Welcome!

Our Goal

- Design Conceptual Data Models (E-R model and diagram)
- Execute models by converting them to relational databases (Normal Forms)

Data Modeling Occurs in Three Steps

- Data analysis (what do we need)
- Conceptual Data model
- Converting to a database (in our case, relational)

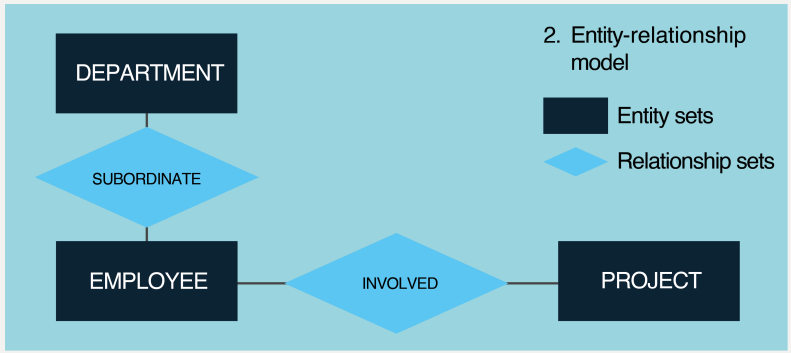
How Do We Execute the Steps?

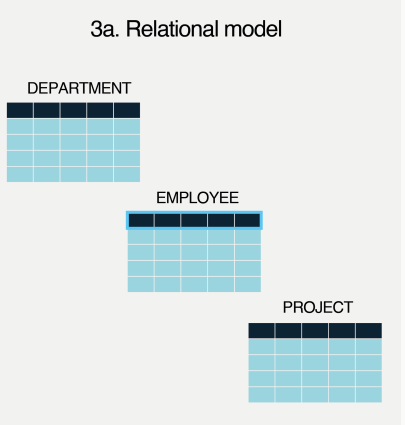
- Find required data from the information system
- Find a way to organize the relationship between features in your data
- Provide a formal description of the database object

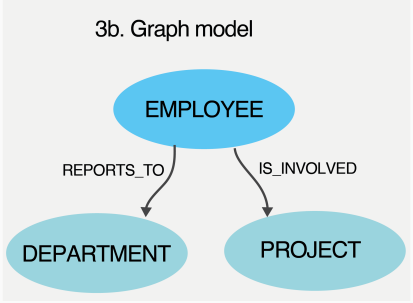
An Example: Project Monitoring

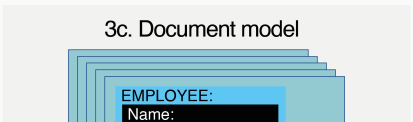
1. Requirements analysis	Order no. 113 Date: 07/14/2023
Goal	For project monitoring purposes, employees, work, and project times should periodically be reported per department.

1. Employees report to departments, with each employee being assigned to exactly one department.
2. Each project is centrally assigned a unique project number.
3. Employees can work on multiple projects simultaneously; the respective percentages of their time are logged.
4. ...









Bad Ways of Doing This

- Redundancy
- Incompleteness

Conceptual Data Model

We will look at the Entity-Relationship model, which we discussed Monday

E-R Model

- Entity sets
- Relationship sets
- Attributes

Entity Sets

- Entities are things in the real world that we can distinguish from another object
- An entity set is a set of entities of the same type that share the same attributes
- The **extension** is the concrete actual collection of entities in the entity set
- They do not need to be disjoint
  - Tom can be a student and an instructor

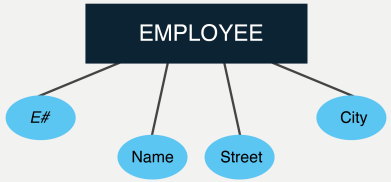
Entity Sets

- Entities are described by attributes
- Attributes have values

Entity Sets Example

Entity:	Employee Murphy, lives on Morris Road in Kent
Entity set:	Set of all employees with the attributes Name, Street, and City
Identification key:	Employee number as an artificial key

Representation in the entity-relationship model



Relationship Sets

- A relationship is an association among several entities

Relationship Sets

- A relationship is an association among several entities
- Formally

Relationship Sets

- A relationship is an association among several entities
- Formally
- If $E_1, E_2, ...E_n$ are n entity sets...

Relationship Sets

- A relationship is an association among several entities
- Formally
- If $E_1, E_2, ...E_n$ are n entity sets...
- Then a relationship set R is a subset of...

Relationship Sets

- A relationship is an association among several entities
- Formally
- If $E_1, E_2, ...E_n$ are n entity sets...
- Then a relationship set R is a subset of...
- $\{(e_1, e_2, ...e_n)|e_1 \in E_1, ...e_n \in E_n\}$

Relationship Sets

- A relationship is an association among several entities
- Formally
- If $E_1, E_2, ...E_n$ are n entity sets...
- Then a relationship set R is a subset of...
- $\{(e_1, e_2, ...e_n)|e_1 \in E_1, ...e_n \in E_n\}$
- where e_i are relationships

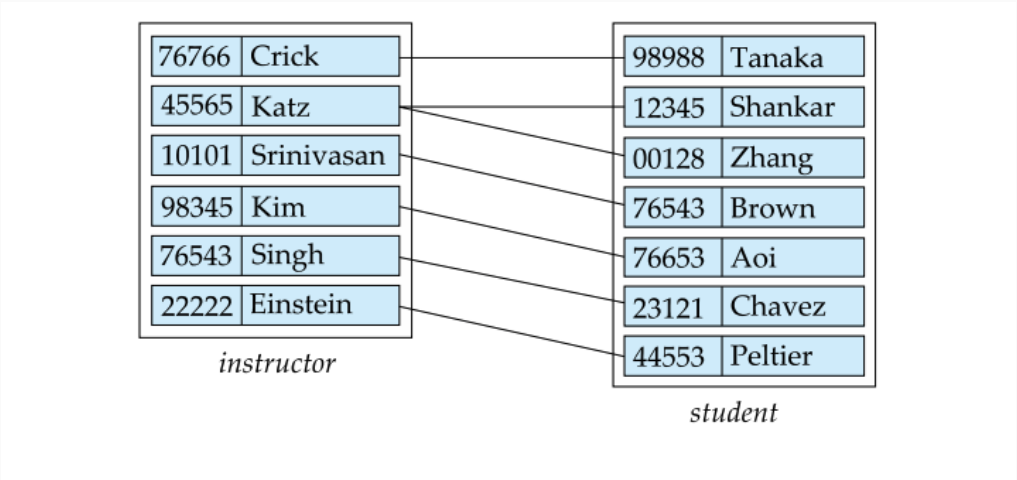
Relationship Sets

Example:

Student(SID = 112233, Name = Peter)
advisor(SID, IID)
instructor(IID = 332211, Name = John)
(112233, 332211) $\in$ *advisor*

Relationship Sets

Advisor is the relationship set which captures the association between students and their mentors



Relationship Sets

- Another way to think about it is as participation

Relationship Sets

- Another way to think about it is as participation
- $E_1, ...E_n$ participate in R

Relationship Sets

- Another way to think about it is as participation
- $E_1, ...E_n$ participate in R
- The function of that participation is a role

Relationship Sets

- Another way to think about it is as participation
- $E_1, ...E_n$ participate in R
- The function of that participation is a role
- Instructors ADVISED. Students are ADVISED

Relationship Sets

- Some relationship sets are recursive

Relationship Sets

- Some relationship sets are recursive
- Recursive means the same entity set participates in a relationship set more than once, but in different roles

Relationship Sets

- Some relationship sets are recursive
- Recursive means the same entity set participates in a relationship set more than once, but in different roles
- **Example:** Tim is a father to Jim, but he has a father named Steve

Relationship Sets

- Some relationship sets are recursive
- Recursive means the same entity set participates in a relationship set more than once, but in different roles
- **Example:** Tim is a father to Jim, but he has a father named Steve
- If there was a relationship set *parent*, Tim would enter as the father of Jim, but the son of Jim

Relationship Sets

- Some relationship sets are recursive
- Recursive means the same entity set participates in a relationship set more than once, but in different roles
- **Example:** Tim is a father to Jim, but he has a father named Steve
- If there was a relationship set *parent*, Tim would enter as the father of Jim, but the son of Jim
- **Example:** Employees report to a manager

Relationship Sets

- Some relationship sets are recursive
- Recursive means the same entity set participates in a relationship set more than once, but in different roles
- **Example:** Tim is a father to Jim, but he has a father named Steve
- If there was a relationship set *parent*, Tim would enter as the father of Jim, but the son of Jim
- **Example:** Employees report to a manager
- Except the CEO, everyone reports to someone, but middle managers have people who report **to them** .

Relationship Sets

- Some relationship sets are recursive
- Recursive means the same entity set participates in a relationship set more than once, but in different roles
- **Example:** Tim is a father to Jim, but he has a father named Steve
- If there was a relationship set *parent*, Tim would enter as the father of Jim, but the son of Jim
- **Example:** Employees report to a manager
- Except the CEO, everyone reports to someone, but middle managers have people who report **to them** .
- Middle managers would enter as subordiantes and managers

Relationship Sets

- Relationship sets have descriptive attributes

Relationship Sets

- Relationship sets have descriptive attributes
- **Example:** We may add `Date` to signify when an instructor started advising a student

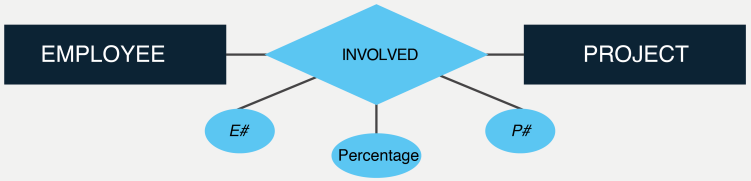
Relationship Sets

- Relationship sets have descriptive attributes
- **Example:** We may add `Date` to signify when an instructor started advising a student
- **Example 2:** Student *takes* course, *takes* may have a grade as a descriptive attribute

Example Relationship Set

Relationship:	Employee Murphy spends 70 % of their time working on project P17
Relationship set:	Set of all employee project involvements with the attributes Employee number, Project number, and Percentage
Identification key:	Concatenated key consisting of employee number and project number

Representation in the entity-relationship model



Attributes

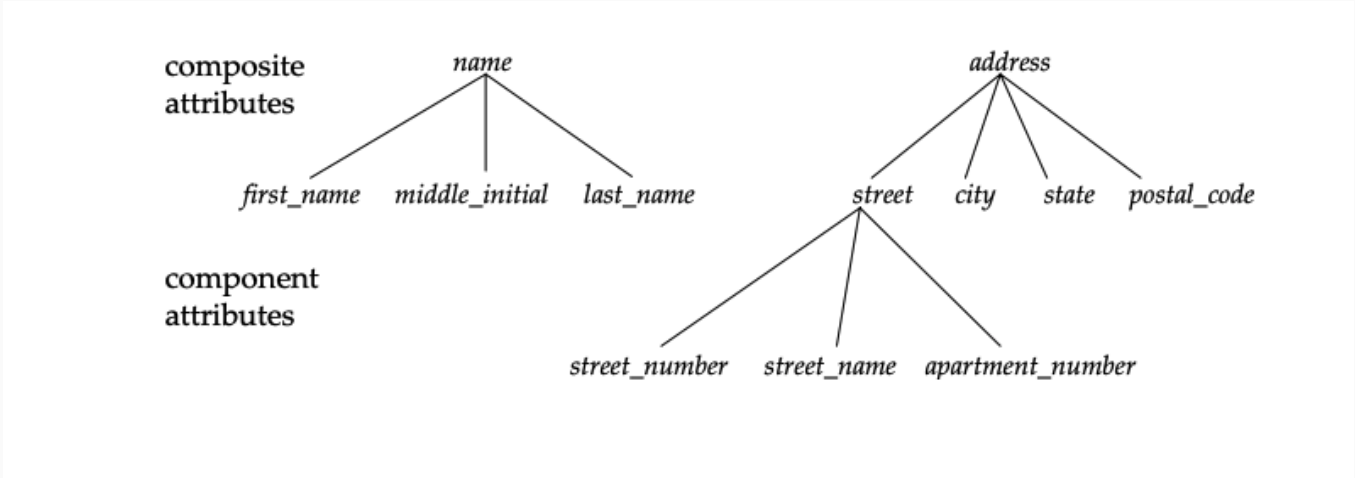
- The domain of an attribute is predetermined values that the attribute can take on
- The domain of a `Semester` attribute may be $\{Fall, Spring\}$
- An attribute is a function that maps from the entity set into the domain
- Each entity with arity m can be described by a set of m (attribute, data value) pairs
- Instructor(<u>ID</u>, Name, Salary)
- $\{(ID, 123), (name, Einstein), (Salary, 100,000)\}$

Types of Attributes

- Simple: no subparts
- Composite: can be divided into subparts.
- Single-valued: only take on one value
- Multivalued: values are a set $\{phone\_number\}$
- Note these may overlap: a single-valued attribute can be simple, but it could also be composite (Birthdate into a regular date).

Types of Attributes

Composite attributes can be broken into components



Association Types

Association Types

- Every association from E_1 to E_2 has a type
- An association type from E_1 to E_2 indicates how many entities of the associated entity set in E_2 can be assigned to a specific entity in E_1
- There are four types
  - Unique (type 1)
  - Conditional association (type c)
  - Multiple association (type m)
  - Multiple-conditional association (type mc)

Unique Associations

- Each entity from the entity set E_1 is assigned exactly one entity from the set E_2
- Assume no employee can be cross-listed
- Then each employee is subordinate to ONE and ONLY ONE department
- The subordinate relationship from employees to departments is unique (type 1)

Conditional Association (Type C)

- Each entity from E_1 is assigned 0 or 1 entity from the entity set E_2. The entity set is optional, so it is conditional
- Each department has one head, who is an employee, but not every employee is a department head

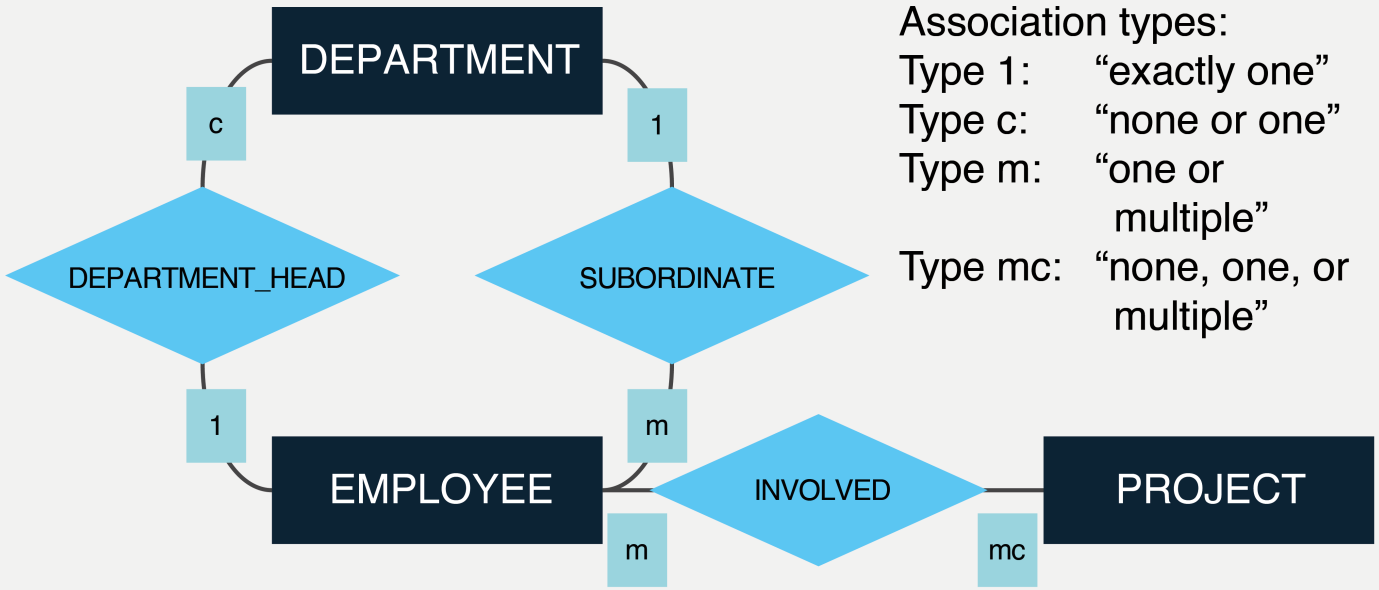
Multiple Association

- Each entity from entity set E_1 is assigned one or more entities from the entity set E_2
- This is often called complex, since one entity from E_1 can be related to an arbitrary number of entities from E_2
- From projects to employees - each project has at least one, but possibly more, employees

Multiple-conditional association (mc)

- Each entity from the entity set E_1 is assigned 0, 1, or ¿1 entities from the entity set E_2.
- This is also called conditional complex
- its different from m in that not every entity from E_1must have a relationship to E_2
- From employees to projects - not every employee is involved in every project, but an employee can be involved in one or more

Association Types Example



Association types:
Type 1: “exactly one”
Type c: “none or one”
Type m: “one or multiple”
Type mc: “none, one, or multiple”

Example for department heads:
Typ c: “Each employee may or may not be a department head.”
Typ 1: “Each department has exactly one department head.”

Constraints

An E-R schema may define constraints to which the database must conform

- Mapping cardinalities
- Participation Constraints
- Keys

Cardinality

- Each relationship contains two association types (E_1 to E_2, and E_2 to E_1)
- Cardinality := (association type from E_1 to E_2, association type from E_2 to E_1)

Possible Cardinalities

Bj := (A1, A2) Cardinalities of relationships with the association types A1 and A2

A1 \ A2	1	c	m	mc
1	(1,1)	(1,c)	(1,m)	(1,mc)
c	(c,1)	(c,c)	(c,m)	(c,mc)
m	(m,1)	(m,c)	(m,m)	(m,mc)
mc	(mc,1)	(mc,c)	(mc,m)	(mc,mc)

B1: (1,1), (1,c), (c,1), (c,c)
B2: (1,m), (1,mc), (c,m), (c,mc), (m,1), (m,c), (mc,1), (mc,c)
B3: (m,m), (m,mc), (mc,m), (mc,mc)

B1: unique-unique relationships
B2: unique-complex relationships
B3: complex-complex relationships

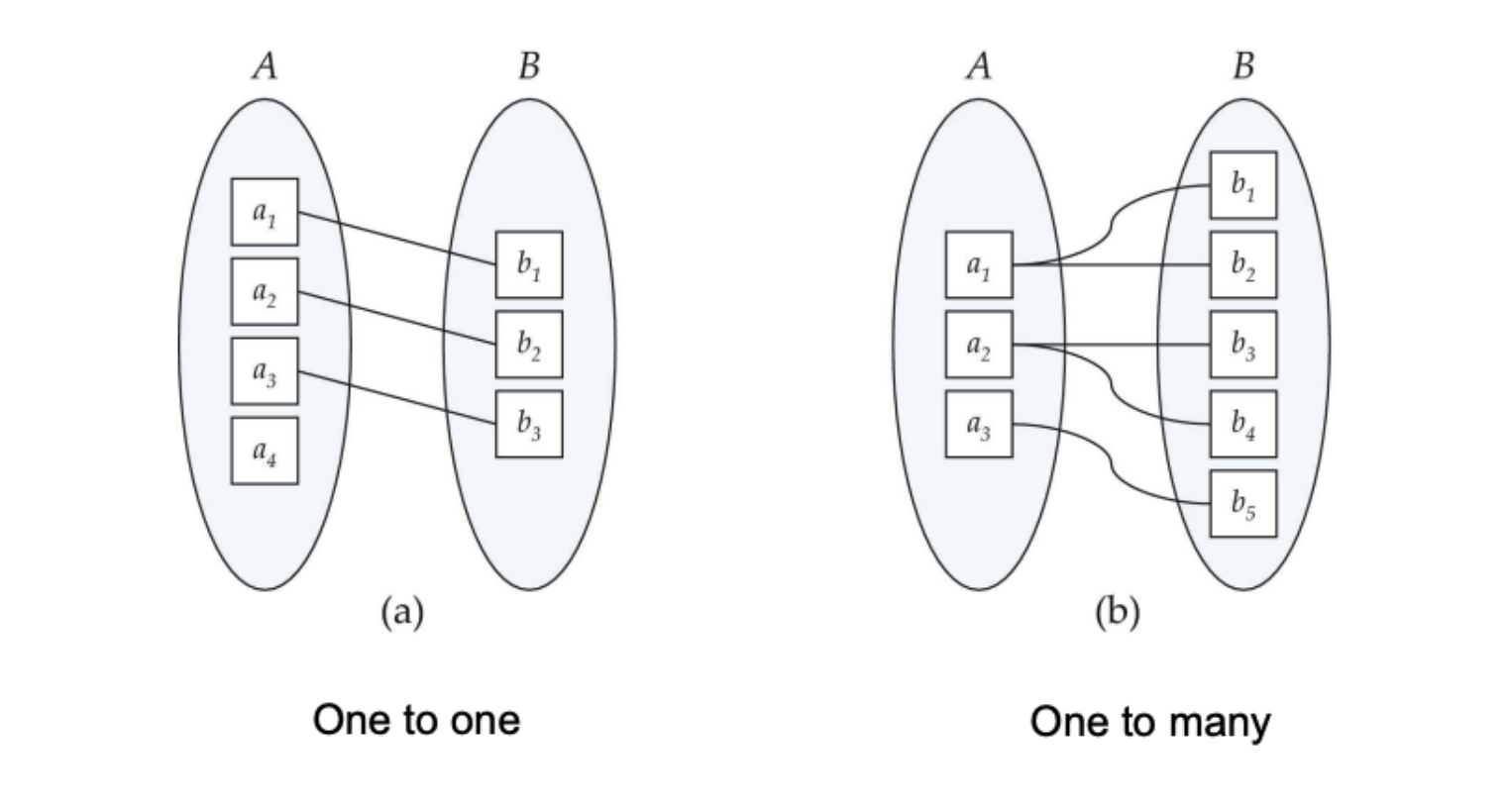
Participation Constraints

- The participation of E in R is said to be total if every entity in E participates in at least one relationship in R
- If only some entities in E participate in R, then participation is partial.

Participation Constraints

B totally participates in A in figure (a).

A partially participates in figure (a)



Keys

- We must have a way to specify how entities within a given entity set are distinguished
- The key for an entity set is a set of attributes that suffice to distinguish entities from each other
- Superkeys, candidate keys, and primary keys apply here as they apply to relational schemas

Keys For Relationship Sets

- Let R be the relationship set involving entity sets $E_1, E_2, ...E_n$
- Let $primary\,key(E_i)$ denote the set of attributes that form the primary key for entity set E_i
- The composition of the primary key for a relationship set depends on the set of attributes associated with the relationship set R.
- If the relationship set R has no attributes associated with it then an individual relationship in set R is

$$primary\,key(E_1) \cup key(E_2)... \cup primary\,key(E_n)$$

- If the relationship set R has m attributes $a_1, a_2, .., a_m$ associated with it, then its

$$primary\,key(E_1) \cup key(E_2)... \cup primary\,key(E_n) \cup \{a_1, a_2, ..., a_m\}$$

Keys For Relationship Sets

- If the attribute names of primary keys are not unique across entity sets, the attributes are renamed to distinguish them
- The name of the entity set is combined with the name of the attribute to form a unique name

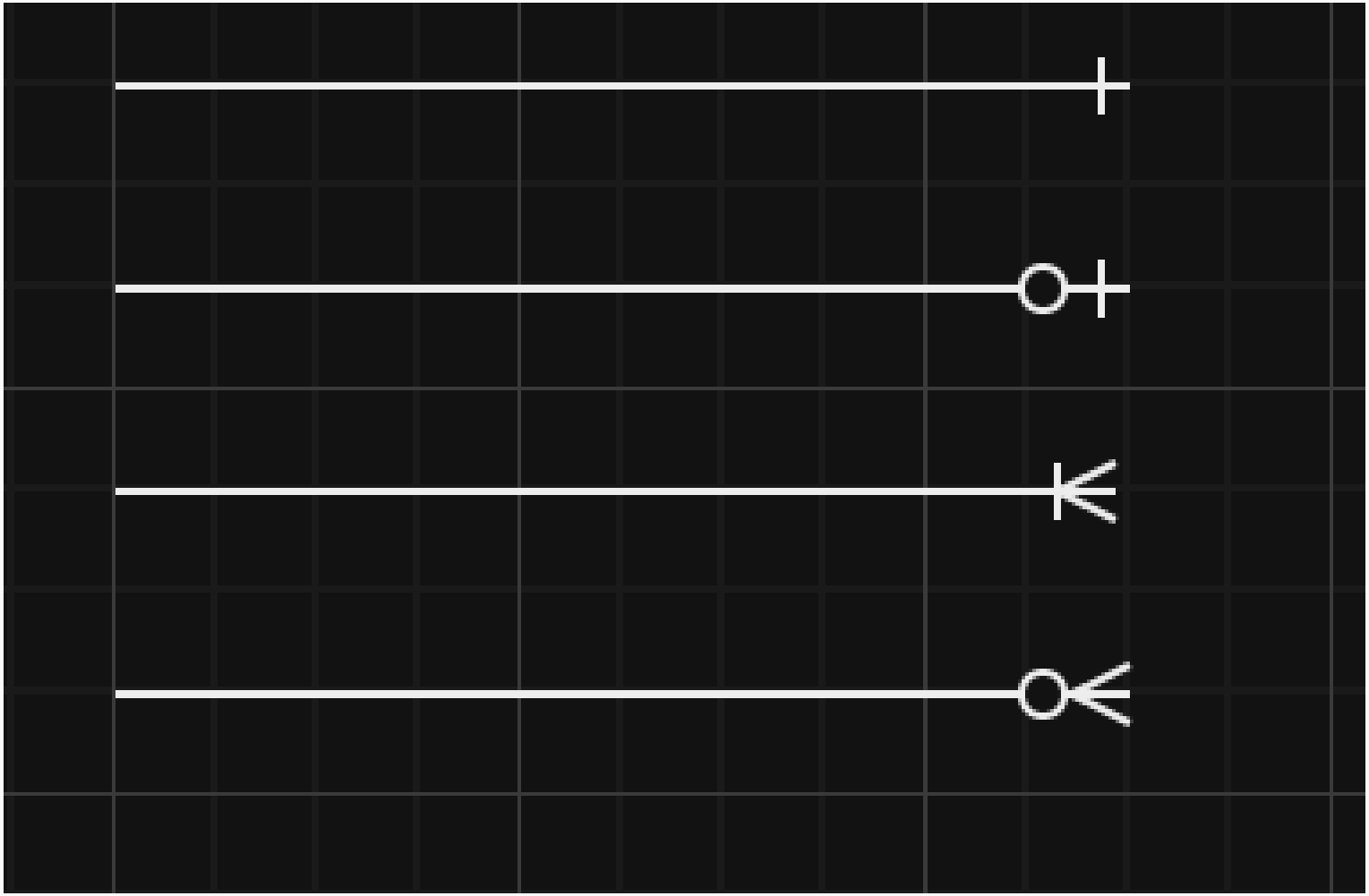
ER Diagrams

- Let's turn these concepts into objects that we can see!
- Power of E-R model is its diagram

ER Diagrams

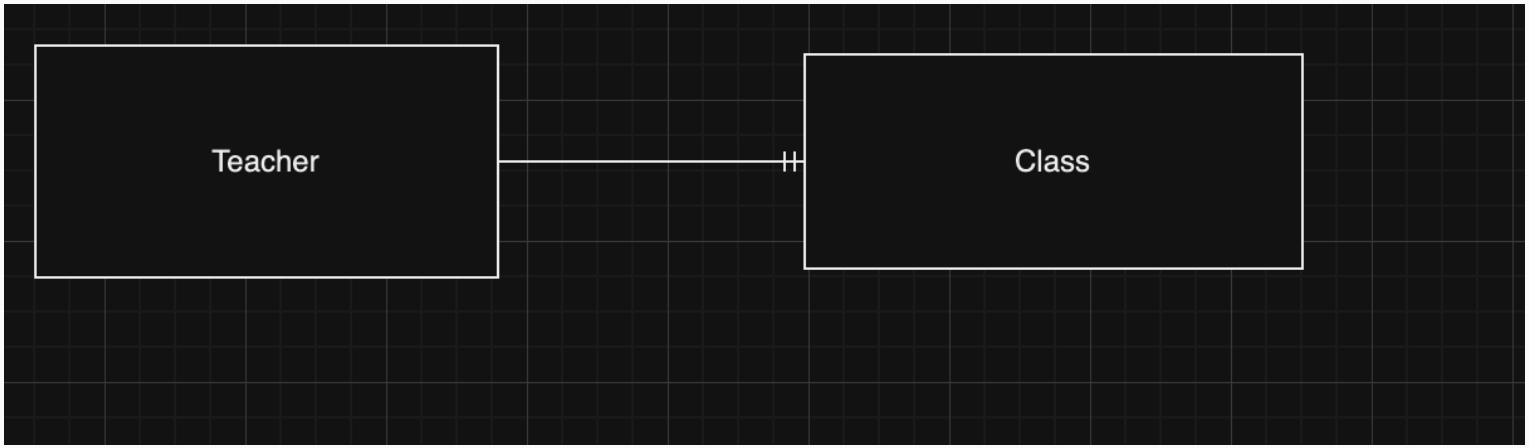
- We represent entities with squares and relationships with diamonds
- We will represent association types using crows foot notation

Crow's Foot Notation



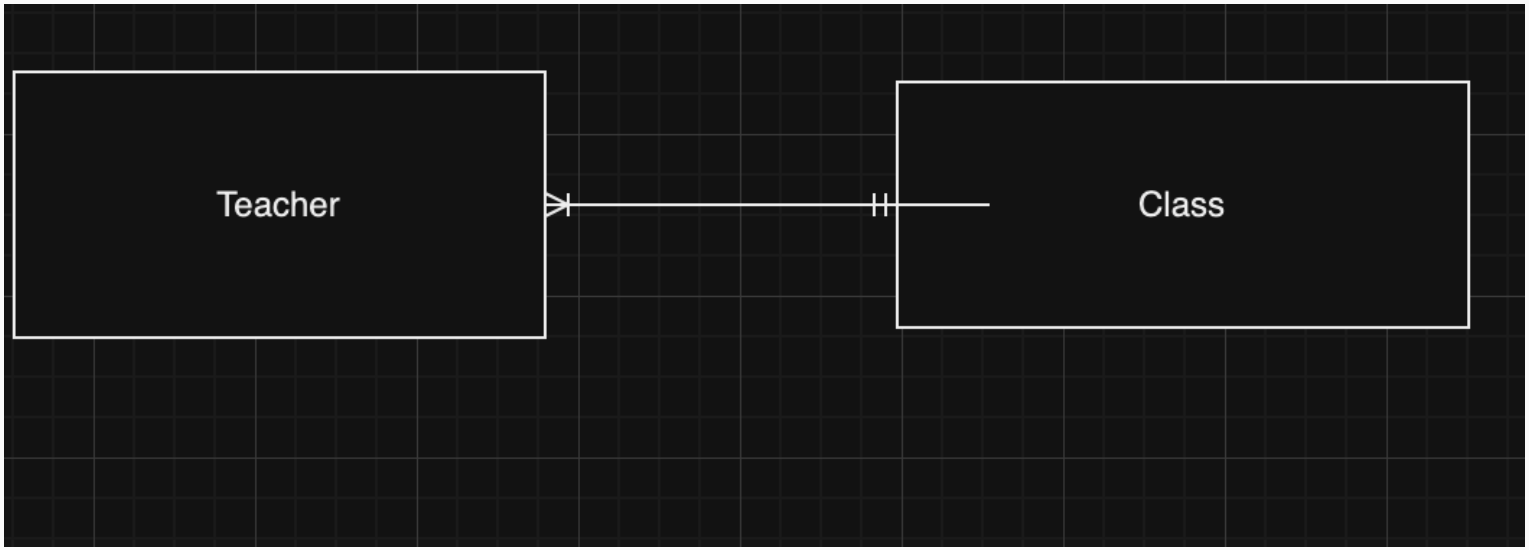
Teacher Has One Class

Pretend teachers are only allowed to teach one class a semester



Teacher Has One Class AND Class has at least 1 Teacher

But a class can be taught by multiple people (lots of biology 101 teachers)



Practice Question

UPS prides itself on having up-to-date information on the processing and current location of each shipped item. To do this, UPS relies on a company-wide information system. Shipped items are the heart of the UPS product tracking information system. Shipped items can be characterized by item number (unique), weight, dimensions, insurance amount, destination, and final delivery date. Shipped items are received into the UPS system at a single retail center. Retail centers are characterized by their type, uniqueID, and address. Shipped items make their way to their destination via one or more standard UPS transportation events (i.e., flights, truck deliveries). These transportation events are characterized by a unique scheduleNumber, a type (e.g, flight, truck), and a deliveryRoute.

Please create an Entity Relationship diagram that captures this information about the UPS system. Be certain to indicate identifiers and cardinality constraints.

Step 1: Flag the Entities and Relationships (Nouns and Verbs) and attributes (adjectives)

UPS prides itself on having up-to-date information on the processing and current location of each shipped item. To do this, UPS relies on a company-wide information system. Shipped items are the heart of the UPS product tracking information system. **Shipped items** can be characterized by item number (unique, weight, and final delivery date. Shipped items are *received* into the UPS system at a single **retail center**. Retail centers are characterized by their type, uniqueID, and address. Shipped items *make their way to their destination* via one or more standard UPS **transportation events** (i.e., flights, truck deliveries). These transportation events are characterized by a unique scheduleNumber, a type (e.g, flight, truck), and a deliveryRoute.

Step 1

- Entities: items, retail center, event
- Relationships: received, shipped

Step 2: Make a Matrix

$$
\begin{bmatrix}
 & \text{Items} & \text{Retail Center} & \text{Event} \\
\text{Items} & \textit{NA} & & \\
\text{Retail Center} & & \textit{NA} & \\
\text{Event} & & & \text{NA}
\end{bmatrix}
$$

Step 2: Make a Matrix

$$\begin{bmatrix} & \text{Items} & \text{Retail Center} & \text{Event} \\ \text{Items} & \textit{NA} & \textit{Recieve} & \textit{Ship} \\ \text{Retail Center} & \textit{Recieve} & \textit{NA} & \\ \text{Event} & \textit{Ship} & & \text{NA} \end{bmatrix}$$

Step 2: Make a Matrix

- Our matrix shows us that there is one entity with two relationships, and two entities with one relationship
- It make make sense then to think about the diagram as having an "L" shape.

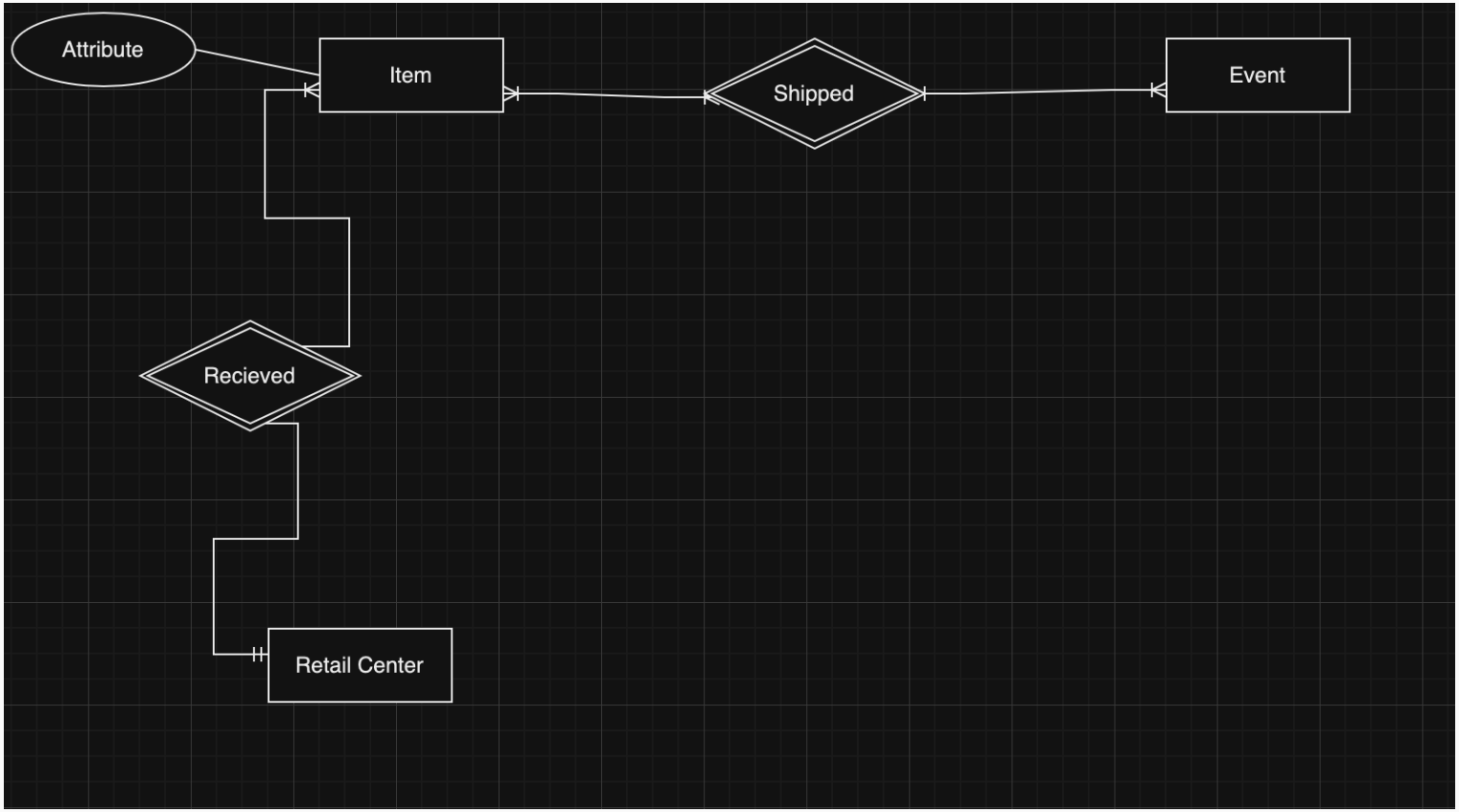
Step 3: Consider the Association Types

- Items - a retail center can only receive one (1)
- Items are shipped via **one or more** transportation methods (m)
- Events - can't happen without an item, but likely have many (m)
- Retail centers - must have one item, but likely have many (m)

Step 4: Define Cardinality

- Items, Retail Center - (1,m): Unique-complex
- Items, Events - (m, m): Complex-complex

Draw the Diagram



- Extended Entity-Relationship (EER) model includes additional features beyond the basic E-R model.
- Two important EER features are **Generalization** and **Aggregation**.
- In this presentation, we'll explore these concepts in detail.

Generalization

- Generalization is a process of defining a more general entity type from a set of more specialized entity types.
- It allows us to model hierarchies and capture common attributes and relationships.
- Key terms:
  - **Superclass**: The more general entity type.
  - **Subclass**: Specialized entity types that inherit from the superclass.
- Example: A "Vehicle" superclass with "Car" and "Motorcycle" subclasses.
- Often called an "is-a" relationship ("Car" is a "vehicle")

Generalization Constellations

- Overlapping entity subsets
- Overlapping-complete entity subsets
- Disjoint entity subsets
- Disjoint-complete entity subsets

Generalization Constellations

- Overlapping entity subsets
- The specialized entity sets overlap with each other
- Consider the entity set Employee with two subsets (Photo_Club, Sports_Club)
- Say members can be in each club, then its overlapping

Generalization Constellations

- Overlapping-complete entity subsets
- Overlap along with COMPLETE coverage of the entity set across the subsets
- Say there is a Chess_Club along with the PHOTO_Club and Sports_Club
- Assume every employee has to join one club
- Overlapping and Complete

Generalization Constellations

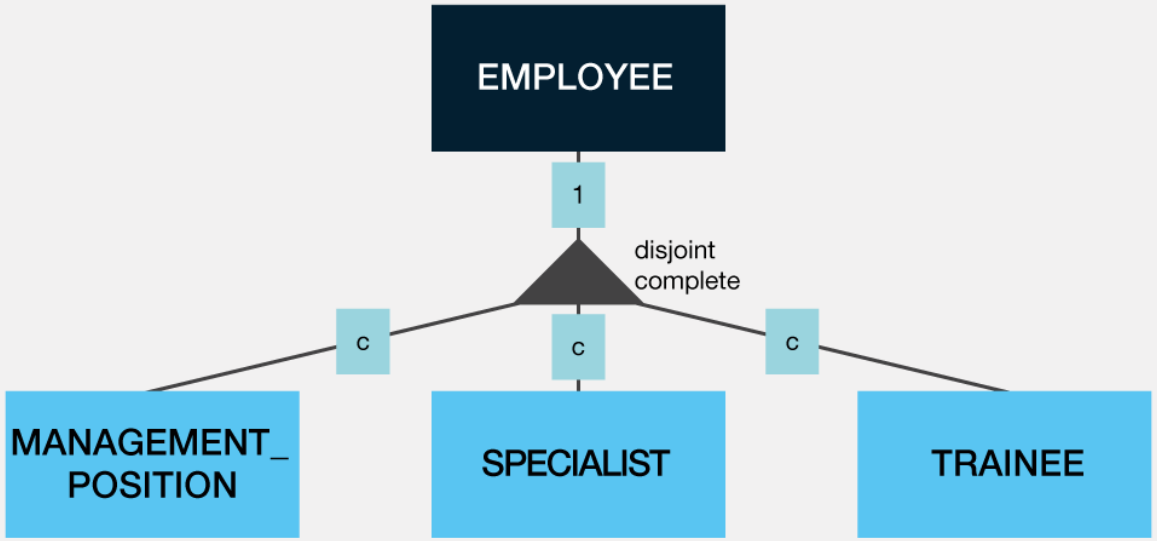
- Disjoint entity subsets
- Entity sets in the specialization are disjoint (mutually exclusive)
- Consider employee entity set with specializations MANAGEMENT and ASSISTANT. Since employees cannot be both, it's disjoint entity subset

Generalization Constellations

- Overlapping and Complete
- Specialization entity sets are disjoint but together completely cover the generalized entity set.
- If the company ONLY had management and assistants

Generalization Constellations

A disjoint complete generalization example



Eyample for employee categorization:
"Each employee is either part of management, a specialist, or a trainee."

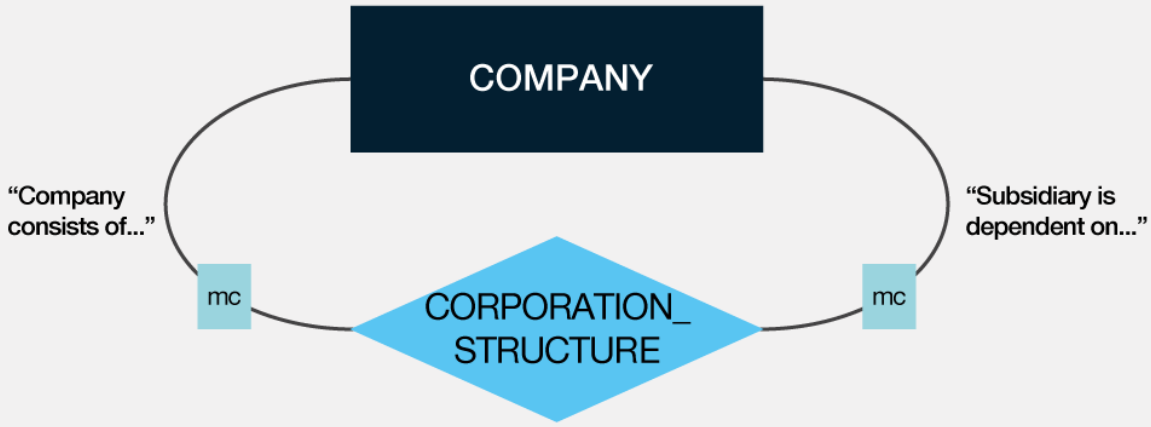
Aggregation

- Aggregation is a modeling concept that represents a “whole-part” relationship between entities.
- It allows us to show that one entity is composed of other entities.
- It is depicted using a diamond shape connecting the whole (the diamond) to its parts.
- Example: An “Order” entity is composed of “Order Items.”
- We call this a part of relationship

Example: Corporations

- Consider a set of companies, some of which are subsidiaries of others
- Company alpha is a subsidiary of company beta, who owns 80% of the shares
- beta is a subsidiary to gamma, who owns 30% of the shares.
- alpha owns 3% of gamma
- This is a network like relationship

Example: Corporations



Example for a corporate structure:

Type mc: "Each company may have multiple superordinate and/or subordinate companies."

Example: Items

Doesn't need to be a network - can also be a hierarchy

Example for an item list:
Type mc: **"Each item may consist of multiple sub-items."**
Type c: **"Each sub-item is dependent on exactly one superordinate item."**

A useful web tool

https://app.diagrams.net/

Implementation in the Relational Model

- The study of the relational model has led to theories that describe how to best convert diagrams into tables
- One of the major fields is in normal forms, which are used to discover and study dependencies within tables in order to avoid redundant information and anomalies

Implementation in the Relational Model

- An attribute in a table is redundant if individual values of this attribute can be omitted without a loss of information

Example of a Redundant Table

DEPARTMENT_EMPLOYEE

E#	Name	Street	City	D#	DepartmentName
E19	Stewart	E Main Street	Stow	D6	Accounting
E1	Murphy	Morris Road	Kent	D3	IT
E7	Howard	Lorain Avenue	Cleveland	D5	HR
E4	Bell	S Water Street	Kent	D6	Accounting

Redundancies are Bad News

- Insertion anomalies
- Deletion anomalies
- Update anomalies

Insertion Anomaly

- Insertion anomaly occurs when adding new data to the database is problematic.
- Example: Adding a new department like "Marketing" with no employees assigned.
- In such cases, there's no good way to add the department information without employees.

Deletion Anomalies

- Deletion anomalies happen when the removal of some data results in the inadvertent loss of other information.
- Example: A 100% employee turnover could result in the loss of all department name and ID information.
- Deletion anomalies can lead to data inconsistencies.

Update/Modification Anomalies

- Update or modification anomalies occur when data updates are inefficient or error-prone.
- Example: Changing "D3" to "Data Processing" might require updating multiple cells instead of just one.
- This can lead to data inconsistencies and increased complexity.

Normal Forms

- To address these problems, we have several types of normal forms in database design.
- Normalization is the process of organizing data in a database to reduce data anomalies and improve data integrity.
- By following normal forms, we can mitigate insertion, deletion, and update anomalies.

First Normal Form

- 1NF
- A table is in the first normal form when the domains of the attributes are atomic
- Each attribute gets its values from an unstructured domain, and there must be no sets, lists, or repressive groups within the individual attributes

Second Normal Form

- A table is 2NF when each nonkey attribute is fully functionally dependent on each key (plus the domains of attributes are atomic)
- An attribute B is functionally dependent on an attribute A if for each value of A, there is exactly one value of B ($A \rightarrow B$)
- A functional dependency of B on A requires that each value of A uniquely identifies ONE value of B
- For an identification key K and an attribute B to be in one table, there is a functional dependency when $K \rightarrow B$
- full functional dependencies for contacted keys requires (K1, K2) $\rightarrow$ B, with neither K1$\rightarrow$B or K2 $\rightarrow$ B
- If a table with a contacted key is not 2NF, it must be split into subtables

Second Normal Form

PROJECT_EMPLOYEE (unnormalized)

E#	Name	City	P#
E7	Howard	Cleveland	[P1, P9]
E1	Murphy	Kent	[P7, P11, P9]

PROJECT EMPLOYEE (first normal form)

E#	*P#*	Name	City
E7	P1	Howard	Cleveland
E7	P9	Howard	Cleveland
E1	P7	Murphy	Kent
E1	P11	Murphy	Kent
E1	P9	Murphy	Kent

What Happened?

- The table 1NF contains the keys E and P, we must test for full functional dependency
- (E,P) $\rightarrow$ Name? Yes
- (E,P) $\rightarrow$ Name? City
- BUT - E $\rightarrow$ City AND Name
- Attributes are functionally dependent on one part of our key
- So we broke it into two tables

Second Normal Form

- How do we know what to break into different tables?
- Attributes that are dependent on part of the key are transferred to a separate table, along with the key part
- The concatenated keys and other attributes remain
- Let's look again

Second Normal Form

PROJECT_EMPLOYEE (unnormalized)

E#	Name	City	P#
E7	Howard	Cleveland	[P1, P9]
E1	Murphy	Kent	[P7, P11, P9]

PROJECT EMPLOYEE (first normal form)

E#	*P#*	Name	City
E7	P1	Howard	Cleveland
E7	P9	Howard	Cleveland
E1	P7	Murphy	Kent
E1	P11	Murphy	Kent
E1	P9	Murphy	Kent

EMPLOYEE (2NF)

E#	Name	City
E7	Howard	Cleveland
E1	Murphy	Kent

INVOLVED (2NF)

E#	*P#*
E7	P1
E7	P9

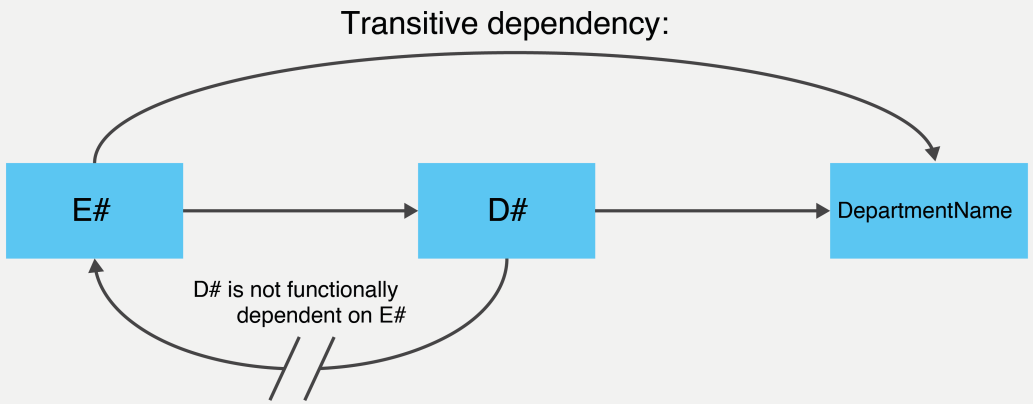
Third Normal Form

- A table is in the third normal form when it meets the criteria for 1NF and 2NF and no nonkey attribute is **transitively dependent** on any key attribute
- Transitive dependency is when an attribute is indirectly functionally dependent on another attribute
- Think the transitive property - transitive dependency is when an attribute is indirectly functionally dependent on an attribute
- An attribute C is transitively dependent on A if B is functionally dependent on A, C is functionally dependent on B, and A is not functionally dependent on B

Third Normal Form

DEPARTMENT_EMPLOYEE (in second normal form)

E#	Name	Street	City	D#	DepartmentName
E19	Stewart	E Main Street	Stow	D6	Accounting
E1	Murphy	Morris Road	Kent	D3	IT
E7	Howard	Lorain Avenue	Cleveland	D5	HR
E4	Bell	S Water Street	Kent	D6	Accounting



EMPLOYEE (in third normal form)

E#	Name	Street	City	D#_Sub
E19	Stewart	E Main Street	Stow	D6
E1	Murphy	Morris Road	Kent	D3
E7	Howard	Lorain Ave	Cleveland	D5
E4	Bell	S Water Street	Kent	D6

DEPARTMENT (3NF)

D#	DepartmentName
D3	IT
D5	HR
D6	Accounting

Third Normal Form

- DepartmentName is transitively dependent on E
- We split off DepartmentName with the other attribute involved in the transitive dependency - D
- We keep D in the employee table as a foreign key (and can add SUB) to make clear it is subordinate

Fourth Normal Form

- Multivalued dependencies
- Consider a table with A, B and C. An attribute C is multivaluedly dependent on an attribute A (written A $\rightarrow\!\!\rightarrow$ C) if any combination of a specific value A with an arbitrary value of B results in an identical set of values C.
- 4NF does not permit multiple true and distinct multivalued dependencies within one table.

Fourth Normal Form

METHOD (unnormalized)

Method	*Author*	*Term*
structogram	{Nassi, Shneiderman}	{sequence, iteration, branching}
data model	{Chen}	{entity set, relationship set}

METHOD (in third normal form)

Method	*Author*	*Term*
structogram	Nassi	sequence
structogram	Nassi	iteration
structogram	Nassi	branching
structogram	Shneiderman	sequence
structogram	Shneiderman	iteration
structogram	Shneiderman	branching
data model	Chen	entity set
data model	Chen	relationship set

METHODOLOGIST (4NF)

Methode	*Autor*
structogram	Nassi
structogram	Shneiderman
data model	Chen

METHODOLOGY (4NF)

Methode	*Begriff*
structogram	sequence
structogram	iteration
structogram	branching
data model	entity

Forth Normal Form

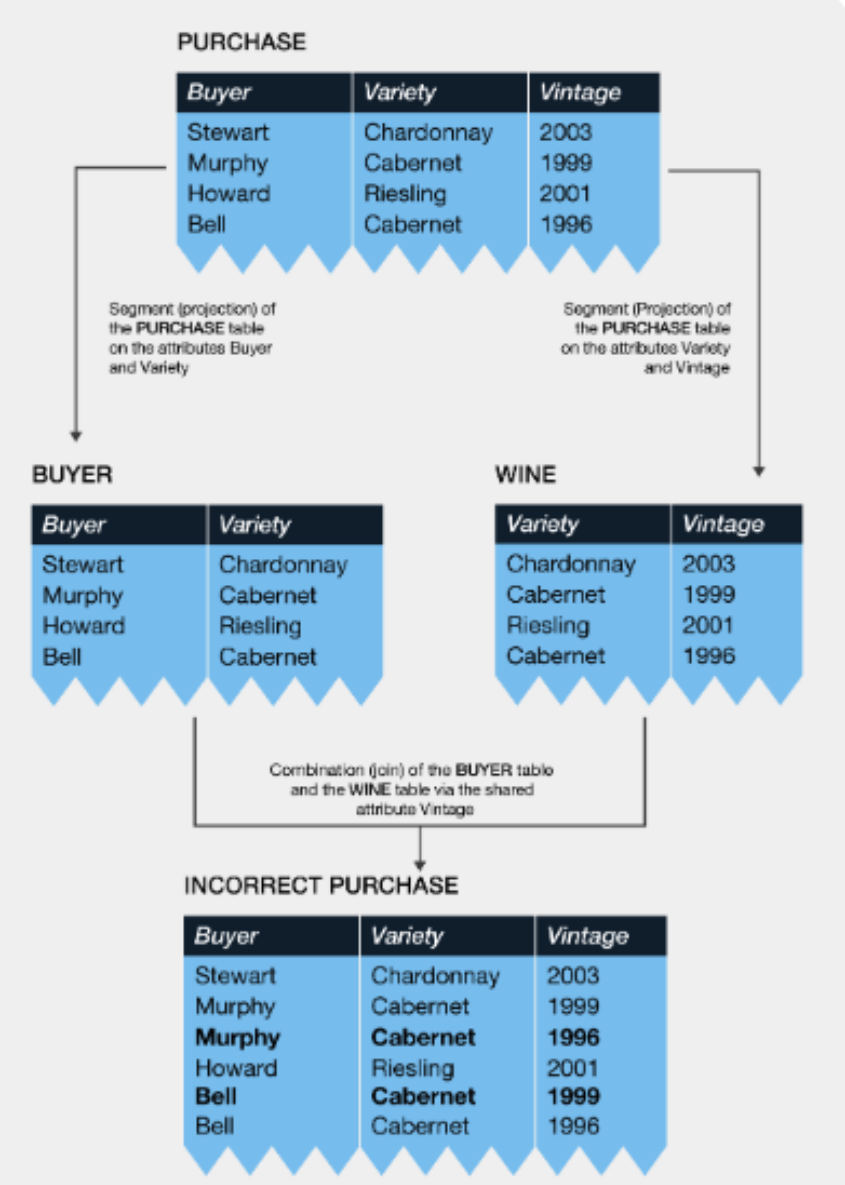
- Term is multivaluedly dependent on Method
- (Method $\rightarrow\!\!\rightarrow$ Term)
- This is because if we combine structogram with either Nassi OR Schneidermn, we receive the same set $\{\textit{sequence}, \textit{iteration}, \textit{branching}\}$
- Therefore, we break them off into two tables Methodologist(Method, Author) and Methodology(Method, Term)

Fifth Normal Form

- You've probably noticed...we are breaking up a lot of tables
- We can lose information this way
- We try to avoid this applying criteria of lossless table splitting, the full preservation of all information via the subtables

Fifth Normal Form

When splitting subtables goes wrong



Fifth Normal Form

- We used the purchase table to make a Buyer and Wine table (arbitrarily)
- Look again closely...there are two purchases that were never made when we combine our tables back again!

Fifth Normal Form

- A tale is in the fifth normal form if it can be arbitrarily split by taking some subset of the attributes and then reconstructed into the original table with join operators.
- This property is commonly called lossless join dependency.
- In short, a table 5NF when it meets lossless join dependency.
- Also called PJNF (project-join normal form)

Fifth Normal Form

- Consider a relation R(a, b, c)
- R has join dependency if the subtables $R_1(a, b)$ and $R_2(b, c)$ can be joined back into R by the shared attribute B

Fifth Normal Form

- We can achieve 5NF by creating three subtables
- Buyer(Buyer, Variety), Wine(Variety, Vintage), Preference(Buyer, Vintage)
- Combine Buyer and Wine by Variety, then combine that with Preference using vintage
- Here is what it looks like

Fifth Normal Form

PURCHASE (in fourth normal form)

Buyer	*Variety*	*Vintage*
Stewart	Chardonnay	2003
Murphy	Cabernet	1999
Howard	Riesling	2001
Bell	Cabernet	1996

BUYER (5NF)

Buyer	*Variety*
Stewart	Chardonnay
Murphy	Cabernet
Howard	Riesling
Bell	Cabernet

WINE (5NF)

Variety	*Vintage*
Chardonnay	2003
Cabernet	1999
Riesling	2001
Cabernet	1996

PREFERENCE (5NF)

Buyer	*Vintage*
Stewart	2003
Murphy	1999
Howard	2001
Bell	1996

Mapping Rules for Relational Databases

MAPPING RULES

Mapping Rules for Relational Databases

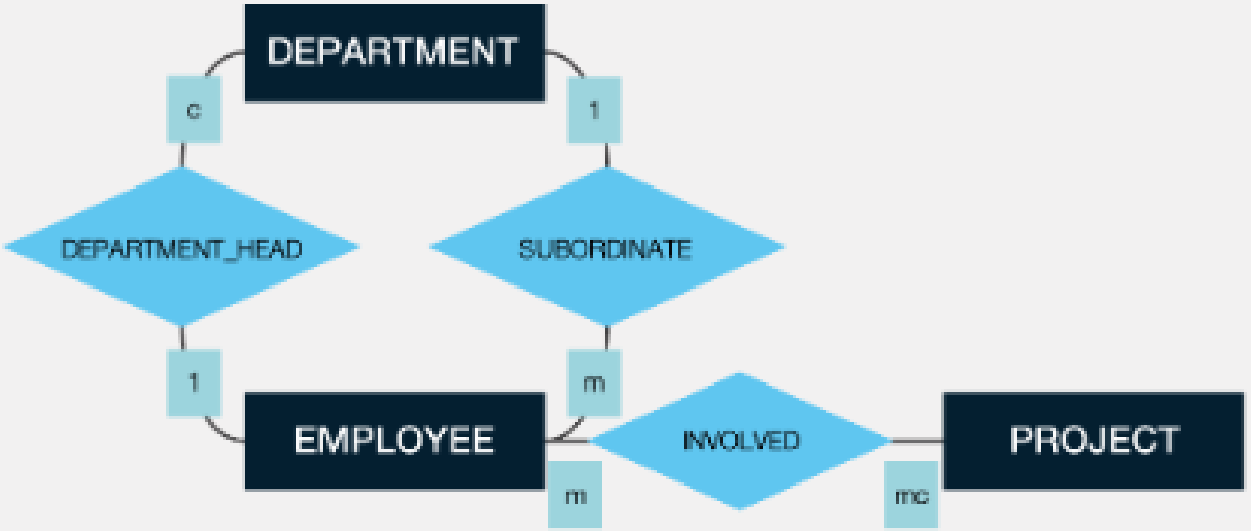
- Now, we discuss how to map the entity-relationship model onto a relational database schema.
- A relational database schema contains definitions of tables, the attributes, and the primary keys
- Integrity constraints set limits for the domains, the dependencies between tables, and for the actual data

Mapping Rules for Relational Databases

First two rules

- Rule 1: Each entity set has its own table, with a unique primary key. Remaining attributes (aside from the primary key) are attributes
- (We know at times many candidate keys can be minimal and unique)
- Rule 2: Each relationship set can be defined as a separate table, primary keys from corresponding entity sets must be included as foreign keys

Rule 1 and 2 Example



RULE R1

DEPARTMENT

D#	DepartmentName

PROJECT

P#	Content

EMPLOYEE

E#	Name	Street	City

RULE R2

DEPARTMENT_HEAD

D#	E#

SUBORDINATE

E#	D#

INVOLVED

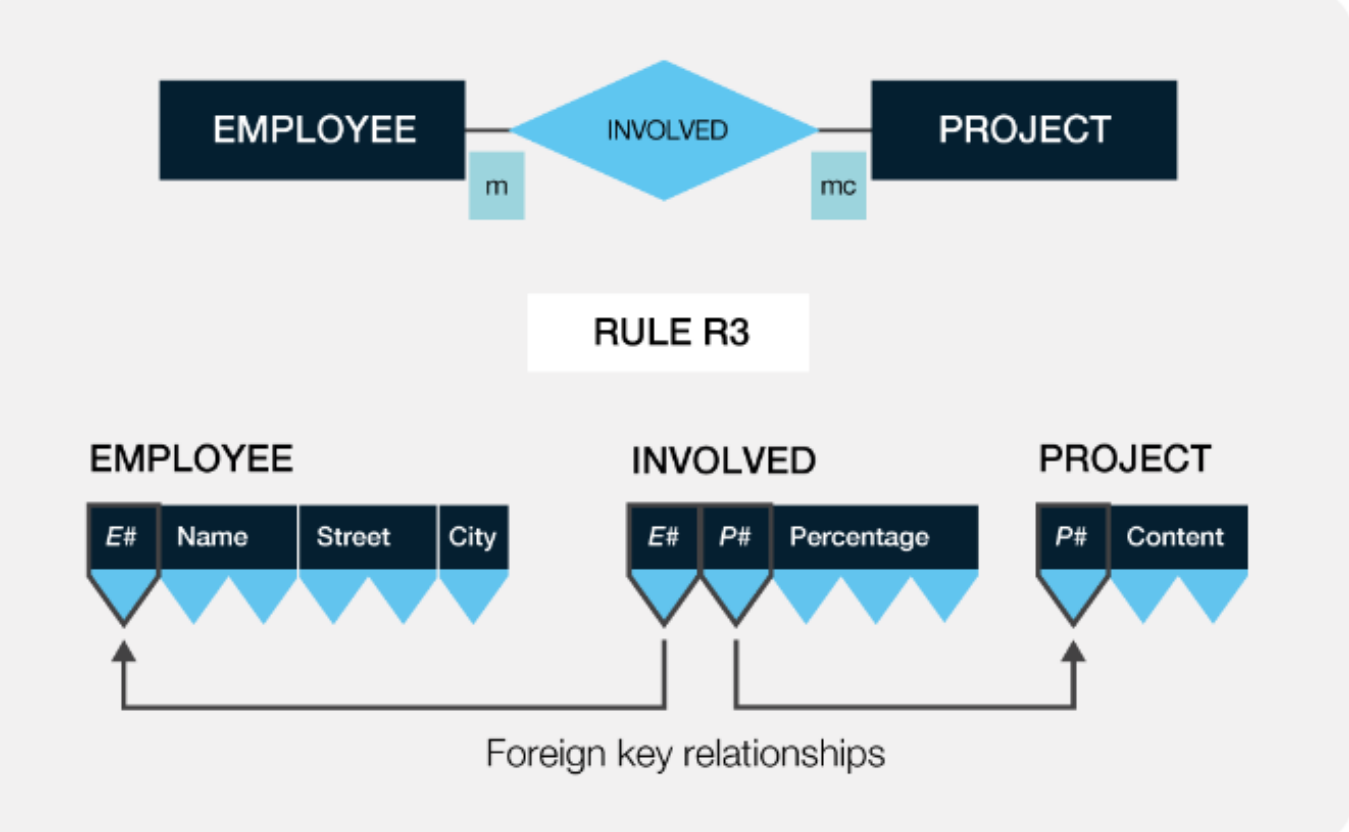
E#	P#	Percentage

Cardinality Based Mapping Rules

Based on the cardinality of relationships, we can define three mapping rules for representing relationship sets from the entity relationship model as tables in corresponding relational database schema.

- Rule 3 (network-like relationship sets) Every complex-complex relationship set must be defined as a separate table which contains at least the identification keys of the associated entity sets as foreign keys. the primary key of a relationship set table is either a concatenated key form the foreign keys or another candidate key. any further characteristics of the relationship set become attributes in the table.

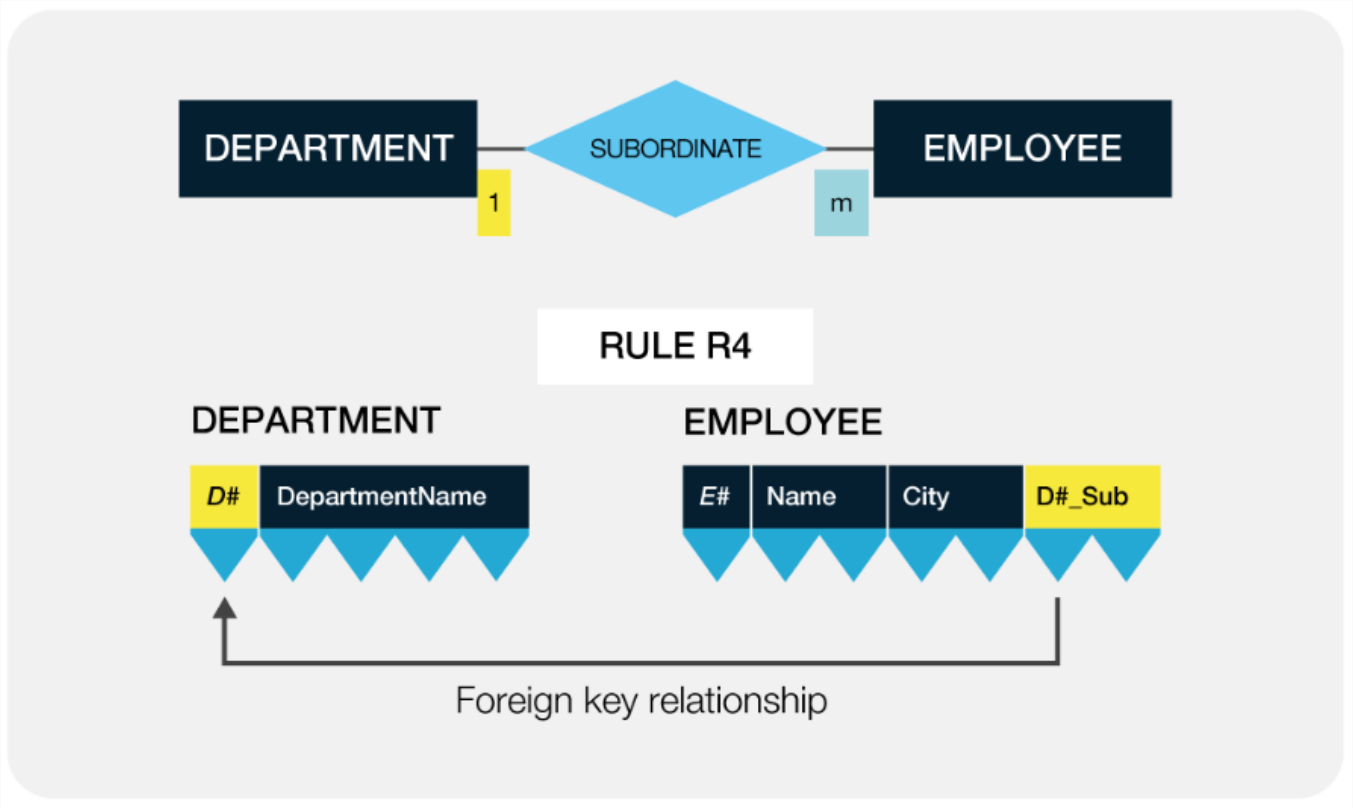
Rule 3 Example



Cardinality Based Mapping Rules

- Rule 4 (hierarchical relationship sets) Unique-complex relationship sets can be represented without a separate relationship set table by the tables of the two associated entity sets. The unique association (association type 1 or c) allows for the primary key of the reference table to simply be included in the referencing table as a foreign key with an appropriate role name.

Rule 4 Example

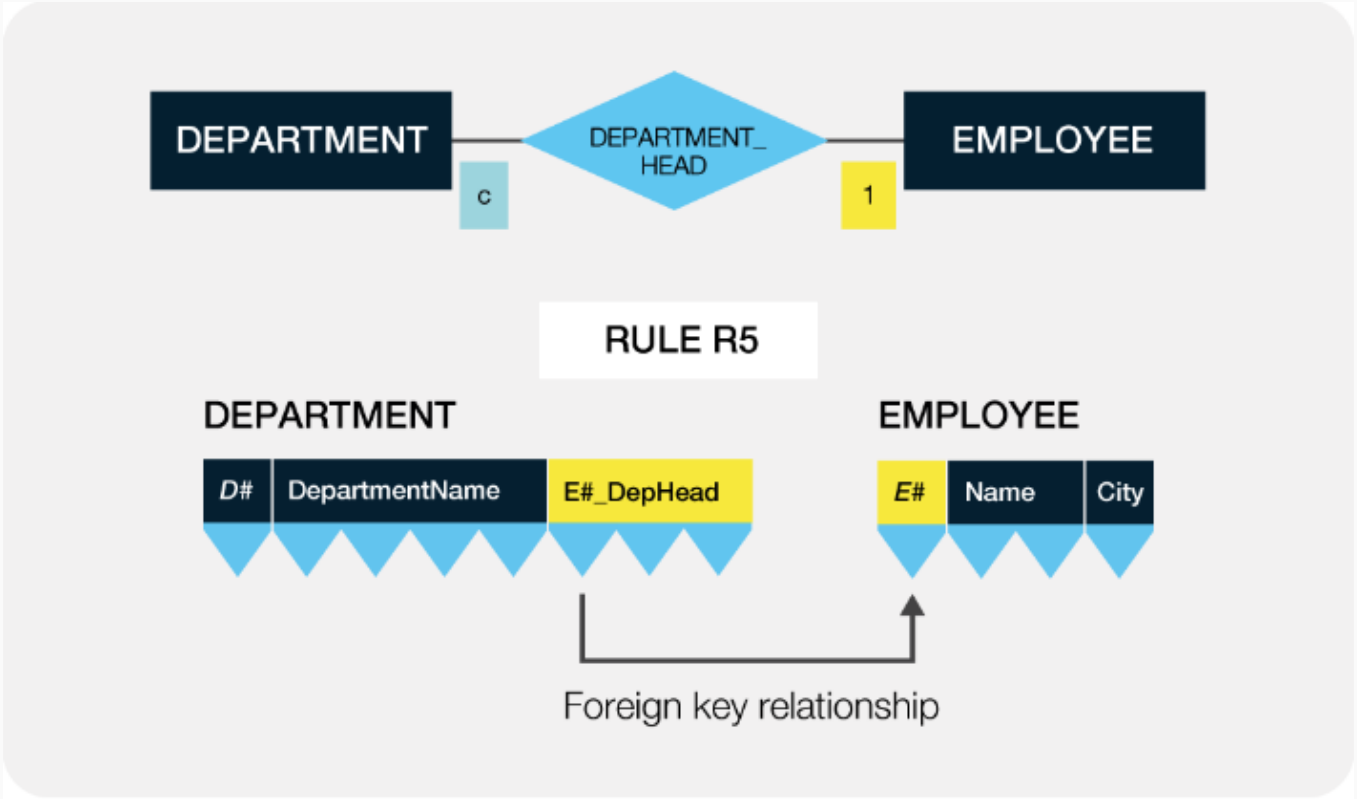


Cardinality Based Mapping Rules

Rule 5 (unique-unique relationship sets) - unique-unique relationship sets can be represented without a separate table by the tables of the two associated entity sets. Again, an identification key from the referenced table can be included in the referencing table along with the role name

- It is best to take the foreign key from the type 1 association so the foreign key with its role name can be included in each tuple of the referencing table
- Avoids having a bunch of null values

Rule 5 Example



Cardinality Based Mapping Rules

- How can you keep this straight?
- Once you identify the cardinality, identify the nature of the relationship
- From there, you can implement the rules based on the matrix

Normalization Example

- Great normalization example from our TA Pavan!

Database Schema Normalization

Database schema normalization is the process of organizing database tables to reduce redundancy and dependency, leading to a more efficient and maintainable database. It's done in a series of steps called normal forms.

Initial Schema

Book (BookID, Title, Author, ISBN, Publisher, Publication Year, Category)

Member (MemberID, First Name, Last Name, Email, Phone, Address, Membership Date)

Loan (LoanID, MemberID, BookID, Due Date)

In this schema, there is redundancy, and dependencies exist.

1st Normal Form (1NF)

To achieve 1NF, ensure that each column contains atomic (indivisible) values:

Book (BookID, Title, ISBN, Publisher, Publication Year)

Author (AuthorID, Author Name)

Category (CategoryID, Category Name)

Now, each column contains atomic values.

2nd Normal Form (2NF)

To achieve 2NF, make sure that non-key attributes are fully functionally dependent on the primary key:

Book (BookID, Title, ISBN, Publication Year)

Author (AuthorID, Author Name)

Category (CategoryID, Category Name)

Publisher (PublisherID, Publisher Name)

Non-key attributes are fully functionally dependent.

3rd Normal Form (3NF)

In 3NF, eliminate transitive dependencies:

Book (BookID, Title, ISBN, Publication Year)

Author (AuthorID, Author Name)

Category (CategoryID, Category Name)

Publisher (PublisherID, Publisher Name)

Member (MemberID, First Name, Last Name, Email, Phone, AddressID, Membership Date)

Address (AddressID, Street Address, City, State, Zip Code)

Now, 'Address' is in its table, and 'Member' refers to 'Address' via 'AddressID.'

4th Normal Form (4NF)

To achieve 4NF, consider multi-valued dependencies, such as books with multiple authors:

Books (BookID, Title, ISBN, Publisher, Publication Year)

Authors (AuthorID, Author Name)

Book_Author_Relationship (BookID, AuthorID)

'Authors' are in a separate table with their own primary key, eliminating the multi-valued dependency.

5th Normal Form (5NF)

Consider join dependencies, e.g., books divided into volumes:

Books (BookID, Title, ISBN, Publisher, Publication Year)

Volumes (VolumeID, Volume Title, ISBN, Volume Publisher, Volume Publication Year)

Book_Volume_Relationship (BookID, VolumeID)

'Books' and 'Volumes' are in separate tables, and their relationship is represented by the 'Book_Volume_Relationship' table, eliminating join dependencies.

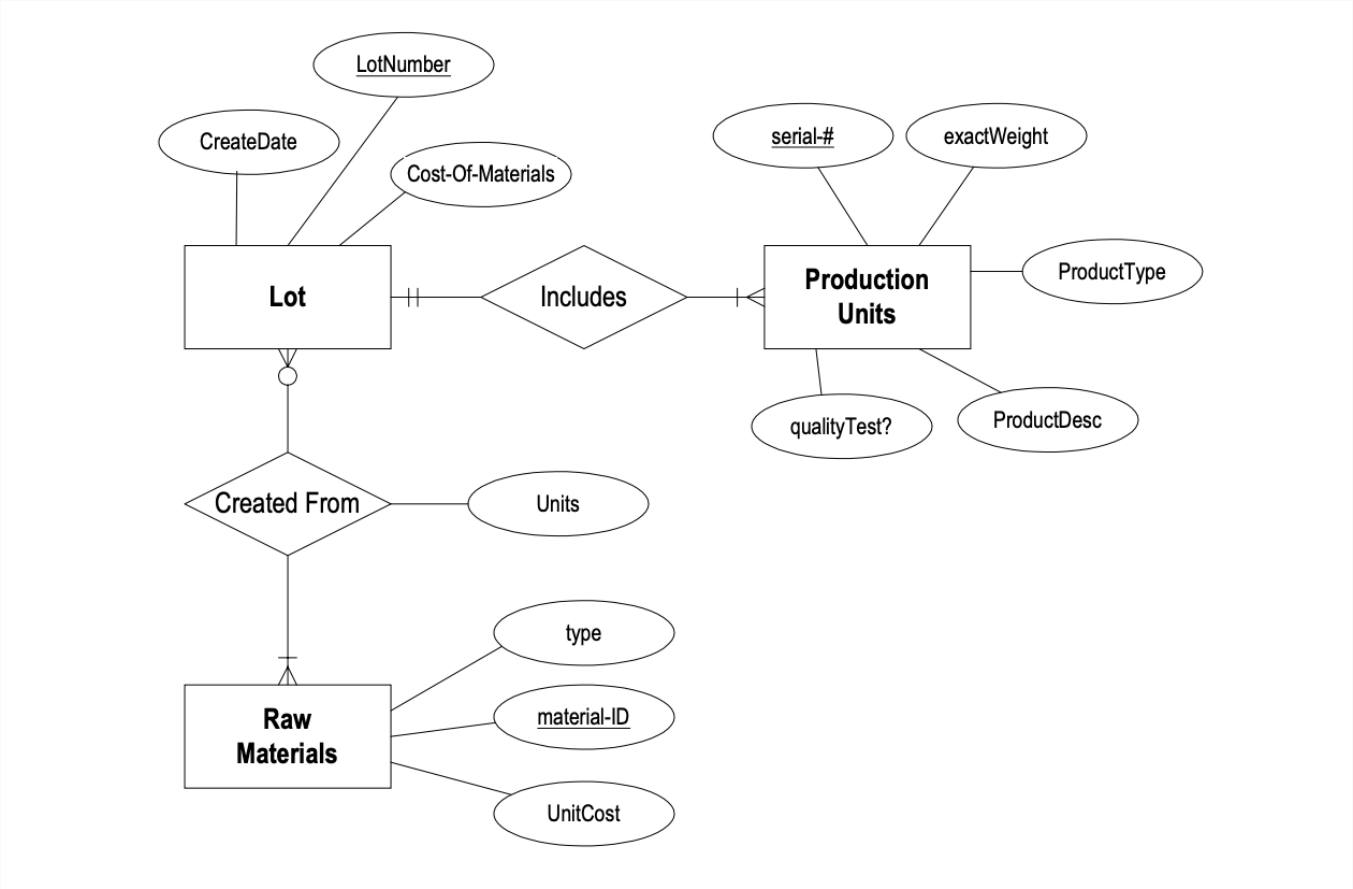
Conclusion

Achieving the 5th Normal Form results in a highly normalized structure, eliminating various anomalies and dependencies. The database is efficient and maintainable. Further normalization depends on specific requirements.

Let's Practice

- Let's convert an E-R diagram into a relational schema
- Here are our steps:
- 1. Identify the cardinality by looking at the crows foot notation
- 2. Identify what mapping rules need to be applied based on that cardinality
- 3. Based on 1 and 2, decide what tables must exist, what primary keys should be used, and what foreign keys should be used

Let's Look at a Diagram



First, the Basics

- What are our entities?
- What are our relationships?
- What are our attributes?

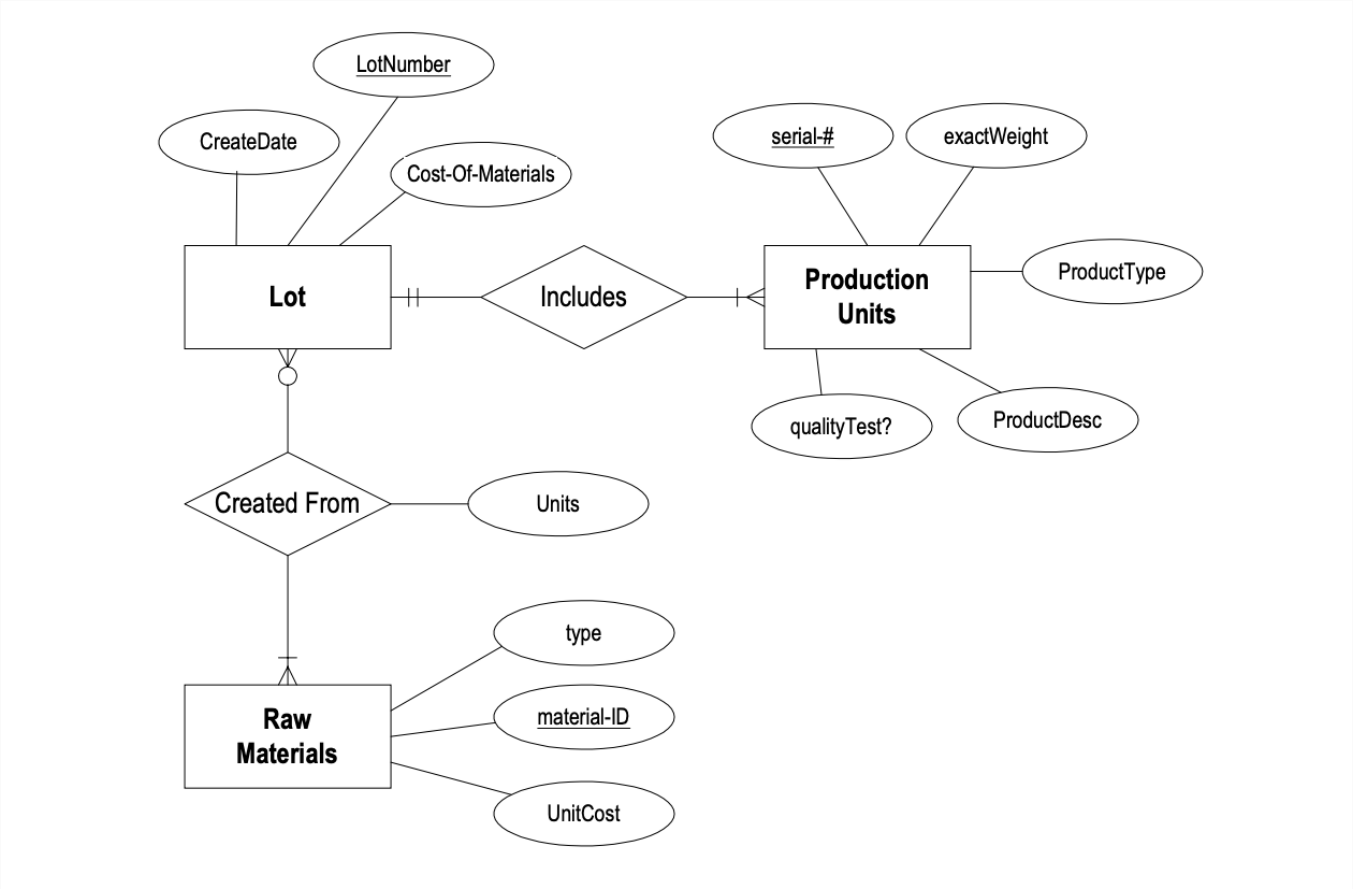
Second, the Associations

- What are our association types?
- What are the cardinalities?
- What types of relationships do we have?

Third, the Rules

- Apply the rules for relational mapping based on our answers from parts 1 and parts 2

Let's Look Again



Step 1

- Entities?
- Relationships?
- Attributes?

Step 1

- Entities: Production Units, Lot, Raw Materials
- Relationships: Created From, Includes
- Attributes: CreateDate, LotNumber, CostMaterial, SerialNumber, exactWeight,...you get it

Step 2

- What are our association types?
- What are the cardinalities?
- What types of relationships do we have?

Step 2

What are our association types:

Step 2

What are our association types:

- Lot to Production Units: m
- Production units to Lot: 1
- Lot to Raw Materials: m
- Raw Materials to Lot: mc

Step 2

What are the cardinalities:

- (Lot, Production Units) = (m, 1) - unique-complex
- (Lot, Raw Materials) = (m, mc) - complex-complex

Apply our Rules

- We have unique-complex, so apply Rule 4 (Primary key of reference table goes to referencing table)
- We have complex-complex, so apply Rule 3 (relationship set must be defined as a separate table which contains at least the identification keys of the associated entity sets as foreign keys)

Apply our Rules

- We can put the primary key from the reference table (Lot) in the referencing table (Production Units)
- We break out the Created From relation into a new Table, include the primary keys from both entities

Write out the Schema

- Production Units(Serial, exactWeight, Type, Desc, LotNumber)
- Lot(LotNumber, CreateDate, Cost)
- Created From(LotNumber, MaterialID, Units)
- Raw Materials(MaterialID, Type, UnitCost)

Structural Integrity Constraints

Structural Integrity Constraints

- Integrity or consistency of data means that stored data does not contradict itself
- A database has integrity/consistency if the stored data is free of errors and currently represents the anticipated informational value
- Data integrity is impaired if there are ambiguities or conflicting records
- Ex: a consistent EMPLOYEE table requires that the names of employees, streets, and cities really exists and are correctly assigned

Structural Integrity Constraints

- Uniqueness constraint : Each table has an identification key (attribute or combination of attributes) that uniquely identifies each tuple within the table. Need a primary key
- Domain constraint: the attributes in a table can only take on values from a predefined domain (supported by enumeration types)
- Referential integrity constraint: Each value of a foreign key must actually exist as a key value in the reference table