Red	No	F	1300



View

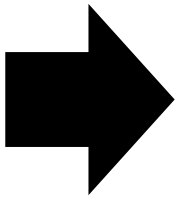
Horses				
Name	Color	Spots	Gender	Weight
Amy	Yellow	Yes	F	920
Sally	Black	No	F	1200
Sarah	Grey	No	F	1700
Jane	Red	No	F	1300

Horse View - Projection

- For someone allocating stables who only needs to know the name and weight of the horses:
  - SELECT Name, Weight FROM Horses

Relation

Horses				
Name	Color	Spots	Gender	Weight
Amy	Yellow	Yes	F	920
Dave	Grey	No	M	1800
Ed	Yellow	No	M	1100
Sally	Black	No	F	1200
Sarah	Grey	No	F	1700
Tom	Red	Yes	M	1500
Jane	Red	No	F	1300



View

Horses	
Name	Weight
Amy	920
Dave	1800
Ed	1100
Sally	1200
Sarah	1700
Tom	1500
Jane	1300

Views

- We’re showing users (or applications) only the data they need to see
  - Provides security by restricting access to data
  - Hides complexity by making data from several relations appear as a single object
  - Allows the same data to be seen in different ways by different users

- What does this sound like?
  - External Schema!

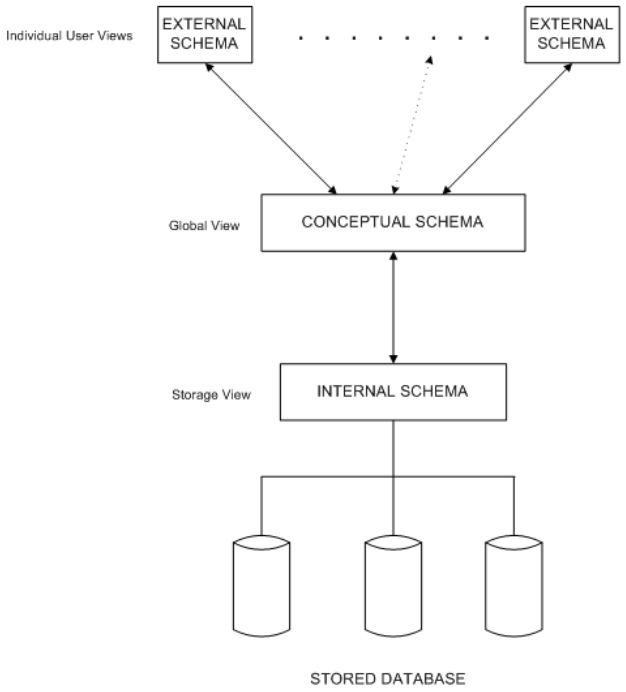


Figure 1.2 The ANSI/SPARC three-schema Architecture

Growth and Restructuring

- By using views, changes to the database structure may be made without affecting users
  - What does this sound like?

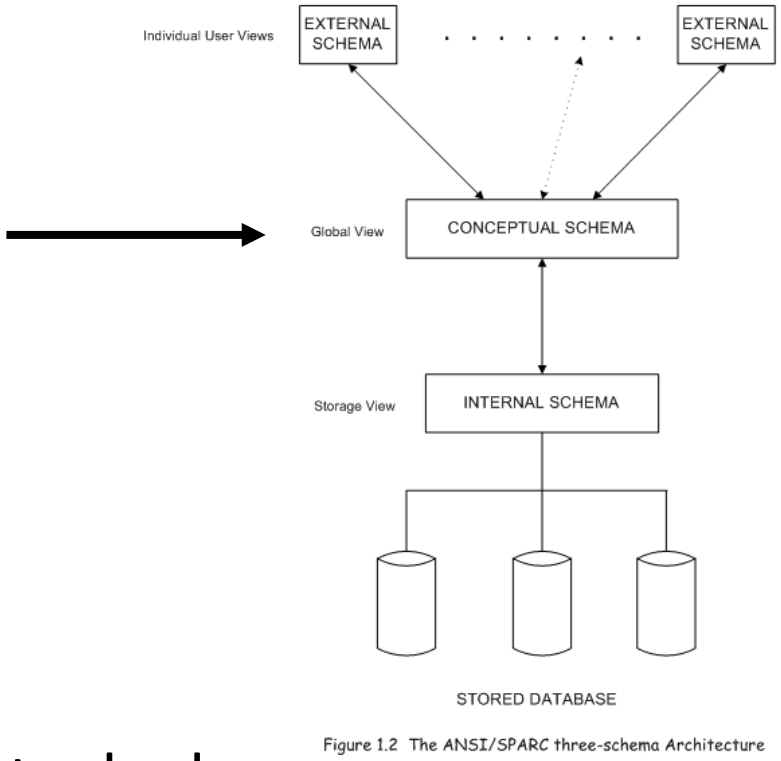


Figure 1.2 The ANSI/SPARC three-schema Architecture

- Two primary types of changes:
  - **Growth**: Adding new attributes to a relation, or new relations to a data model
  - **Restructuring**: Changing the conceptual schema
    - Should be “Information Equivalent” – which facilitates the restructuring being reversible

Growth – Let's start here

(Note that we've added an "owner" attribute)

- We want name, gender, weight, and owner of female horses

SELECT Name, Gender, Weight, Owner FROM Horses WHERE Gender='F'

Horses

Name	Color	Spots	Gender	Weight	Owner
Amy	Yellow	Yes	F	920	Mgrimes
Dave	Grey	No	M	1800	Mparks
Ed	Yellow	No	M	1100	Mgrimes
Sally	Black	No	F	1200	Rcooper
Sarah	Grey	No	F	1700	Mgrimes
Tom	Red	Yes	M	1500	Mgrimes
Jane	Red	No	F	1300	Rcooper

Horses-View

Name	Gender	Weight	Owner
Amy	F	920	Mgrimes
Sally	F	1200	Rcooper
Sarah	F	1700	Mgrimes
Jane	F	1300	Rcooper

What is this part doing? PROJECTING a subset of attributes (Reducing degree)

What is this part doing? SELECTING a subset of tuples (Reducing cardinality)

Growth - Attributes

- Adding attributes does not affect the views

SELECT Name, Gender, Weight, Owner FROM Horses WHERE Gender='F'

Horses						
Name	Color	Spots	Gender	Weight	Owner	Phone#
Amy	Yellow	Yes	F	920	Mgrimes	5551112345
Dave	Grey	No	M	1800	Mparks	5552229876
Ed	Yellow	No	M	1100	Mgrimes	5551112345
Sally	Black	No	F	1200	Rcooper	5553335656
Sarah	Grey	No	F	1700	Mgrimes	5551112345
Tom	Red	Yes	M	1500	Mgrimes	5551112345
Jane	Red	No	F	1300	Rcooper	5553335656

Horses-View			
Name	Gender	Weight	Owner
Amy	F	920	Mgrimes
Sally	F	1200	Rcooper
Sarah	F	1700	Mgrimes
Jane	F	1300	Rcooper

No change to the view!

Growth - Relations

- Adding relations to the schema does not affect the views

SELECT Name, Gender, Weight, Owner FROM Horses WHERE Gender='F'

Horses						
Name	Color	Spots	Gender	Weight	Owner	Phone#
Amy	Yellow	Yes	F	920	Mgrimes	5551112345
Dave	Grey	No	M	1800	Mparks	5552229876
Ed	Yellow	No	M	1100	Mgrimes	5551112345
Sally	Black	No	F	1200	Rcooper	5553335656
Sarah	Grey	No	F	1700	Mgrimes	5551112345
Tom	Red	Yes	M	1500	Mgrimes	5551112345
Jane	Red	No	F	1300	Rcooper	5553335656

Horses-View			
Name	Gender	Weight	Owner
Amy	F	920	Mgrimes
Sally	F	1200	Rcooper
Sarah	F	1700	Mgrimes
Jane	F	1300	Rcooper

Owners		
OwnerID	OwnName	Balance
101	Mgrimes	825.00
102	Mparks	0
103	Rcooper	210.00

No change to the view!

Restructuring

- Restructuring should have an information equivalent outcome
- The view must be updated, but no impact to users/applications

  SELECT Name, Gender, Weight, OwnName as Owner FROM Horses
  JOIN Owners on FKOwnerID = OwnerID WHERE gender = ‘F’

Horses

Name	Color	Spots	Gender	Weight	FKOwnerID
Amy	Yellow	Yes	F	920	101
Dave	Grey	No	M	1800	102
Ed	Yellow	No	M	1100	101
Sally	Black	No	F	1200	103
Sarah	Grey	No	F	1700	101
Tom	Red	Yes	M	1500	101
Jane	Red	No	F	1300	103

Owners

OwnerID	OwnName	Phone	Balance
101	Mgrimes	5551112345	825.00
102	Mparks	5552229876	0
103	Rcooper	5553335656	210.00

Horses-View

Name	Gender	Weight	Owner
Amy	F	920	Mgrimes
Sally	F	1200	Rcooper
Sarah	F	1700	Mgrimes
Jane	F	1300	Rcooper

No change to the view!

Materialized Views

- Normal views are temporary
  - Only exist while the data is being access
  - This is what most views are

- Materialized views (snapshots) are constructed from one or more relations ahead of time
  - Used to “freeze” data at a certain point in time
  - Improves performance
  - Periodically recreated to reflect changes in the data
  - Often deleted if not used for a while, then recreated when needed again

Module 6.5
Views and Materialized Views

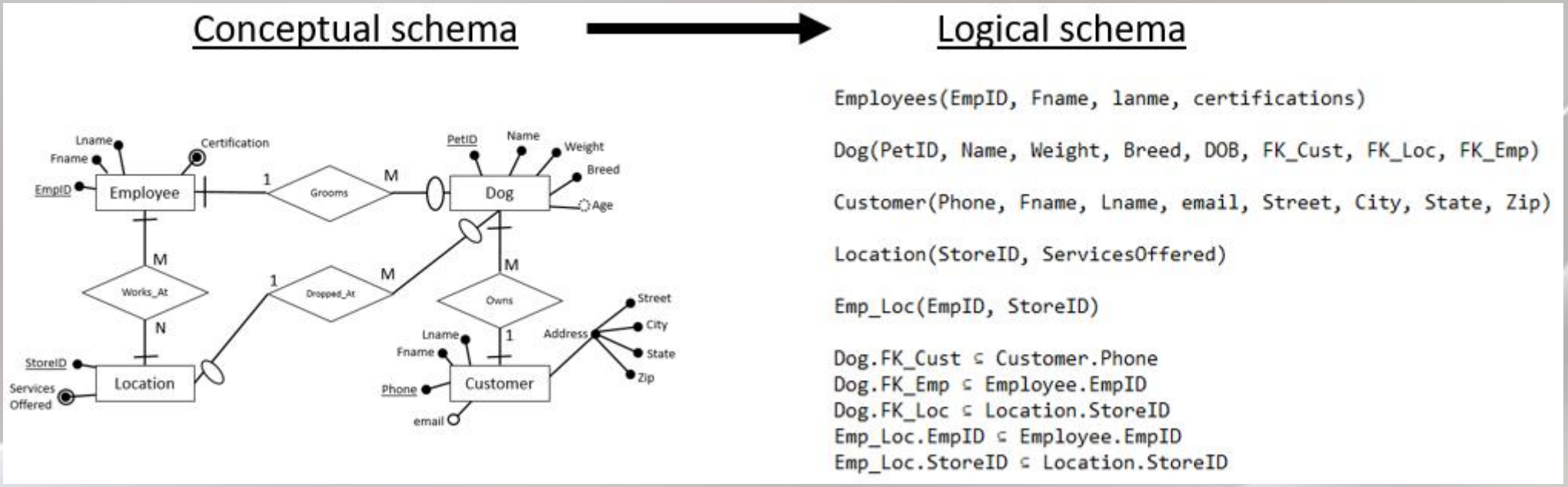
- What do views do and why are they useful?
- How do growth and restructuring impact views?
- What is the difference in a normal view and a materialized view?

Break

Module 6.7
Mapping an ER Model to a Logical Schema

Next step in transforming business rules into an actual DB

ERD → Design Specific ERD → Logical Schema



Mapping an ER Model to a Logical Schema

- Next step in transforming business rules into an actual DB
  - ERD → Design Specific ERD → Logical Schema

- Unfortunately, it is often difficult to represent all attributes of one schema in another
  - “Information Reducing”
  - Must carry forward all requirements, either by modeling or in semantic integrity constraints

Steps for Mapping Entity Types

- Create a **relation schema** for each entity type in the ER diagram.

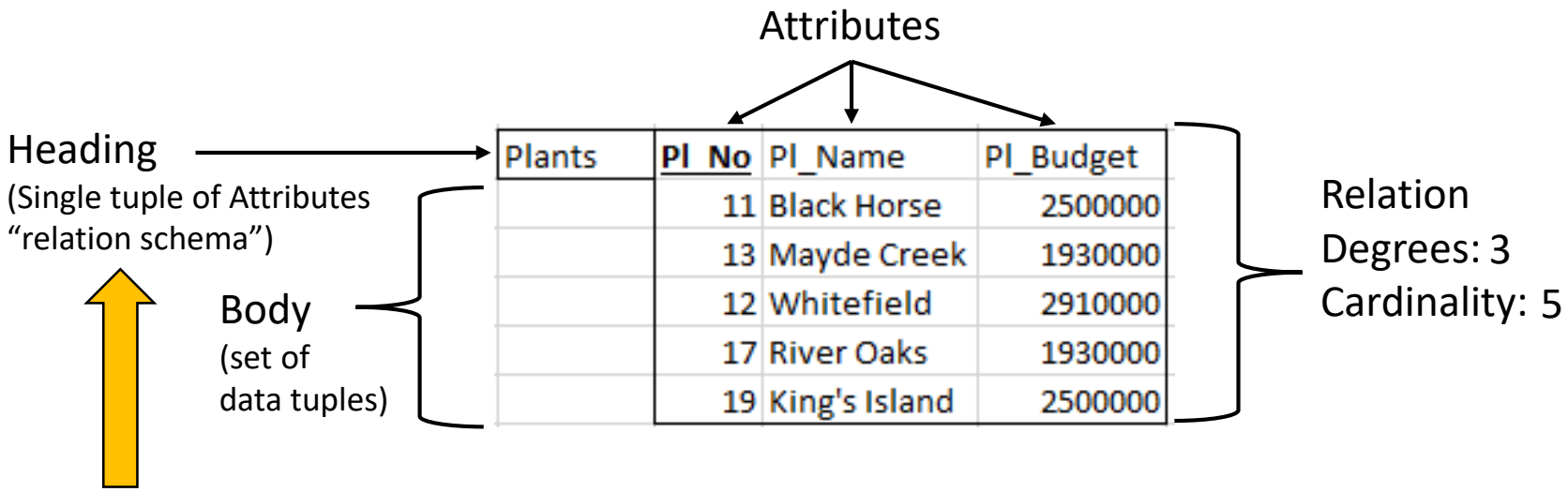
- Create an attribute for every stored attribute. This implies:
  - For composite attributes only their atomic components are recorded
  - Derived attributes are not recorded
  - Multi-valued attributes do not exist in a Design-Specific ER model

- Choose a primary key from among the candidate keys by underlining the attribute(s) making up the primary key.

- For each weak entity type, add the primary key of the identifying parent entity type as attribute(s) in the relation schema.
  - The attribute(s) thus added plus the partial key of the weak entity type form the primary key of the relation schema representing the weak entity type.

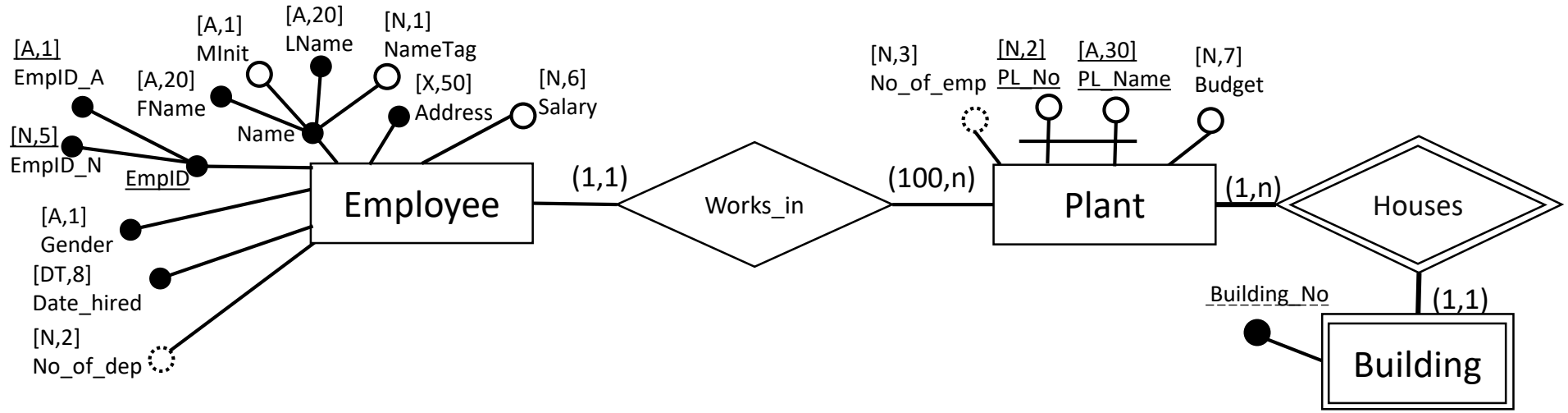
Review: what is a relation?

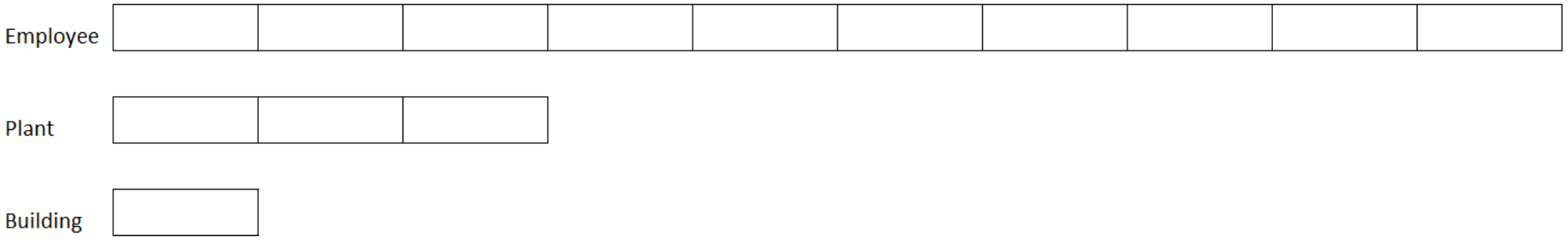
- A mathematical terms approximated by a two dimensional table:
  - Heading – a single tuple listing the attributes (Relation Schema)
  - Body – collection of data tuples



Plants	Pl_No	Pl_Name	Pl_Budget
	11	Black Horse	2500000
	13	Mayde Creek	1930000
	12	Whitefield	2910000
	17	River Oaks	1930000
	19	King's Island	2500000

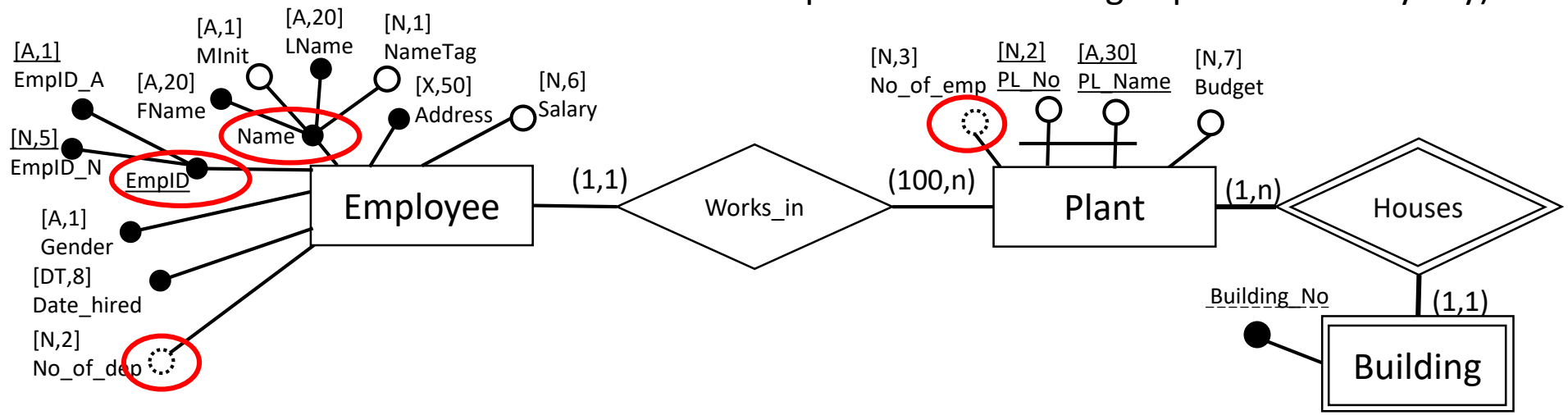
Step 1: Create relation schemas for entities





Step 2: Create attributes

No derived or composite attributes allowed
No MV attributes or M:N relationship (should not be present in the Design Specific ERD anyway)



Employee	EmpID_A	EmpID_N	FName	MInit	LName	NameTag	Gender	Address	Salary	Date_hired

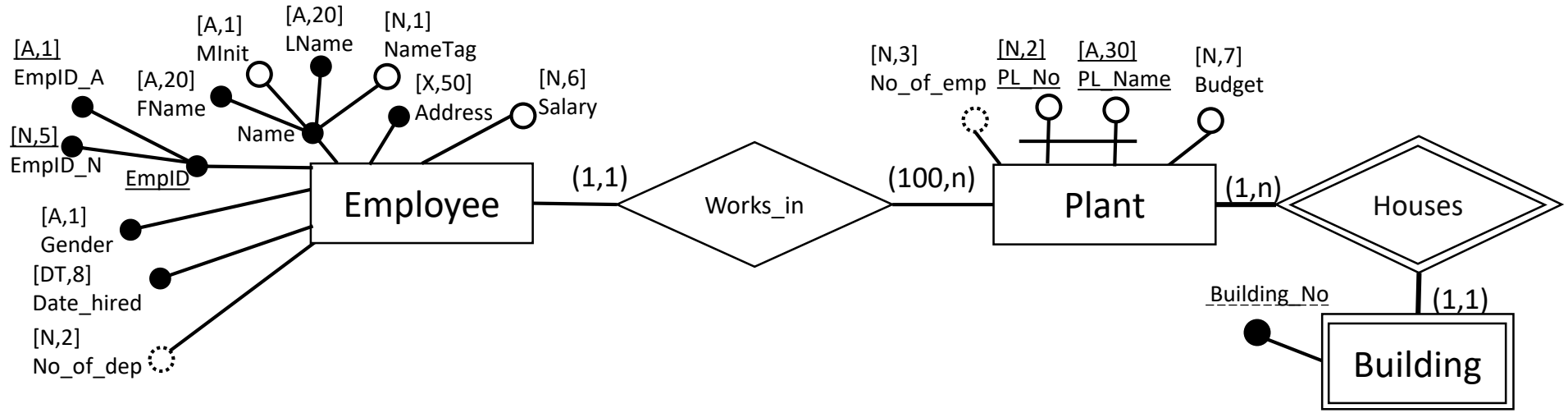
Plant	PL_Name	PL_No	Budget

Building	Building_No

Note 1: Only the atomic attributes constituting EmpID and Name in the ERD are recorded in EMPLOYEE.

Note 2: The derived attributes, No_of_dep in EMPLOYEE and No_of_emp in PLANT are not captured here. They will be derived in the external schema (i.e., a view).

Step 3: Choose PK from the CK by underlining them



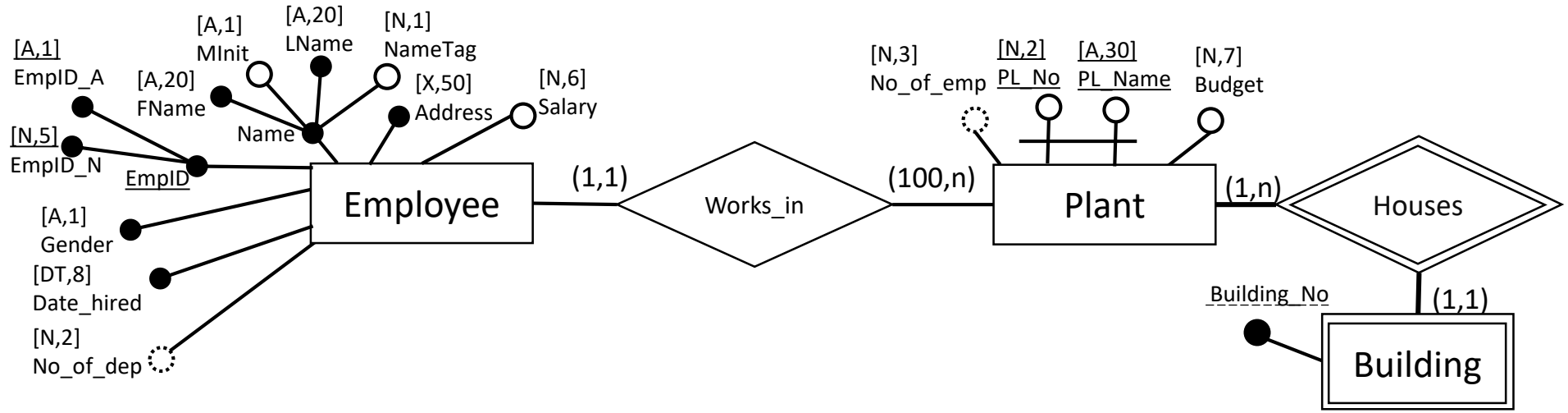
Employee	**EmpID_A**	**EmpID_N**	FName	MInit	LName	NameTag	Gender	Address	Salary	Date_hired

Plant	PL_Name	**PL_No**	Budget

Building	Building_No

???

Step 4: Add Primary Key of strong entity to weak entities



Employee	**EmpID_A**	**EmpID_N**	FName	MInit	LName	NameTag	Gender	Address	Salary	Date_hired

Plant	PL_Name	**PL_No**	Budget

Building	**Building_No**	**FK_PL_No**

Logical Schemas Defined!

- Next step is to map the relationships

Employee	**EmpID_A**	**EmpID_N**	FName	MInit	LName	NameTag	Gender	Address	Salary	Date_hired

Plant	PL_Name	**PL_No**	Budget

Building	**Building_No**	**FK_PL_No**

Mapping Relationship Types

- Characteristic of the Design-Specific ER model:
  - Only binary or recursive relationships are present – why?
  - Only 1:1 or 1:n cardinality constraints are present – why?

- Relationship type mapped by enforcing a foreign key constraint

- The foreign key constraint requires that the referenced attribute(s) be a candidate key of (advisably, primary key) the referenced relation

- Reduction in information ☹
  - Participation constraints are not mapped.
  - Cardinality constraint of (1:n) is implicit by default

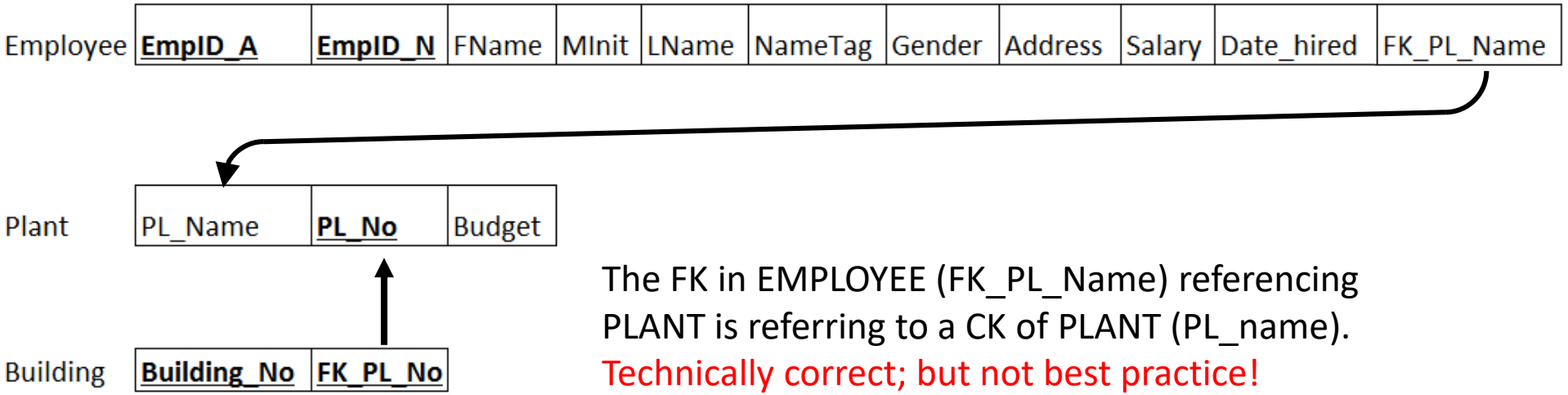
Mapping Relationship Types [1:n]

- Add a foreign key attribute to the referencing schema (Child)
  - Foreign key must share the same domain with a candidate key in the referenced schema (Parent)
  - Best Practice: use primary key as the referenced attribute

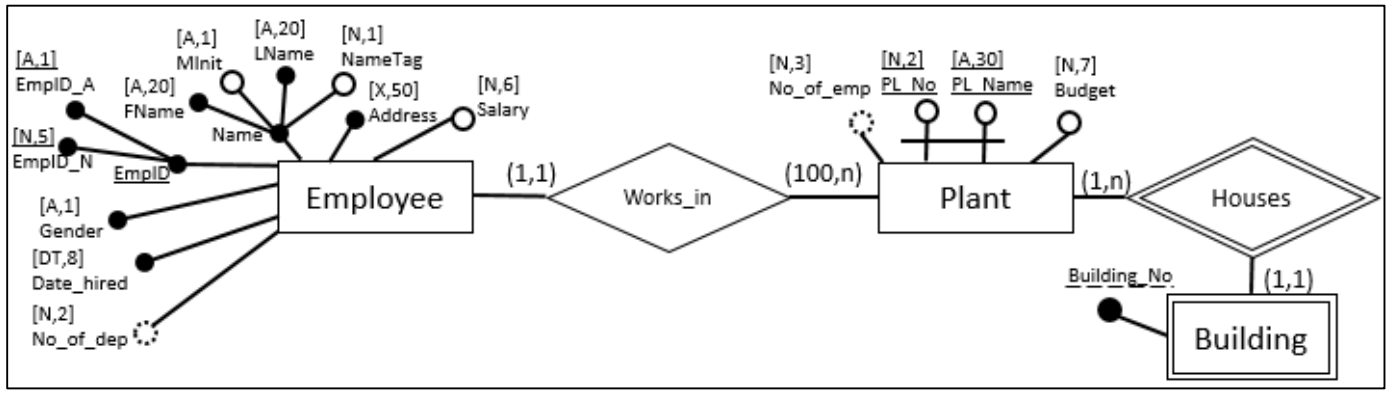
- Remember:
  - The entity type on the “many-side” of the relationship type is the referencing relation schema (“child”)
  - The entity type on the “one-side” is the referenced relation schema (“parent”)

- Represented with a directed arc (i.e., a line) with arrowhead pointing at the referenced candidate key

Mapping Relationship Types [1:n]



The FK in EMPLOYEE (FK_PL_Name) referencing PLANT is referring to a CK of PLANT (PL_name).
Technically correct; but not best practice!



Note 1: Participation constraints are not mapped.
Note 2: Cardinality constraint of (1:n) is implicit by default

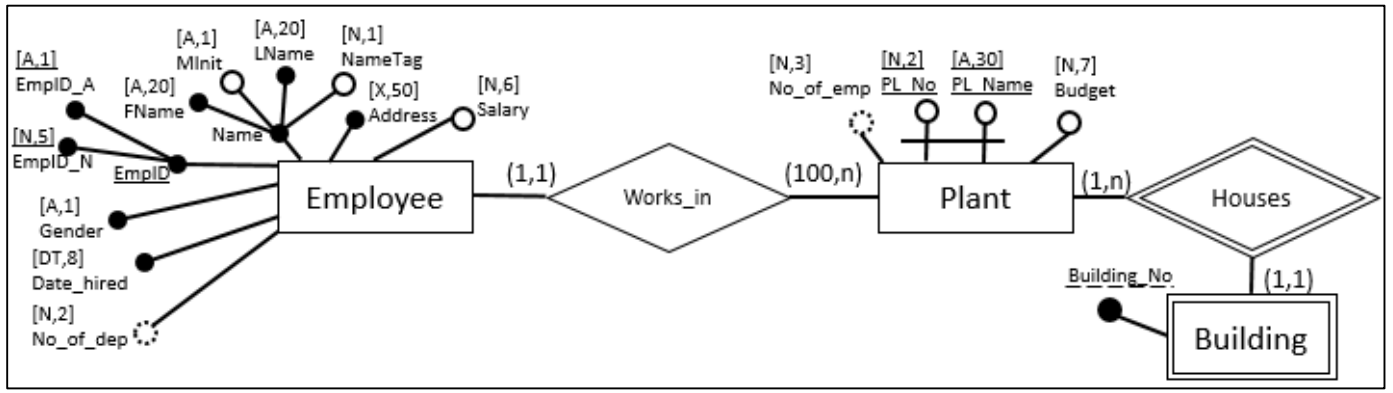
Mapping Relationship Types [1:n]

Employee	**EmpID_A**	**EmpID_N**	FName	MInit	LName	NameTag	Gender	Address	Salary	Date_hired	FK_PL_No

Plant	PL_Name	**PL_No**	Budget

Building	**Building_No**	**FK_PL_No**

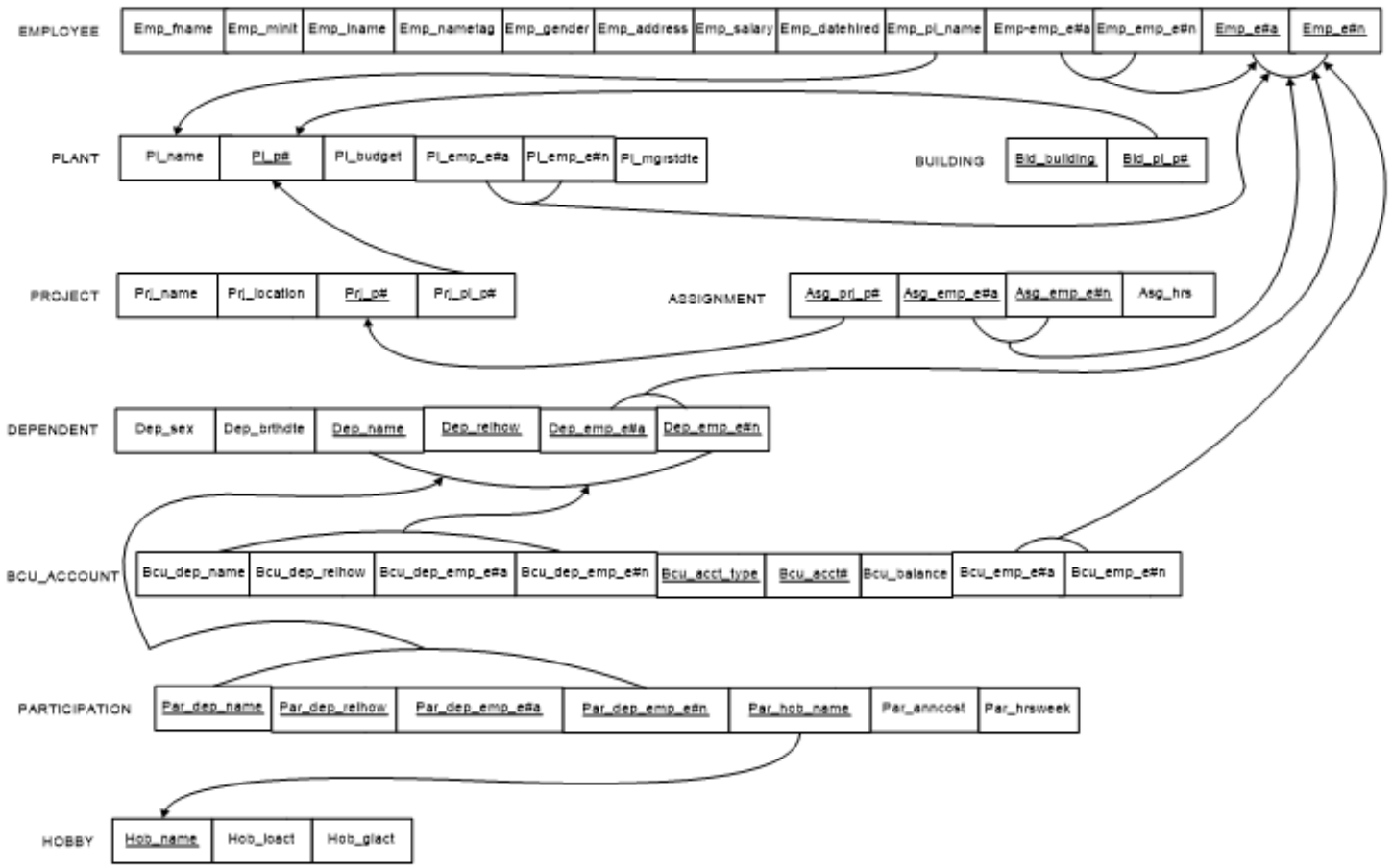
This is better!



Note 1: Participation constraints are not mapped.
Note 2: Cardinality constraint of (1:n) is implicit by default

Alternative Notation

- The lines are useful, but may get messy for large models
- We can represent Inclusion Dependency as **R2.{A2} ⊆ R1.{A1}**



Alternative Notation

EMPLOYEE(EmpID_A, EmpID_N, FName, MInit, LName, NameTag, Gender, Address, Salary, Date_hired, FK_PL_No)

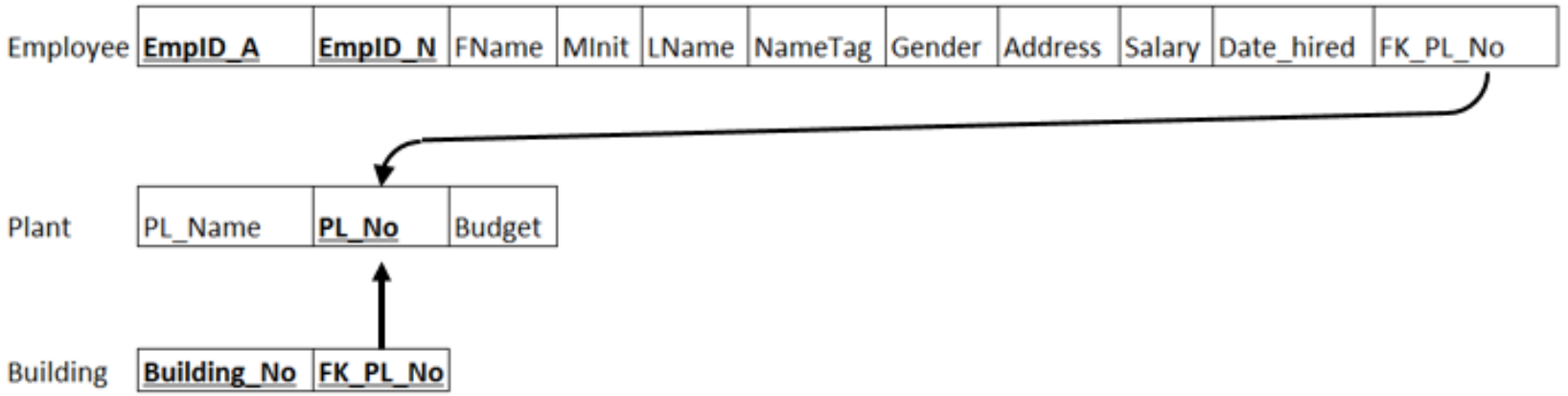
#EMPLOYEE.{FK_PL_No} ⊆ PLANT.{PL_No}

PLANT (PL_Name, PL_No, Budget)

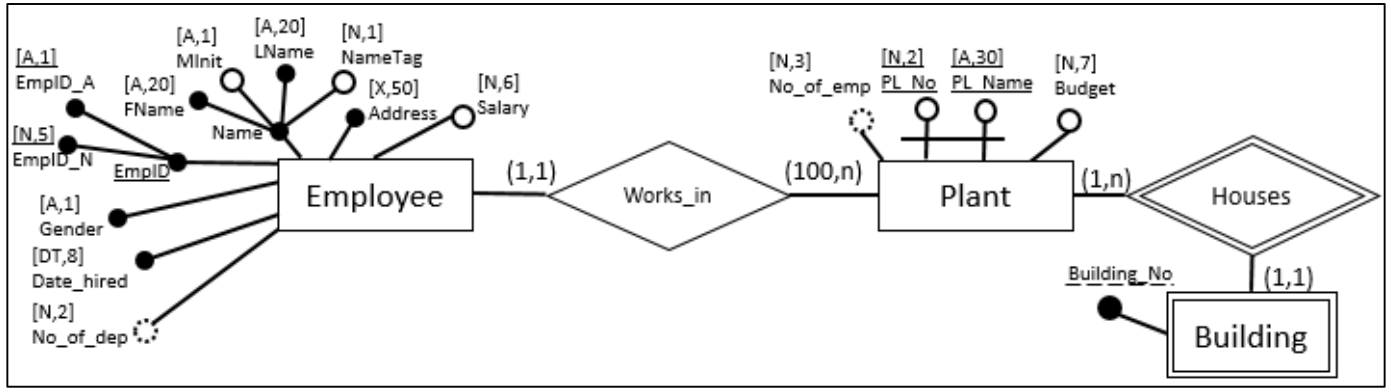
BUILDING (Building_No, FK_PL_No)

#BUILDING.{FK_PL_No} ⊆ PLANT.{PL_No}

Equivalent

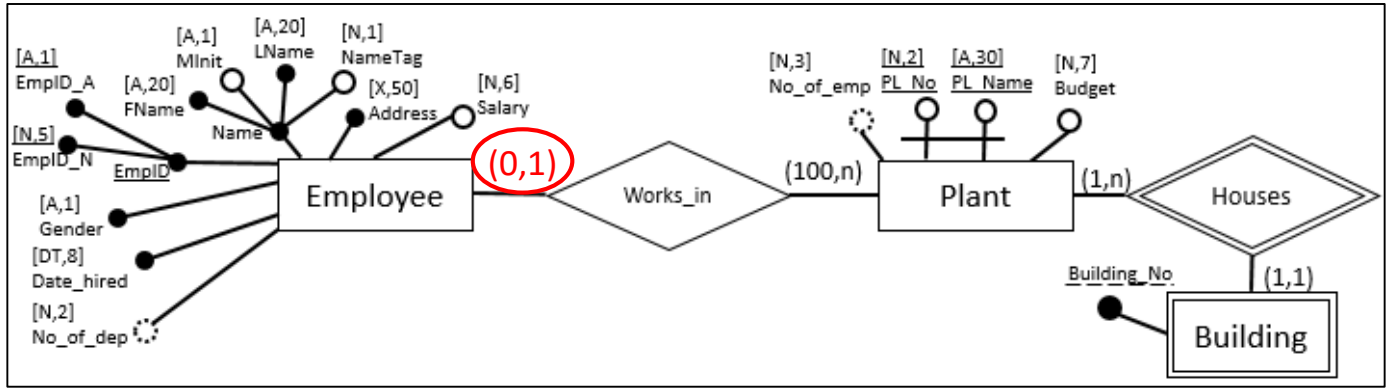


An alternate mapping for [1:n]

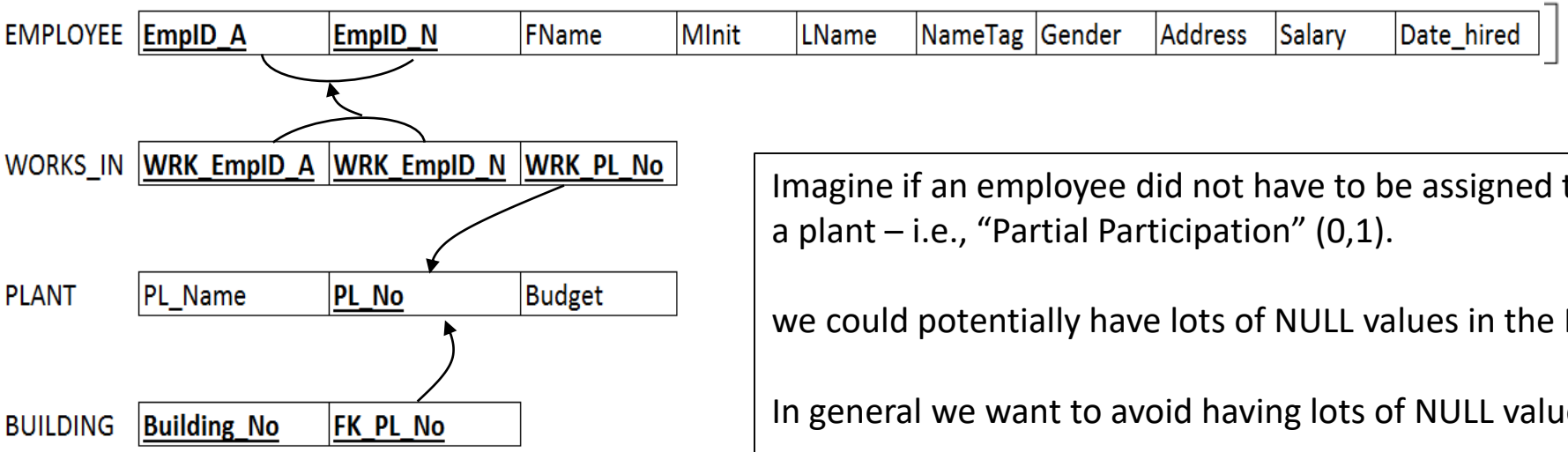


- While not mapped, participation does impact the design
  - Reminder: Participation is “Partial” (optional) or “Total” (mandatory)

An alternate mapping for [1:n]



- Instead of adding a FK to the child table, create a gerund



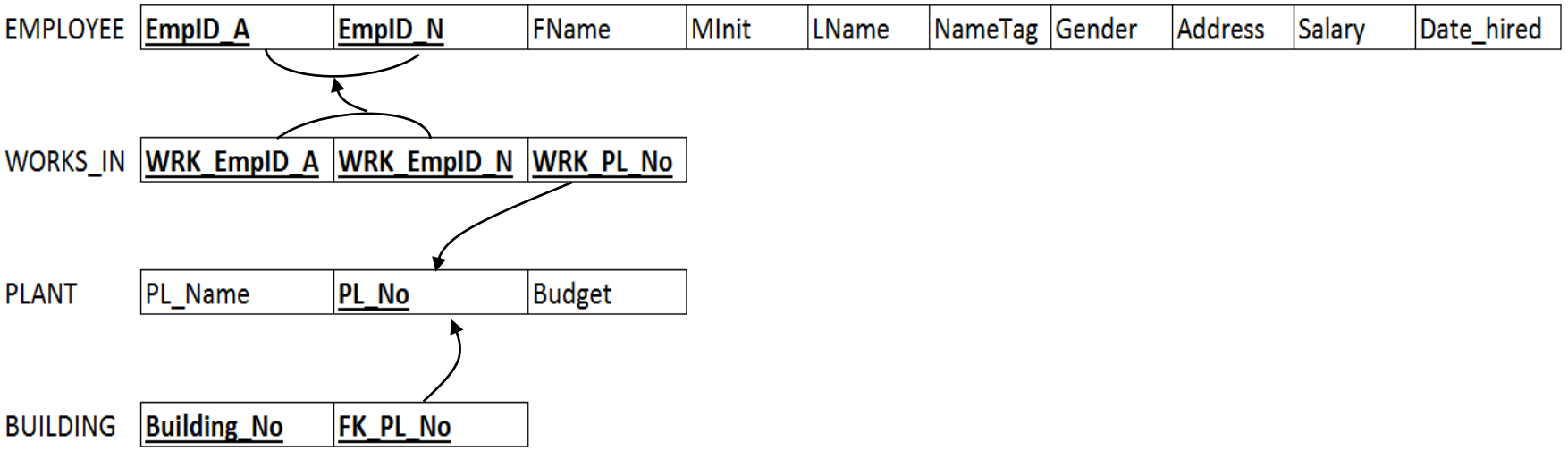
Imagine if an employee did not have to be assigned to a plant – i.e., “Partial Participation” (0,1).

we could potentially have lots of NULL values in the FK.

In general we want to avoid having lots of NULL values

You do not HAVE to model 1:n this way, but it is useful in some scenarios

An alternate mapping for [1:n]

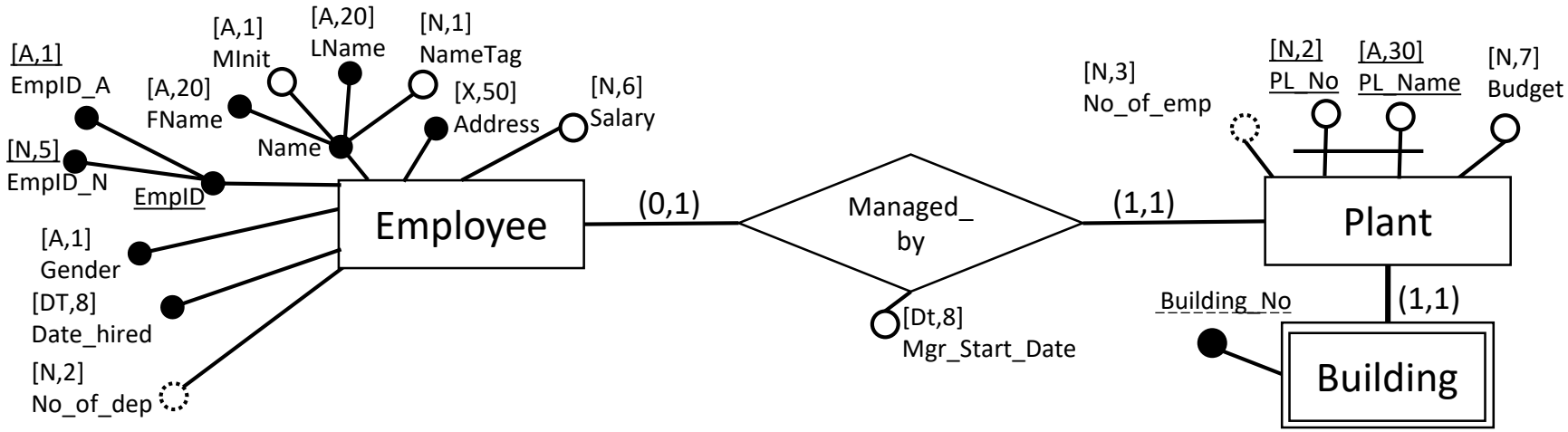


WORKS_IN.{WRK_EmpID_A,WRK_EmpID_N} ⊆ EMPLOYEE.{EmpID_A,EmpID_N}
WORKS_IN.{WRK_PL_No} ⊆ PLANT.{PL_No}

BUILDING.{FK_PL_No} ⊆ PLANT.{PL_No}

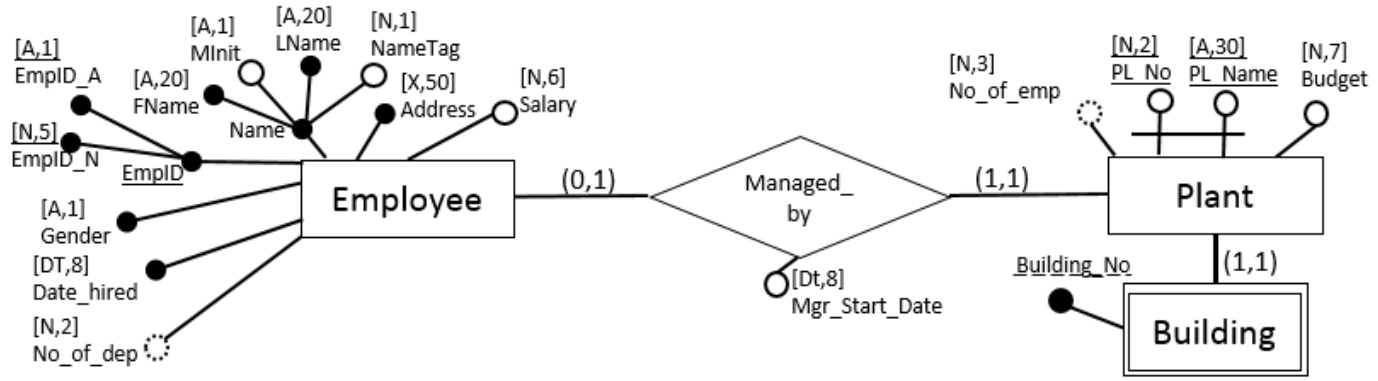
- Using this design, if an employee is not assigned to a plant, we simply do not create an instance in the WORKS_IN gerund, rather than having NULL values in the FK in the Employee relation

Mapping a 1:1 relationship

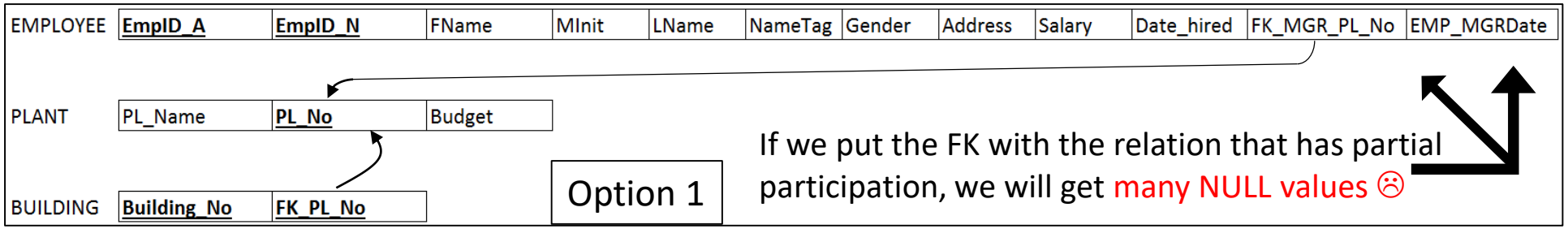


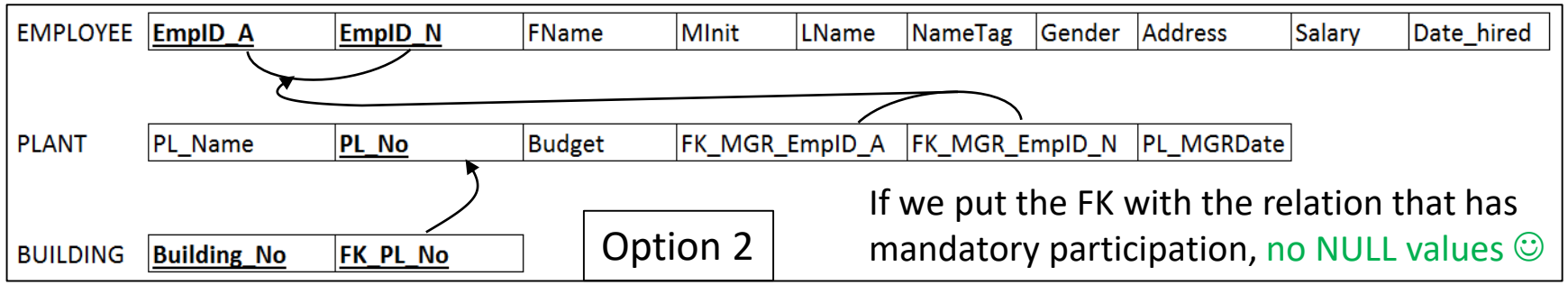
- Which side is the “child” in a 1:1 relationship?
  - The are equivalent, so technically either can have the FK
- So which side to use?
  - The relation with partial participation (employee) will have many NULL values – should be considered the “parent”
  - The relation with total participation (plant) will have no NULL values – should be considered the “Child”

Mapping a 1:1 relationship

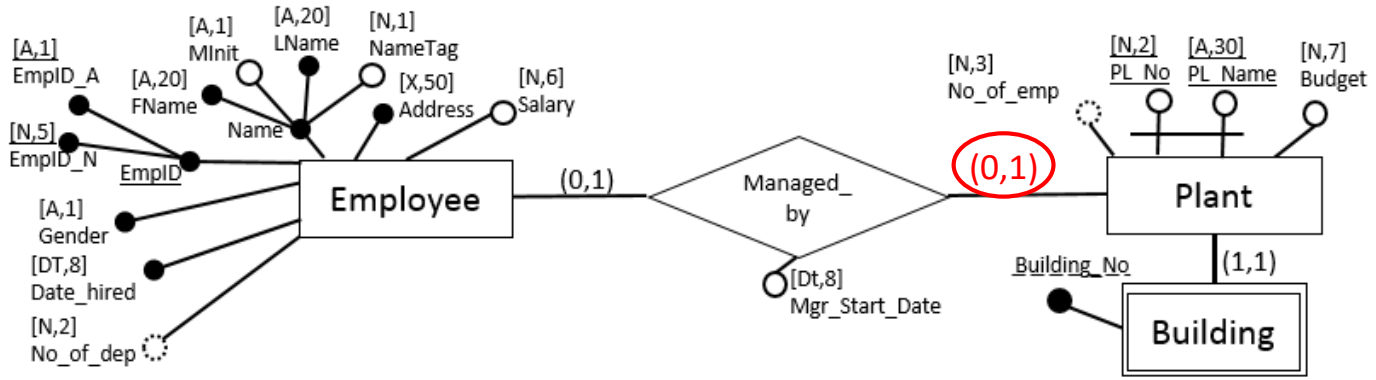


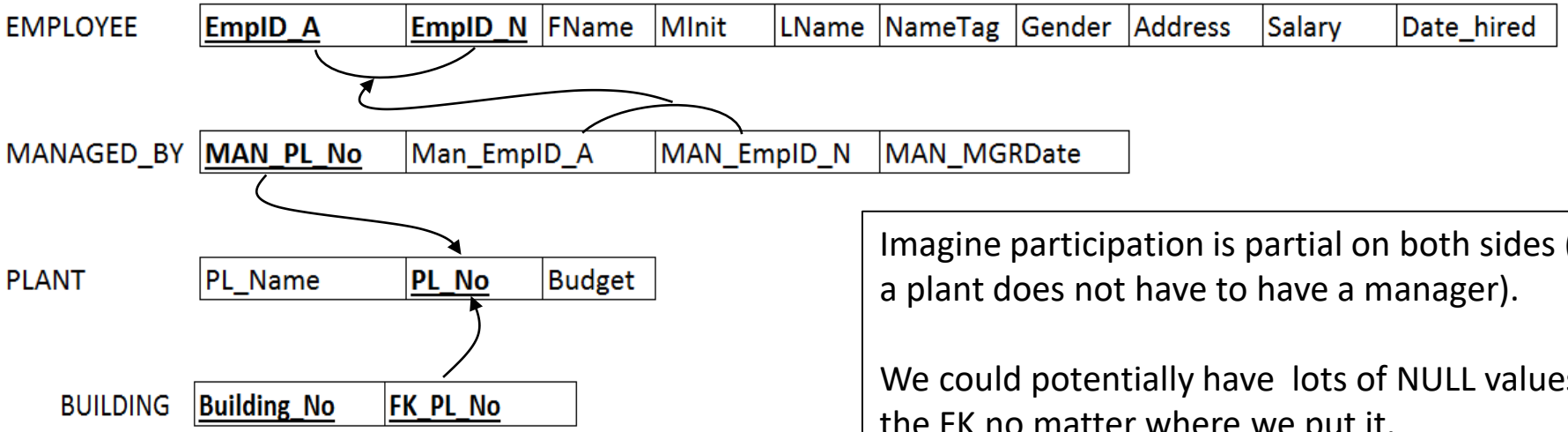
- Both technically work, but option 2 is MUCH better





One more twist on 1:1 relationships



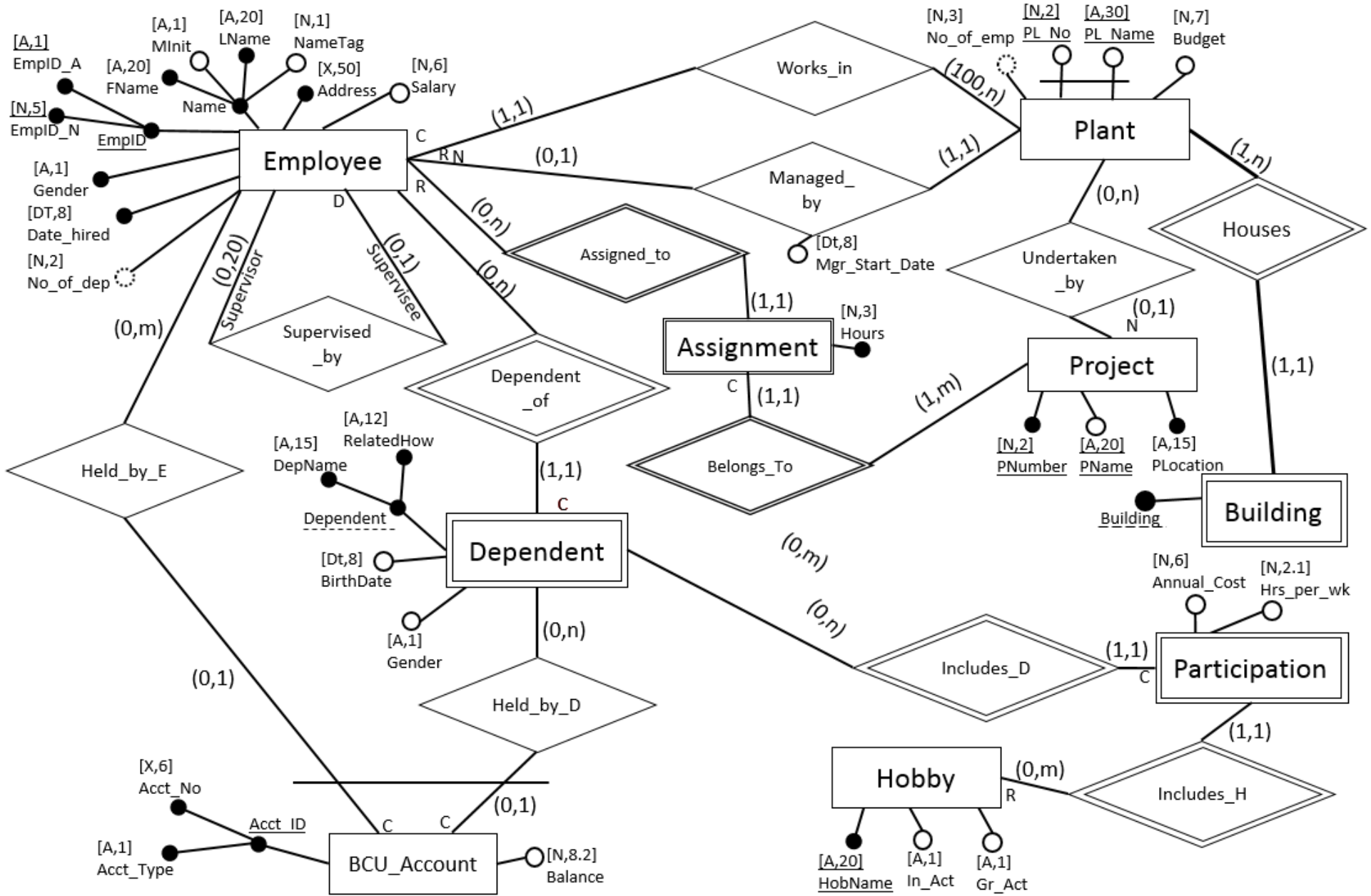


Imagine participation is partial on both sides (i.e., a plant does not have to have a manager).

We could potentially have lots of NULL values in the FK no matter where we put it.

Again, we can use a gerund to fix this situation.

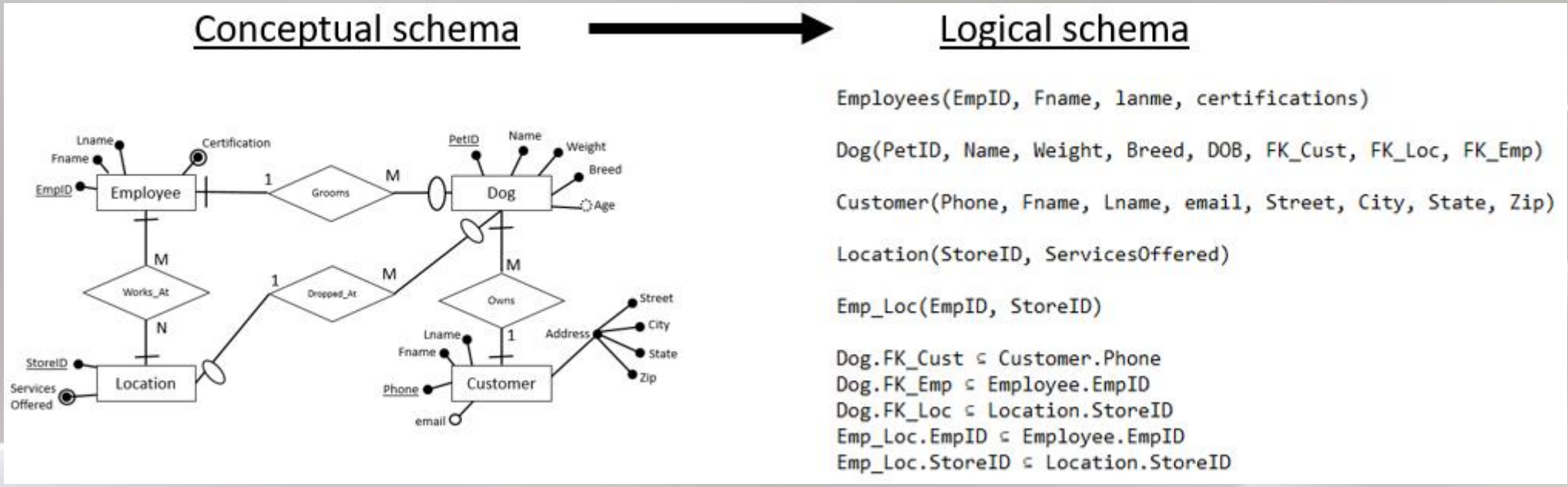
Creating a Logical Model – Page 313-315



Module 6.7
Mapping an ER Model to a Logical Schema

Next step in transforming business rules into an actual DB

ERD → Design Specific ERD → Logical Schema



Question for you….

```
CREATE TABLE instructors (
 Inst_id        char(15) CONSTRAINT pk_inst PRIMARY KEY,
 Inst_name      varchar(50) CONSTRAINT nn_Instname NOT NULL,
 Inst_office    varchar(8)
);
```

```
CREATE TABLE courses(
 Crs_id         char(8) CONSTRAINT pk_crs PRIMARY KEY,
 Crs_name       varchar(50) CONSTRAINT nn_crs NOT NULL,
 Crs_hours      numeric(1)
);
```

```
CREATE TABLE sections (
 Sec_id          char(5) CONSTRAINT pk_sec PRIMARY KEY,
 Sec_time        datetime,
 Sec_location    nvarchar(10),
 Sec_term        char(8) CONSTRAINT nn_secterm NOT NULL,
 Sec_inst_id     char(15) CONSTRAINT fk_inst FOREIGN KEY REFERENCES instructors (inst_id),
 Sec_crs_id      char(8) CONSTRAINT fk_cou FOREIGN KEY REFERENCES  courses (crs_id),
 CONSTRAINT nn_sec_crs NOT NULL
);
```

```
CREATE TABLE students (
 Stu_id         char(15) CONSTRAINT pk_stu PRIMARY KEY,
 Stu_name       nvarchar(50) CONSTRAINT nn_stuname NOT NULL,
 Stu_phone      char(12),
 Stu_GPA        numeric(1,3),
 CONSTRAINT chk_gpa CHECK (Stu_GPA <= 4.0)
);
```

```
CREATE TABLE enrollment(
Enr_stu_id      char(15),
Enr_sec_id      char(8),
Enr_Grade       char(2),
CONSTRAINT fk_stu FOREIGN KEY Enr_stu_id REFERENCES students (stu_id),
CONSTRAINT fk_sec FOREIGN KEY Enr_sec_id REFERENCES sections (sec_id),
CONSTRAINT pk_enr PRIMARY KEY (Enr_stu_id, Enr_sec_id),
);
```

Could you create an ERD or Logical Schema based on this DDL code?

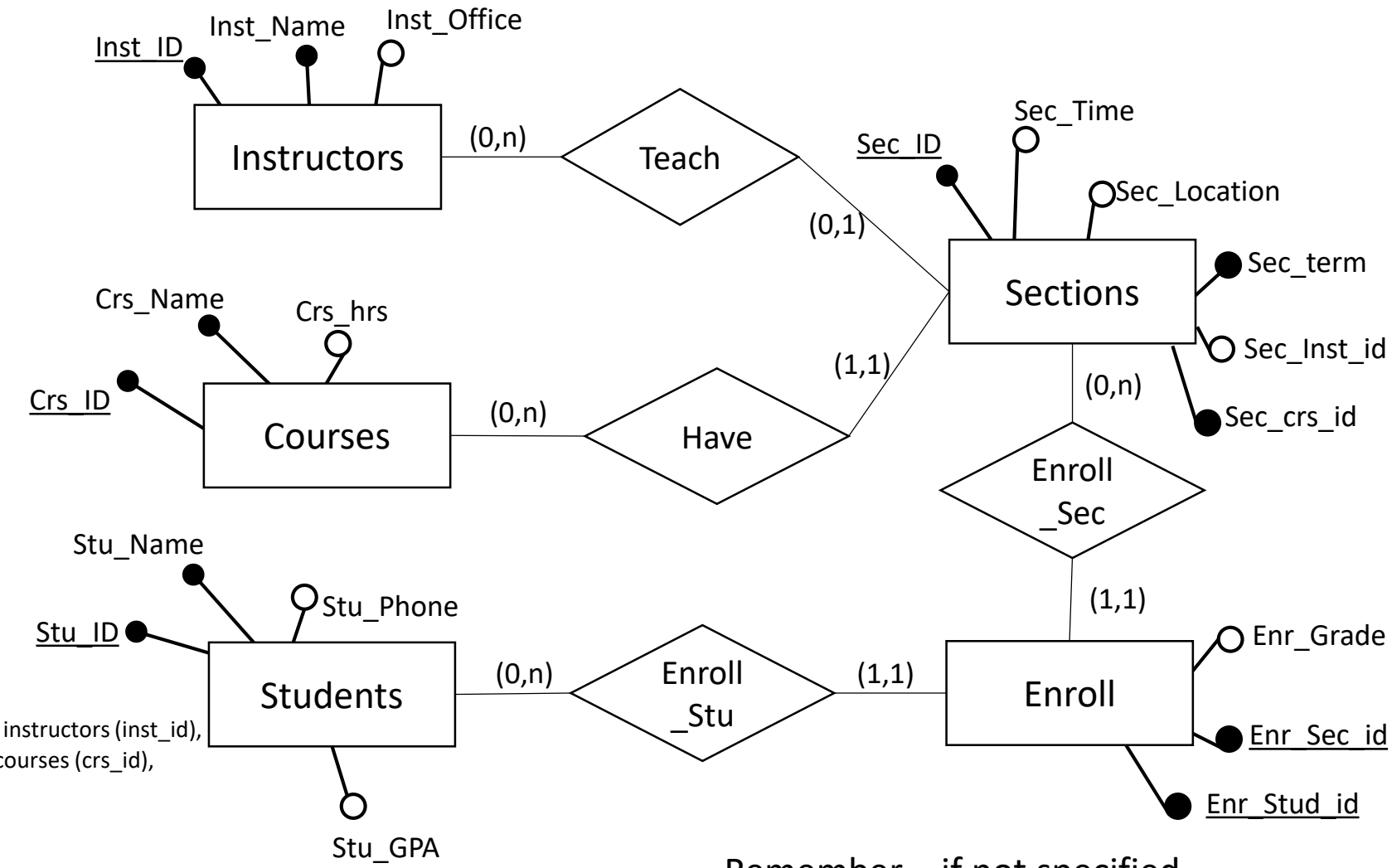
Could you create this code based on an ERD or logical schema?

Question for you….

```
CREATE TABLE instructors (
 Inst_id        char(15) CONSTRAINT pk_inst PRIMARY KEY,
 Inst_name      varchar(50) CONSTRAINT nn_Instname NOT NULL,
 Inst_office    varchar(8)
);
```

```
CREATE TABLE courses(
 Crs_id         char(8) CONSTRAINT pk_crs PRIMARY KEY,
 Crs_name       varchar(50) CONSTRAINT nn_crs NOT NULL,
 Crs_hours      numeric(1)
);
```

```
CREATE TABLE sections (
 Sec_id          char(5) CONSTRAINT pk_sec PRIMARY KEY,
 Sec_time        datetime,
 Sec_location    nvarchar(10),
 Sec_term        char(8) CONSTRAINT nn_secterm NOT NULL,
 Sec_inst_id     char(15) CONSTRAINT fk_inst FOREIGN KEY REFERENCES instructors (inst_id),
 Sec_crs_id      char(8) CONSTRAINT fk_cou FOREIGN KEY REFERENCES  courses (crs_id),
 CONSTRAINT nn_sec_crs NOT NULL
);
```



Remember – if not specified
1:n and partial participation is implied

```
CREATE TABLE students (
 Stu_id         char(15) CONSTRAINT pk_stu PRIMARY KEY,
 Stu_name       nvarchar(50) CONSTRAINT nn_stuname NOT NULL,
 Stu_phone      char(12),
 Stu_GPA        numeric(1,3),
 CONSTRAINT chk_gpa CHECK (Stu_GPA <= 4.0)
);
```

```
CREATE TABLE enrollment(
Enr_stu_id      char(15),
Enr_sec_id      char(8),
Enr_Grade       char(2),
CONSTRAINT fk_stu FOREIGN KEY Enr_stu_id REFERENCES students (stu_id),
CONSTRAINT fk_sec FOREIGN KEY Enr_sec_id REFERENCES sections (sec_id),
CONSTRAINT pk_enr PRIMARY KEY (Enr_stu_id, Enr_sec_id),
);
```

Progress Quiz Time!

- The Progress Quiz is available in Canvas
  - You MUST complete the quiz on Canvas by 5:00 on Friday – This in-class activity does not count for points!
  - Each week we will discuss the questions, so for those of you that are in class and keeping up with things, you'll have an extra easy time with it!
- Go to http://kahoot.it and we'll get started momentarily!

Exam 1

- 75 minutes, in class, on paper, closed book, individual effort
  - You only need pencil or pen – I will provide the exam, scratch paper, etc.
  - Once you start the exam you cannot leave the room for any reason – plan accordingly (i.e., use the restroom before coming to class)
  - Be on time – once the first person leaves the room, no one new may start the exam

- Will be approximately one third each of:
  - Multiple choice
  - Short answer / Matching / Fill in the Blank / etc.
  - Drawing an ERD

- I won't ask you to WRITE any SQL but you might have to INTERPRET SQL

Exam 1 (This list of topics is not all inclusive – refer to the slides for more!)

- Topics include anything we have covered up until now:
  - Modules 1, 2, 3, 6, and 10
  - This list of topics is NOT all inclusive – anything we have covered may be on the exam

- All general database concepts
  - Data/information/metadata
  - Schemas
  - File systems vs. databases
- Conceptual data modeling
  - ER Diagrams
  - Characteristics of attributes
  - Cardinality, Participation, Degrees
  - Deletion constraints
  - M:N Relationships / Gerunds

Exam 1 (This list of topics is not all inclusive – refer to the slides for more!)

- Relational modeling
  - Set operations
  - Relational algebra
  - Keys
  - Foreign Keys / Referential Integrity
  - Logical Models
- Database Creation
  - Data definition language
  - Inserting, updating, and deleting data

Questions, comments, or concerns?

Exam 1- Commonly missed concepts

- Foreign keys refer to any candidate key – PREFERABLY (and usually) the primary key, but it can be any candidate key

- There is no “child” in a 1:1 relationship – they are equal. In this case we place the foreign key in the entity that has mandatory participation in order to have fewer NULL values
  - If both sides are optional either side is OK, but better to chose the entity with fewer entities OR use a gerund to eliminate NULL vales.

- In a M:N relationship, EITHER side (or neither side, or BOTH) could be considered the parent, since it is made up of two 1:M relationships, in which each entity is the parent and the gerund is the child

BZAN 6354

Lecture 6

Go forth and do great things!

Dr. Mark Grimes, Ph.D.
gmgrimes@bauer.uh.edu

UNIVERSITY of HOUSTON
C. T. BAUER COLLEGE of BUSINESS
Department of Decision & Information Sciences