COMP3311 Database Management Systems

Spring 2022

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Relational Data Model

Prof Xiaofang Zhou

+ In This Lecture …

- What are the basic concepts in the Relational Model?
  - The relational model was introduced in 1970 by E. F. Codd It has been the turning point for modern database systems. It is the basis for several commercial Database Management Systems, for example, Oracle, MySQL and SQL Server
- What is the theoretical basis of the relational model?
- What are integrity constraints and how are they enforced (in/by relational DBMSs)?

Data integrity is the maintenance of, and the assurance of the accuracy and consistency of data over its entire life-cycle, and is a critical aspect to the design, implementation and usage of any system which stores, processes, or retrieves data - *Wikipedia*

+ Basic Concepts

- Entities and relationships are represented by tables
  - Generalizations/Specializations
  - Composite Attributes
  - Multivalued Attributes
  - Strong Entities
  - Weak Entities
  - Relationships
  - Schema Combination
- Formal semantics and languages for manipulating the tables
- Ease of implementation – write queries on tables without caring about the physical level and optimization issues
- All popular DBMSs today are based on relational data model (or an extension of it, e.g., object-relational data model)

+ Relational Model

- It represents the data as a collection of tables

Relational Model		Representation	Notation
Relation	⇔	table	$R(A_1, A_2, \ldots, A_n)$
Attribute	⇔	column	A_i
Domain	⇔	type and range of attribute values	$dom(A_i)$
Tuple / Record	⇔	row	
Attribute value	⇔	value in table cell	

Examples of attribute domains:

Attribute	Domain
age	[0-100]
name	50 alphabetic characters
salary	non-negative integer

… RM is based on a sound theoretical foundation with a simple and uniform data structure called relation

Schemas And Instances

- A set of relation schemas define a relational database
  - `Employee(emp#, name, address, hkid, works_on_project)`
  - `Project(proj#, name, budget)`
- A table can be used to show the instances of a relation schema

Employee

emp#	name	address	hkid	works_on_ project
1	Holmes D.	86 Queen	A450361	3
5	Chan B.	21 Minto	C461378	2
35	Hui J.	16 Peak	F562916	1
8	Bell G.	53 Water	A417394	2
15	Wing R.	58 Aster	C538294	3

Project

proj#	name	budget
1	E-commerce	200,000
2	Stock control	100,000
3	Web store	500,000

+ Relations, not Tables

- The term table is used interchangeably with relation
  - Every relation is a table
  - Not every table is a relation!
- Relations have specific properties, based on the mathematical set theory

City: Brisbane		Product	Year: 1998			
Region	Suburb		Qtr 1	Qtr 2	Qtr 3	Qtr 4
South	Algester	Disks	32	243	23	246
South	Calam Vale	Labels	4232	65	865	768
West	Taringa	Envelops	3242	543	4554	454
North	McDowell	Toners	23	456	24	434
South	Sunny Bank	Ribbons	324	65	56	657
West	Indooroopilly	Disks	234	6786	324	554

Not a Relation !

+ This Table is not a Relation Either

Employee Report			
Printing (New Territories)			
Smith	50K	Sai Kung	
Dilbert	40K	Yuen Long	
Jones	60K	Sha Tin	Manager
Head Office (Hong Kong Island)			
Trump	65K	Wan Chai	
Biden	78K	Wan Chai	Manager

+ Question:

How to convert the Employee Report table into relations?

+ Naive Implementation

name	salary	address	dept	loc	mgr
Smith	50000	Sai Kung	Printing	New Territories	Jones
Dilbert	40000	Yuen Long	Printing	New Territories	Jones
Jones	60000	Sha Tin	Printing	New Territories	Jones
Trump	65000	Wan Chai	Head Office	Hong Kong Island	Biden
Biden	78000	Wan Chai	Head Office	Hong Kong Island	Biden

- Consider these updates:
  - Printing department moved to Hong Kong Island
  - Smith is new manager of Printing Department

+ A Better Implementation

Employee

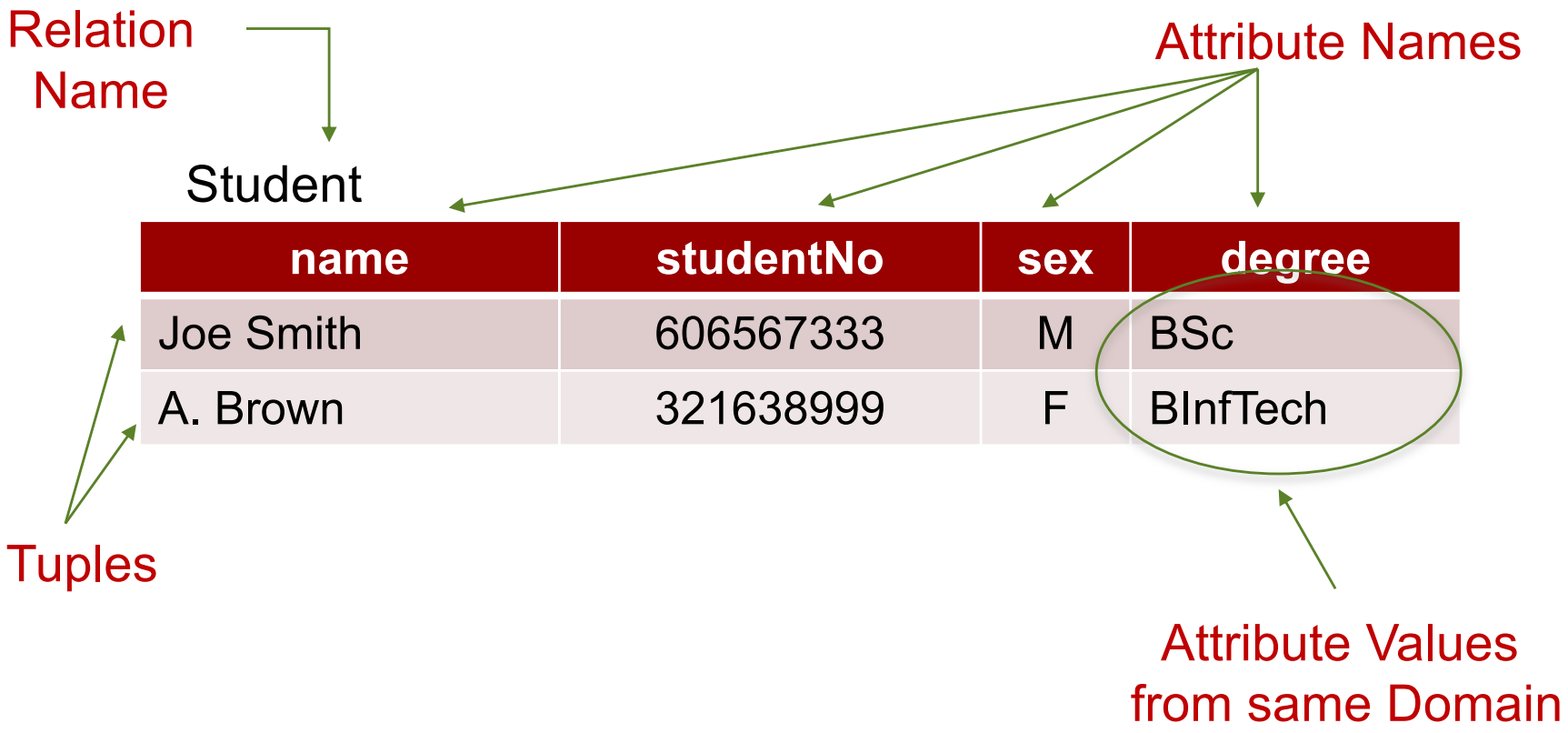
name	salary	address	dept
Smith	50000	Sai Kung	Printing
Dilbert	40000	Yuan Long	Printing
Jones	60000	Sha Tin	Printing
Trump	65000	Wan Chai	Head Office
Biden	78000	Wan Chai	Head Office

Department

dept	loc	mgr
Printing	New Territories	Jones
Head Office	Hong Kong Island	Biden

- Again, consider these updates:
  - Printing department moved to Hong Kong Island
  - Smith is new manager of Printing Department

+ Relation Components



+ Domains

- A domain D is a set of atomic values
- An atomic value is indivisible (as far as the relational data model is concerned)
- Each domain has a data type or format
- Examples:
  - Domain of names:
    - {Joe Smith, Alan Yates, Bob Lovell, Jane Austin, ...}
  - Domain of degrees:
    - {PhD, MPhil, MDataSci, BCompSci, BEng, MSBigData, …}

+ Are These Values Atomic?

name	empNum	salary	phones
Joe Smith	367-90	30800	34900987
			36789817
A. Brown	987-87	65000	39871236
Bob Forrester	884-89	76900	34890699
			0419998472
			34980876

personName	homeAddress
Roberts, Mike	Flat 12, 12/F, Acacia Building, 105 Kennedy Road, Wan Chai, Hong Kong
…	…

+ Domain Types

- Integers
- Numbers and currency
- Fixed or variable length character strings
- Date, time stamp

- Sub-range from a data type
  - e.g., 0 ≤ `age` ≤ 100
- Enumerated data type,
  - e.g. `gender` in {‘Male’, ‘Female’, ‘Other’}

+ Cartesian Product

- A relation (instance) is any subset of the Cartesian product of the domains of values

  Let dom(`name`) = { Lee, Cheung }

  dom(`grade`) = { A, B, C }

  Then the Cartesian product of the domains is

  dom(`name`) × dom(`grade`) = { <Lee, A>, <Lee, B>, <Lee, C>, <Cheung, A>, <Cheung, B>, <Cheung C> }

- A relation `r` according to the relation schema `StudentGrade(name, grade)` can be defined as any subset of the Cartesian product dom(`name`) × dom(`grade`)

  `r(StudentGrade)` = { <Lee, A>, <Cheung C> }
  $\subseteq$ dom(`name`) × dom(`grade`)

Properties of Relations

- Tuples in a relation are not ordered, even though they are represented in a tabular form
  - Recall that a relation is a set of tuples
- All attribute values are atomic
  - Multivalued and composite attribute values are not allowed in relations, although they are permitted in the ER model
- A special null value is used to represent values that are:
  - not applicable (phone number for a client that has no phone)
  - missing (there is a phone number, but we do not know it yet)
  - not known (we do not know whether there is a phone number)

+ Uniqueness Constraint

- All tuples in a relation must be distinct
  - No two tuples can have same values for all attributes

name	address	degree
J Smith	23 Milton Rd	BEng
J Smith	23 Milton Rd	BEng

+ Keys

Superkey

A superkey, $\mathcal{S}$, of relation R={A_1, A_2, …, A_n} is a set of attributes $\mathcal{S} \subseteq$ R such that for any two tuples t_1 and $t_2 \in$ r(R), $t_1[\mathcal{S}] \neq t_2[\mathcal{S}]$

- A superkey is *any* set of attributes that can identify a unique tuple in r(R).

`Employee(emp#, name, address, hkid, works_on_project)`

where `emp#` and `hkid` are unique.

Possible superkeys: `emp#`
`hkid`
`{emp#, name}`
`{hkid, name, address}`
plus many others

Candidate key

A superkey that is minimal (A relation may have several candidate keys)

Primary key

One of the candidate keys (Chosen by the database designer)

+ Example Key

WorksIn

dept	empNum	date	hours
Printing	42	1/1/2022	3
Printing	30	1/1/2022	6
Printing	42	2/1/2022	9
Sales	14	1/1/2022	6
Sales	30	1/4/2022	8

- Are they possible keys for relation WorksIn?
  1. (dept)
  2. (empNum)
  3. (dept, empNum)
  4. (dept, empNum, date, hours)
  5. (dept, empNum, date)

+ Determining the Key

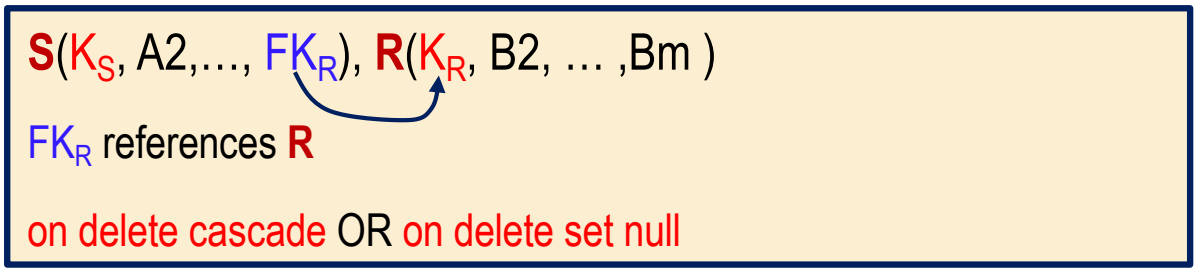
Enrolment

studentNo	subCode	sem	year	status
s3457890	CS181	Spring	1/2020	Pass
s9875604	CS181	Spring	1/2021	Fail
s8901265	CS386	Fall	2/2018	Withdraw
s3457890	CS182	Fall	2/2021	Pass
s3457890	CS271	Spring	1/2020	Pass

- Are they possible keys for relation Enrolment?
  1. (studentNo)
  2. (subCode)
  3. (studentNo, subCode)
  4. (studentNo, subCode, sem, year)

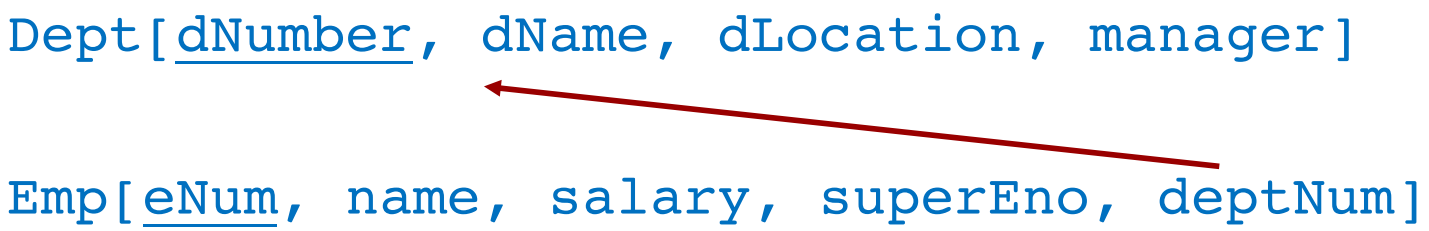
+ Integrity Constraints

- Entity integrity constraint
  - If X is a primary key of R, then X cannot contain null values
- Referential integrity (foreign key) constraint
  - Given two relations R and S, relation S may reference relation R via a set of attributes FK_R that forms the primary key K_R of R
  - The attributes FK_R in S are called a foreign key.
  - The value of the foreign key FKR in a tuple of S must either be equal to the value of the primary key K_R of a tuple in R or be entirely null



Integrity Constraint Examples

- Key and entity integrity constraints are specified on individual relations
- Referential integrity constraints are specified between two relations



Value of `deptNum` must be equal to `dNumber` or be null

Other Integrity Constraints

- There are many application-specific constraints which cannot be specified as the integrity constraints (primary key or florigen key constraints) or domain constraints
- Semantic Constraints
  - "The salary of an employee should not exceed the employee's supervisor's salary"
  - "The maximum number of hours that an employee can work on a project is 56"
- Transition Constraints
  - "The salary of an employee can only increase"
- Such constrains are often implemented in a constraint specification language (e.g., SQL) using triggers and assertions

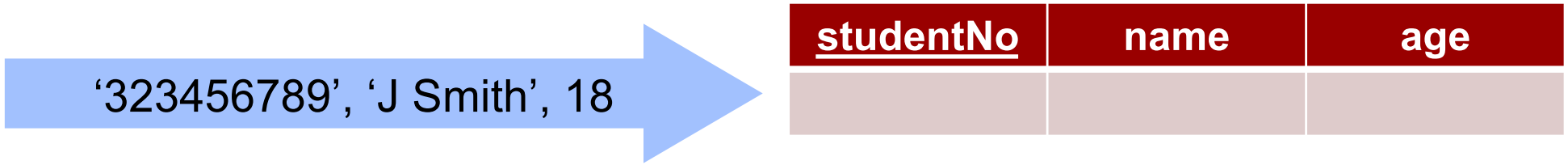
+ Constraint Enforcement

- Enforcement of database constraints ensures that the database remains consistent
- Changes to the database must not violate any declared integrity constraints (leave the database in an inconsistent state)
- If a database update is submitted to the DBMS that would violate integrity, it must be rejected

… at the same time, a constraints can be dropped or added

Constraints & Insertion

- Integrity constraints can be violated by inserting a new tuple
  - studentNo value already exists
  - age is ‘old’ instead of 18
- The insert can be rejected or corrected



+ Constraints & Deletion

- Referential integrity can be violated if the tuple being deleted is referenced by a foreign key from other tuples
  - deleting a dept while there are still employees working in that department
- The deletion can be rejected, cascaded or the referencing attribute values can be modified

delete student ‘323456789’

studentNo	name	age
323456789	J Smith	18

+ Constraints & Modification

- Non key values
  - domain check
- Primary key
  - similar to performing a delete and an insert
- Foreign key
  - DBMS must ensure new value refers to existing tuple in referenced relation

change ‘323456789’ age to 19

studentNo	name	age
323456789	J Smith	18

+ Reduction To Relational Schema

- We need to reduce:
  - attributes → composite, multivalued
  - entities → strong, weak
  - relationships → degree (e.g., unary, binary, ternary, etc.)
  - → cardinality and participation constraints

Cardinality/participation constraints in the E-R model
are reduced to
referential integrity constraints in the relational model

+ (7+1)-Steps for Mapping

Input: an ER model

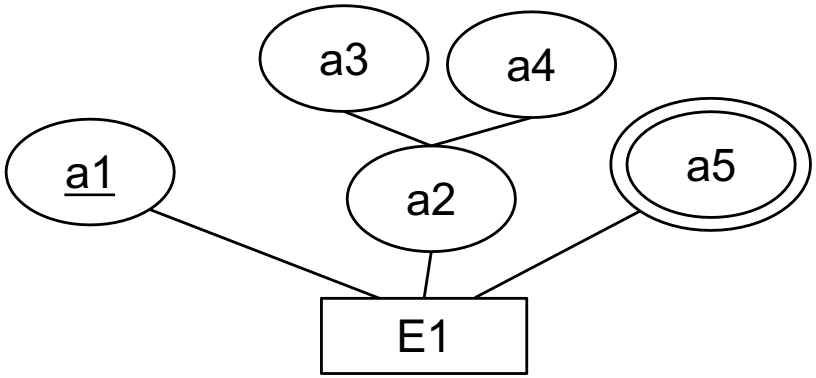
Output: relations with primary/foreign key constraints

Steps:

1. Entity mapping (strong entity with simple/composite attributes)
2. Weak entity mapping
3. Multi-valued attribute mapping
4. Binary relationship mapping (1:1)
5. Binary relationship mapping (1:*)
6. Binary relationship mapping (*:*)
7. n-ary relationship mapping
8. Super & sub-class mapping

Step 1: Regular Entity

- Regular: non-weak entity with simple attributes
- For each entity type E, create a relation with all attributes
  - Include only simple component attributes of a composite attribute
  - Don't include derived attributes
  - Choose one candidate key of E as the primary key
- Note: consideration foreign key relationship, weak entities and multi-valued attributes later

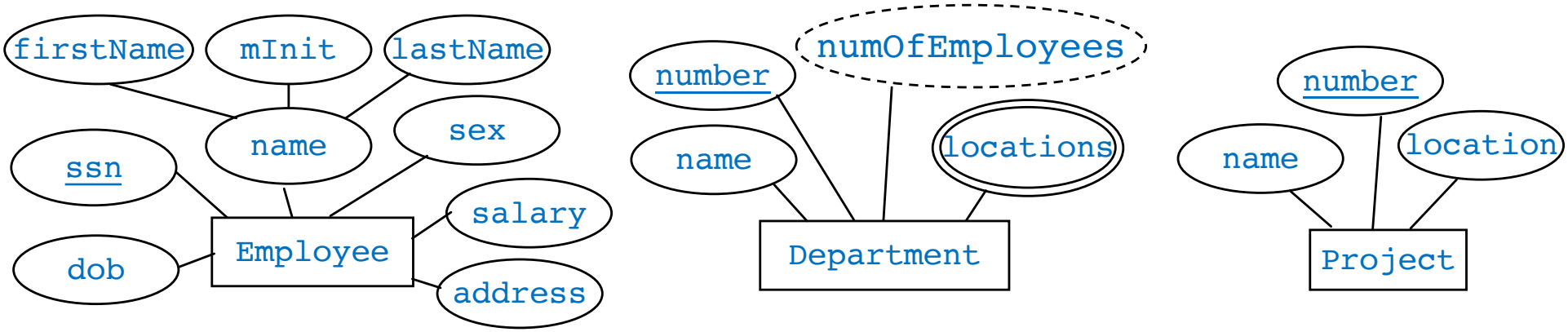


E1[a1, a3, a4]

… what about if a2 is the key? What about a5?

+ Step 1: Examples

- There are three (strong) entity types in the Company Database: Employee, Department, Project



Employee[ssn, firstName, mInit, lastName, dob, address, sex, salary]

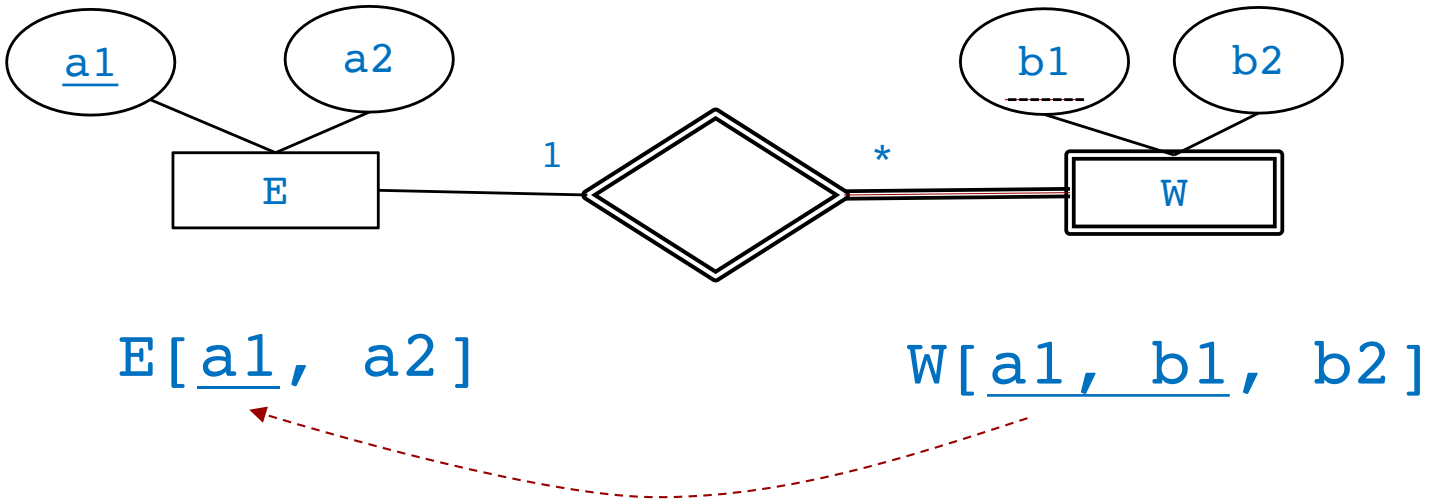
Department[number, name]

Project[number, name, location]

…for Dependent, numOfEmployees and locations are not considered

+ Step 2: Weak Entity

- For each weak entity type `W` with owner entity type `E`
  - Create a relation that includes all simple attributes of `W`
  - Include as foreign key attributes in `W` to the primary key attributes of `E`
  - The primary key of `W` is the combination of the primary key of `E` and the partial key of `W`



…if W has multiple owners, include the primary keys of all owner relations

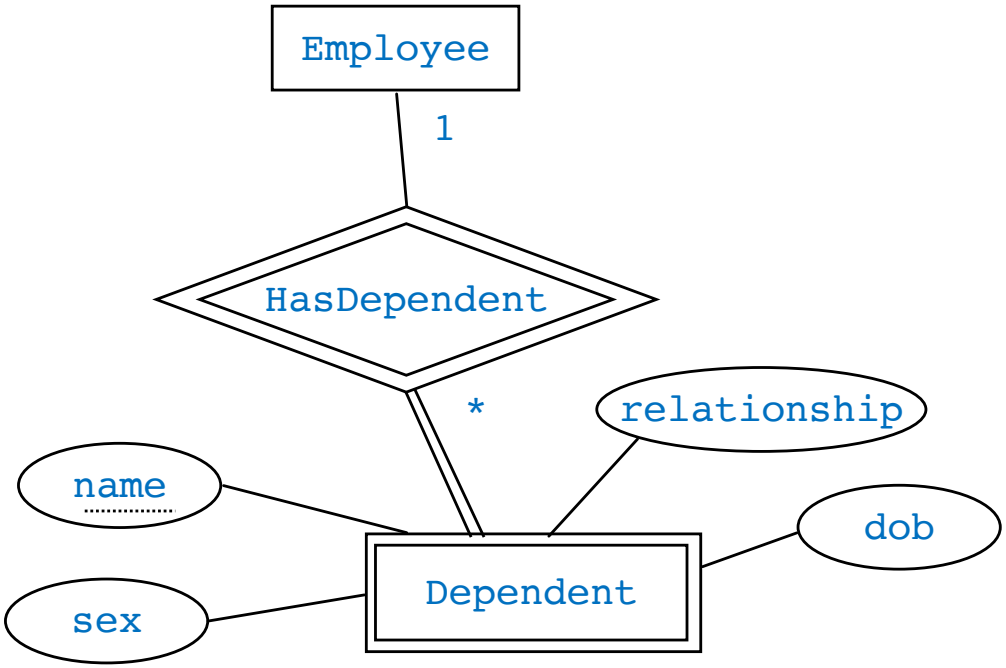
+ Step 2: Example

- This is one weak entity type : `Dependent`

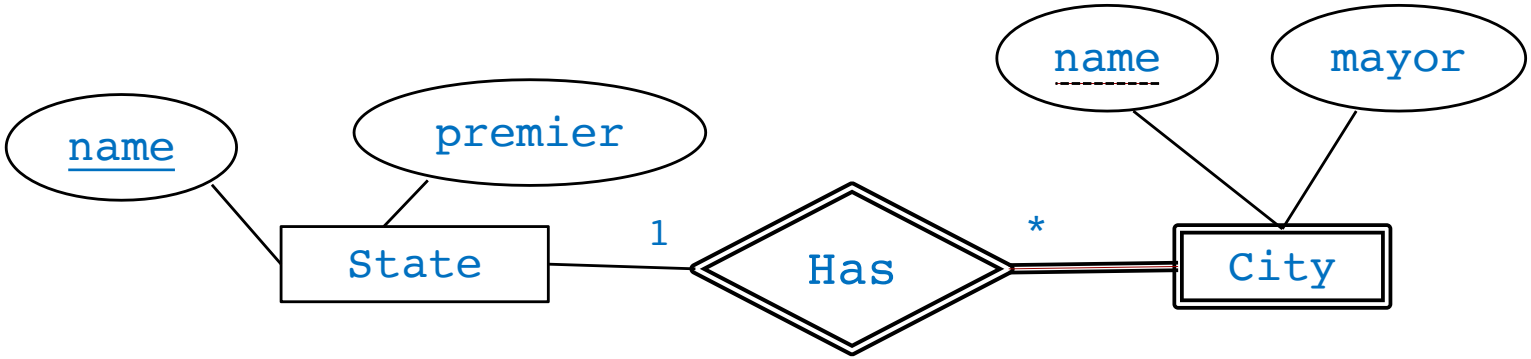
  `Dependent[ssn, name, sex, dob, relationship]`

- Foreign Key:

  `Dependent.ssn → Employee.ssn`



+ Multiple-Choice Question:



Convert this ER diagram to relations, resolving the dual use of "name" in some reasonable way. Which schema below is the most reasonable translation from ER to relations? (*foreign key attributes are shown in bold*)

A. City[name, mayor], State[name, premier]

B. City[cName,**sName**,mayor], State[sName,premier]

C. City[cName,**sName**,mayor], State[sName,premier]

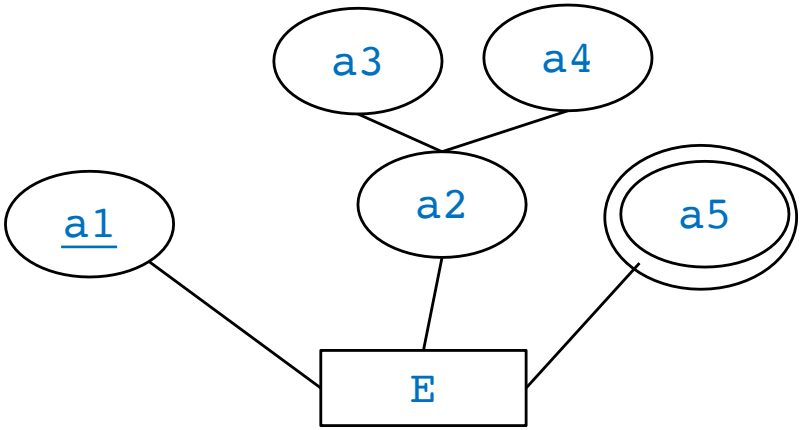
D. City[cName,**sName**,mayor], Has[cName,sName], State[name,premier]

E. None of the above

A doesn't consider weak entity; B is incorrect as its key is cName only; C is correct. D has an extra table Has.

+ Step 3: Multivalued Attributes

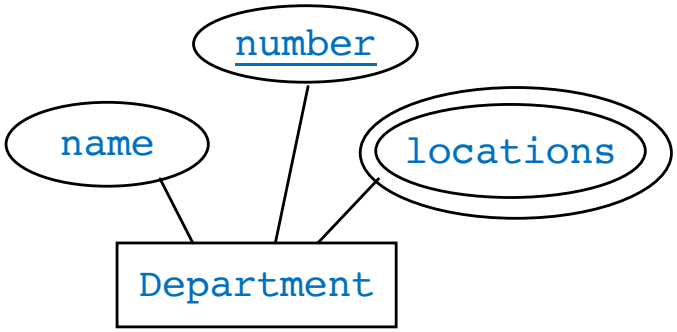
- For each multi-valued attribute A, create a new relation R that includes an attribute corresponding to A plus the primary key K (as a foreign key of R) of the relation that represents the entity type or relationship type that has A as an attribute
  - The primary key of R is the combination of attributes A & K



E[a1, a3, a4]

R[a1, a5]

+ Step 3: Example

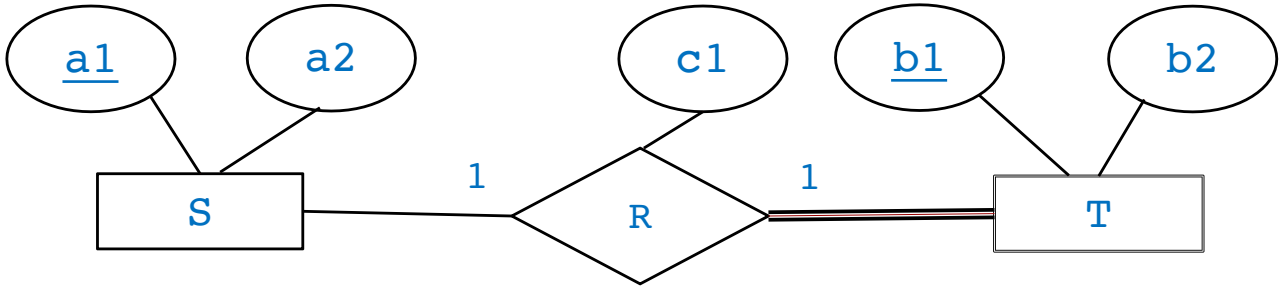


DeptLoc[dNumber, dLocation]

Notice the foreign key relationship

Step 4: Binary 1:1 Relationship

- For each binary 1:1 relationship type R, with participation entities (relations) S & T
  - Choose one relation (say T) and include as foreign key in T to the primary key of S
    - It is better to choose T, the entity type with total participation in R
  - Include all the simple attributes (or the simple components of composite attributes) of R as attributes of T

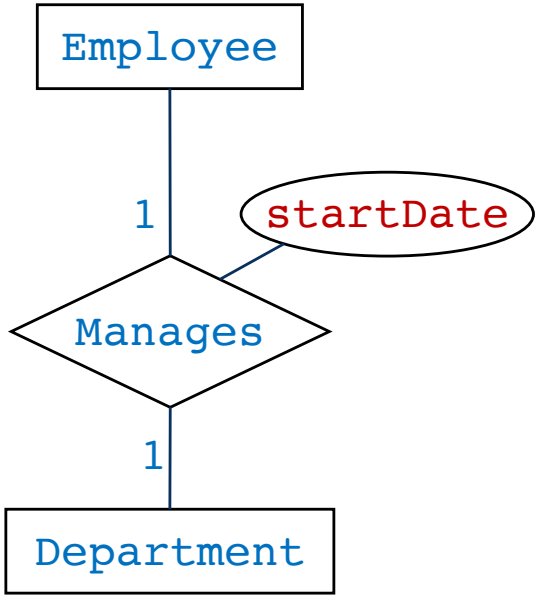


S[a1, a2]

T[b1, b2, a1, c1]

…why it's better to choose the side with total participation?

+ Step 4: Example



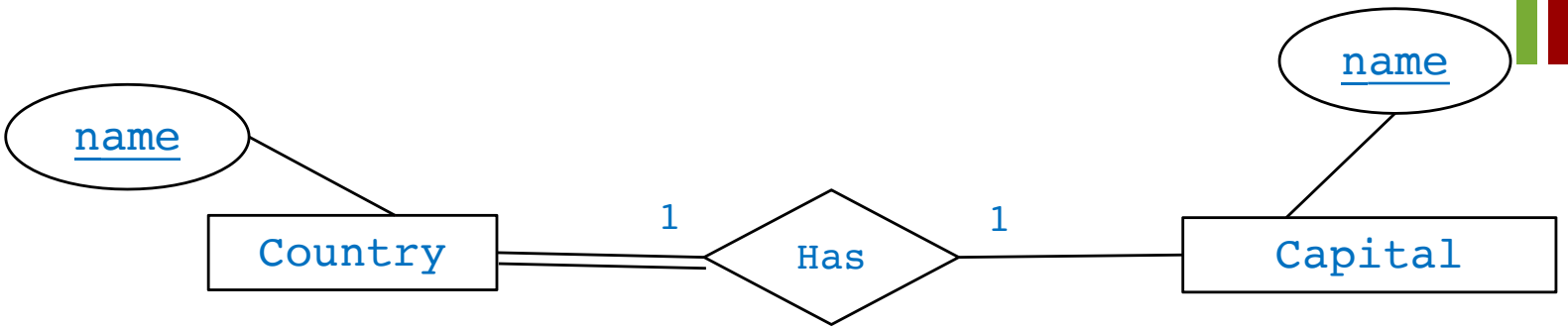
Old: `Department[number, name]`

New: `Department[number, name, mgrSSN, mgrStartData]`

Added: `Department.mgrSSN → Employee.ssn`

…there is no total participation here, so this choice doesn't matter. If a department must have a manager, this is a better choice

+ Multiple-Choice Question



Which schema below is a reasonable translation from ER to relations? (foreign key attributes are shown in bold)

A. `Country[coName, caName], Capital[caName]`

B. `Country[name], Capital[name]`

C. `Country[coName, caName]`

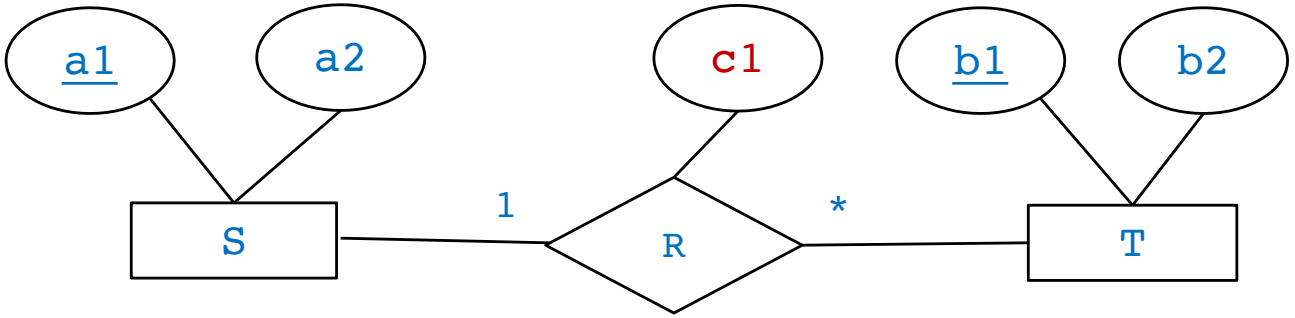
D. Both A and C

E. All of A, B, and C

A is correct.

+ Step 5: Binary 1:N Relationship

- For each (non-weak) binary 1:* relationship type R, identify relation T that represents the participating entity type at the n-side of the relationship type
  - Include as foreign key of T the primary key of relation S that represents the other entity type participating in R
  - Include any simple attributes (or the simple components of composite attributes) of R as attributes of T



S[a1, a2]

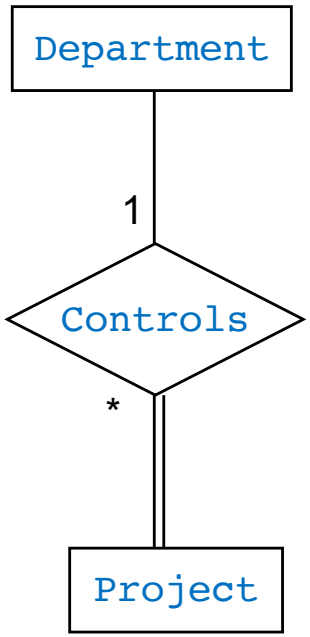
T[b1, b2, a1, c1]

+ Step 5: Example 1


```
Employee[ssn, firstName, mInit,
  lastName, dob, address, sex, salary,
  dNum]
```

where primary key of `Department` is included as a foreign key in the `Employee` relation (`dNum`)

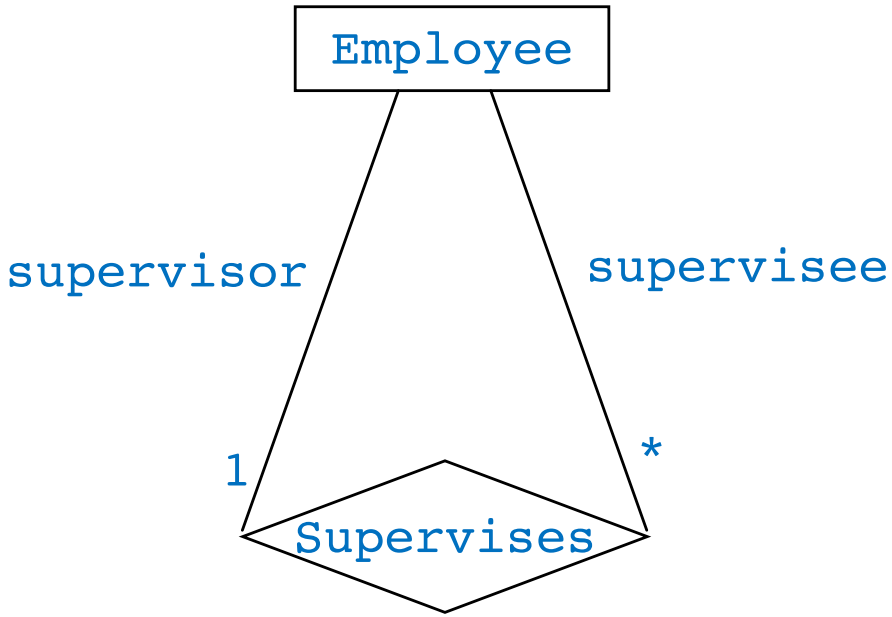
+ Step 5: Example 2



`Project[number, name, location, dNum]`

Where primary key of `Department` is included as a foreign key in the `Project` relation (`dNum`)

+ Step 5: Example 3

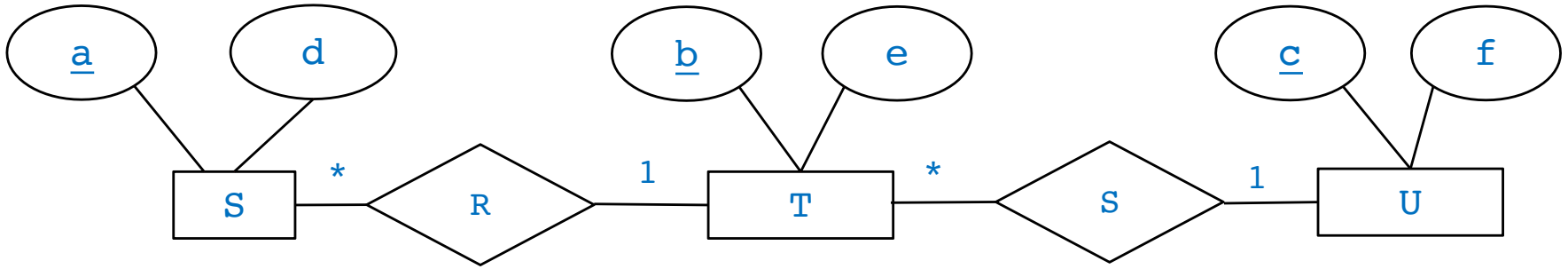


Employee[ssn, firstName, mInit, lastName, dob, address, sex, salary, dNo, **mgrSSN**]

Where primary key of Employee is included as a foreign key within Employee (called mgrSSN)

Note the recursive relationship!

+ Multiple-Choice Question



Translate the ER diagram to relational schema. Which of the following can appear in your relational schema? (foreign key attributes in bold):

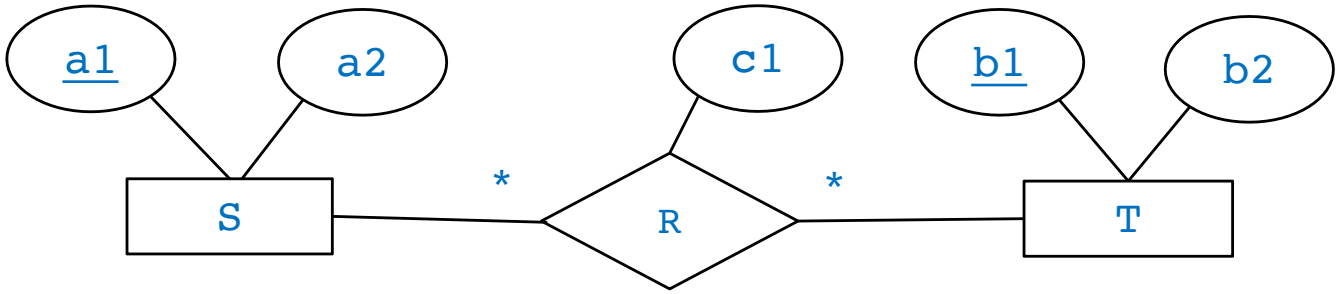
1. S[a,**b**,d]
2. T[b,**c**,e]
3. U[b,**c**]
4. All of these
5. None of these

Old: S[a, d] T[b, e] U[c, f]

(1) is not correct as b should not be part of the key. The correct design is S(a, b, d), T(b, c,e) and U(c, f). So only (2) can appear in the final schema.

Step 6: Binary M:N Relationship

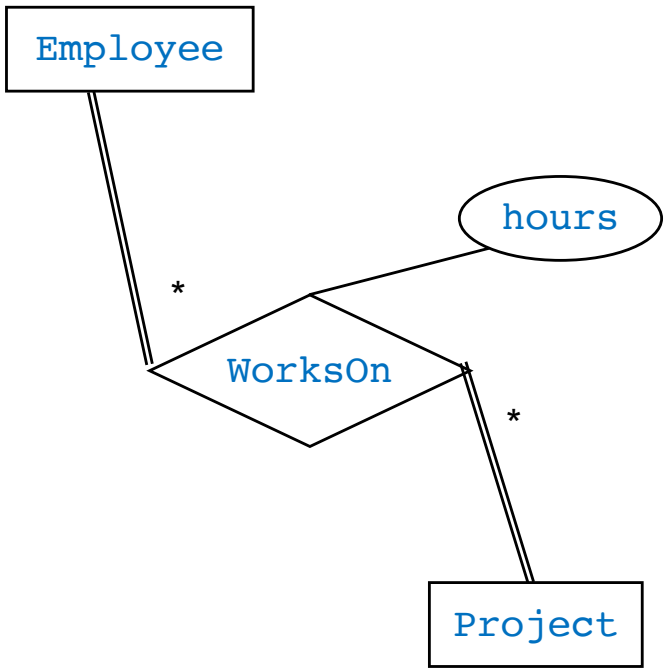
- For each binary *:* relationship type R, create a new relation R
  - Include the primary keys of the relations that represent the participating entity types (i.e., S and T) as foreign key of R
  - The combination of foreign keys will form the primary key of R
    - Note: cannot represent the *:* using a single foreign key in one relation because of the *:* cardinality ratio
  - Include any simple attributes of R as attributes of R



S[a1, a2] R[**a1, b1**, c1] T[b1, b2]

…what about the case if one side 1?

+ Step 6: Example



`WorksOn[eSSN, pNo, hours]`

Where `WorksOn` includes the primary keys of `Project` and `Employee` as foreign keys

The primary key of `WorksOn` is the combination of the foreign key attributes

`hours` in `WorksOn` represents the attribute of the relationship type

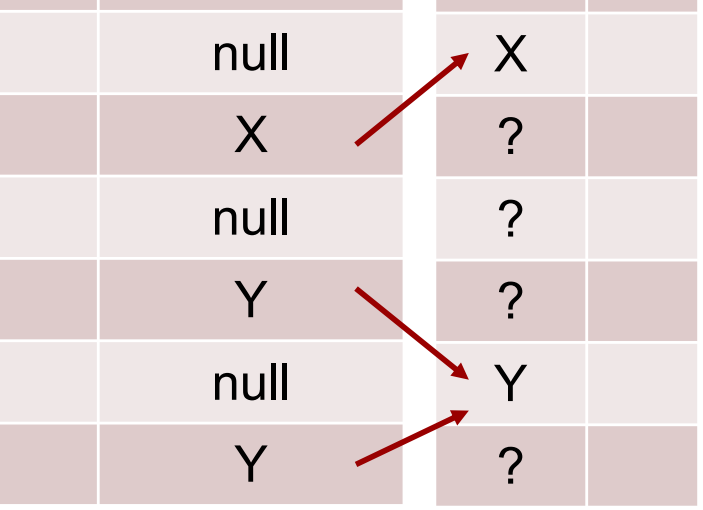
More on *:* Mapping

- Note that 1:1 and 1:* relationships can always be mapped in the same way as *:*
- It is advantageous to have few relationship instances exist as it reduces the number of “nulls” that appear as foreign key values

+ Sparse 1:1 Relationship

PK2	…	PK1 as FK
?		null
?		null
A		X
?		null
B		Y
?		null
C		Y

PK1	…
?	
X	
?	
?	
?	
Y	
?	



Standard Implementation

PK2	…
?	
?	
A	
?	
B	
?	
C	

PK1	…
?	
X	
?	
?	
?	
Y	
?	

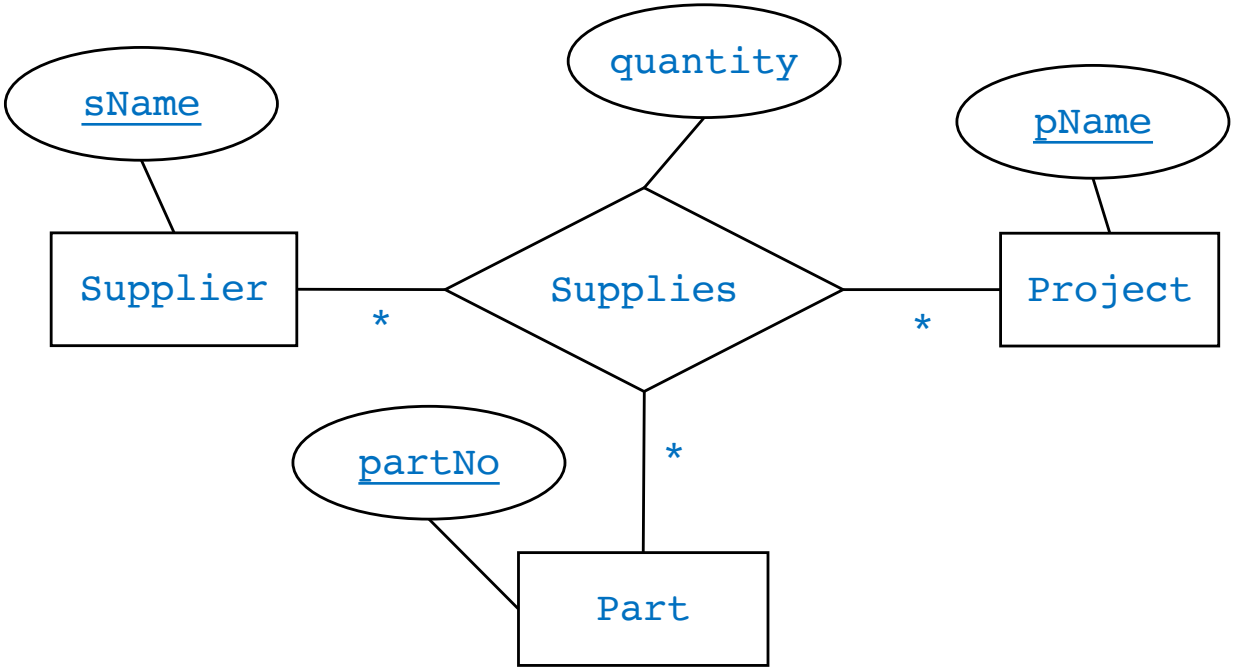
PK2	PK1
A	X
B	Y
C	Y

: Implementation

+ Step 7: N-ary Relationship Type

- For each “n-ary” relationship type `R` with participating entities, create a new relation `R`
  - Include as foreign key attributes of `R` the primary keys of the relations that represent the participating entity types in `R`
  - Include any simple attributes of the n-ary relationship type
  - The combination of foreign keys referencing the relations representing the participating entity types is used to form primary key of `R`

+ Example: 3-ary Relationship



Supplier[sName, ...]

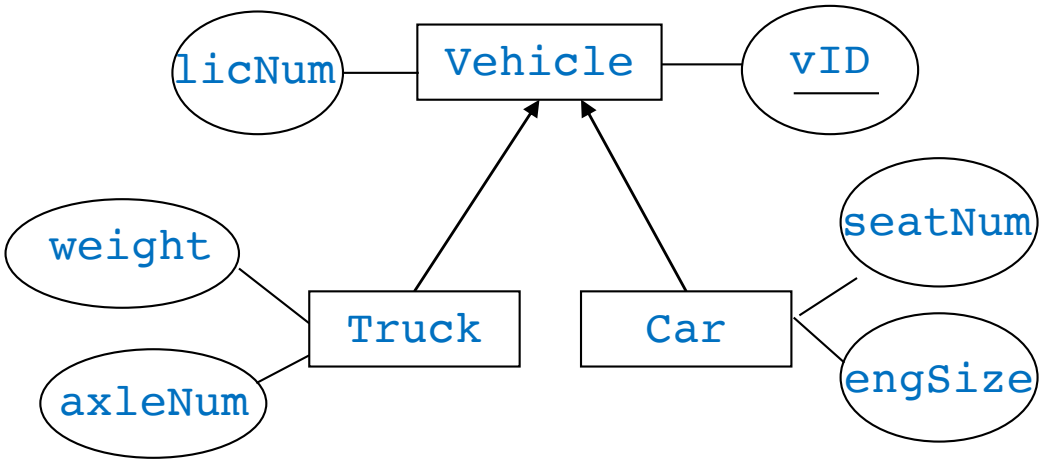
Project[pName, ...]

Part[partNo, ...]

Supplies[**sName, pName , partNo**, quantity]

+ Step 8: Super & Sub-classes

- A general case)
  - We create a relational table for the superclass and create a relational table for each subclass
  - The primary key of each of the subclass is the primary key of the superclass, which also become foreign keys



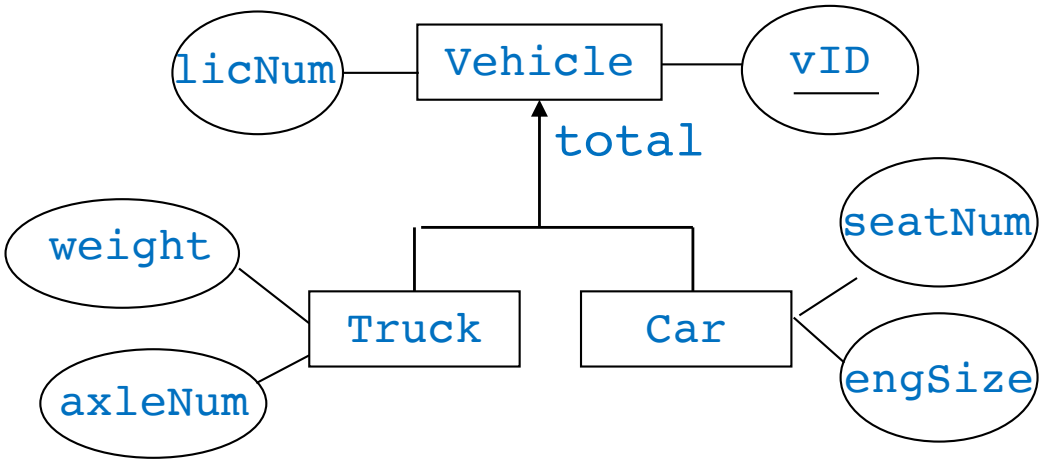
Vehicle(vID, licNum)

Truck(**vID**, weight, axleNum)

Car(**vID**, seatNum, engSize)

+ Step 8 (Special Case)

- A special case for disjoint-total only
  - We create a relational table for each subclass. The attributes of the superclass are merged into each of the subclasses.
  - The primary key of the subclass table is the primary key of the superclass.



Overlapping: redundancy
Partial: may lose superclass entities not in any subclass

Other design options also possible

Truck(vID, licNum, weight, axleNum)

Car(vID, licNum, seatNum, engSize)

+ Design Choices

- ER model is a means of capturing user's data requirements. However, different designers may interpret the semantics of the user's requirements differently
- This may result in the same UoD being represented by different ER diagrams because of different design choices
- These design choices in the ER Model impact on the resulting relational schema

+ Summary

- Relational Data Model (RDM)
  - Representation and terminologies
- Translating ERM into RDM
  - From design to implementation
  - The 8-steps

Input: an ER model

Output: relations with primary/foreign key constraints

Steps:

1. Entity mapping (strong entity with simple/composite attributes)
2. Weak entity mapping
3. Multi-valued attribute mapping
4. Binary relationship mapping (1:1)
5. Binary relationship mapping (1:*)
6. Binary relationship mapping (*:*)
7. n-ary relationship mapping
8. Super & sub-class mapping

+ Readings

- Textbook
  - RDM: Ch. 2 for both 6th and 7th ed)
  - EDR to Relational Reduction: Ch. 7/6ed, or Ch. 6/7ed