E/R Diagram and Relational Data Model

CEE412 / CET522

Transportation Data Management and Visualization

WINTER 2020

Announcements

TA

- Ziyuan Pu, Ph.D Candidate
- STAR Lab, More Hall 101
- Email: ziyuanpu@uw.edu

Assignment 1 due today

Assignment 2 out today

- Related contents will be covered today & this Friday.
- There is more than one good designs for a particular database. Be creative!
- The required reading is also a good reference for this assignment.

Announcements

Find your team member for class projects

- Till now, 28 undergrads and 22 grads registered.
- **Four members** per team (two team with three members).
  - 4 * 11 teams + 3 * 2 teams = 50 teams
- At least **one grad** and **two undergrads** in each team
  - A team with one/two grads.

Entity/Relationship Diagrams

Entity/Relationship Diagrams

What is an E/R diagram?

- A data modeling technique
- A visual representation of entities and relationships between them
- A way for you to sketch out a database design

Entity/Relationship Diagrams

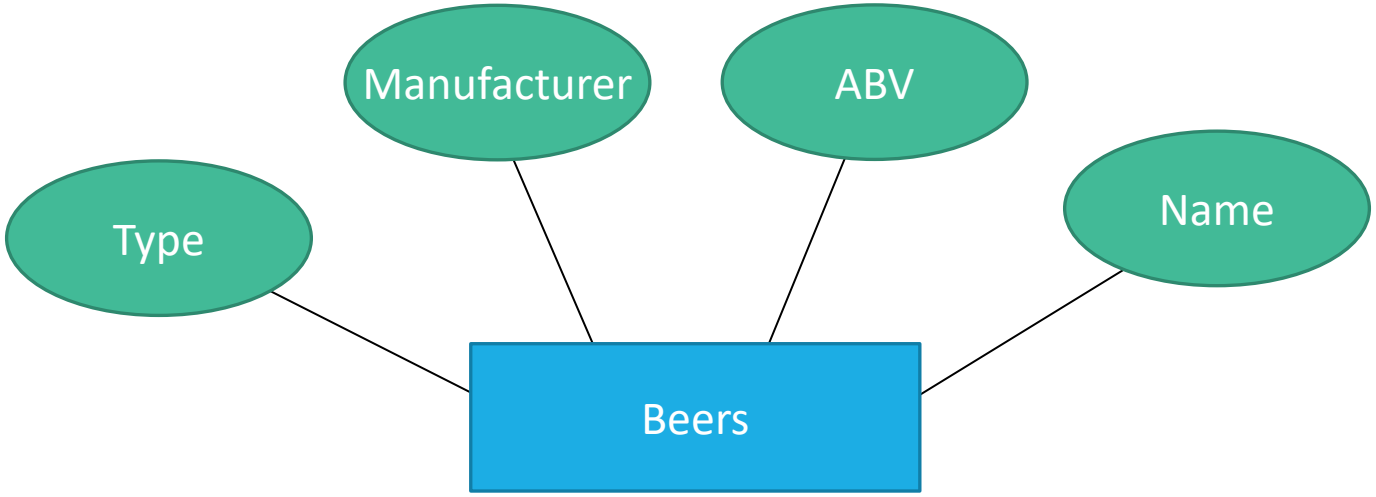
- Entity sets: represented by rectangles

- Attributes: represented by ovals

- Relationships: represented by diamonds

Attributes in ER Diagrams

Attributes are shown connected to the associated entity set by a line



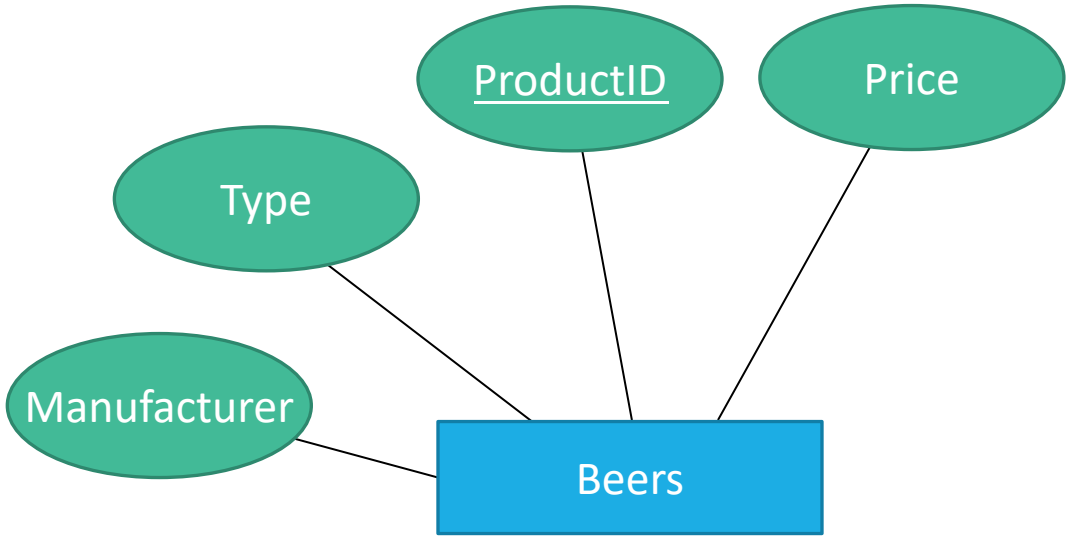
Each entity has values for each attribute, e.g.

Name	Manufacturer	Type	ABV
Old Rasputin	North Coast	Imperial Stout	9.0

Keys in E/R Diagrams

Every entity set must have a key.

A key for an entity set E is a set K of one or more attributes such that, given any two distinct entities e1 and e2 in E, e1 and e2 cannot have identical values for each of the attributes in the key K.



Which attribute set can serve as a key for all students in this class?

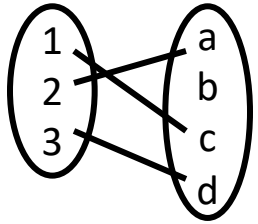
Relationships

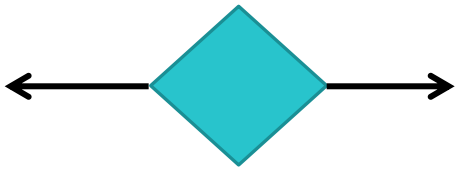
Relationships are connections among two or more entity sets.

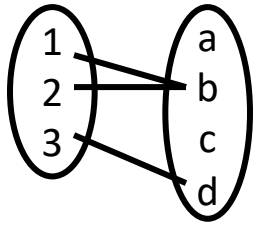
- one-to-one

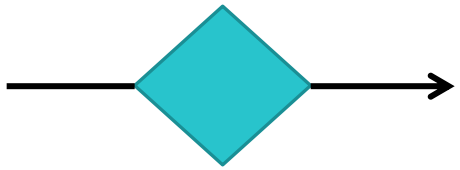
A special case of

- many-to-one

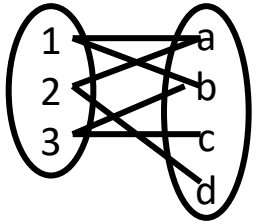


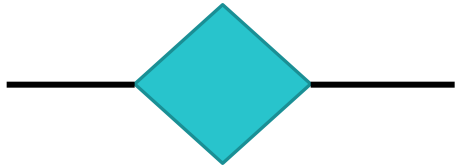






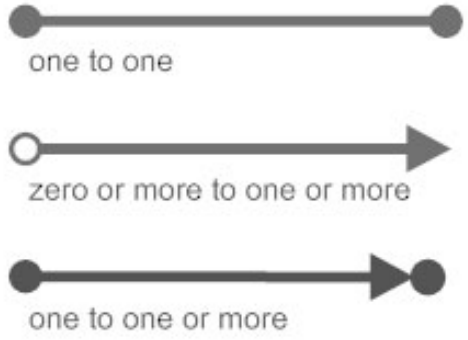
- many-to-many

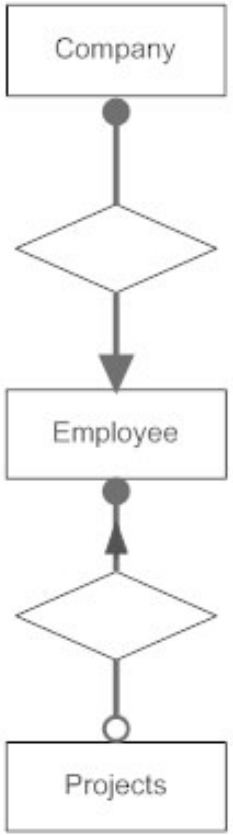




Relationship Styles

Bachman Style





Martin Style

1 - one, and only one (mandatory)

* - many (zero or more - optional)

1...* - one or more (mandatory)

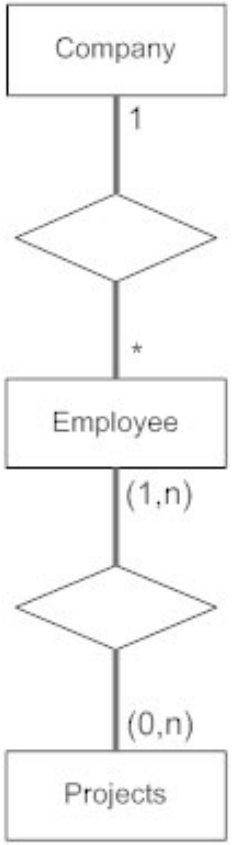
0...1 - zero or one (optional)

(0,1) - zero or one (optional)

(1,n) - one or more (mandatory)

(0,n) - zero or more (optional)

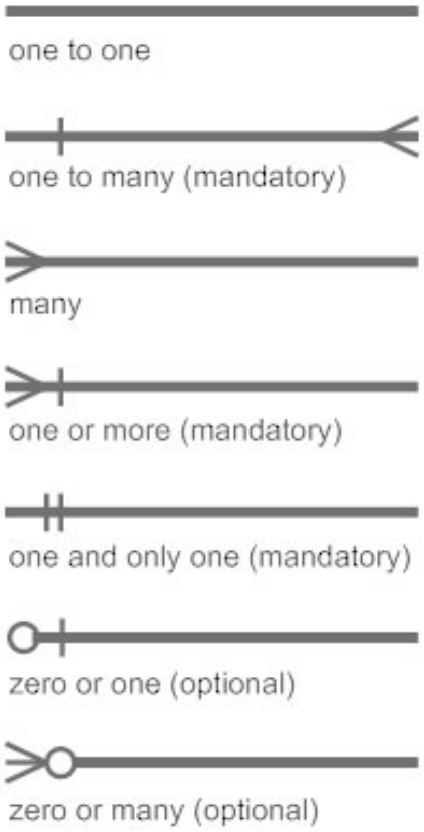
(1,1) - one and only one (mandatory)



https://www.smartdraw.com/entity-relationship-diagram/

Relationship Styles

Information Engineering Style



Chen Style

Ordinality - describes the minimum (optional vs mandatory)

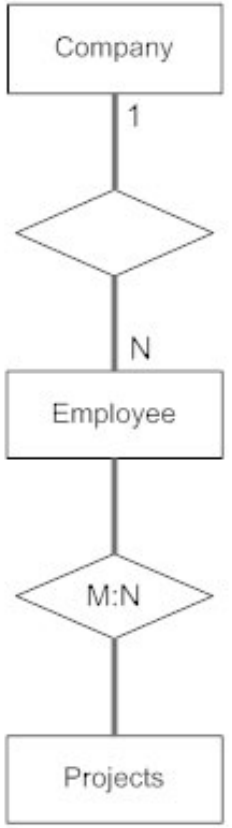
M:N

Cardinality - describes the maximum

1:N (n=0,1,2,3...)
one to zero or more

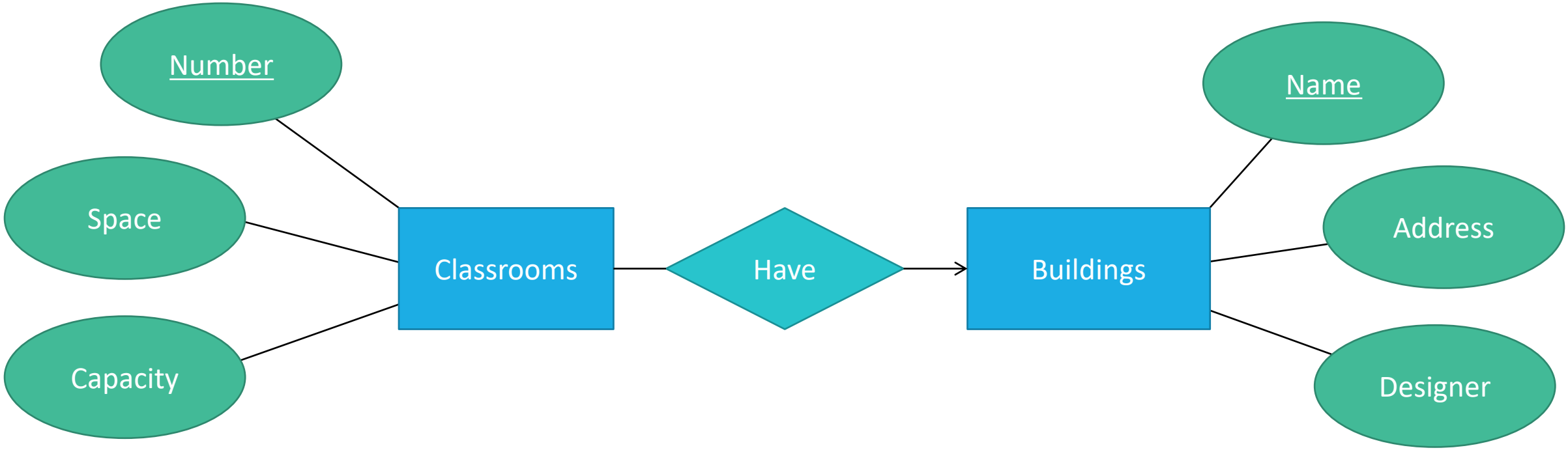
M:N (m and n=0,1,2,3...)
zero or more to zero or more
(many to many)

1:1
one to one



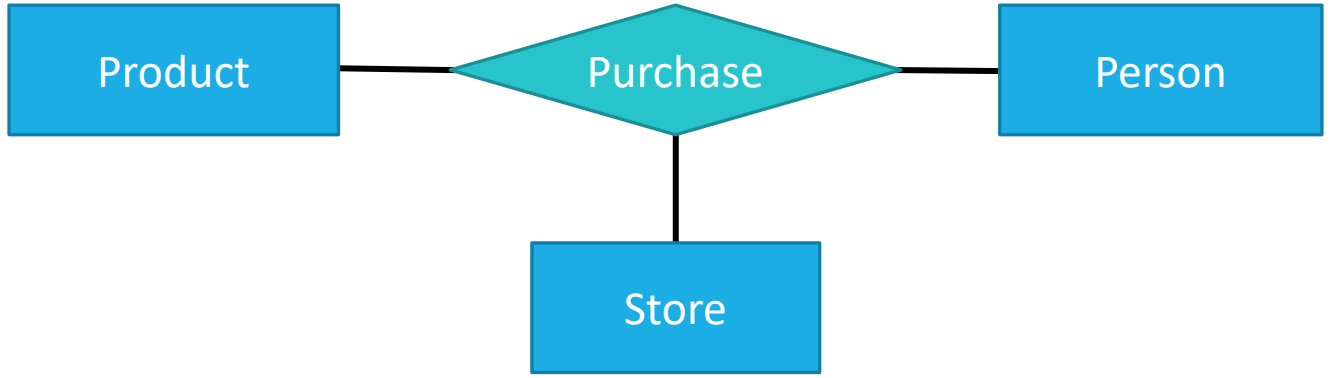
https://www.smartdraw.com/entity-relationship-diagram/

Relationships



Multi-Way Relationships

Model the purchase relationship between buyers, products, and stores:

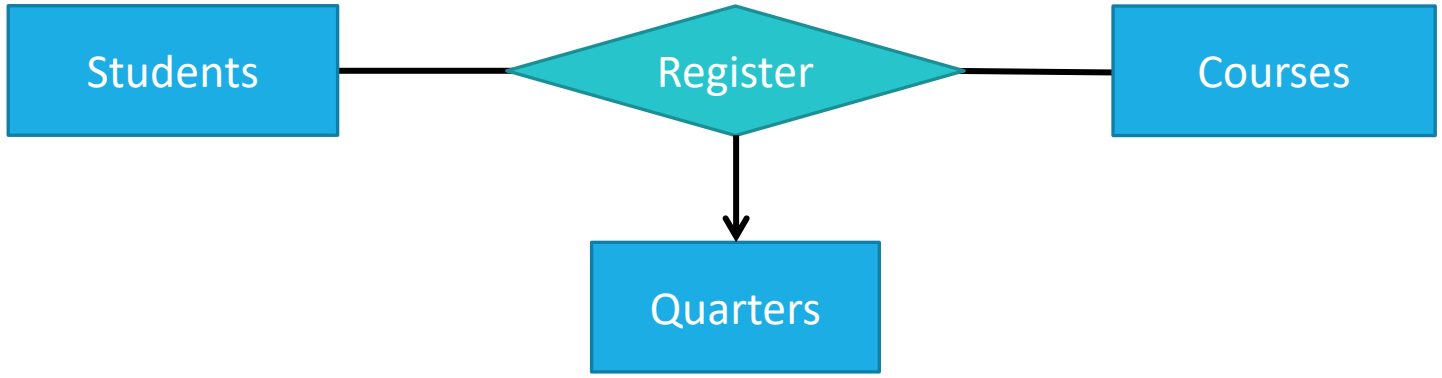


Three-way many-to-many relationship:

Requires knowledge of all three entities to make a unique combination

Multi-Way Relationships

What does the arrow mean?



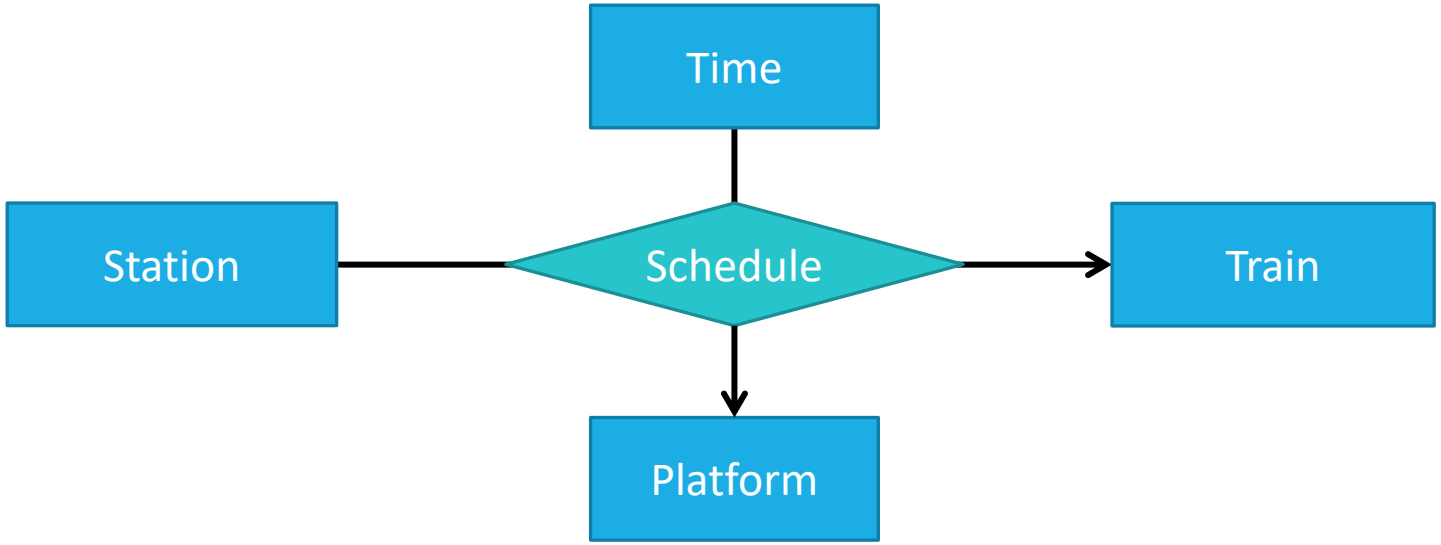
Three-way many-to-one relationship:

Any unique combination of **Student** and **Course** will be associated with only a unique **Quarter**

(assuming no retaking is allowed).

Multi-Way Relationships

4-way relationship, what does it mean?

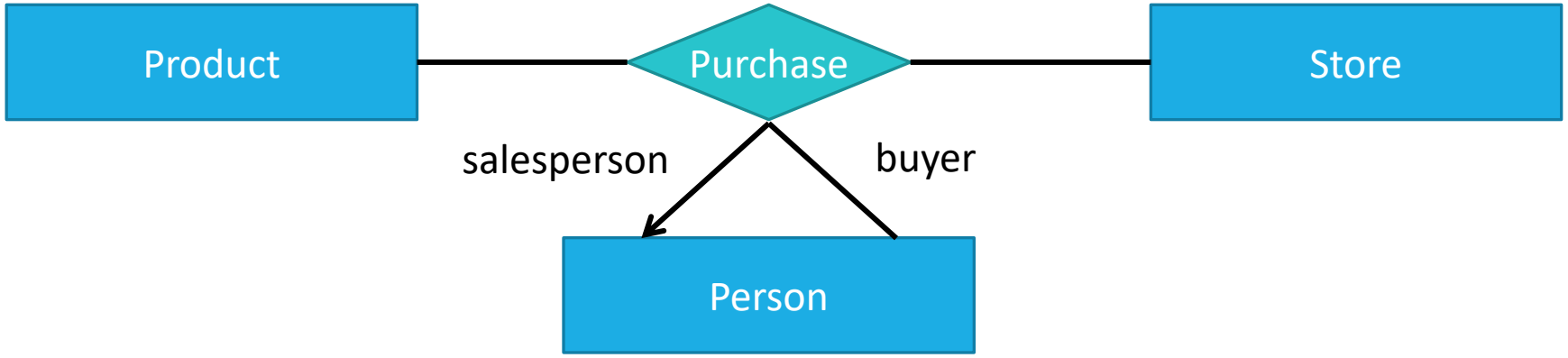


Station, **Time**, and **Train** determine **Platform** and

Station, **Time**, and **Platform** determine **Train**.

Roles in Relationships

What if we need an entity set twice in one relationship?



Recursive relationship: We draw as many lines from the entity set as the entity set appears in the relationship.

An E/R Diagram Example

A database will be created to manage music review data

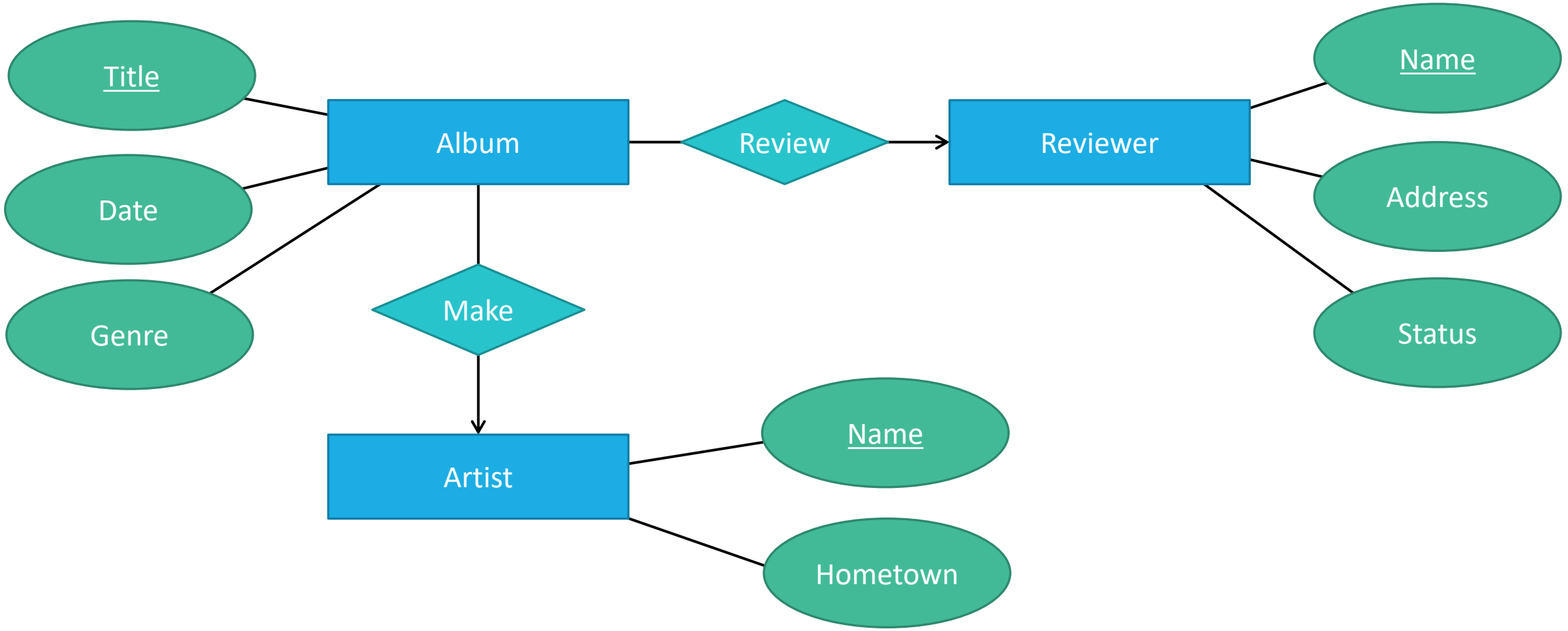
Store information such as:

- Albums: name, date, genre, etc.
- Reviewers: name, status, address, etc.
- Artists: name, type, hometown, etc.

Enforce rules including:

- Each album is made by a single artist, but an artist can make many albums.
- Each album can be reviewed by a single reviewer.

An E/R Diagram Example

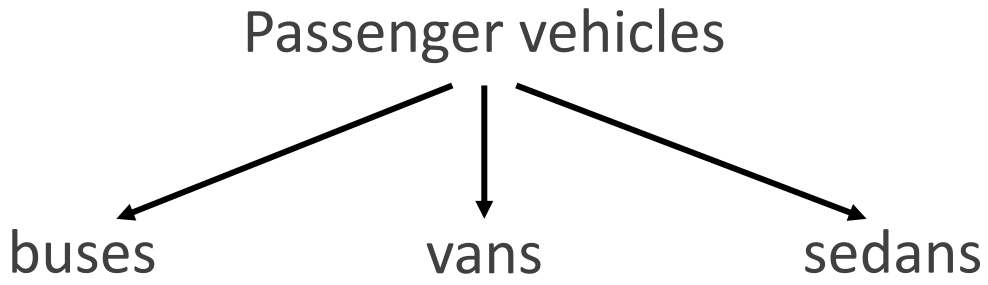


Subclasses in the E/R Model

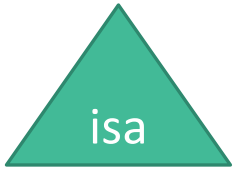
The world is inherently hierarchical. Some entities are special cases of others. So we need the idea of a subclass to improve storage efficiency in this case.

Example: varieties of passenger vehicles.

- A lot of common attributes shared by all types (e.g., Model, Make, Year).
- Additional Fields only associated with certain types (e.g., license class for busses).



Subclasses in the E/R Model

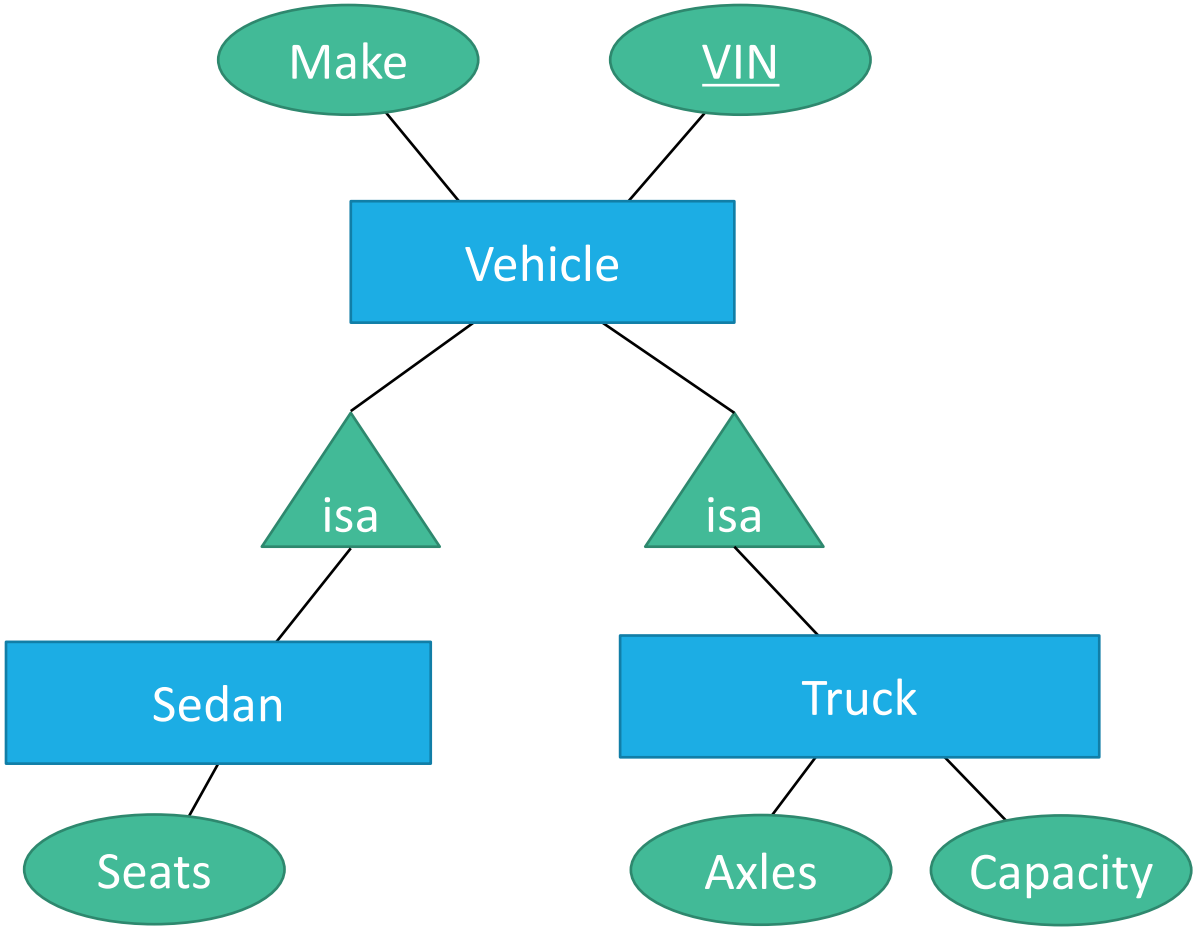
A subclass will be represented by 

The entity connected to the top point is the parent entity.

The entity connected to the bottom is the child entity.

The child entity inherits its parents attributes, including any keys, though it may have a key of its own.

Subclasses in the E/R Model



Subclasses in the E/R Model

Think in terms of records:

We use subclasses because certain types of information will not be relevant to all child classes

For example, here we want to store **Axles** and **Capacity** for **Trucks**, but this does not apply to **Sedans** and other **Vehicles**.

- Vehicle

Field 1	Field 2
VIN	Make

- Sedan

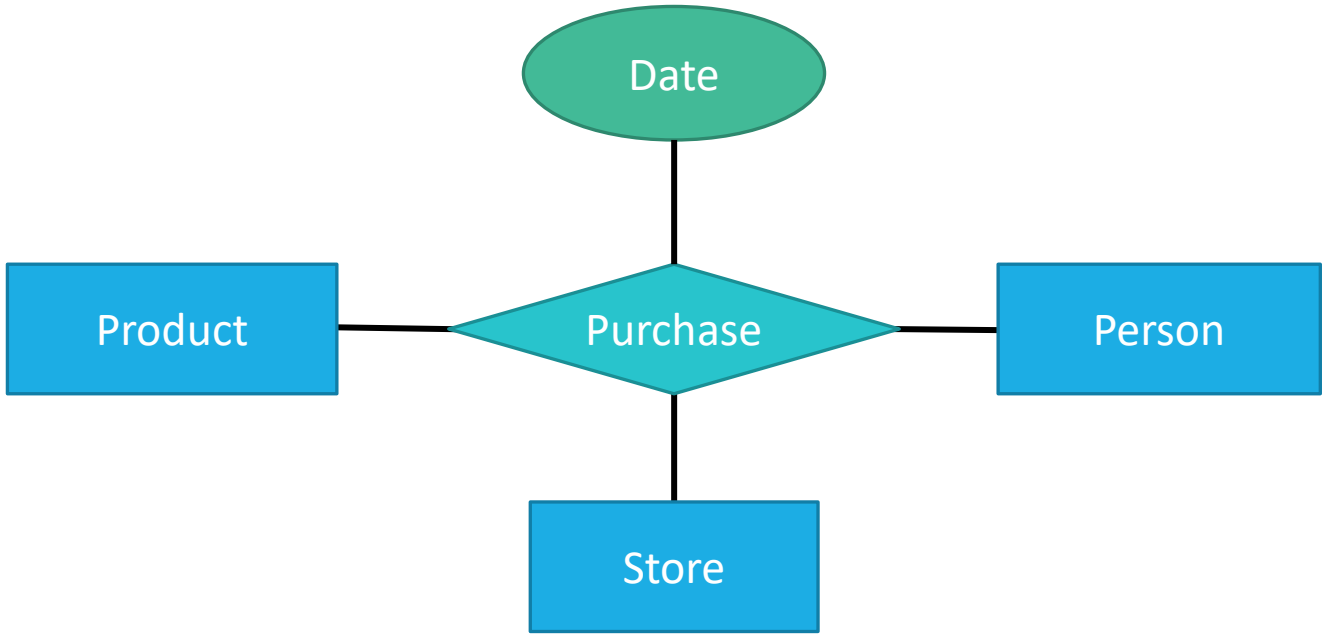
Field 1	Field 2	Field 3
VIN	Make	Seats

- Truck

Field 1	Field 2	Field 4	Field 5
VIN	Make	Axles	Capacity

Attributes on Relationships

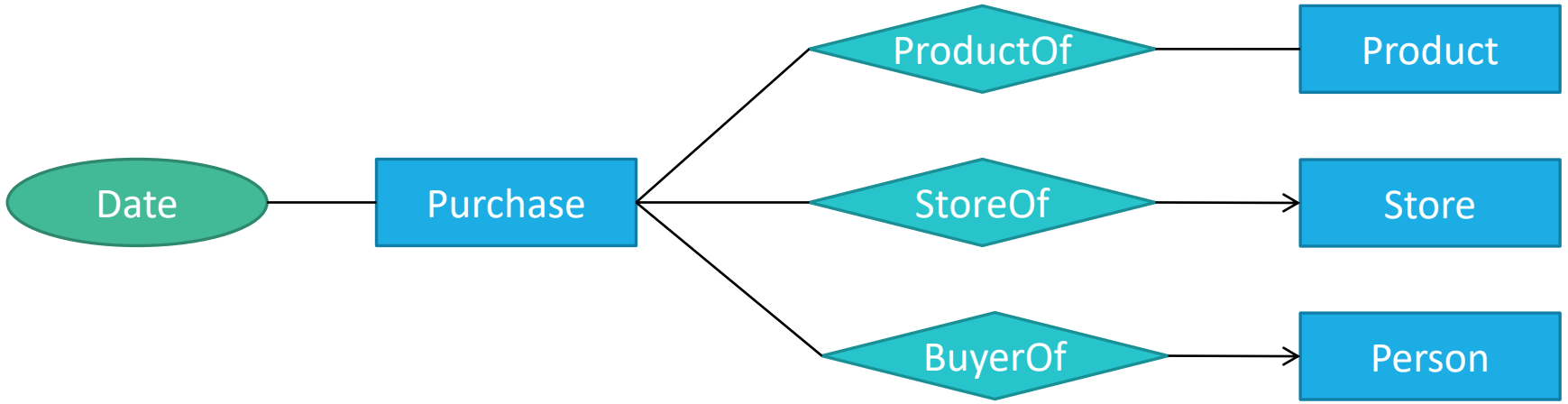
Date is only defined as a descriptor of the purchasing event.



Is this a good solution?

- Attributes on relationships can easily make a diagram confusing.
- Some data modeling methods (e.g., ODL) do not support multi-way relationships.

Converting Multi-Way Relationships to Binary

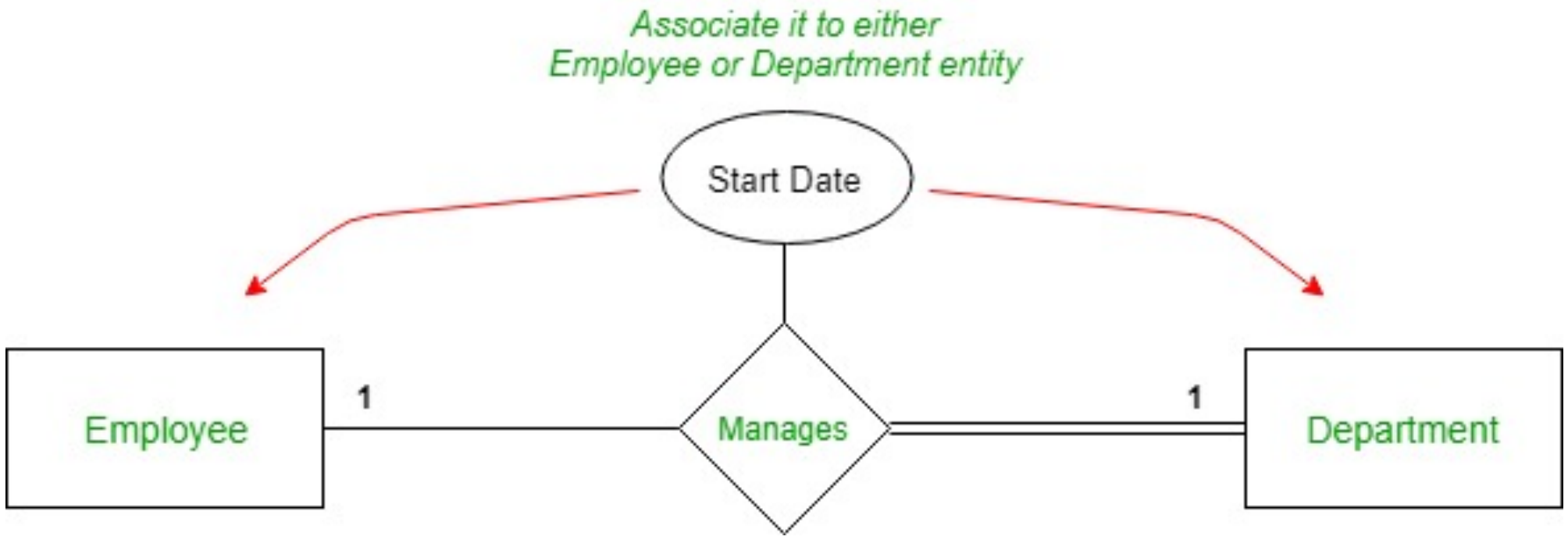


Steps

1. Create a new entity set, named similar to the relationship
2. Create relationships between the new entity set and those that existed in the multi-way relationship

Attributes on Relationships

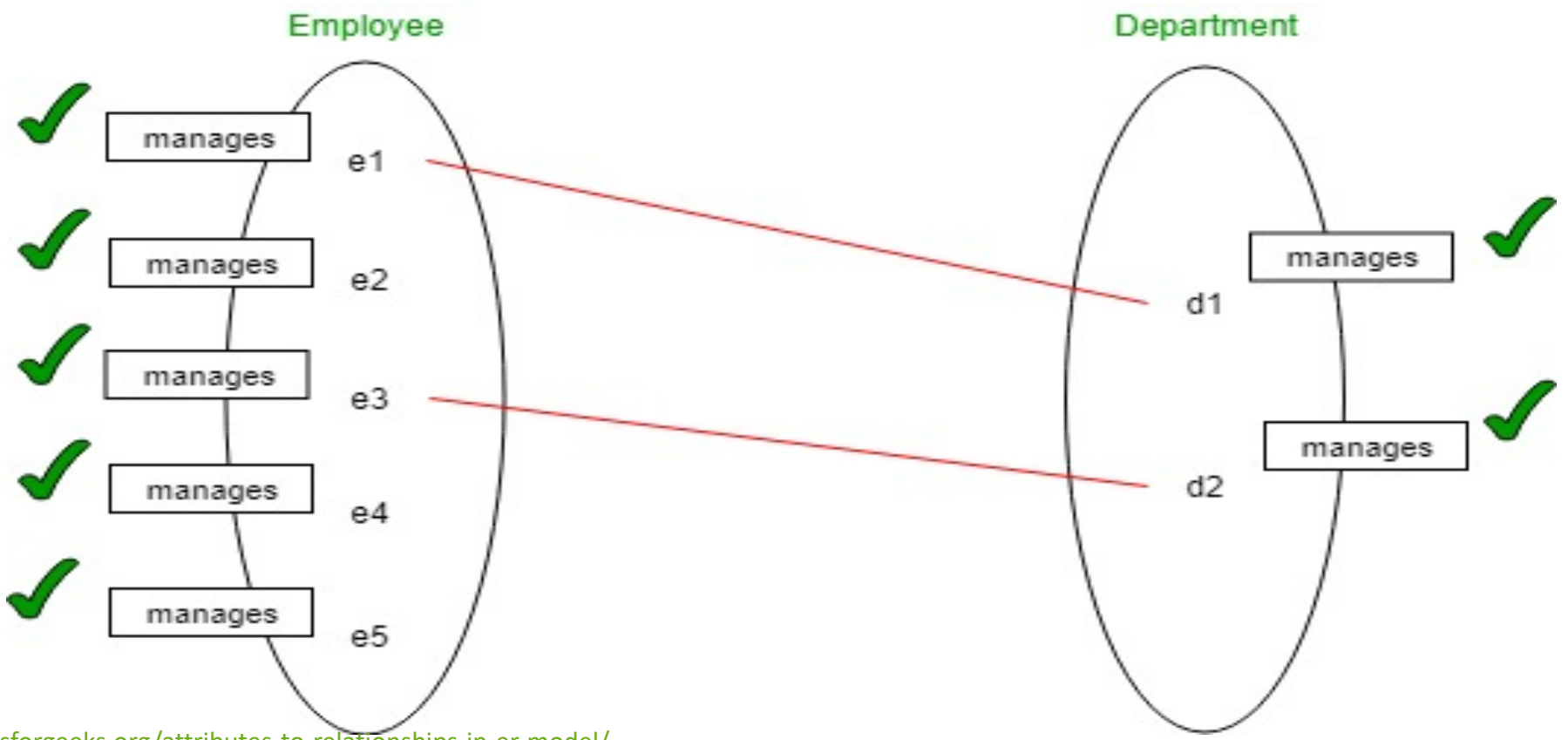
One to one relationship



https://www.geeksforgeeks.org/attributes-to-relationships-in-er-model/

Attributes on Relationships

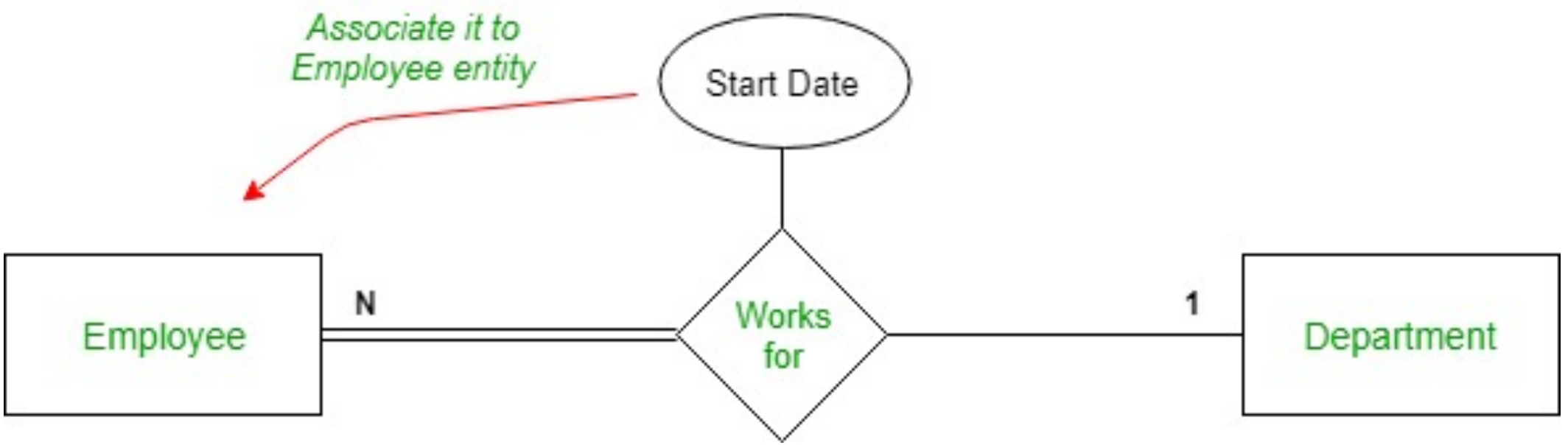
One to one relationship



https://www.geeksforgeeks.org/attributes-to-relationships-in-er-model/

Attributes on Relationships

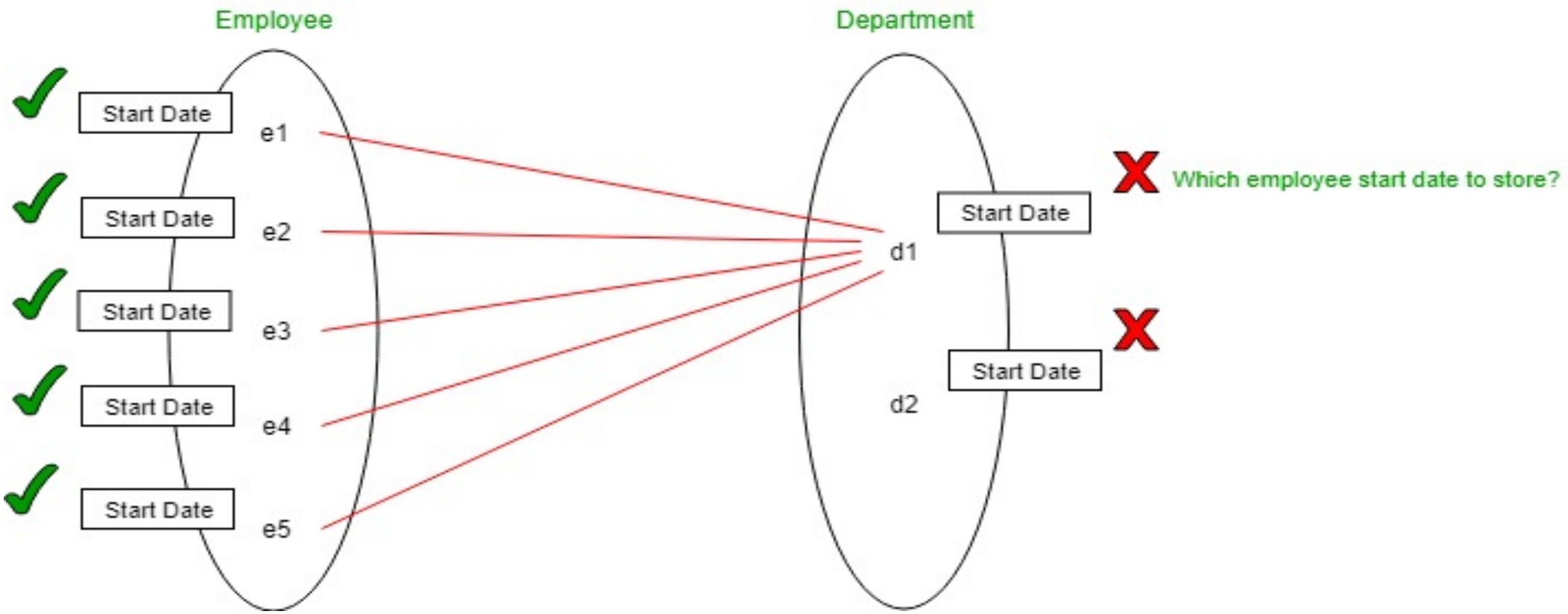
One to many relationship



https://www.geeksforgeeks.org/attributes-to-relationships-in-er-model/

Attributes on Relationships

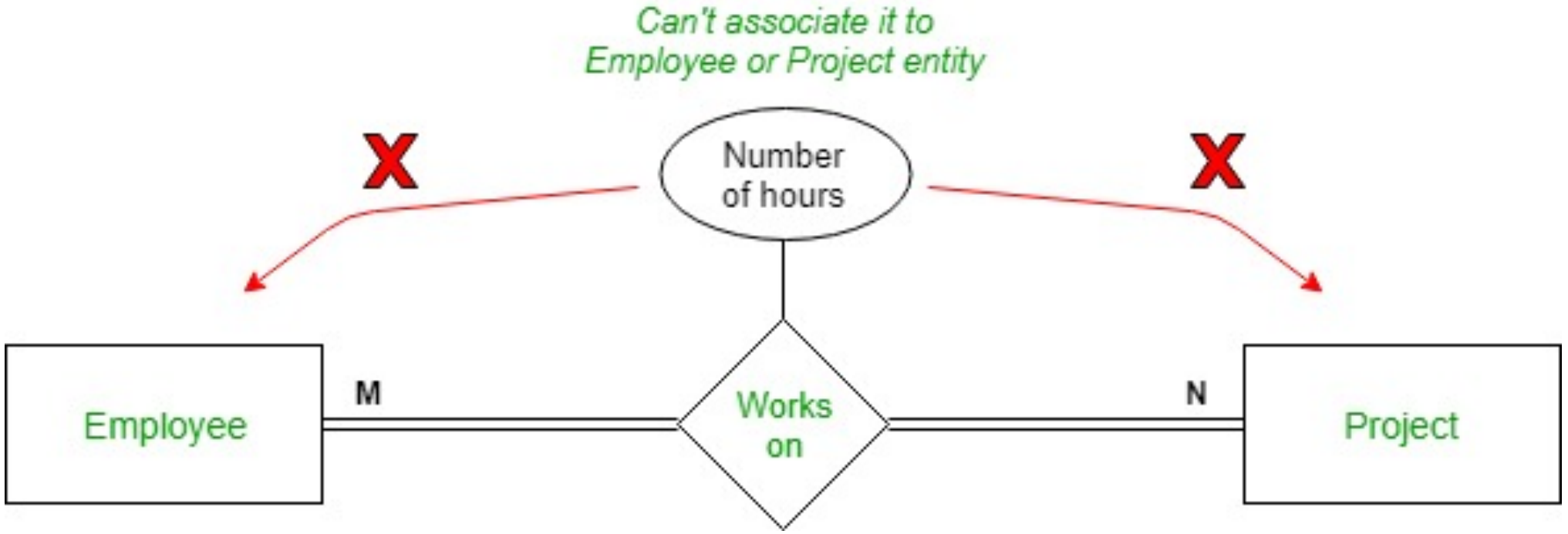
One to many relationship



https://www.geeksforgeeks.org/attributes-to-relationships-in-er-model/

Attributes on Relationships

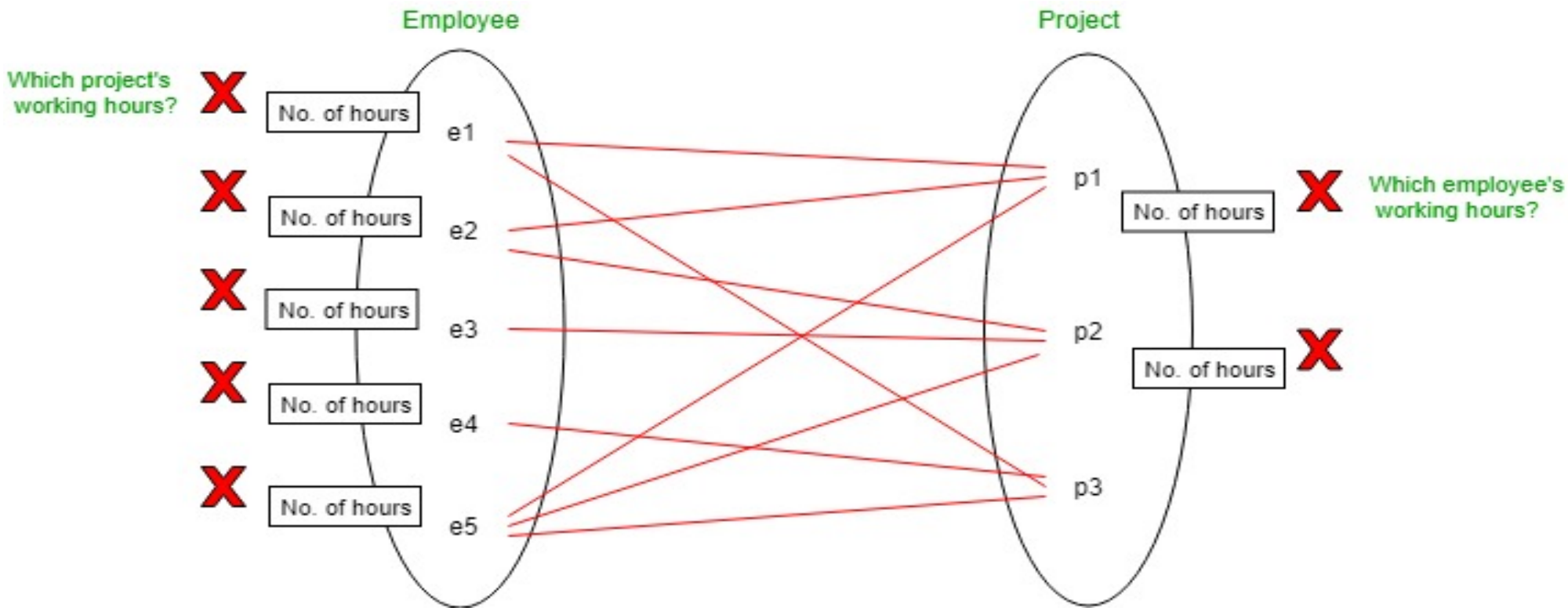
Many to many relationship



https://www.geeksforgeeks.org/attributes-to-relationships-in-er-model/

Attributes on Relationships

Many to many relationship



https://www.geeksforgeeks.org/attributes-to-relationships-in-er-model/

Design Principles

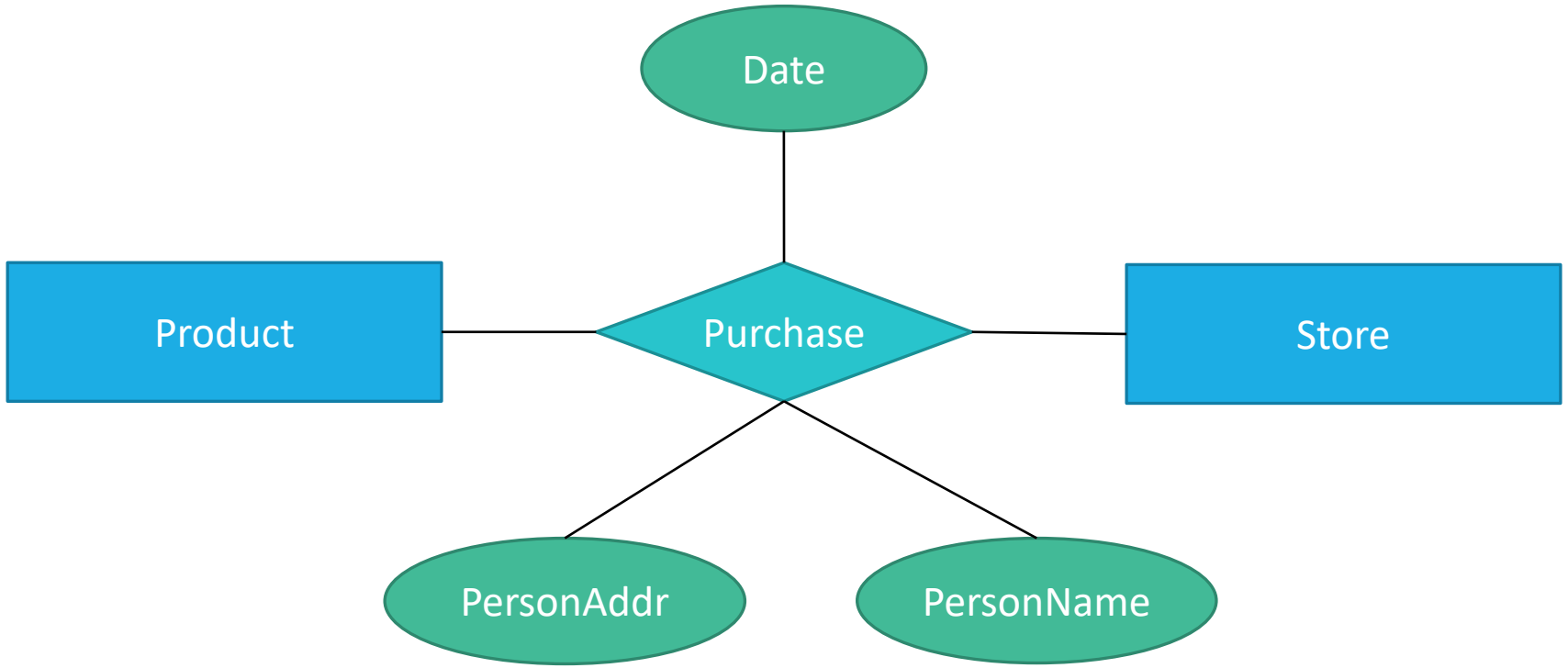
1. Faithful! Entity sets and their attributes should reflect reality.

2. Avoid redundancy. We should be careful to say everything once only.

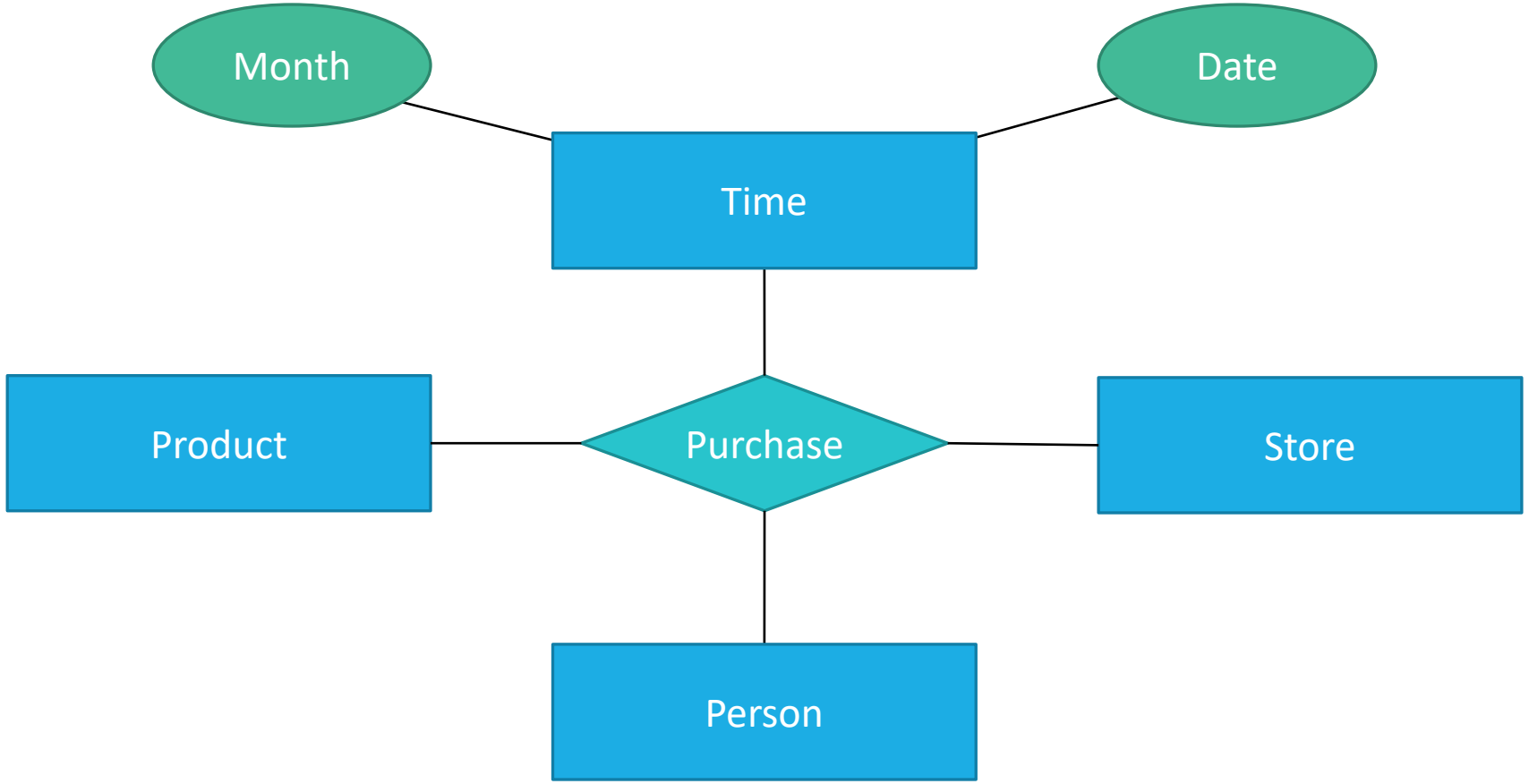
3. Choose the right relationships. Unnecessary relationships, though they may be true, can cause redundancy and complexity.

4. Pick the right kind of element. Is it an entity set, relationship, or attribute?

Design Principles: What's Wrong?



Design Principles: What's Wrong?



Example

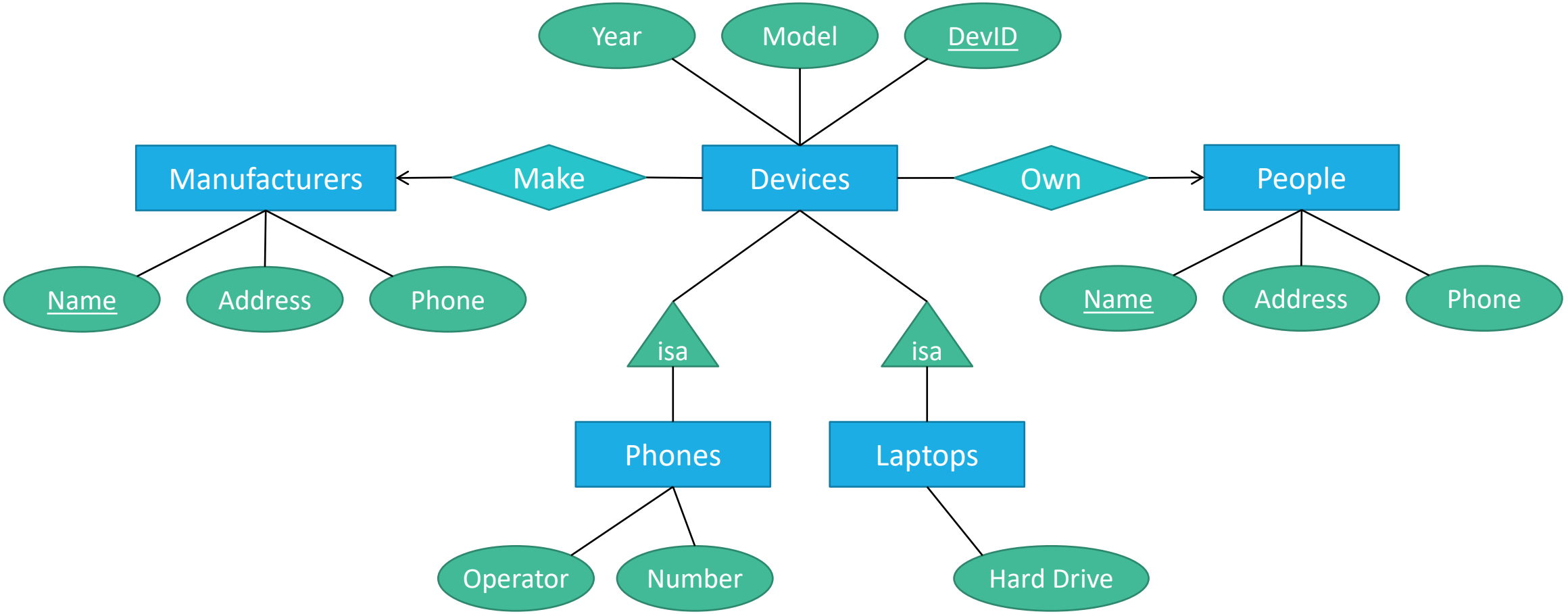
Let us say I want to make a database of all the electronic devices owned by the people in this room, including:

- Device descriptions
- Information about manufacturers and owners
- Other information?

Sketch your database design using the E/R diagram:

- Include Entities, Attributes and Relationships.
- Each entity should have a key.
- Make use of multi-way relationships and subclasses only if necessary

Example


Year
Model
DevID
Manufacturers
Make
Devices
Own
People
Name
Address
Phone
Name
Address
Phone
isa
isa
Phones
Laptops
Operator
Number
Hard Drive

Constraints in E/R Diagrams

Some commonly used constraints.

Key constraints:

- Attributes or sets of attributes must have unique and not null values in the relational table.
- Example: one course must have a unique SLN.

Single-value constraints:

- Each attribute of an entity set has a single value.
- For example, a person can have only one birthdate.

Constraints in E/R Diagrams

Is a key for an entity E single-value constrained?

- Definitely. Keys are a major source of single-value constraints.

Single value implied in many to one relationship:

How is single value constraint different from key?

- Non-key single value attributes can take a null value (keys cannot be null)

Constraints in E/R Diagrams

Domain constraints:

- Value of each attribute X must be taken from the domain associated with that attribute
- Example: a person's name cannot be a number.

General constraints:

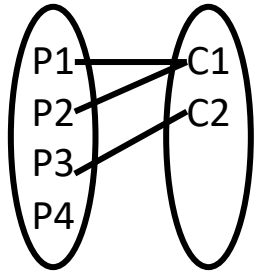
- Other arbitrary constraints that should hold in the database, such as Check constraints and Range constraints.
- Example: state names, age of a person.

Referential integrity constraints:

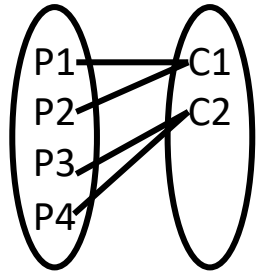
- Require that a value referred to by some object actually exists in the database.
- For example, if you take a course at UW, your course must exist in the UW course database.

Referential Integrity Constraints

Is the referential integrity constraint enforced in the following E/R diagram?



No. Why? How about the following diagram?



This one does assert that exactly one company exists as the maker of a product.

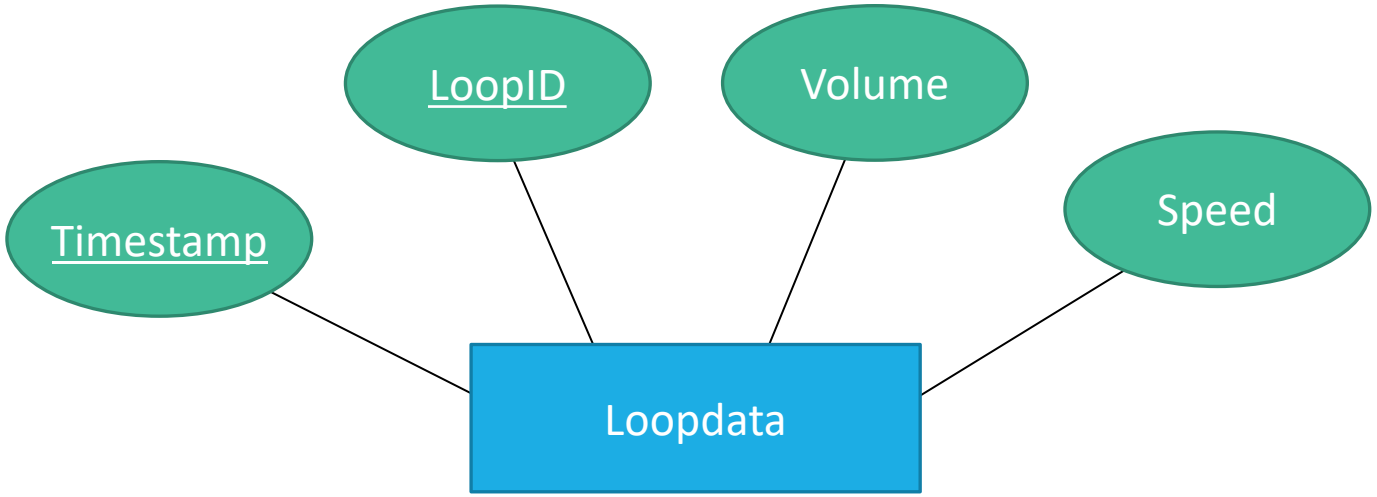
Referential Integrity Constraints

What are the possible results of referential integrity constraints in database?

- Prohibit deletion of an entity required by another table.
  - Cannot delete a course if there are students who have registered for that course.
- Cascade update/delete:
  - If I delete a company from the companies table, all products associated with that company will also be deleted.

Depends on your choice when creating the relationship.

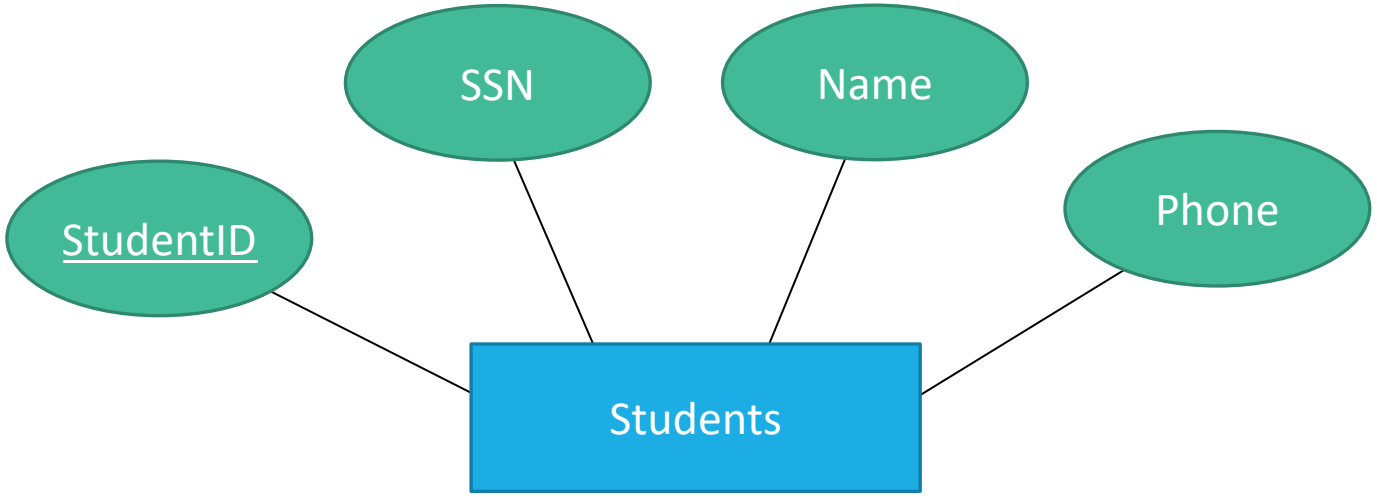
Keys in E/R Diagrams



The combination of **Timestamp** and **LoopID** is required to uniquely identify a single observation

How to represent: underline all attributes that are required to form the key

Keys in E/R Diagrams



Both **StudentID** and **SSN** can serve as keys.

How do we specify multiple keys in E/R diagrams?

No formal way! We have to choose one.

Weak Entity Sets

Entity sets are weak when their key comes from other entities to which they are related.

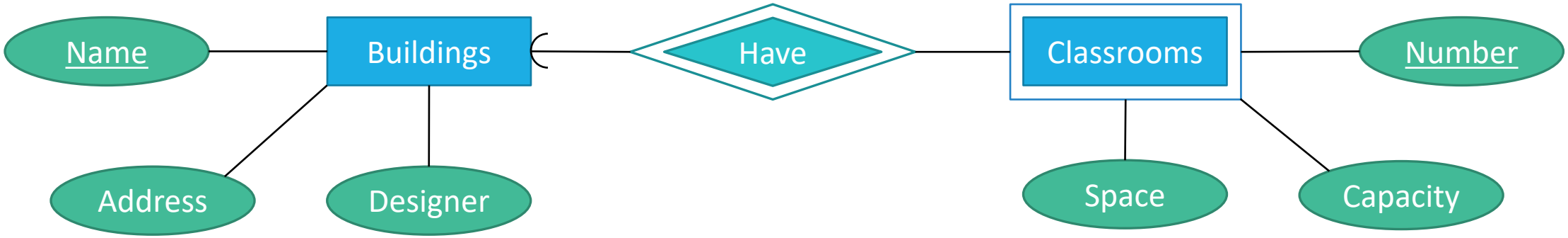
Causes of weak entity sets:

- Entity sets fall into a hierarchy based on classifications unrelated to the subclass (e.g., buildings and classrooms)
- Converting multi-way relationships to binary ones (e.g., invoice of a purchase)

In the E/R Diagram

- Weak entity sets are shown in double border rectangles.
- A weak entity's supporting relationships will be shown as diamonds with a double border.

Weak Entity Sets



If E is a weak entity set, then its key consists of:

- Zero or more of its own attributes, and
- Key attributes from entity sets that are reached by certain many-one relationships from E to other entity sets.

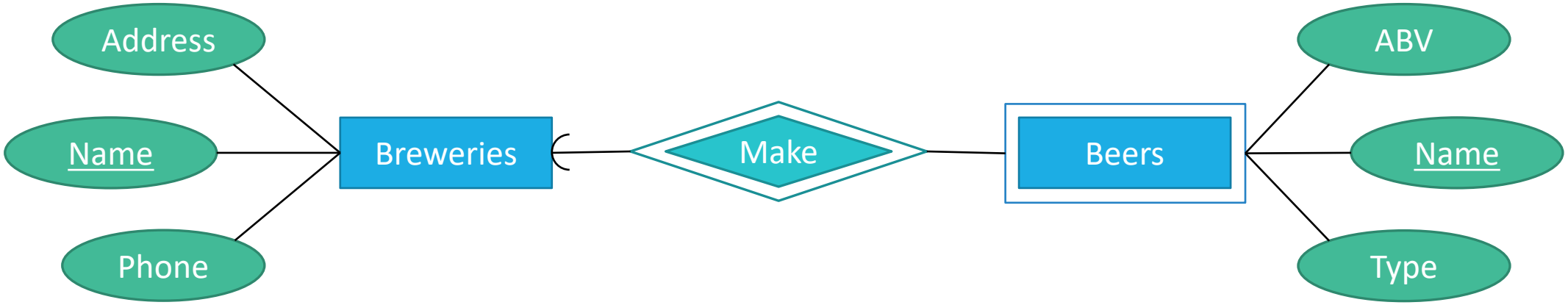
Weak entity sets usually imply many-to-one relationships (including one-to-one relationships as special cases), and referential integrity.

Weak Entity Sets

What is a supporting many-one relationship R from E to F?

- R must be a binary, many-one relationship from E to F.
- R must have referential integrity from E to F.
- The attributes that F supplies for the key of E must be key attributes of F.
- If F is itself weak, then some or all of the key attributes of F supplied to E will be key attributes of one or more entity sets G to which F is connected by a supporting relationship.
- If there are several different supporting relationships from E to F, then each relationship is used to supply a copy of the key attributes of F to help form the key of E

Weak Entity Sets



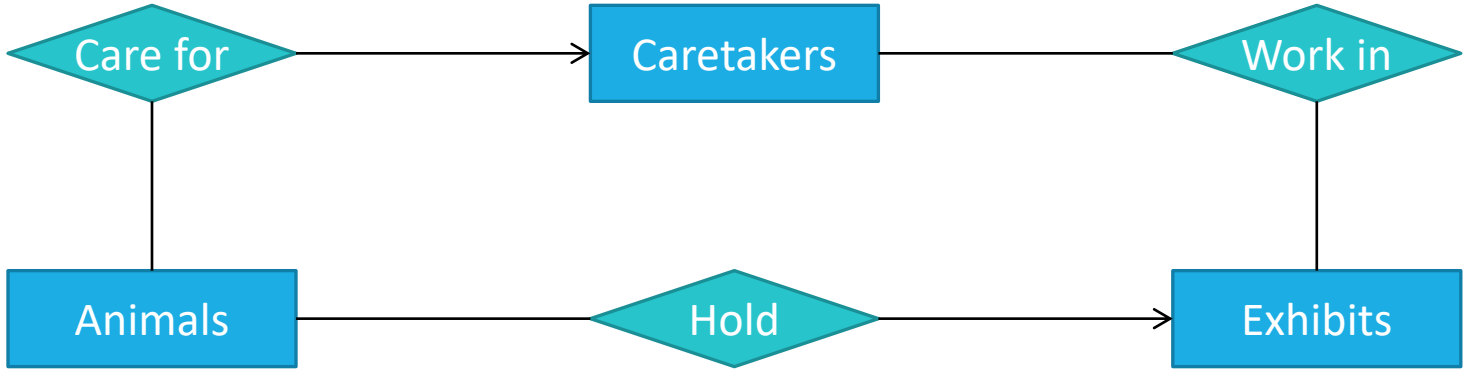
Important: Do not overuse weak entity sets.

In many cases it is sufficient to introduce an attribute to act as key (e.g., create some ProductID for **Beers**)

ER Diagram Review

- Every entity set should have a key.
- If there are multiple candidate keys, choose one.
- Don’t add unnecessary relationships.

Which relationship is unnecessary?



ER Diagram Review

- Don’t include unnecessary entity sets.

- Direction of arrow in many to one relationship
- Double rectangles and diamonds designate weak entity sets
- Correctly assign attributes, don’t just assign attributes when you should add an entity set.
- Referential integrity is a pretty strict constraint – use wisely
- Don’t over complicate!

ER Diagram Review

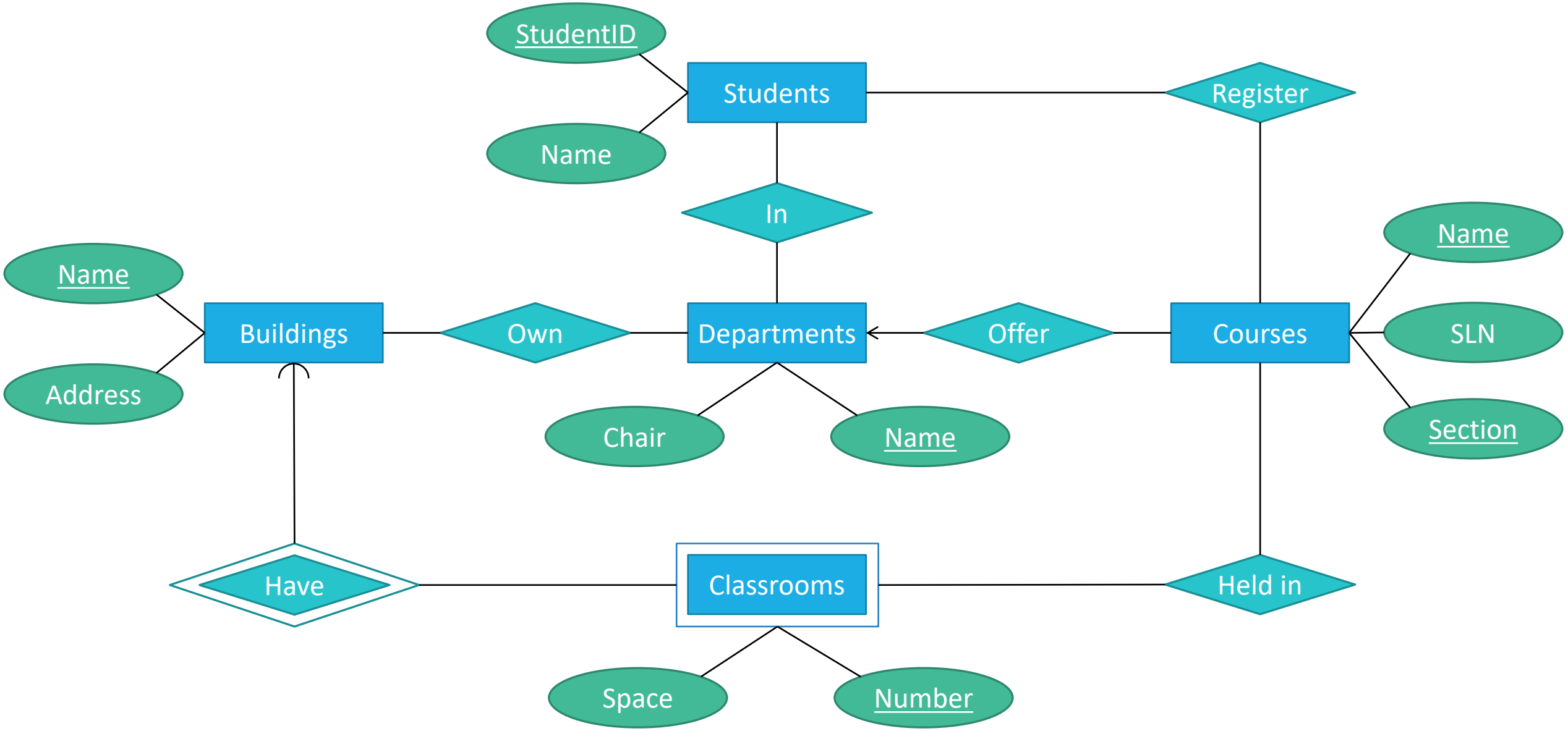
Draw an ER diagram for a database that represent the course offering and registration system in UW.

At minimum, we want to represent the following:

- Students: StudentID, name, etc.
- Courses: SLN, name, section, etc.
- Departments: name, chair, etc.
- Buildings: name, address, etc.
- Classrooms: number, space, etc.
- Other information you are interested to save.

Assumptions you need to consider:

- Classrooms can only be uniquely identified by the combination of building name and classroom number.
- One course can only be offered by a unique department.
- Other assumptions in the UW system.


StudentID
Students
Register
Name
In
Name
Buildings
Own
Departments
Offer
Courses
Name
SLN
Address
Chair
Name
Section
Have
Classrooms
Held in
Space
Number

The Relational Data Model

Announcement

Assignment 2 is released

Find your group member

- Canvas → Discussion
- Piazza

Late homework

- Inform the instructor or TA in advance

Instance vs. Schema

Instance:

- Particular data contained in a database.

Database schema is quite different from an instance of the resulting database.

- A Schema is stable with over time – attributes almost never change.
- A DB instance changes continuously as data is added, removed, and updated.
- For database design, only the schema is important, although knowledge of typical instances helps guide our design.

From E/R Diagrams to Relational Schema

Processes of creating a new database.

- Design phase – considering information we need to store, relationships and constraints.
- Implementation phase – building the database in a real DBMS.

Most commercial DBMS use the relational model, we need to make our design consistent with this model too.

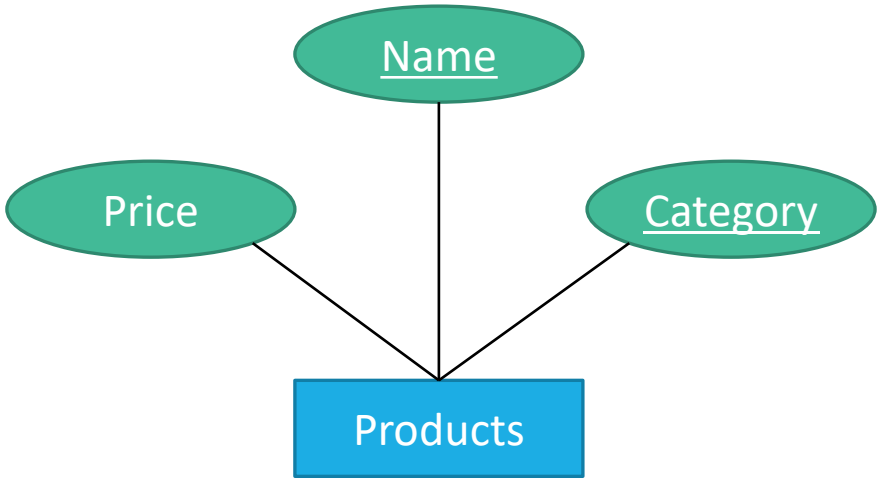
This involves converting an E/R design to a relational database, which is also very straightforward.

- Turn each entity set into a relation with the same set of attributes.
- Replace a relationship by a relation whose attributes are the keys for the connected entity sets.

From Entity Sets to Relations

First consider strong entity sets.

Create a relation of the same name and with the same set of attributes



Products(Name, Category, Price)

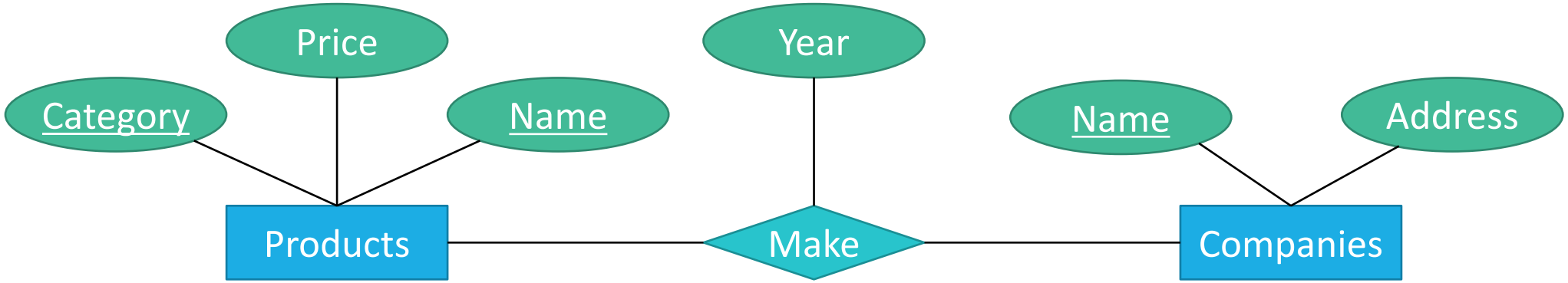
Name	Category	Price
VX720	Monitor	$199.99

From Relationships to Relations

When E has a many-many relationship R to F, relationship R can be converted to a relation with the following attributes:

- The key attributes of E
- The key attributes of F
- Any attributes belonging to relationship R

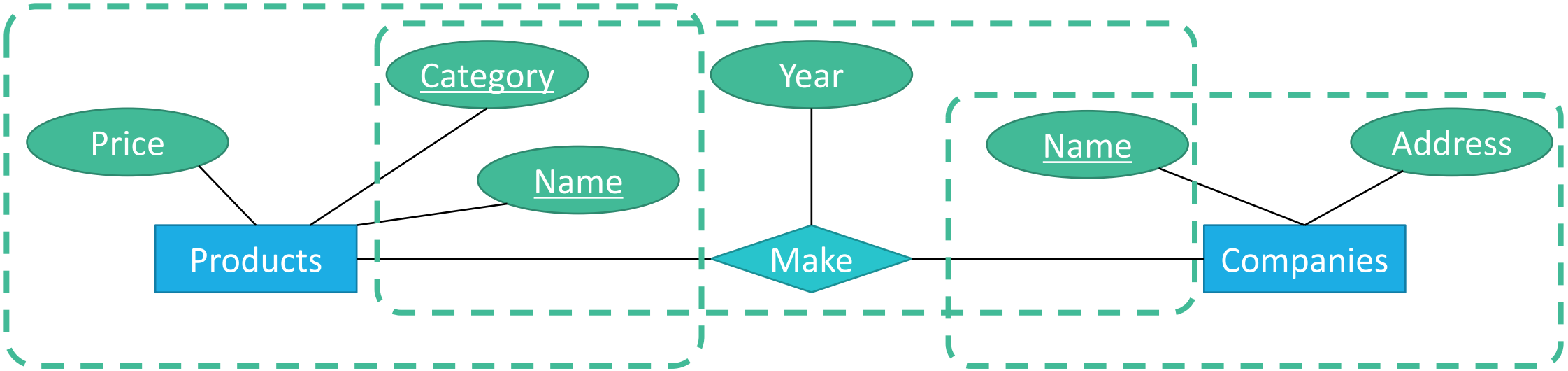
From Relationships to Relations



Make(Products.Name, Products.Category, Companies.Name, Year)

Products.Name	Products.Category	Companies.Name	Year
VX720	Monitor	Gateway	1999

From Relationships to Relations



Companies(Name, Address)

Products(Name, Category, Price)

Make(Products.Name, Products.Category, Companies.Name, Year)

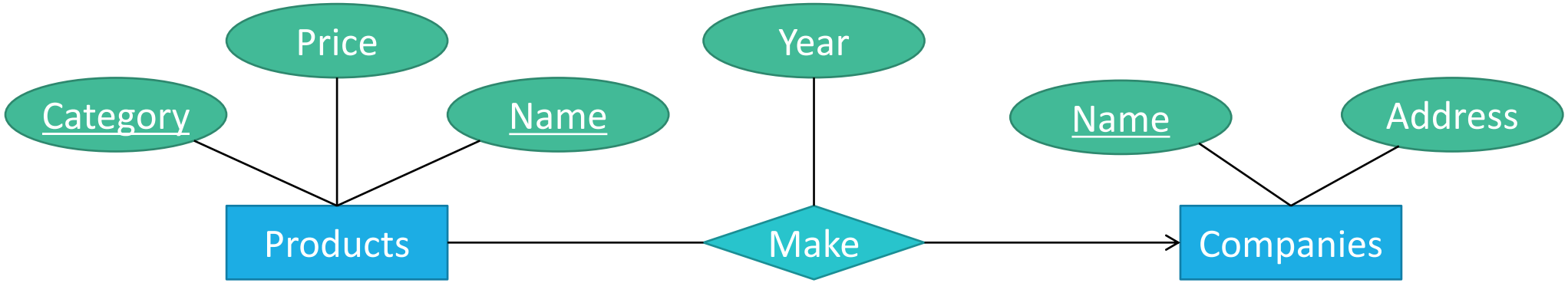
Combining Relations

How to deal with many-to-one relationships?

When E has a many-to-one relationship R to F, R and E need to be combined into one relation with a schema consisting of:

- All attributes of E
- The key attributes of F
- Any attributes belonging to R

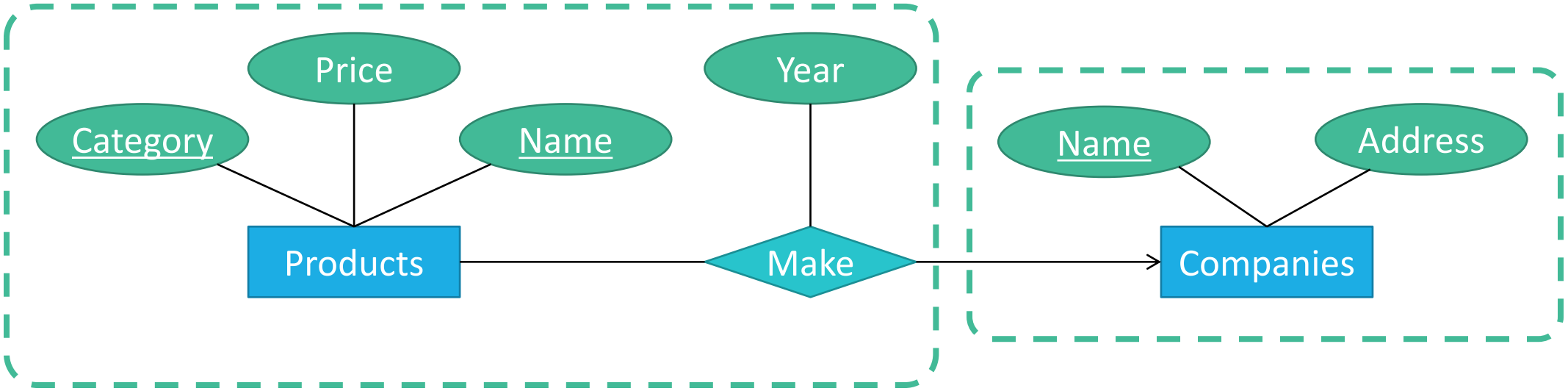
Combining Relations



How many relations do we need?

Two! No need for **Make**. **Products** can be modified to include **Year** and **Companies.Name**. **Companies** should be the other relation.

Combining Relations



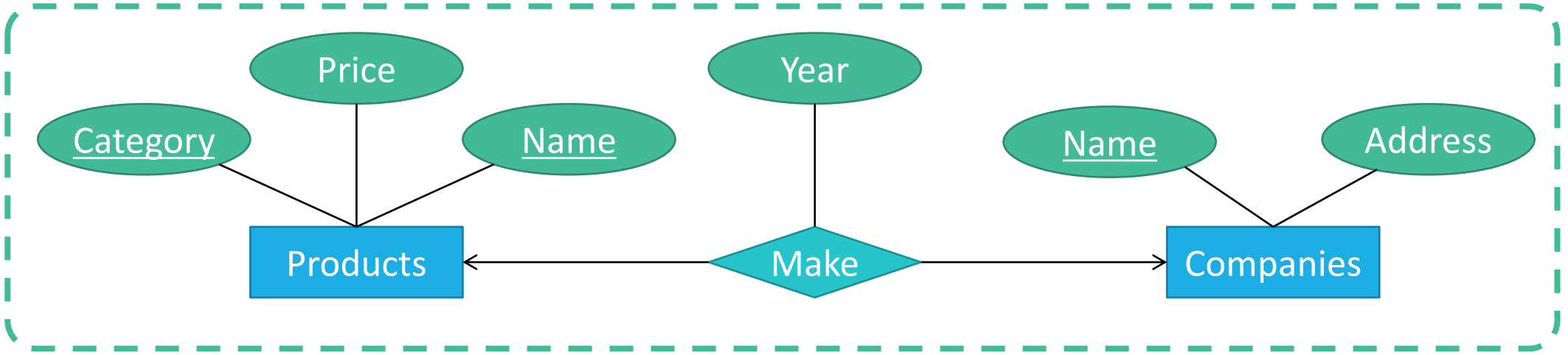
Companies(Name, Address)

Products(Name, Category, Price, Companies.Name, Make.Year)

Combining Relations

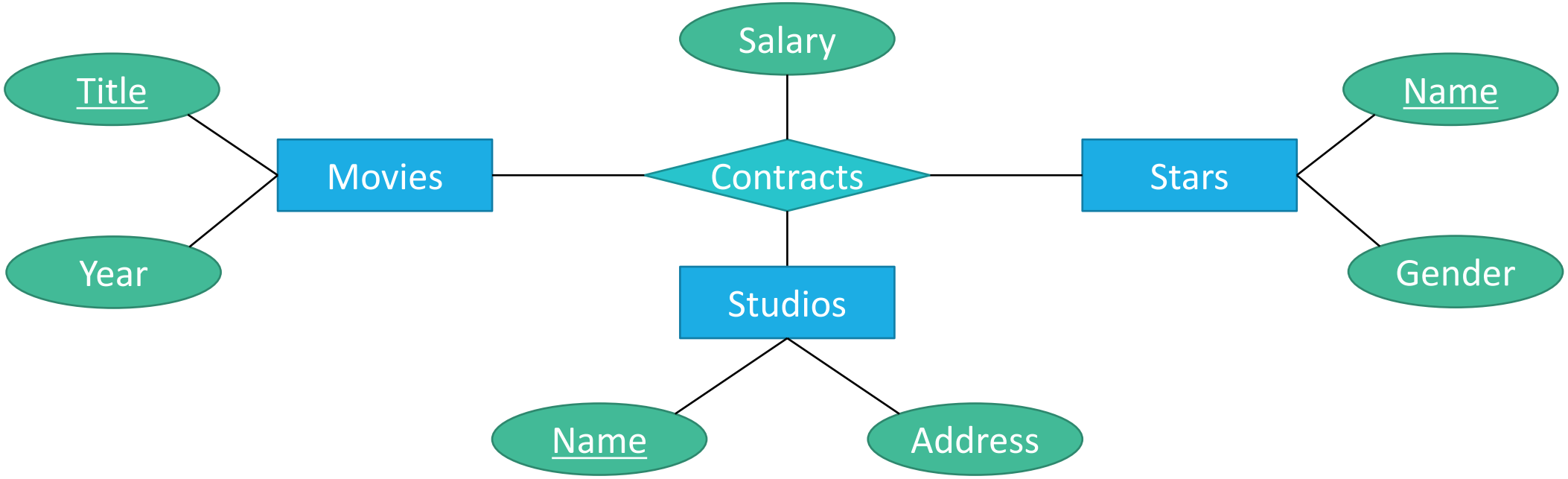
One-to-one relationships

- The entire relationship with both entity sets becomes one big relation with all attributes
- The key can be taken from either entity which we choose as the primary entity.



CompaniesAndProducts(Companies.Name, Companies.Address, Products.Name, Products.Category, Products.Price, Make.Year)

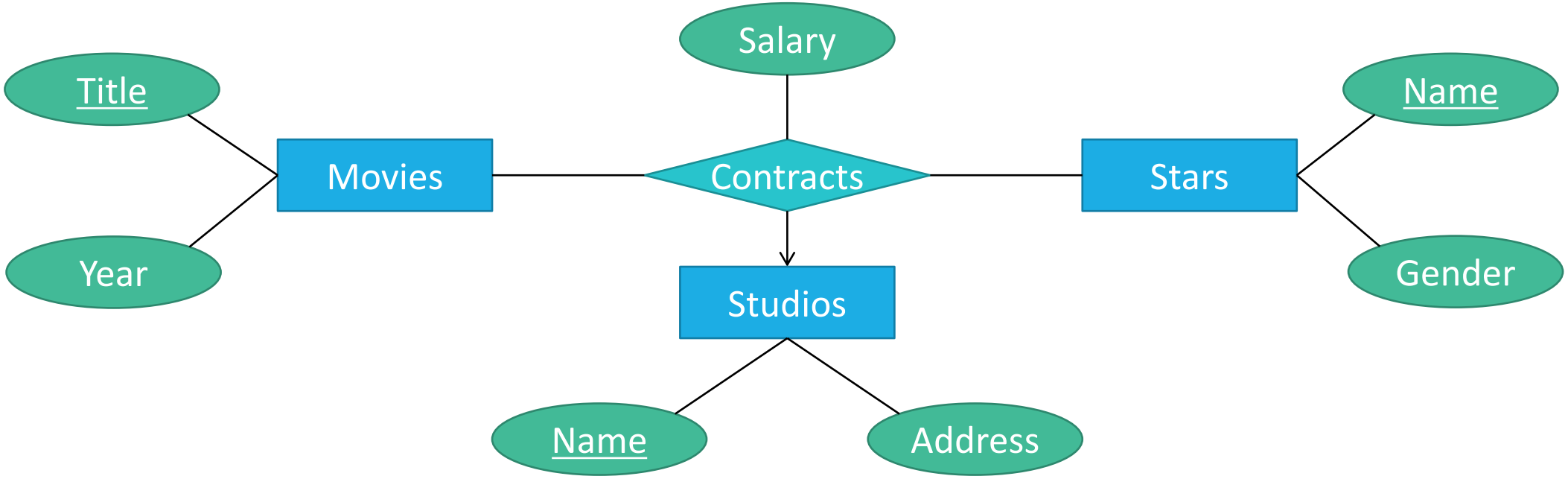
From Relationships to Relations



All entity sets become relations, as does the relationship **Contracts**:

Contracts(Movies.Title, Stars.Name, Studios.Name, Salary)

From Relationships to Relations

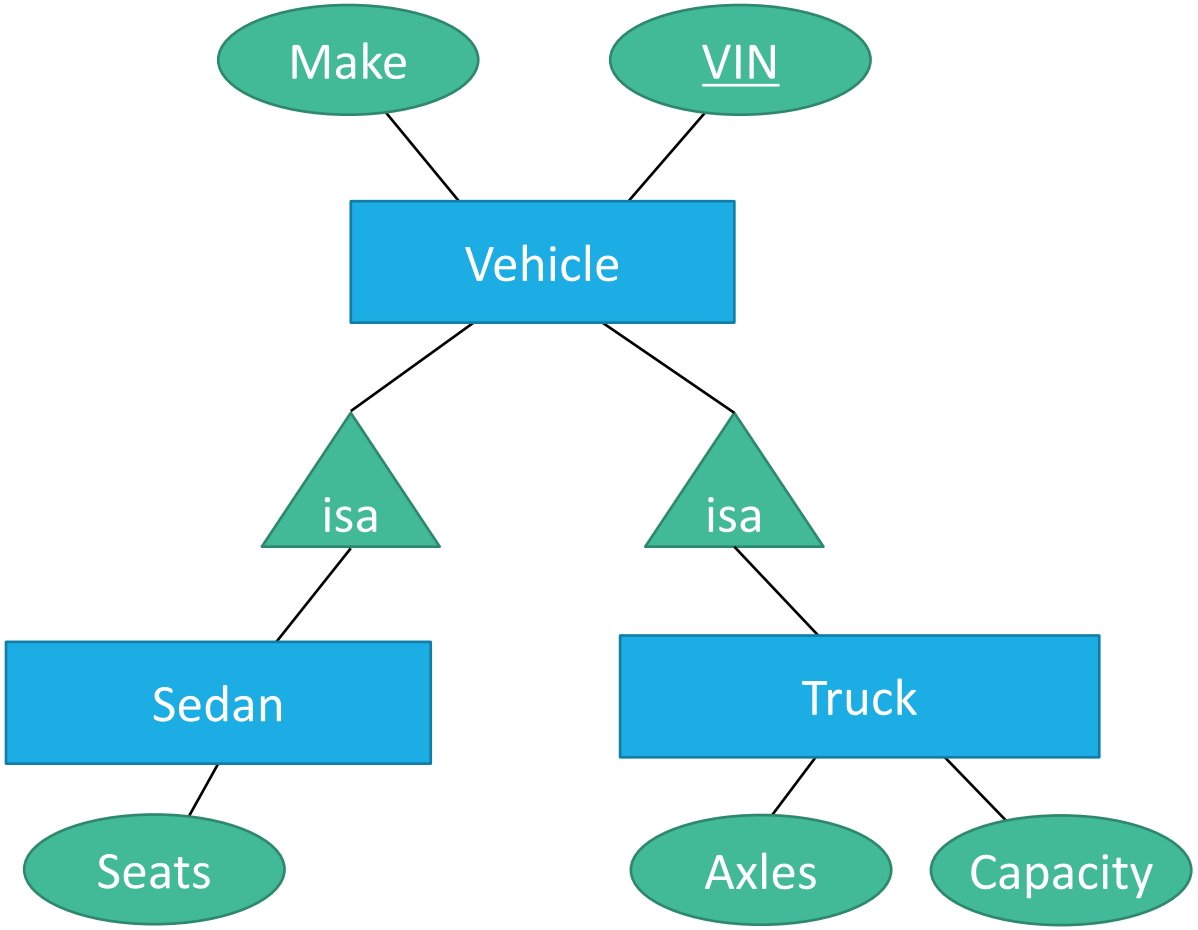


Three way relationship with many on two sides but one on one side.

Contracts(Movies.Title, Stars.Name, Studios.Name, Salary)

What's the difference with the previous slide?

From Subclasses to Relations



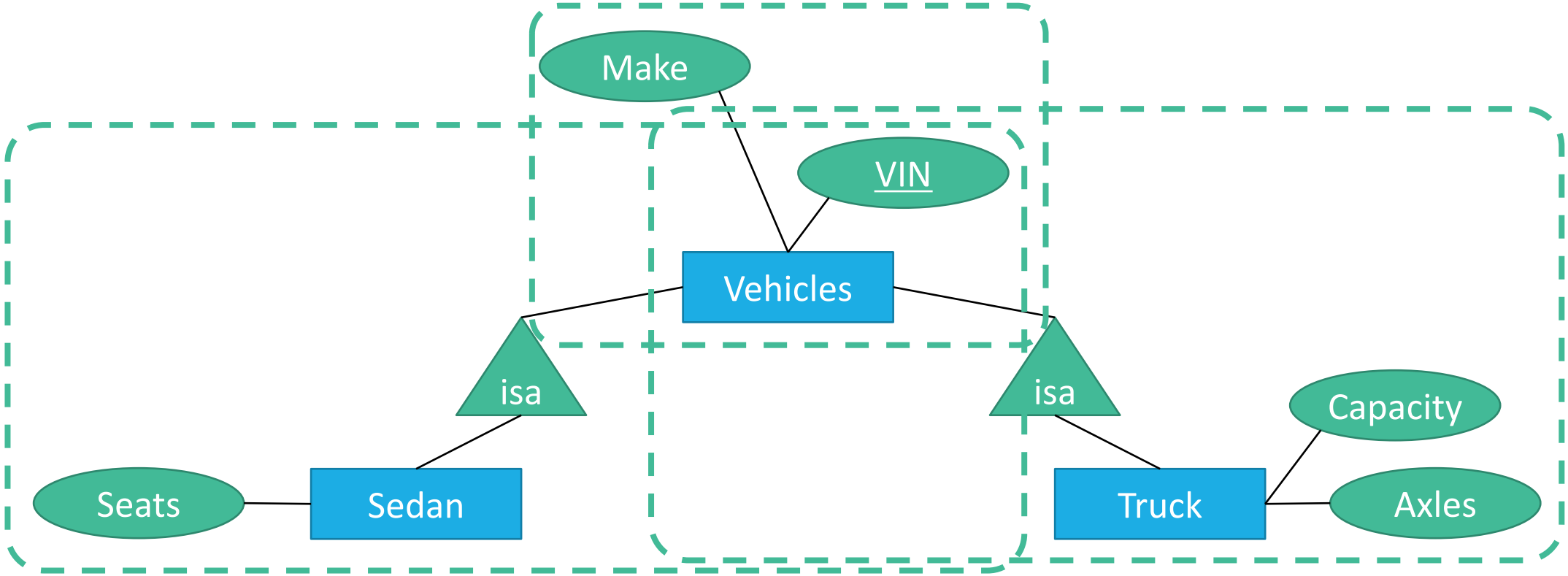
From Subclasses to Relations

There are three ways to convert a subclass entity to a relation:

- Object oriented – each entity set becomes a relation, with all applicable attributes
- NULL values – one table to represent entire hierarchy, and NULL values for inapplicable attributes
- ER approach – what we will use

For each entity set E in the hierarchy, create a relation that includes the key attributes from the root and any attributes belongs to E.

From Subclasses to Relations

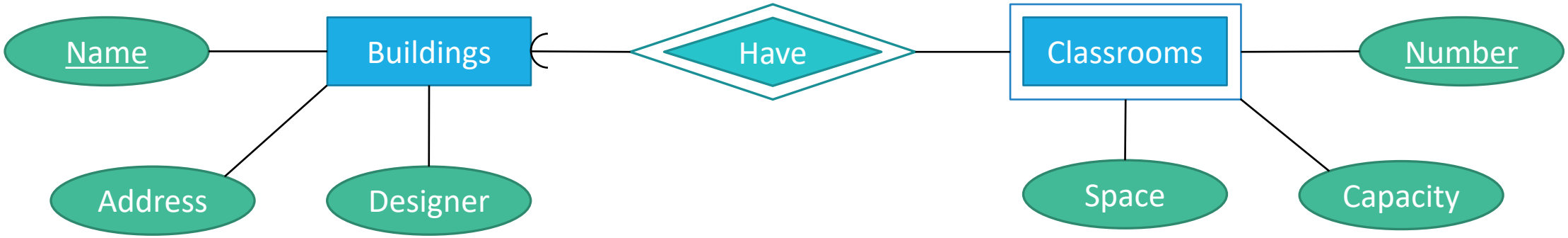


Vehicle(VIN, Make)

Sedan(Vehicle.VIN, Seats)

Truck(Vehicle.VIN, Axles, Capacity)

Handling Weak Entity Sets



For a weak entity set, the supporting relationships must be many-to-one relationships with referential integrity.

Similar to a many-to-one relationship, we combine the relationship and the weak entity set to create a single relation.

Buildings(Name, Address, Designer)

Classrooms(Buildings.Name, Number, Space, Capacity)

Normalization

Relational Schema Design

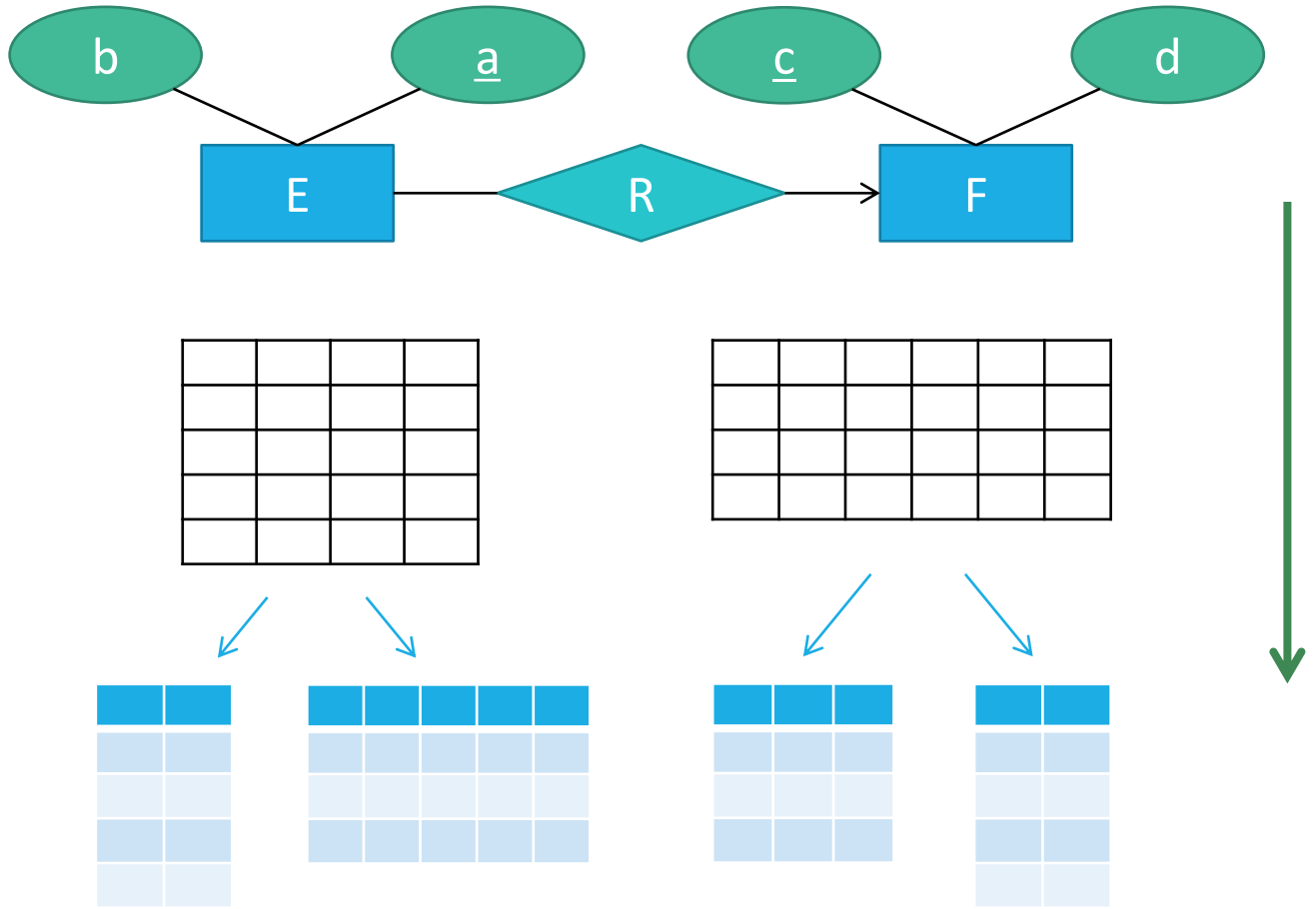
Conceptual Model:

Relational Model:

- May contain undesired functional dependencies

Normalization:

- Eliminate anomalies
- Reduce redundancy and dependency
  - Divide larger tables to smaller tables
  - link them using relationships



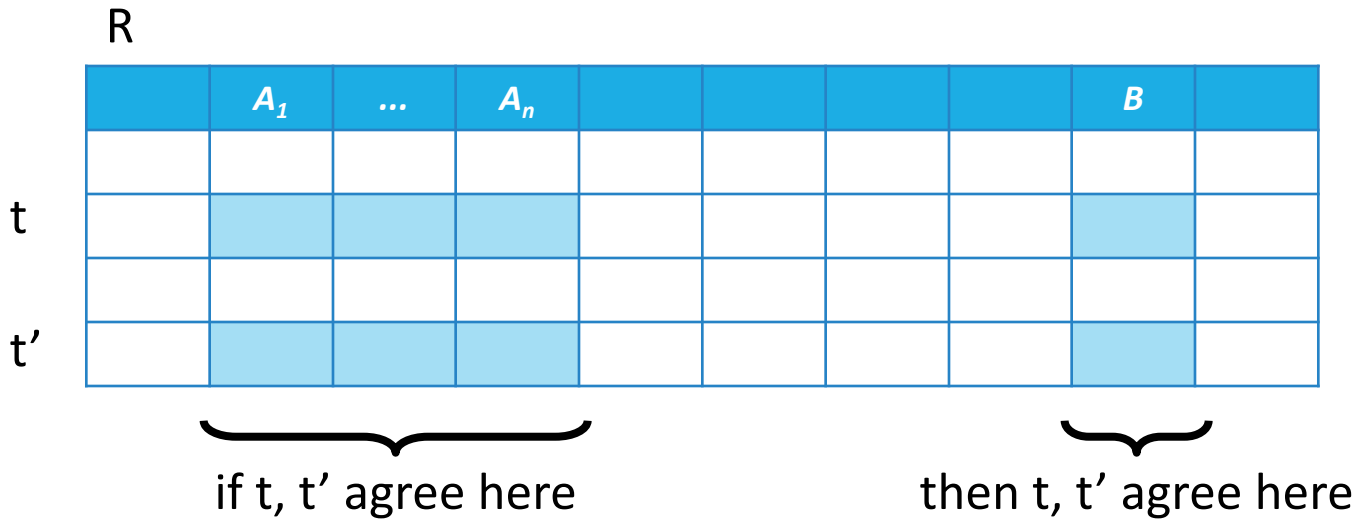
Functional Dependencies

A functional dependency (FD) on a relation R is a statement of the form:

"If two tuples of R agree on attributes A_1, A_2, …, A_n (i.e., the tuples have the same values in their respective components for each of these attributes), then they must also agree on another attribute, B."

Written as:

$A_1, A_2, \ldots, A_n \rightarrow B$



Functional Dependencies

Anomalies

- Redundancy
  - Unnecessary repeated faculty information.
- Insert Anomalies
  - What if this person is not teaching a course? Inability to insert new faculty member.
- Update Anomalies
  - May result in multiple conflicting copies of information.
- Delete Anomalies
  - What if a course is no longer offered? We might lose other information we want to save…

Faculty and Their Courses

Faculty ID	Faculty Name	Faculty Hire Date	Course Code
389	Dr. Giddens	10-Feb-1985	ENG-206
407	Dr. Saperstein	19-Apr-1999	CMP-101
407	Dr. Saperstein	19-Apr-1999	CMP-201
424	Dr. Newsome	29-Mar-2007	?

Employees' Skills

Employee ID	Employee Address	Skill
426	87 Sycamore Grove	Typing
426	87 Sycamore Grove	Shorthand
519	94 Chestnut Street	Public Speaking
519	96 Walnut Avenue	Carpentry

Faculty and Their Courses

Faculty ID	Faculty Name	Faculty Hire Date	Course Code
389	Dr. Giddens	10-Feb-1985	ENG-206
407	Dr. Saperstein	19-Apr-1999	CMP-101
407	Dr. Saperstein	19-Apr-1999	CMP-201

DELETE

Functional Dependencies

Problems that occur when we try to cram too much into a single relation are called anomalies. The principle types of anomalies that we encounter are:

- Redundancy. Information may be repeated unnecessarily in several tuples.
- Insert Anomalies. Too many attribute requirements for an entity to be inserted.
- Update Anomalies. Information need to be changed in several places when updating.
- Delete Anomalies. We may lose data when we don't want to.

Functional Dependencies

Employees(EmpID, Name, Phone, Position)

EmpID	Name	Phone	Position
E0045	Smith	1234	Clerk
E1247	John	9876	Salesrep
E1597	Smith	9876	Salesrep
E5239	Mary	1234	Lawyer
E6411	Peter	1234	Lawyer

Are there any FDs in the relation above?

- EmpID determines everything
- Position determines Phone (extension)
- But, Phone does not determine Position

Functional Dependencies

Example: people with multiple phone numbers

Name	SSN	PhoneNumber	City
Fred	123-45-6789	206-555-1234	Seattle
Fred	123-45-6789	206-555-6543	Seattle
Joe	987-65-4321	908-555-2121	Westfield
Joe	987-65-4321	908-555-1234	Westfield

- PhoneNumber → everything
- SSN → Name, City

Any kinds of anomalies?

- Redundancy = repeat data
- Update Anomalies = Fred moves to “Bellevue”
- Delete Anomalies = Fred drops all phone numbers

Decomposing Relations

The approach to eliminate these anomalies is to decompose relations.

Break the previous relation into two:

Name	SSN	City
Fred	123-45-6789	Seattle
Joe	987-65-4321	Westfield

SSN	PhoneNumber
123-45-6789	206-555-1234
123-45-6789	206-555-6543
987-65-4321	908-555-2121
987-65-4321	908-555-1234

Functional Dependencies

The goal of decomposition is to replace one relation with redundancy using two or more relations that do not exhibit anomalies.

Normalization process:

- Start with some relational schema
- Figure out the Functional Dependencies (FD's)
- Use them to design a better relational schema

Normalization

Edgar Codd, the inventor of the relational model, proposed the theory of normalization

- First Normal Form
- Extend to Second and Third Normal Form
- Boyce-Codd Normal Form (BCNF): Raymond F. Boyce joined with Edgar Codd

Terminology

Student (Student_ID, Name, Gender, SSN, Teacher_ID)

Teacher (Teacher_ID, Name, Salary)

Candidate key:

- Set of attributes which can uniquely identify a single record, but is not necessarily designated as the key

Candidate Key of Student:

- (Student_ID)
- (SSN)

Prime Attribute:

- An attribute which is a member of a candidate key

Super Key:

- A set of attributes that contains a (candidate) key is called a superkey, short for “superset of a key”.

Super Key of Student:

- (Student_ID)
- (Student_ID, Name)
- (SSN, Gender)
- Etc.

Terminology

Trivial Functional Dependencies:

- A functional dependency $A_1, A_2, \ldots, A_n \rightarrow B$ is trivial if B is one of the A's.
  - Example: Student_ID : SID; Course_ID : CID
- (SID, CID) → CID (SID, CID) → SID

Non-trivial Functional Dependencies:

- A functional dependency $A_1, A_2, \ldots, A_n \rightarrow B$ is trivial if B is not one of the A's.
  - Example: Student_ID : SID; Course_ID : CID
- (SID, CID) → Grade

Transitive Functional Dependencies:

- $A \rightarrow B, B \rightarrow C : A \rightarrow C$

Database Normalization Example

Assume a video library maintains a database of movies rented out

Full Names	Physical Address	Movies rented	Salutation	Category
Janet Jones	First Street Plot No 4	Pirates of the Caribbean, Clash of the Titans	Ms.	Action, Action
Robert Phil	3rd Street 34	Forgetting Sarah Marshal, Daddy's Little Girls	Mr.	Romance, Romance
Robert Phil	5th Avenue	Clash of the Titans	Mr.	Action

Movies Rented column has multiple values

Not Relational Database

First Normal Form (1NF)

Rules:

- Each table cell should contain a single value.
- Each record needs to be unique.

FULL NAMES	PHYSICAL ADDRESS	MOVIES RENTED	SALUTATION
Janet Jones	First Street Plot No 4	Pirates of the Caribbean	Ms.
Janet Jones	First Street Plot No 4	Clash of the Titans	Ms.
Robert Phil	3rd Street 34	Forgetting Sarah Marshal	Mr.
Robert Phil	3rd Street 34	Daddy's Little Girls	Mr.
Robert Phil	5th Avenue	Clash of the Titans	Mr.

A composite key is a primary key composed of multiple columns used to identify a record uniquely

Second Normal Form (2NF)

Rule 1: Be in 1NF

Rule 2: Single Column Primary Key

- Permits transitive FD's in a relation but forbids a nontrivial FD with a left side that is a proper subset of a key.

Primary Key

MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3rd Street 34	Mr.
3	Robert Phil	5th Avenue	Mr.

Foreign Key

MEMBERSHIP ID	MOVIES RENTED
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

Foreign Key

Will talk about Foreign Key in Next Week

Example:

- Insert a record in Movie_Rent Table, where Membership ID = 101

MEMBERSHIP ID	MOVIES RENTED
101	Mission Impossible

- But Membership ID = 101 does not exist in Membership Table

MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3rd Street 34	Mr.
3	Robert Phil	5th Avenue	Mr.

- Database will throw an Error. This helps in referential integrity.

Second Normal Form (2NF)

Rule 1: Be in 1NF

Rule 2: Single Column Primary Key

- Permits transitive FD's in a relation but forbids a nontrivial FD with a left side that is a proper subset of a key.

MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3rd Street 34	Mr.
3	Robert Phil	5th Avenue	Mr.

MEMBERSHIP ID	MOVIES RENTED
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

Change in Name → May Change Salutation

Is there any transitive FD in the Membership Table?

Third Normal Form (3NF)

Rule 1: Be in 2NF

Rule 2: No transitive functional dependencies

MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION ID
1	Janet Jones	First Street Plot No 4	2
2	Robert Phil	3rd Street 34	1
3	Robert Phil	5th Avenue	1

MEMBERSHIP ID	MOVIES RENTED
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

SALUTATION ID	SALUTATION
1	Mr.
2	Ms.
3	Mrs.
4	Dr.

Boyce-Codd Normal Form (BCNF)

Even when a database is in 3rd Normal Form, still there would be anomalies resulted if it has more than one **Candidate** Key.

There is a simple condition under which the anomalies can be guaranteed not to exist. This condition is called:

Boyce-Codd normal form, or BCNF

- has no more than one **Candidate** Key

Sometimes is BCNF is also referred as **3.5 Normal Form.**

Boyce-Codd Normal Form

Boyce Codd Normal Form (BCNF):

- A relation R is in BCNF if and only if: whenever there is a nontrivial FD $A_1, A_2, \ldots, A_n \rightarrow$ B for R, it is the case that $\{A_1, A_2, \ldots, A_n\}$ is a superkey for R.

That is, the left side of every nontrivial FD must be a superkey.

Boyce-Codd Normal Form

Class	Professor	EmployeeID
CEE 327	Anne Goodchild	12994
CEE 367	Pedro Arduino	12223
CEE 377	Gregory Miller	12889
CEE 410	Ryan Avery	12786
CEE 454	Ryan Avery	12786

What are the functional dependencies?

- Class → Everything – Only candidate key
- EmployeeID → Professor – Non-trivial functional dependency

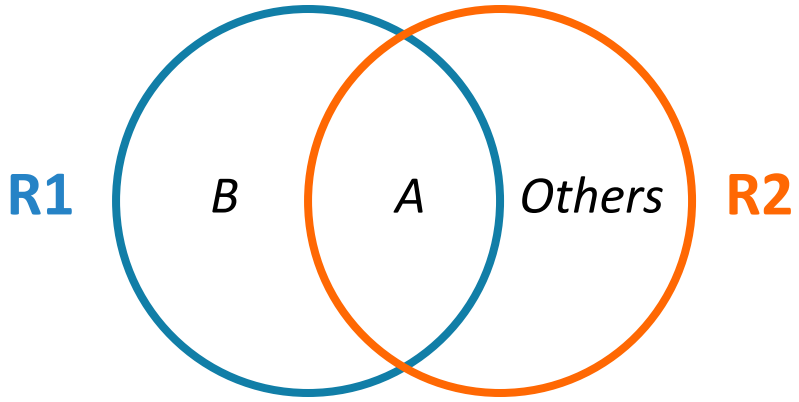
EmployeeID is not a superkey

Conclusion: This relation is **NOT** in BCNF

Decomposition into BCNF

Algorithm (for each relation):

1. Pick any functional dependency $A \rightarrow B$ that violates the chosen normal form.
2. Decompose into R1(A, B) and R2(A, Rest).
3. Identify Keys for R1 and R2.
4. Repeat.



Decomposition into BCNF

Example: decompose the following relation describing purchases in a retailer system.

Purchase (InvoiceNumber, Date, Time, CustomerID, CustomerName, StoreName, StoreAddress)

Identify functional dependencies.

1. InvoiceNumber → Everything
2. Date, Time, CustomerID → Everything
3. StoreAddress → StoreName
4. CustomerID → CustomerName

Decomposition into BCNF

Pick a FD that violates the BCNF.

4. CustomerID → CustomerName

Decompose R into two relations, and identify the key for each.

Purchase(InvoiceNumber, Date, Time, CustomerID, StoreName, StoreAddress)

Customer(CustomerID, CustomerName)

Decomposition into BCNF

Purchase(InvoiceNumber, Date, Time, CustomerID, StoreName, StoreAddress)

Customer(CustomerID, CustomerName)

Identify functional dependencies.

Purchase:

1. InvoiceNumber → Everything
2. Date, Time, CustomerID → Everything
3. StoreAddress → StoreName

Customer:

4. CustomerID → CustomerName

Decomposition into BCNF

Pick a FD that violates the BCNF.

3. StoreAddress → StoreName

Decompose the relation, and identify the key for each created relations.

Purchase(InvoiceNumber, Date, Time, CustomerID, StoreAddress)

Customer(CustomerID, CustomerName)

Store(StoreAddress, StoreName)

All relations are in BCNF.

Normal Forms

First Normal Form (1NF):

- The condition that every component of every tuple is an atomic value.

Second Normal Form (2NF):

- Permits transitive FD's in a relation but forbids a nontrivial FD with a left side that is a proper subset of a key.

Third Normal Form (3NF):

- A relation R is in third normal form (3NF) if: whenever $A_1, A_2, \ldots, A_n \rightarrow B$ is a nontrivial FD, either $\{A_1, A_2, \ldots, A_n\}$ is a superkey, or B is a member of some key.

Normal Forms

Put another way:

Second Normal Form (2NF):

- In 1NF and every non-prime attribute is dependent on the entirety of every candidate key.

Third Normal Form (3NF):

- In 2NF and every non-prime attribute is (non-transitively) dependent on the entirety of all candidate keys.

Relation Examples

Class	Professor	EmployeeID
CEE 327	Anne Goodchild	12994
CEE 367	Pedro Arduino	12223
CEE 377	Gregory Miller	12889
CEE 410	Ryan Avery	12786
CEE 454	Ryan Avery	12786

- Class → Everything – Only candidate key
- EmployeeID → Professor – Non-trivial functional dependency
- Class → EmployeeID → Professor – Transitive FD (allowed by 2NF)

Class itself is the entire key
Everything can be determined by Class
} The relation is in 2NF

EmployeeID is not a superkey,
Professor is not a member of a candidate key
} The relation is **NOT** in 3NF

Boyce-Codd and the Third Normal Form

Boyce-Codd Normal Form:

- For all non-trivial functional dependencies A → B, A must be a super key

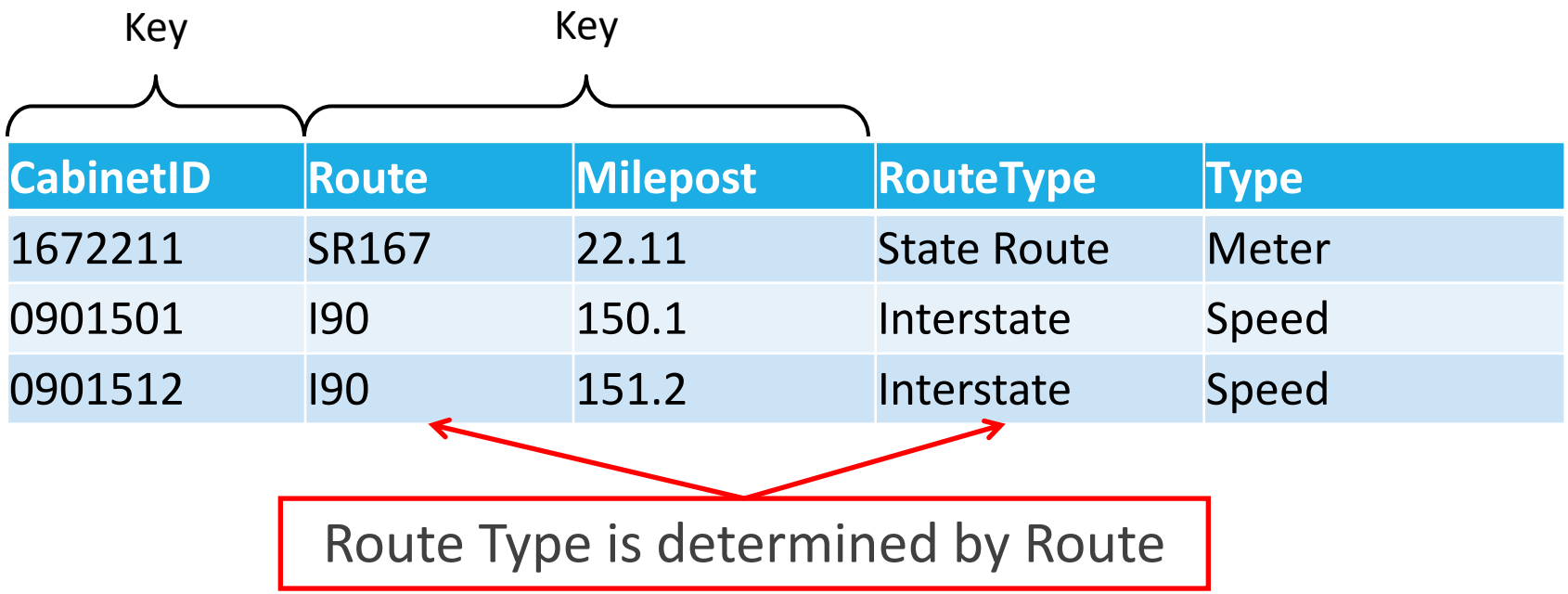
Third Normal Form:

- For all non-trivial functional dependencies A → B, A must be a super key or B is a prime attribute.

Prime attribute: Member of a candidate key

Relation Examples

2NF: every non-prime attribute is dependent on the entirety of every candidate key.



Not in 2NF: non-prime attribute determined by subset of a key.

Relation Examples

3NF: For all non-trivial functional dependencies A → B, A must be a super key or B is a prime attribute.

Key

AccidentID	Route	Milepost	Date	Season
1672211	SR167	22.11	1/12/2013	Winter
0901501	I90	150.1	5/29/2013	Summer
0901512	I90	151.2	11/19/2013	Winter

Season is determined by Date

In 2NF: Everything is (directly or transitively) determined by AccidentID.

Not in 3NF: Date is not a superkey and Season is not a prime attribute.

Relation Examples

BCNF: For all non-trivial functional dependencies A → B, A must be a super key.

Player	Number	Team	Stadium
Marshawn Lynch	24	Seahawks	CenturyLink
Peyton Manning	18	Broncos	Sports Authority
Russell Wilson	3	Seahawks	CenturyLink

A lot of overlapping candidate keys: {Player}, {Number, Team}, {Stadium, Number}, etc., so no non-prime attributes
BUT: Team → Stadium

In 3NF: Stadium is a prime attribute (a member of a candidate key).

Not in BCNF: Team is not a superkey.

Boyce-Codd and the Third Normal Form

Two important properties of a decomposition:

1. Recovery: it should be possible to project the original relations onto the decomposed schema, and then reconstruct the original.
2. Dependency preservation: it should be possible to check in the projected relations whether all the given FD's are satisfied.

3NF is an alternative set of requirements that is slightly less strict than BCNF, and is good enough for most cases.

- We can get both 1 and 2 with a 3NF decomposition.
- But we can't always get 1 and 2 with a BCNF decomposition.