Database Design

7-1
Arcs

ORACLE ACADEMY

What is a Constraint?

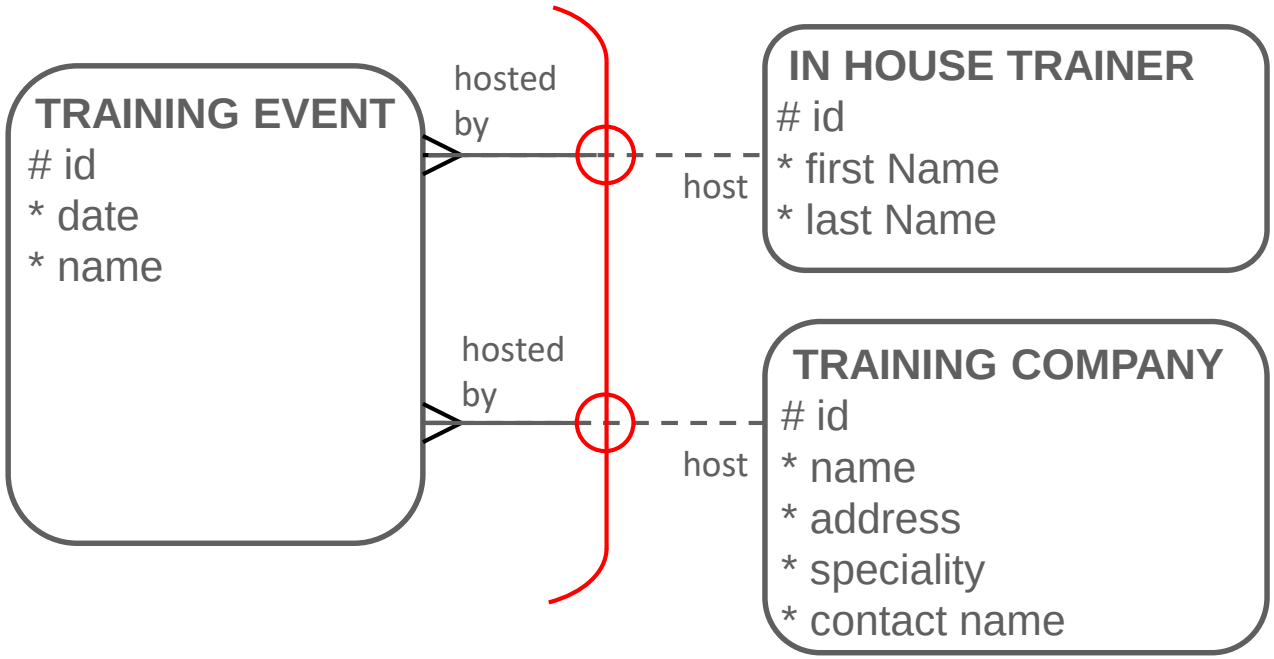
- Every business has restrictions on which attribute values and which relationships are allowed.
- These restrictions are called constraints.
- They may refer to a single attribute of an entity, or to relationships between entities.
- We already know about several kinds of constraints; for example, every EMPLOYEE must work in one and only one DEPARTMENT.
- In this lesson, we will see another kind of constraint—an exclusive OR constraint.

Exclusive OR Relationship

- Mutually exclusive relationships sometimes exist between entities and are also known as Exclusive OR Relationships
- An Exclusive OR relationship is a relationship between one entity and two (or more) other entities where only one of the relationships can exist at a time
- In ERDs, we model this type of relationship with an Arc

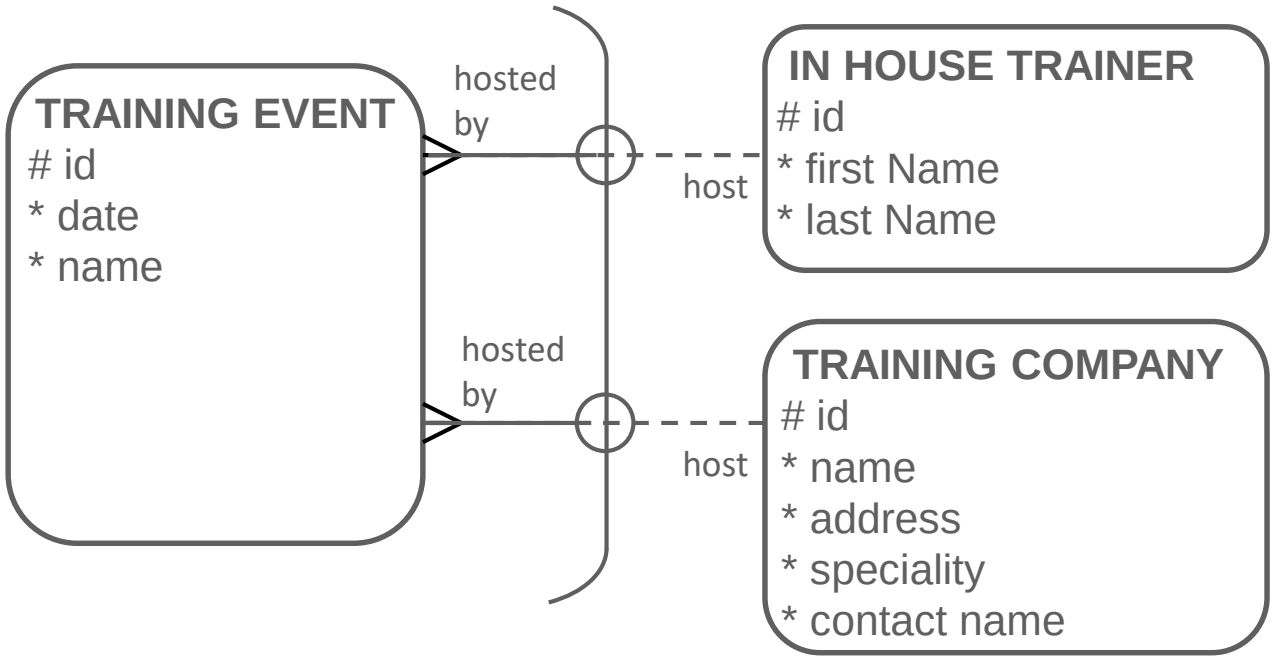
Exclusive OR Relationship

- For example: a TRAINING EVENT can be hosted by either an IN HOUSE TRAINER or an external TRAINING COMPANY.



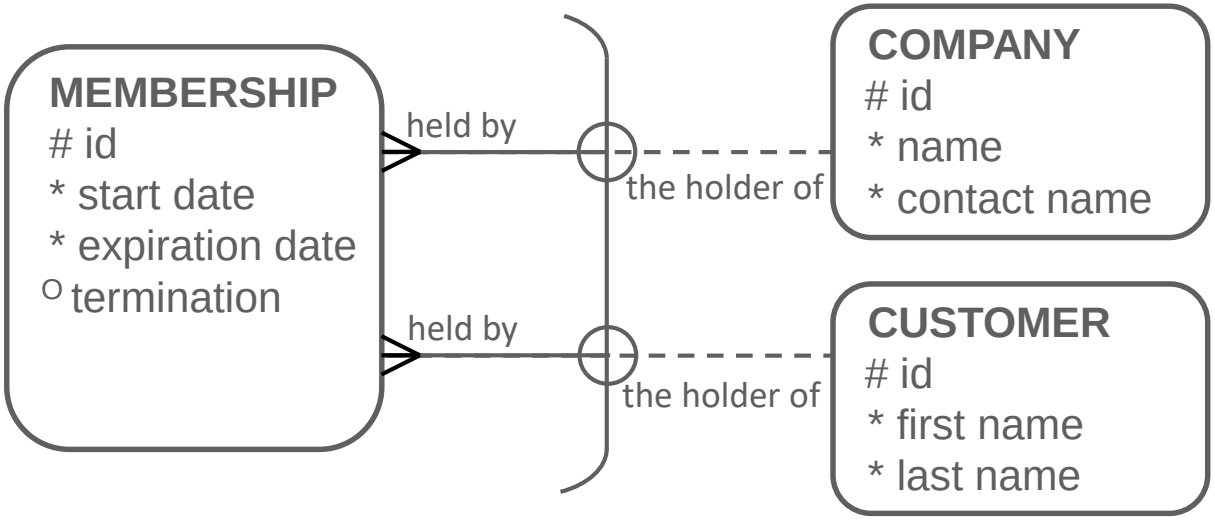
Exclusive OR Relationship

- Each TRAINING EVENT must be hosted by one and only one IN HOUSE TRAINER OR one and only one TRAINING COMPANY.



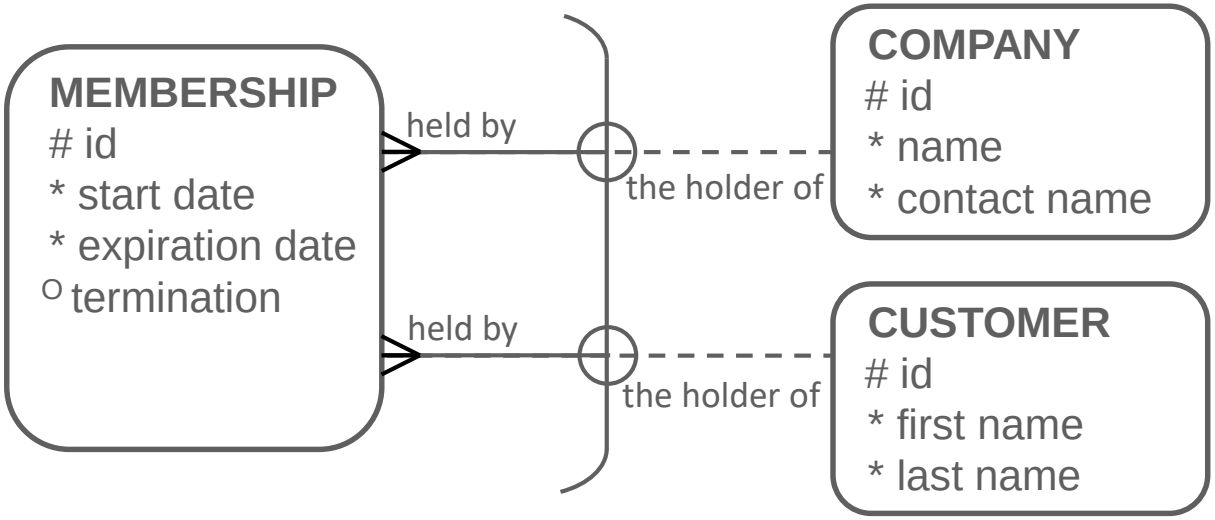
Representing Exclusive OR Relationships in the ERD

- Arcs are a way to represent mutually exclusive relationships in the ERD.



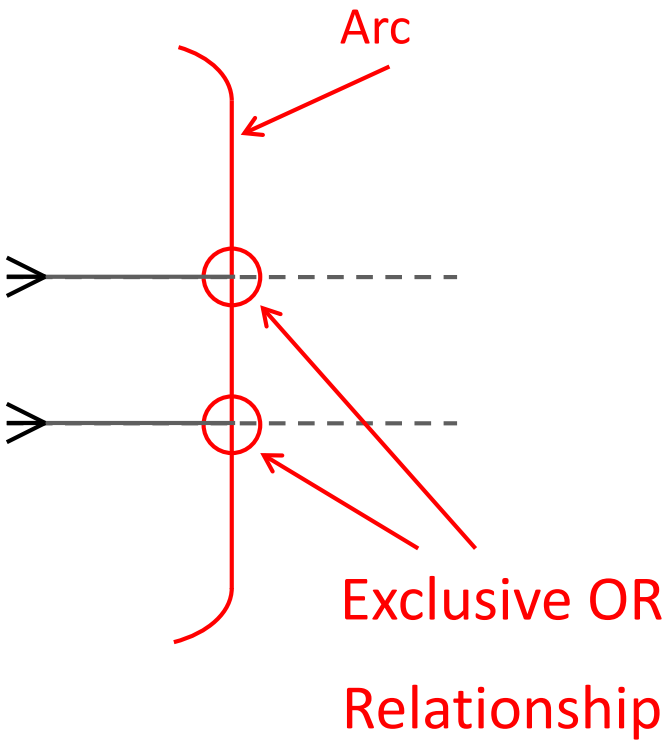
Representing Exclusive OR Relationships in the ERD

- This arc represents the exclusive OR relationship - each MEMBERSHIP must be held by one COMPANY or must be held by one CUSTOMER, but not both.



Representing Exclusive OR Relationships in the ERD

- An arc is represented on an ERD as a solid line with curved ends.
- A circle is drawn on the arc for every relationship that is part of the arc.



Arcs

- An arc always belongs to one entity.
- Arcs can include more than two relationships.
- Not all relationships of an entity need to be included in an arc.
- An entity may have several arcs.
- An arc should always consist of relationships of the same optionality.

Arcs

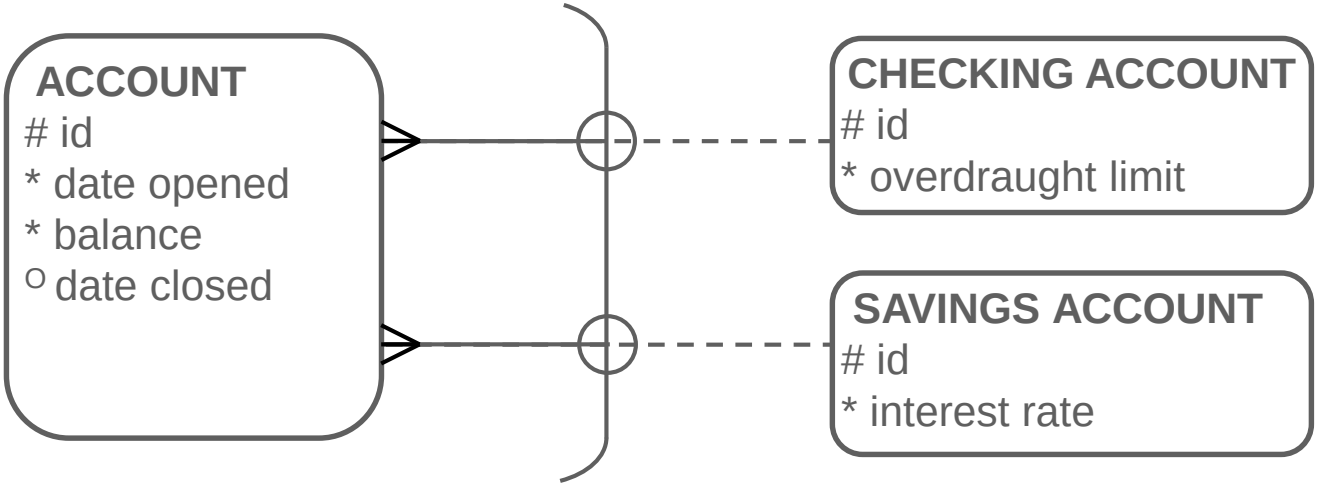
- All relationships in an arc must be mandatory or all must be optional.
- Relationships in an arc may be of different cardinality, although this is rare.

Arcs, Supertypes, and Subtypes

- Arcs and Super/subtypes both model mutual exclusiveness.
- Certain situations are best modeled as an arc, and others as supertype and subtypes.

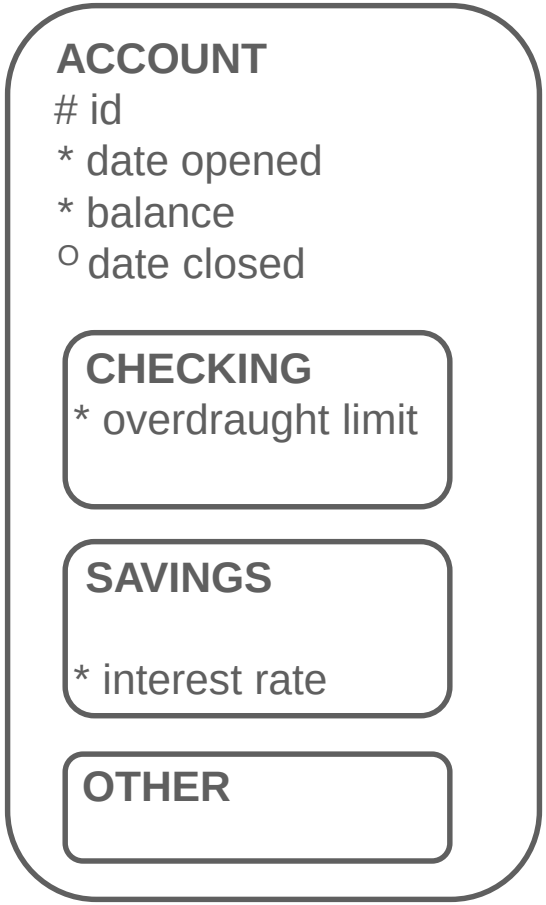
Arcs, Supertypes, and Subtypes

- Example 1: CHECKING ACCOUNT and SAVINGS ACCOUNT are "types" of ACCOUNT.



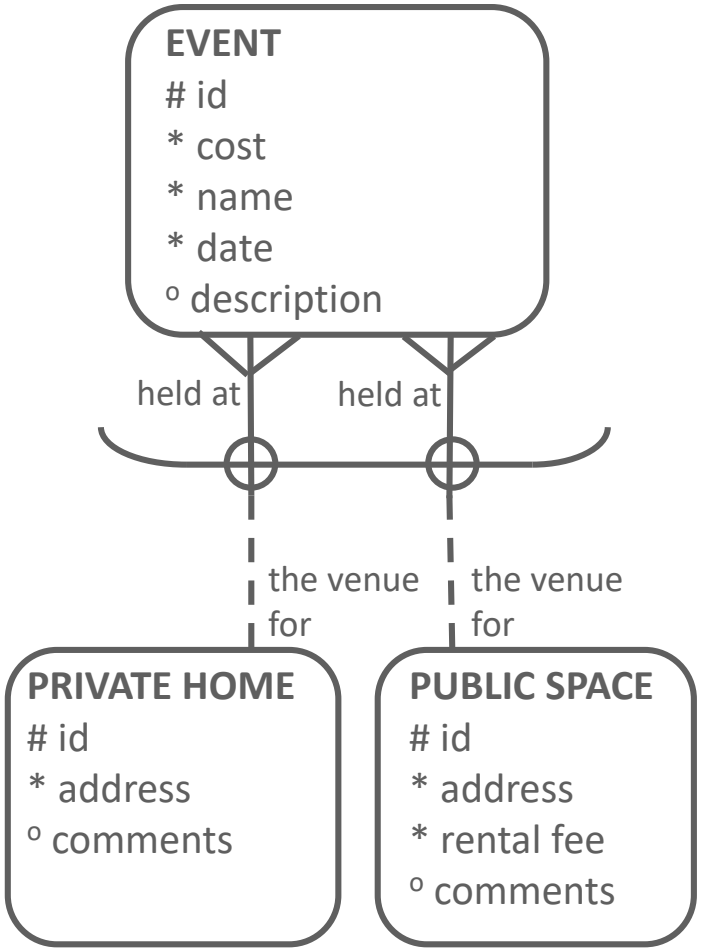
Arcs, Supertypes, and Subtypes

- This should be modeled as supertype and subtypes



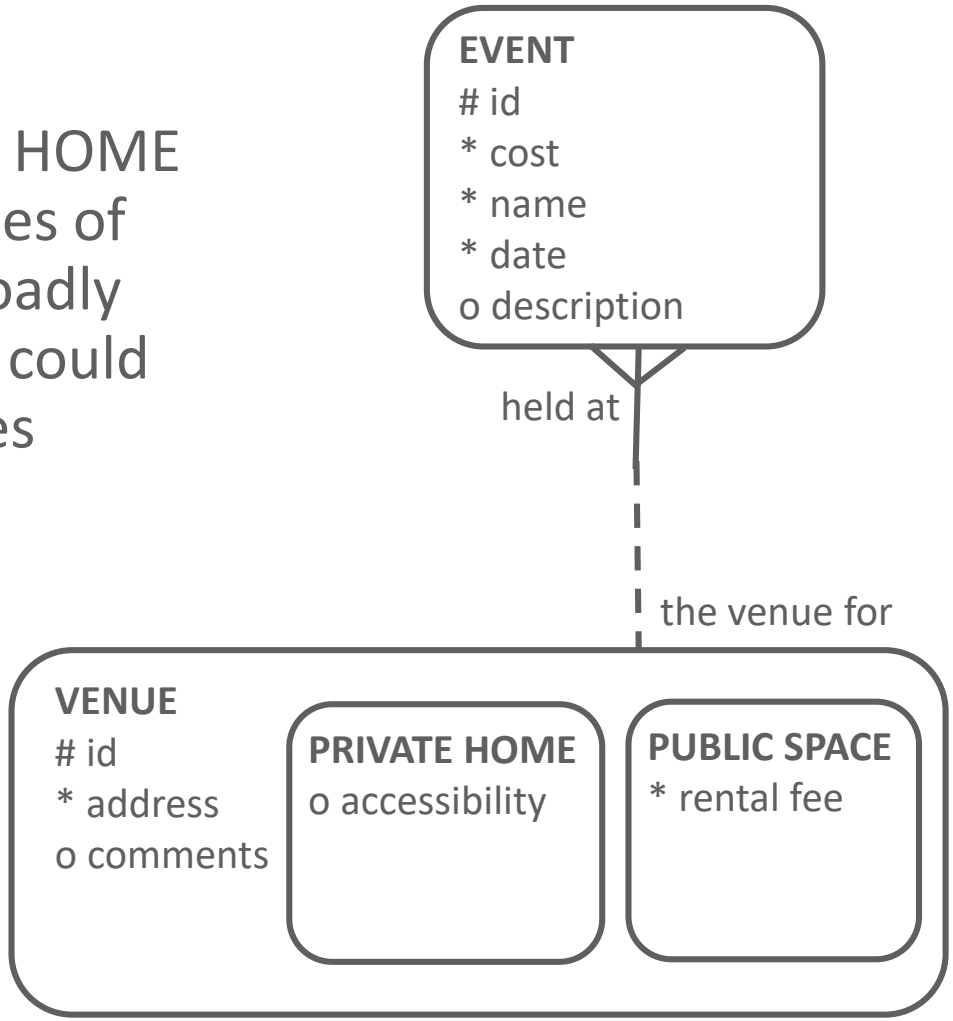
Arcs, Supertypes, and Subtypes

- Example 2: An EVENT can be held at either a PRIVATE HOME or a PUBLIC SPACE.
- If the entities that are related through the arc are similar, there may be a case for creating a super/subtype without an arc.



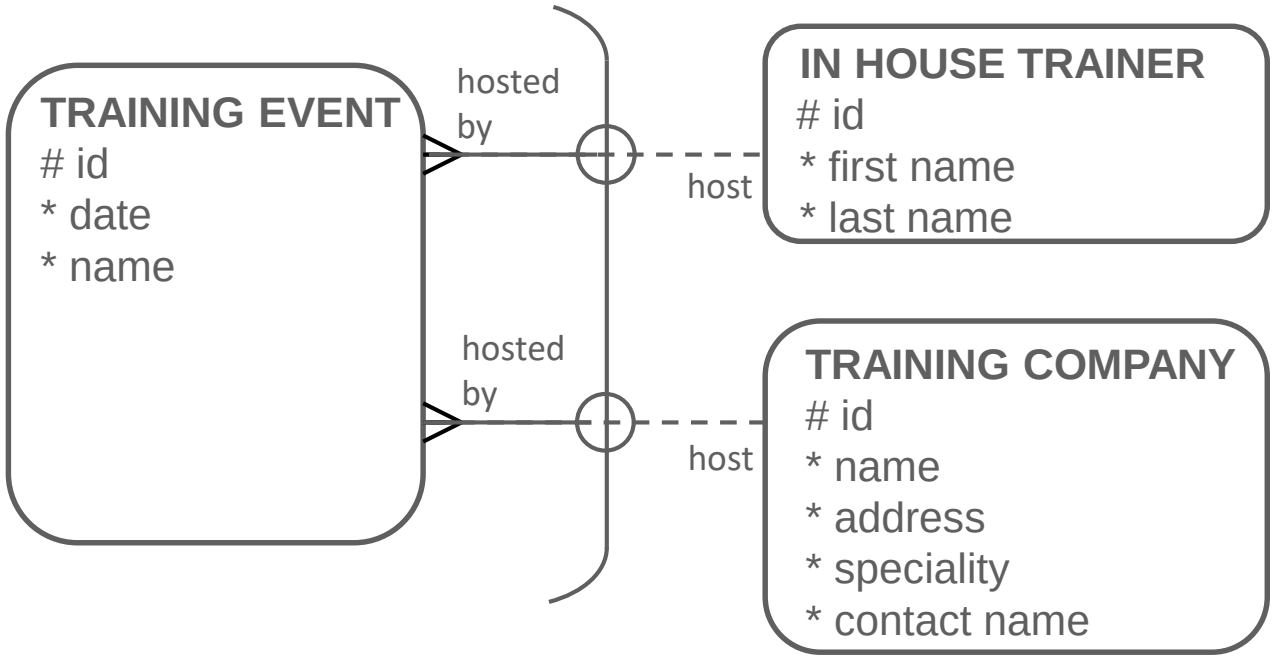
Arcs, Supertypes, and Subtypes

- In this case, both PRIVATE HOME and PUBLIC SPACE are types of VENUE, and they have broadly similar attributes, so they could be supertype and subtypes



Arcs, Supertypes, and Subtypes

- Example 3: IN HOUSE TRAINER and TRAINING COMPANY are NOT types of TRAINING EVENT, and they do not share common attributes. This is best to model with an arc.



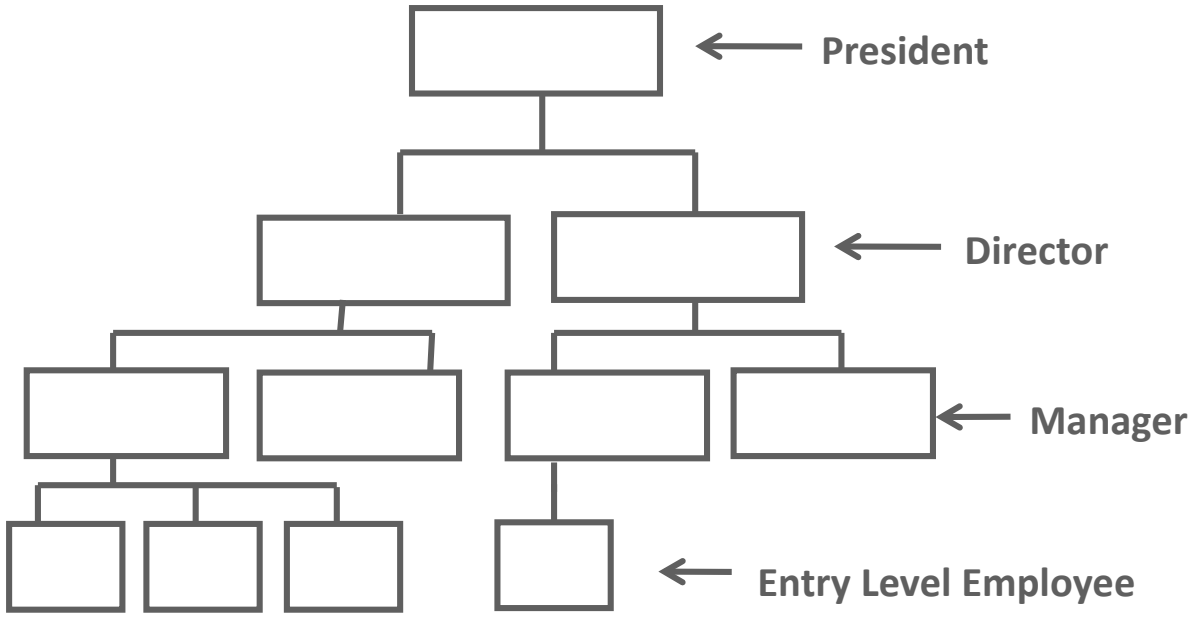
Database Design

7-2
Hierarchies and Recursive Relationships

ORACLE ACADEMY

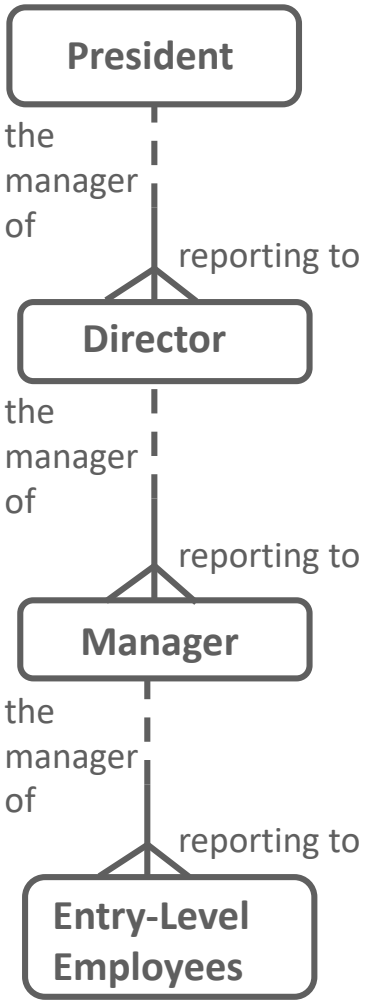
Relationships in an Organizational Chart

- An Organization's reporting hierarchy can be represented by this organizational chart .



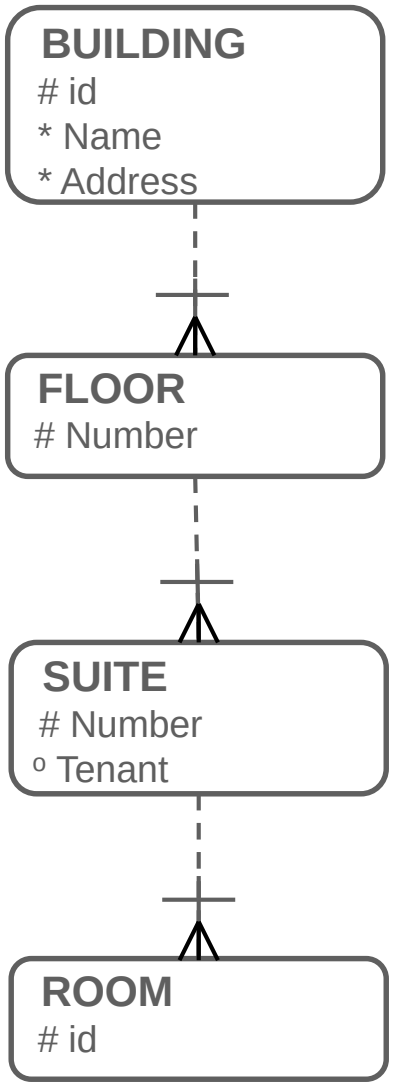
Relationships in an Organizational Chart

- An organizational chart can be represented by this data model.
- We create an entity for each level, with a relationship to the next level.
- What are the UIDs for each entity?



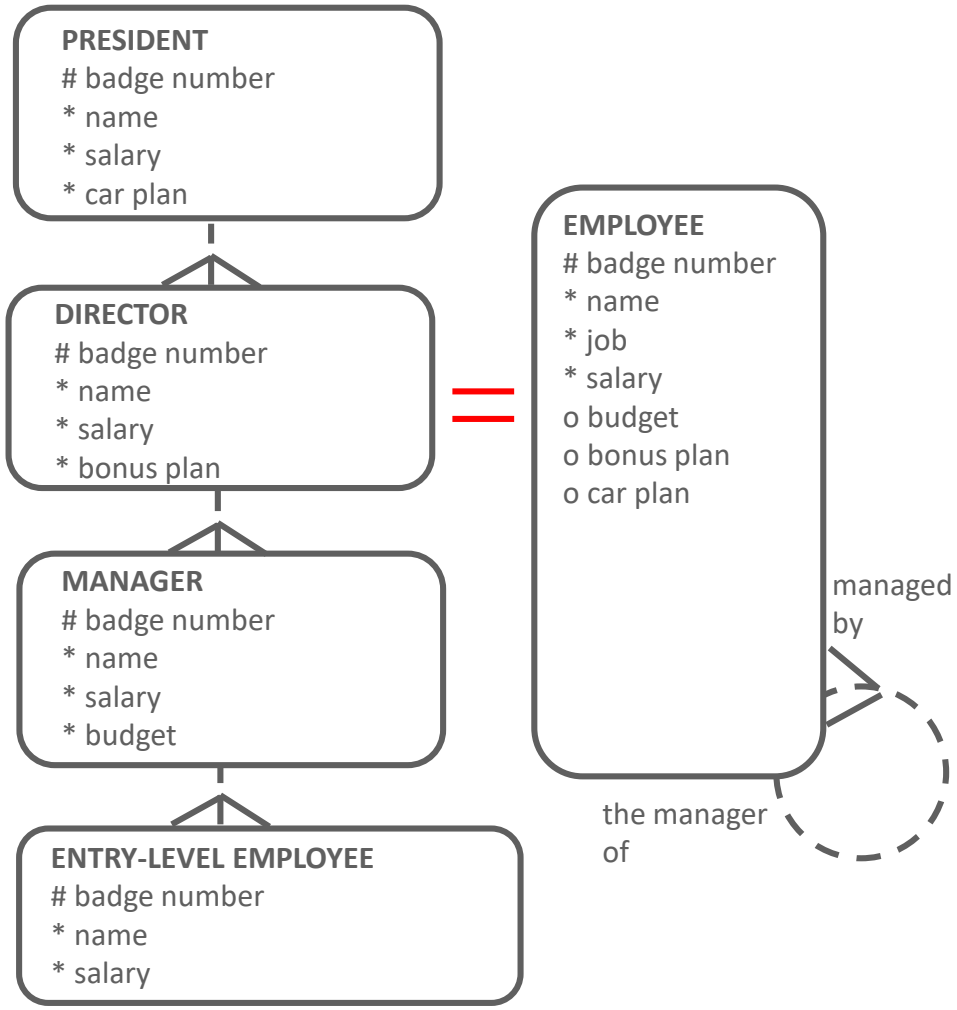
Another Relationship Example

- Notice the barred relationships.
- Here you have a case of the cascading UIDs:
  - the UID of FLOOR is the combination of FLOOR number and the BUILDING id
  - the UID of SUITE is the combination of SUITE number and the FLOOR number and the BUILDING id
  - the UID of ROOM is the combination of ROOM id and SUITE number and FLOOR number and the BUILDING id



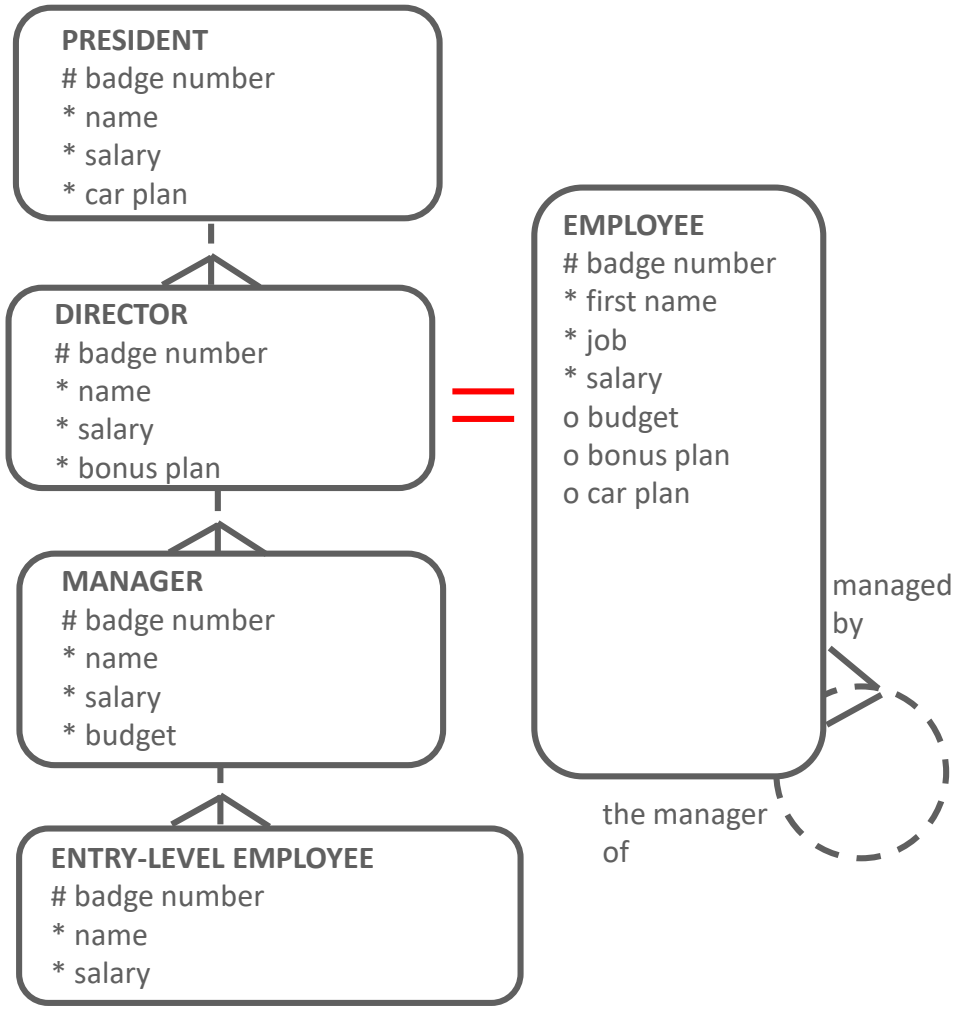
Hierarchy Versus Recursive Relationship

- Both of these models represent all employees.
- The one on the left is a hierarchical structure.
- The one on the right uses a recursive relationship.



Hierarchy Versus Recursive Relationship

- A relationship cannot be both hierarchical and recursive at the same time.
- Which one do you think is better?



Hierarchy Versus Recursive Relationship

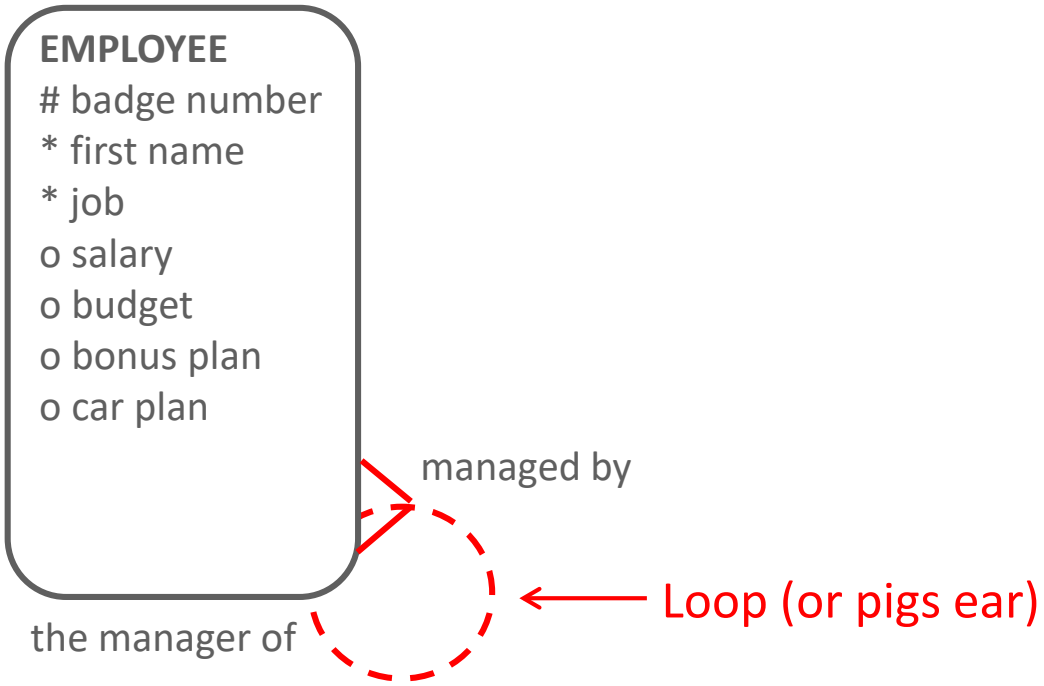
- Hierarchical: Hierarchical structures are more explicit and are easier for most people to understand because they are very similar to an organizational chart.
- Each entity can have its own mandatory attributes and relationships, if the business requires this (instead of all optional attributes and relationships, as you would have in a recursive).
- In this way, your data model truly reflects the business rules.

Hierarchy Versus Recursive Relationship

- Recursive: Recursive relationships tend to be simpler because you are using only one entity.
- Your diagram will be less “busy.”
- However, they are less specific – you cannot have mandatory attributes or relationships unless they are mandatory in all instances of the entity.

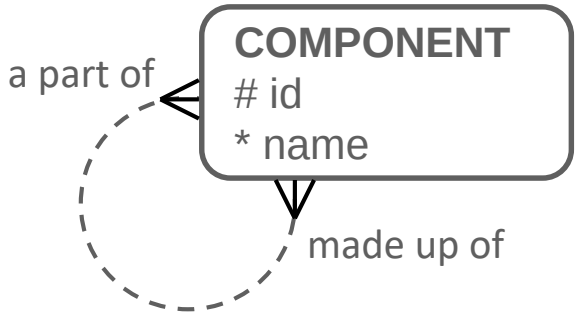
Drawing Convention

- The ERD convention to show a recursive relationship is drawn as a loop, also known as a “pig’s ear”.



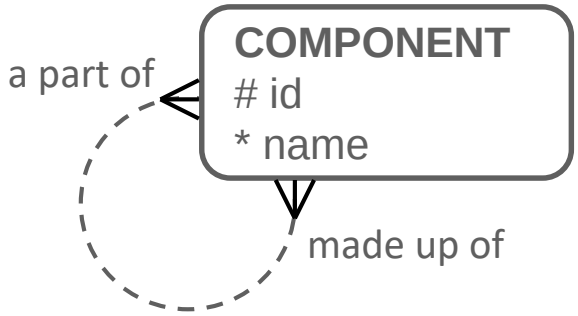
Automobile Manufacturing Business Scenario

- For an automobile manufacturing organization, consider all elementary parts, subassemblies, assemblies, and products as instances of an entity called COMPONENT.
- The model can be created as a simple recursive relationship.



Automobile Manufacturing Business Scenario

- Model Bill of Materials data as a many-to-many recursive relationship:
  - Each COMPONENT may be a part of one or more COMPONENTS.
  - Each COMPONENT may be made up of one or more COMPONENTS.



Database Design

8-1
Modeling Historical Data

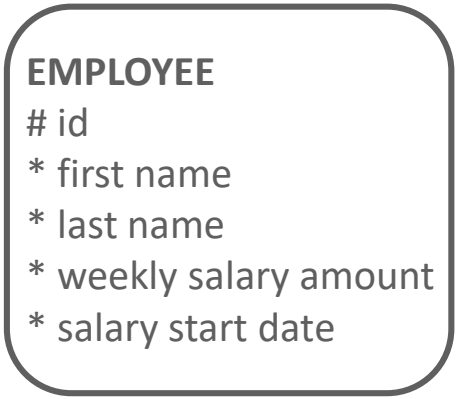
ORACLE ACADEMY

Model Data Over Time

- When is it necessary to model data over time?
- Ask your client:
  - Is an audit trail required?
  - Can attribute values change over time?
  - Can relationships change over time?
  - Do you need to produce reports on older data?
  - Do you need to keep previous versions of the data? If so, for how long?

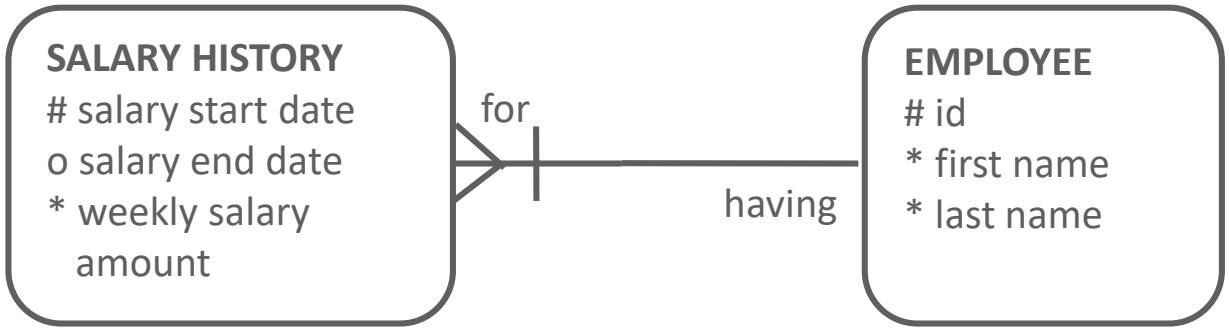
Data Over Time Example

- An organization needs to keep data about employees' salaries.
- All employees are paid weekly.
- Initially, the following EMPLOYEE entity was modeled.
- Additional requirements now specify that the organization needs to keep a historical record of how and when employees' salaries have changed during their employment.



Model Salary Changes

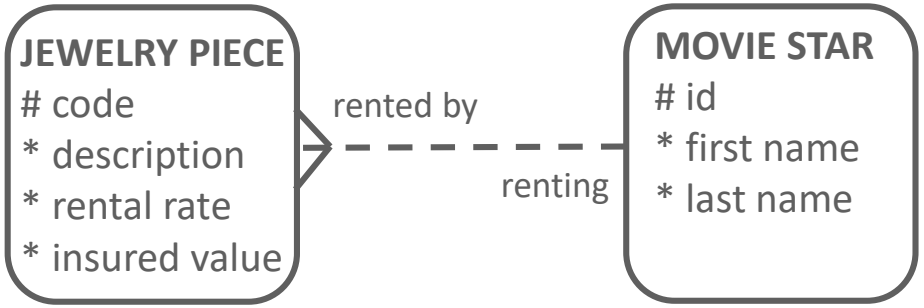
- To model salary changes over time, add a SALARY HISTORY entity.



- The UID of the SALARY HISTORY entity is the related EMPLOYEE id and the salary start date.

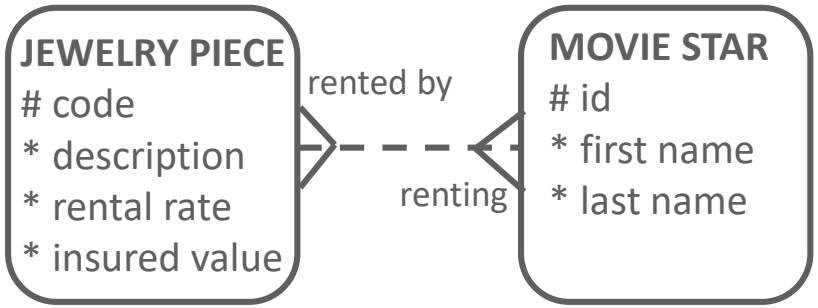
Model Rental Over Time

- A jewelry store rents pieces (necklaces, bracelets and so on) to movie stars for special occasions, such as award ceremonies or movie premieres.
- They would like to track the rental history of a jewelry piece.
- The following ER model will only track the current renter of a piece of jewelry.
- How would you revise the relationship to track history?

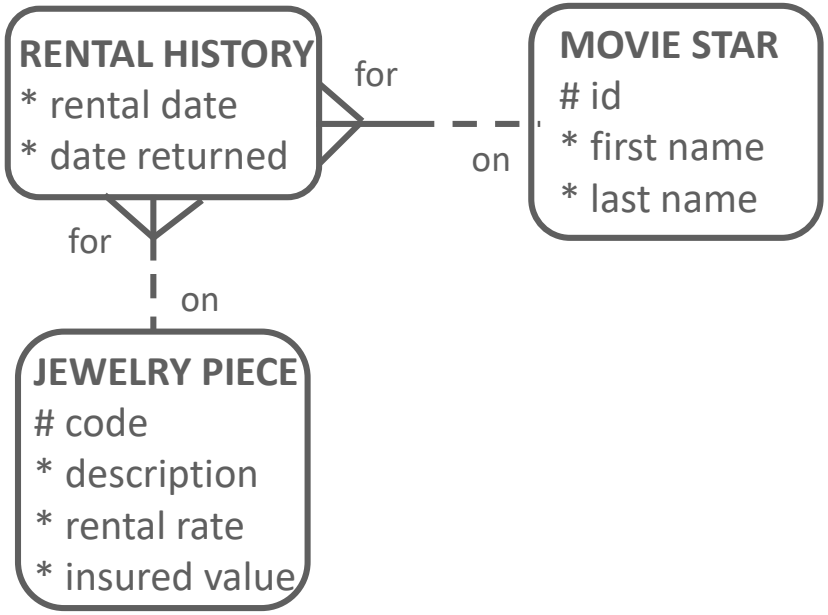


Resolve M:M

- The relationship between JEWELRY PIECE and MOVIE STAR should be revised to a M:M, which is then resolved with an intersection entity RENTAL HISTORY.
- Next we need to determine the UID of RENTAL HISTORY.

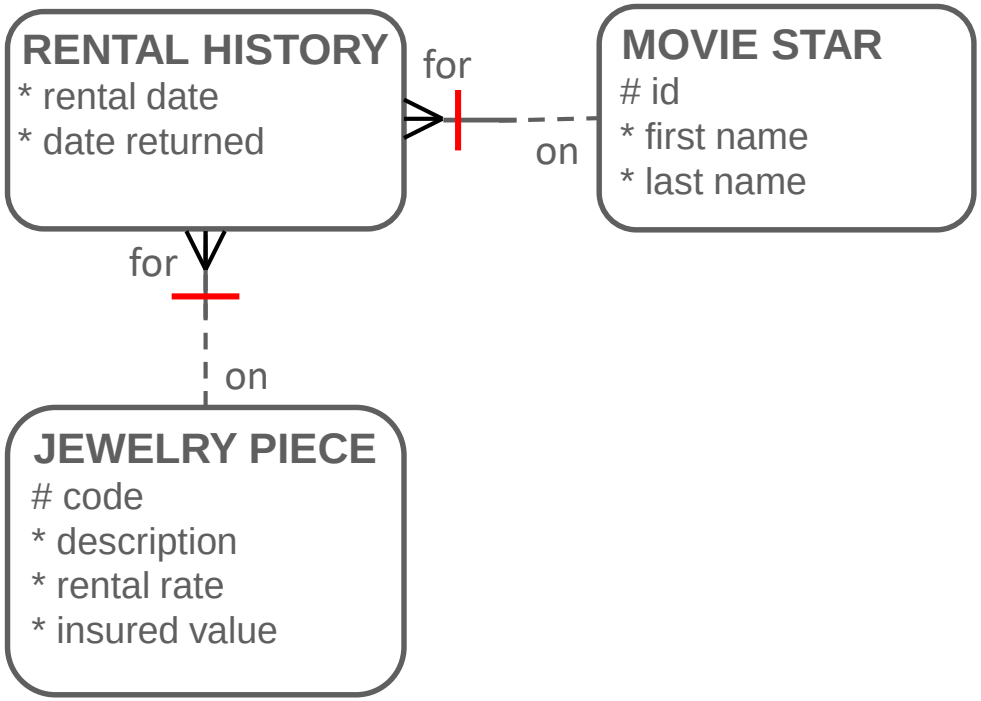


The M:M relationship is resolved with an intersection entity.



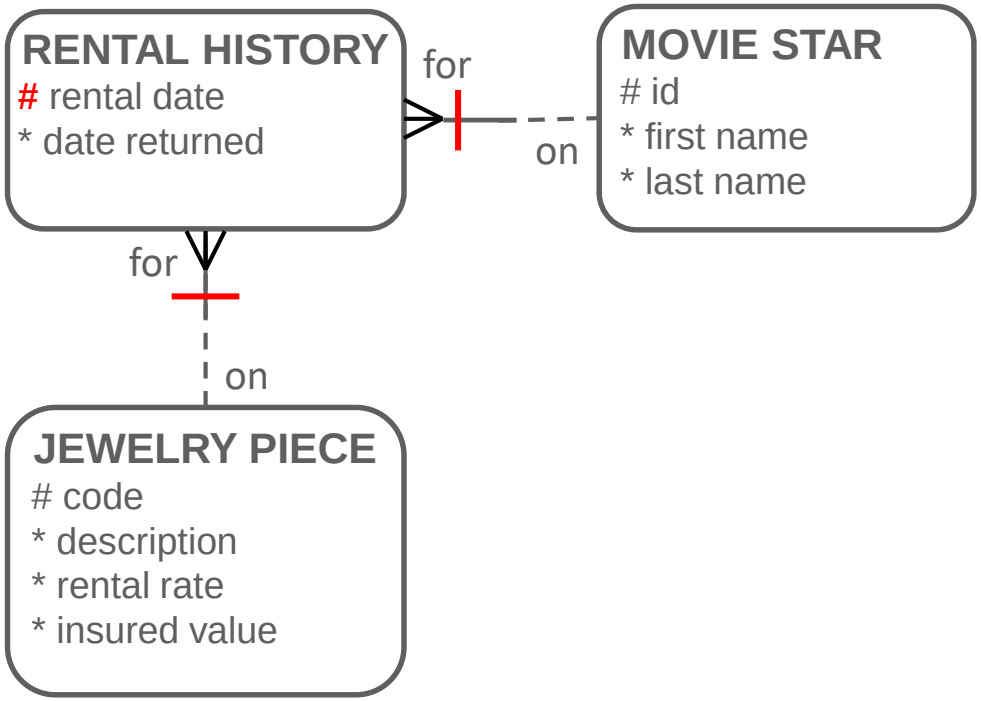
Determine UID

- Option 1: Barred relationship.
- Drawing a Barred relationship is not a suitable UID here, as this would not allow a MOVIE STAR to rent the same JEWELRY PIECE on different dates



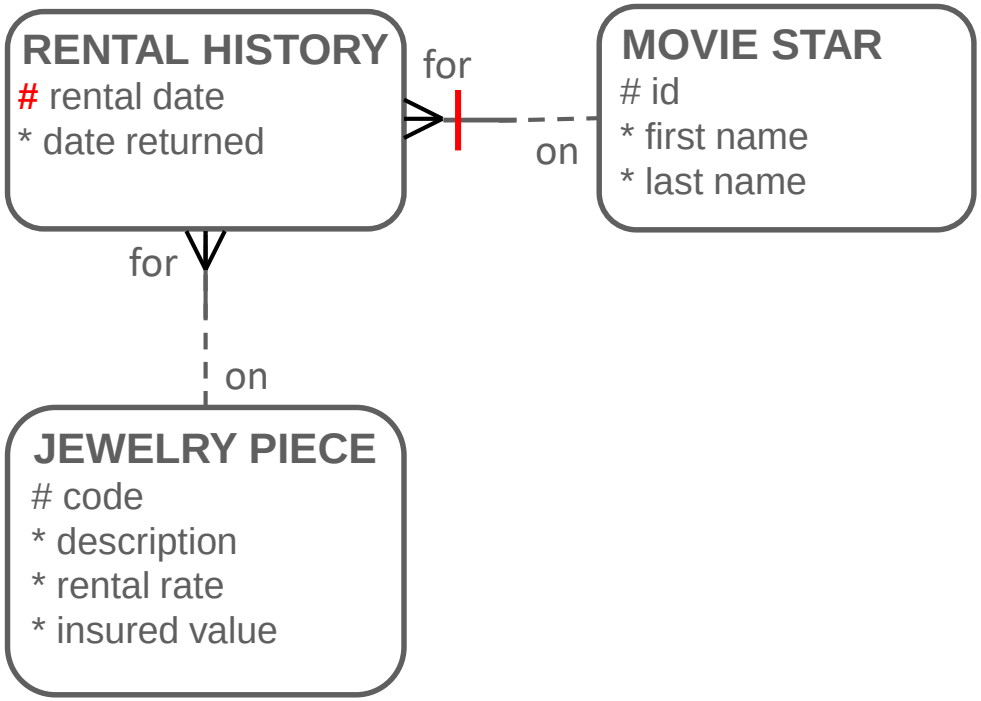
Determine UID

- Option 2: Barred relationship and Rental Date.
- Adding rental date to the UID would allow a MOVIE STAR to rent the same JEWELRY PIECE on different dates, but would also permit different MOVIE STARS to rent the same JEWELRY PIECE on the same date!



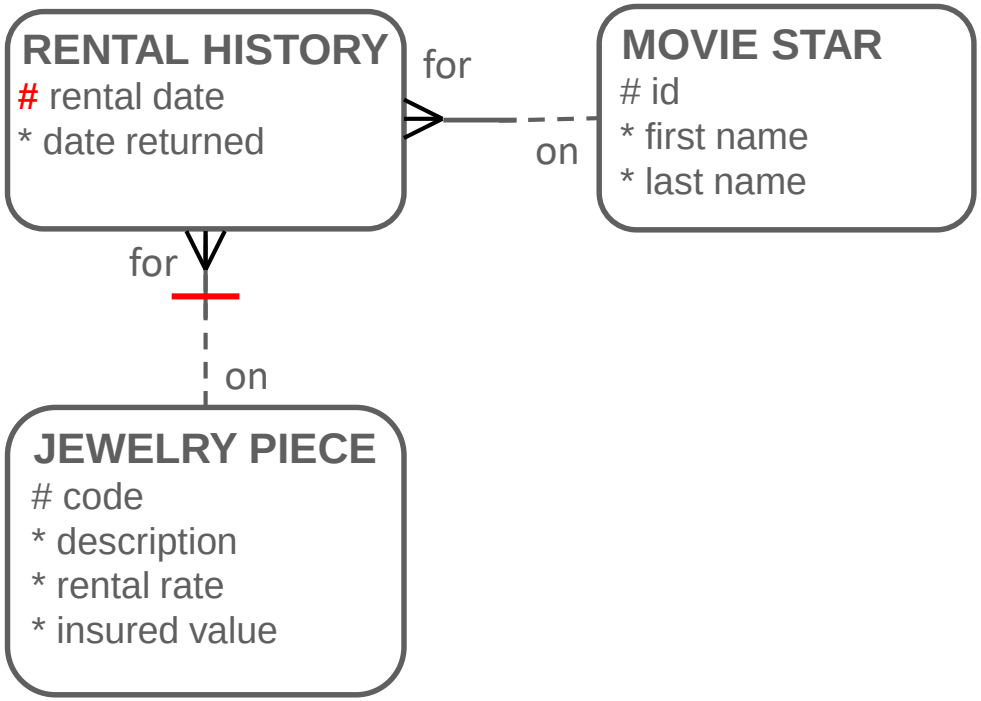
Determine UID

- Option 3: Barred relationship between MOVIE STAR and RENTAL HISTORY with Rental Date.
- This model would not permit the same MOVIE STAR to rent more than one JEWELRY PIECE on a given day.



Determine UID

- Option 4: Barred relationship between JEWELRY PIECE and RENTAL HISTORY with Rental Date.
- This model says that a JEWELRY PIECE can be rented only once on the same date.



Database Design

8-2
Modeling Change: Time

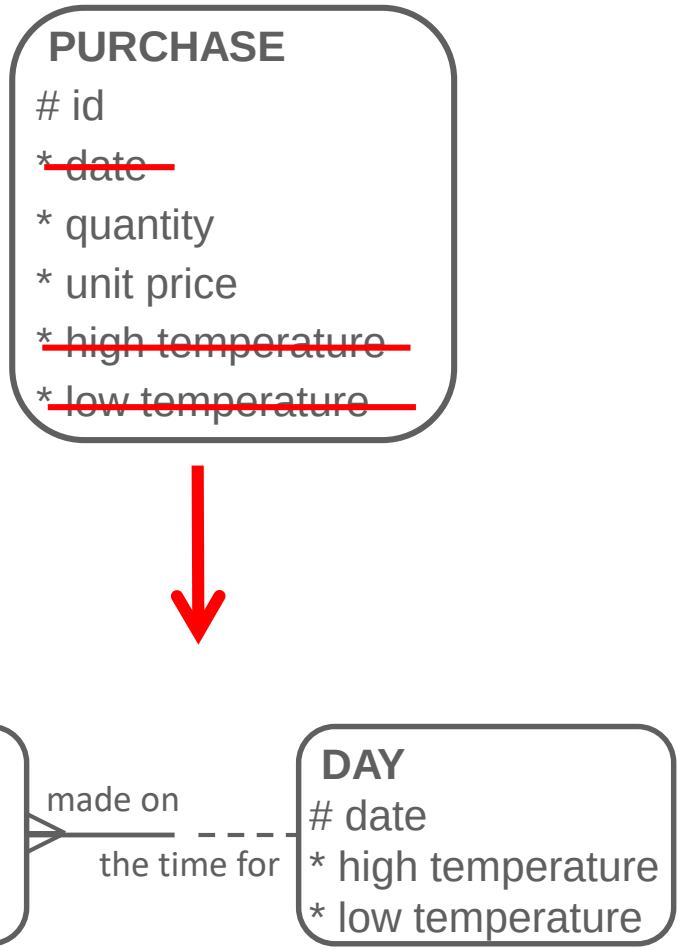
ORACLE ACADEMY

Entity DAY vs. Attribute Date

- Consider the entity PURCHASE.
- You would include an attribute “date” if you wanted to know when the item was purchased.
- However, if we want to identify trends -- such as purchasing coats vs. bathing suits vs. sneakers -- we may want to know the temperature during that time.
- If we add the temperature attributes to the PURCHASE entity it creates a problem.

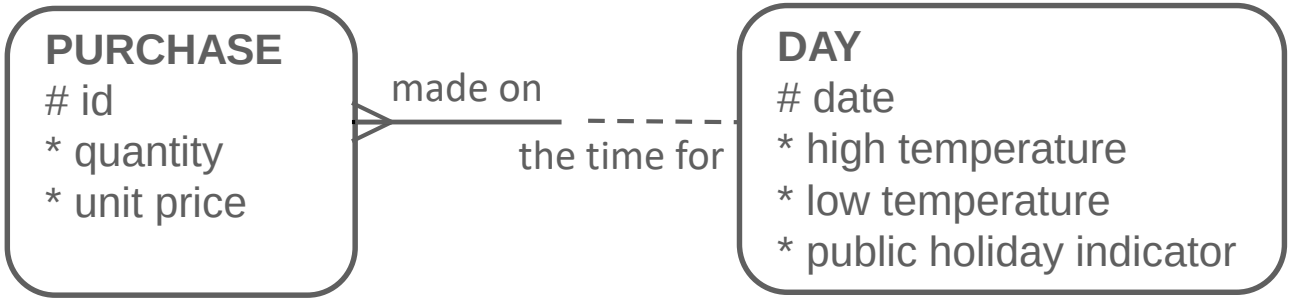
Entity DAY vs. Attribute Date

- Remember Third Normal Form: a non-UID attribute cannot have attributes of its own.
- Because high and low temperature are attributes of the date, we need a separate entity DAY.



Entity DAY vs. Attribute Date

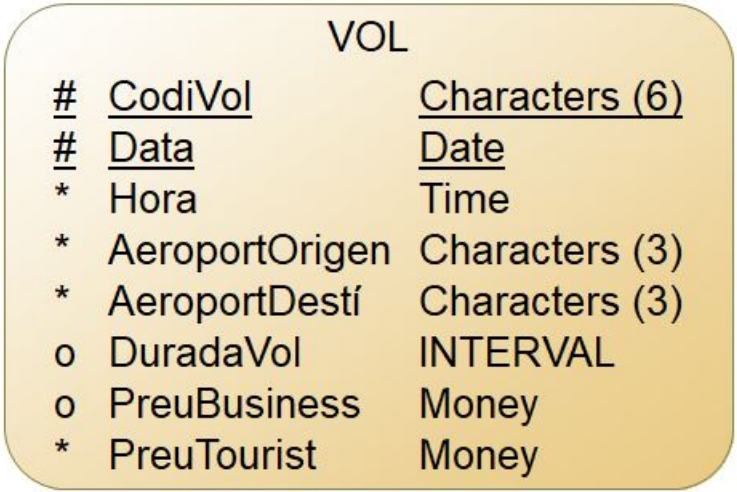
- Having a separate DAY entity allows us to track more information that may be useful to a business, for example which days were public holidays.



Database Design

8-3
Modeling Change: Price

ORACLE ACADEMY


VOL
CodiVol Characters (6)
Data Date
* Hora Time
* AeroportOrigen Characters (3)
* AeroportDestí Characters (3)
o DuradaVol INTERVAL
o PreuBusiness Money
* PreuTourist Money

The Importance of Price Changes

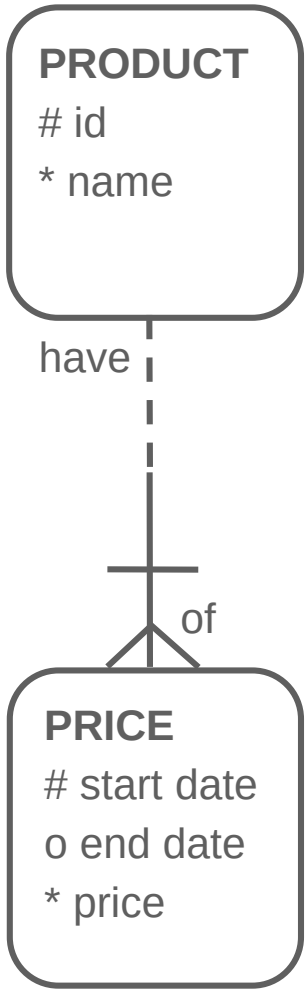
- Changes in price are often an important consideration when modeling business requirements.
- Some examples would be:
  - The stock market: Prices are changing by the second and you are watching the reader board, wondering when to buy and when to sell. What factors would you consider?
  - The fuel industry: Why would you want to track the price changes in fuel if you are thinking of buying a car or heating your home during the winter?
  - Construction businesses: Why are price changes important to a contractor of a five-year bridge-construction project?

What's the Price Today?

- The prices of products change over time.
- Some go up, some go down, and others fluctuate up and down.
- Food, clothing and school fees are more expensive now than they were twenty years ago.
- Technology often gets cheaper over time.
- You can buy a standard specification laptop computer today for around half the price of ten years ago.
- Gold, silver and currency are examples of commodities whose prices fluctuate.

Model Historical Price

- It is often useful to have information on past prices.
- The model shown here tracks the historical price of a product.

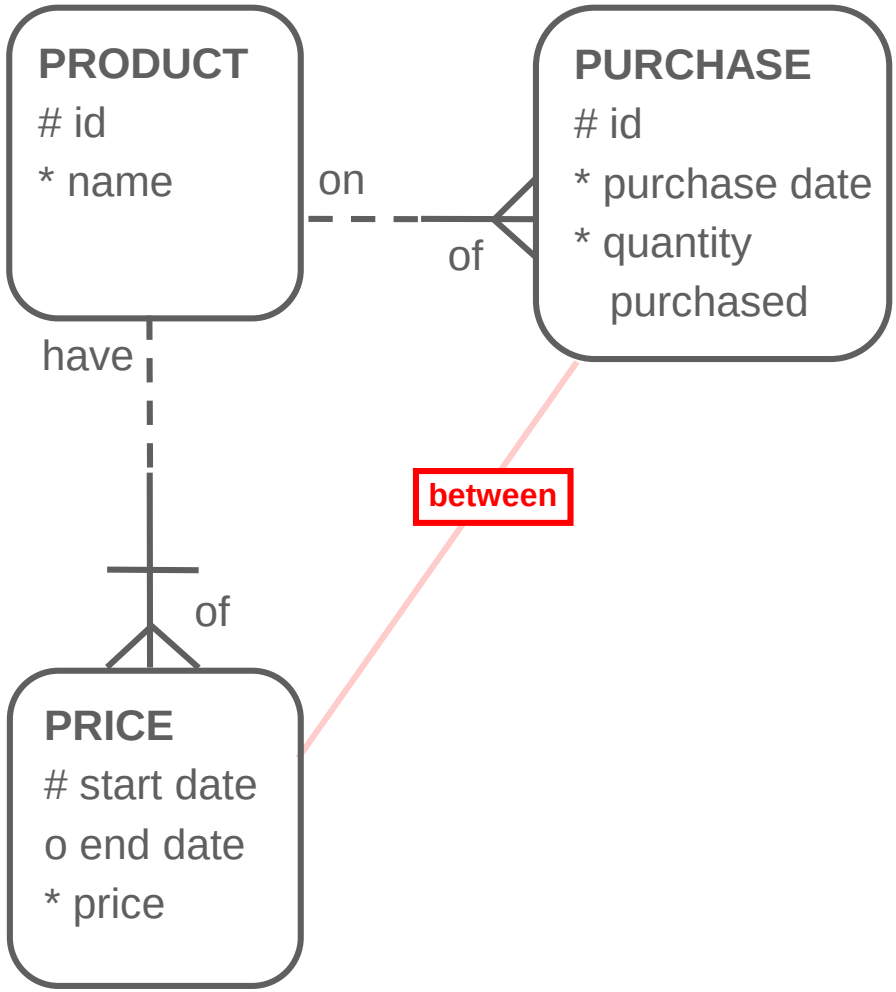


Historical Price Changes

- Consider what happens when you return an item to a store.
- You purchased the item at a certain price, but it has gone on sale since then.

Tracking Price Changes

- Businesses often need to keep a record of price changes.
- In this model, we assume that each PURCHASE is of only one product.
- The price that was paid can be found by matching the purchase date between the start date and the end date of PRICE.

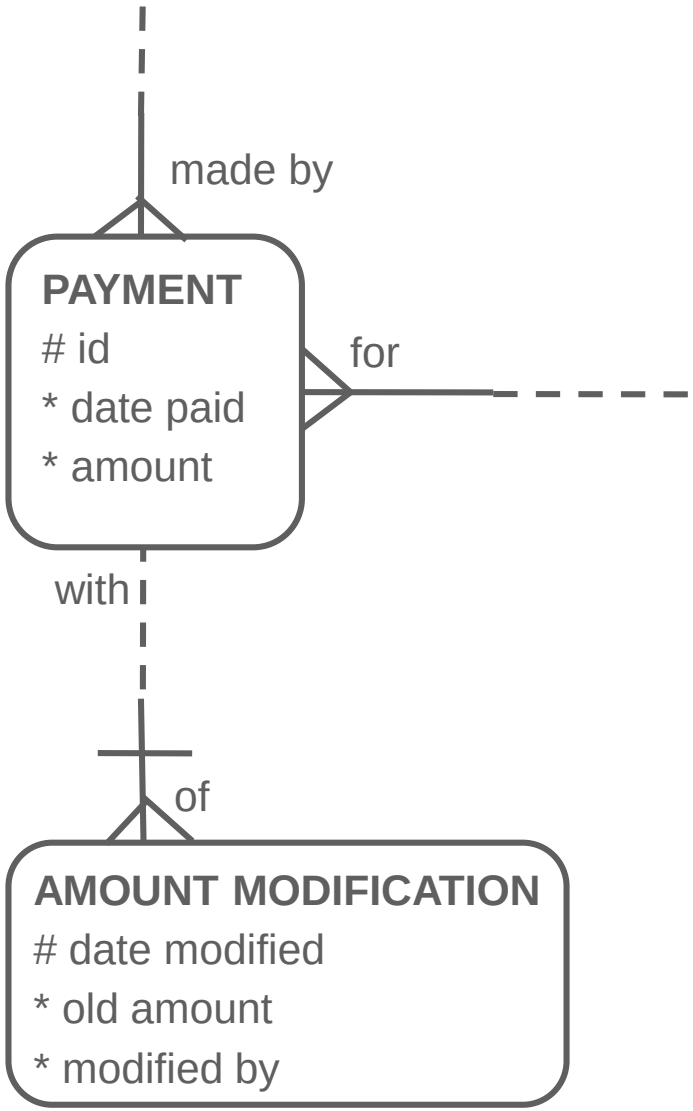


Other Data Changing Over Time

- We've seen that prices change over time.
- Other types of information can also change, for different business reasons.

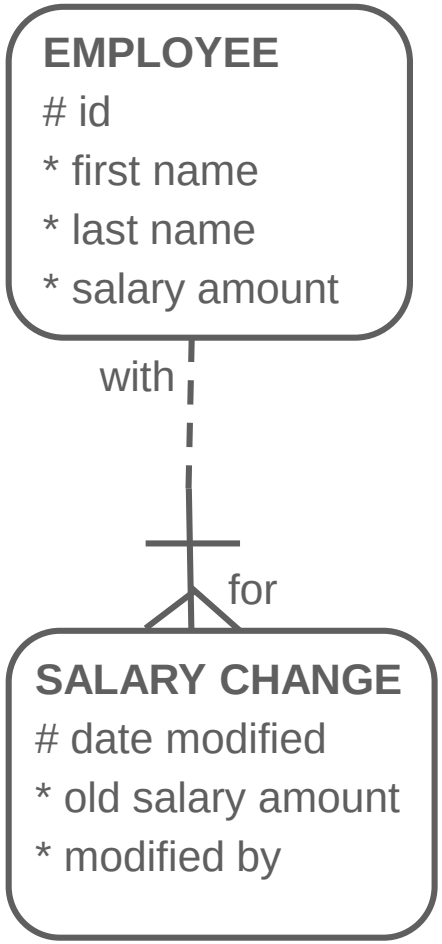
Journaling

- Whenever a system allows a user to modify or remove particular information, the question should be asked, “Do the old values need to be kept on record?”
- This is called "logging" or "journaling."
- This is often an issue when the information is financial or of a sensitive nature, such as a student grade change.



Journal Content

- A journal usually consists of both the modified value and the information about who did the modification and when it was done.
- This extra information can, of course, be expanded if you wish.



Database Design

8-4
Drawing Conventions for Readability

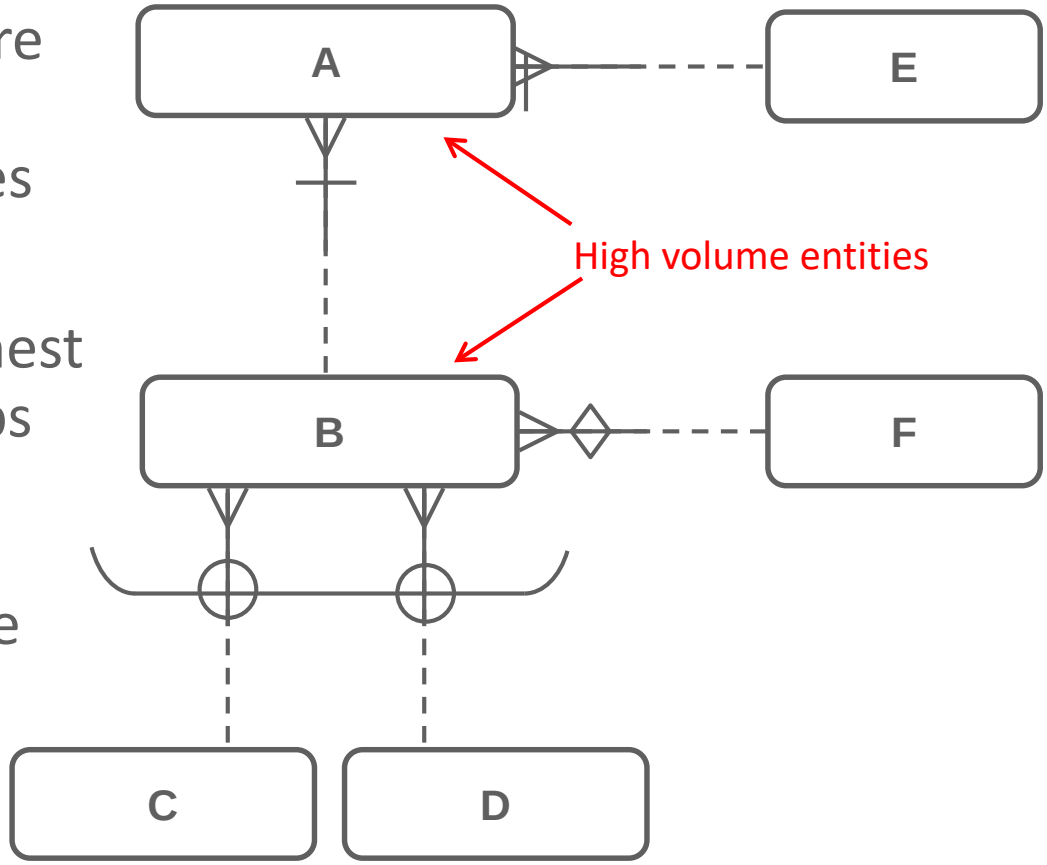
ORACLE ACADEMY

Large ERD Drawing Conventions

- The bigger and more complicated an ERD gets, the more challenging it becomes to lay out the pieces in a clear and readable format.
- There are two drawing conventions that are widely in use:
  - one that places high volume entities towards the top left of the page, and one that places high volume entities towards the bottom right of the page.
- It is not important which convention you follow, but chose one and try to use it consistently.
- A High-volume entity is an entity that will have a large number of instances.

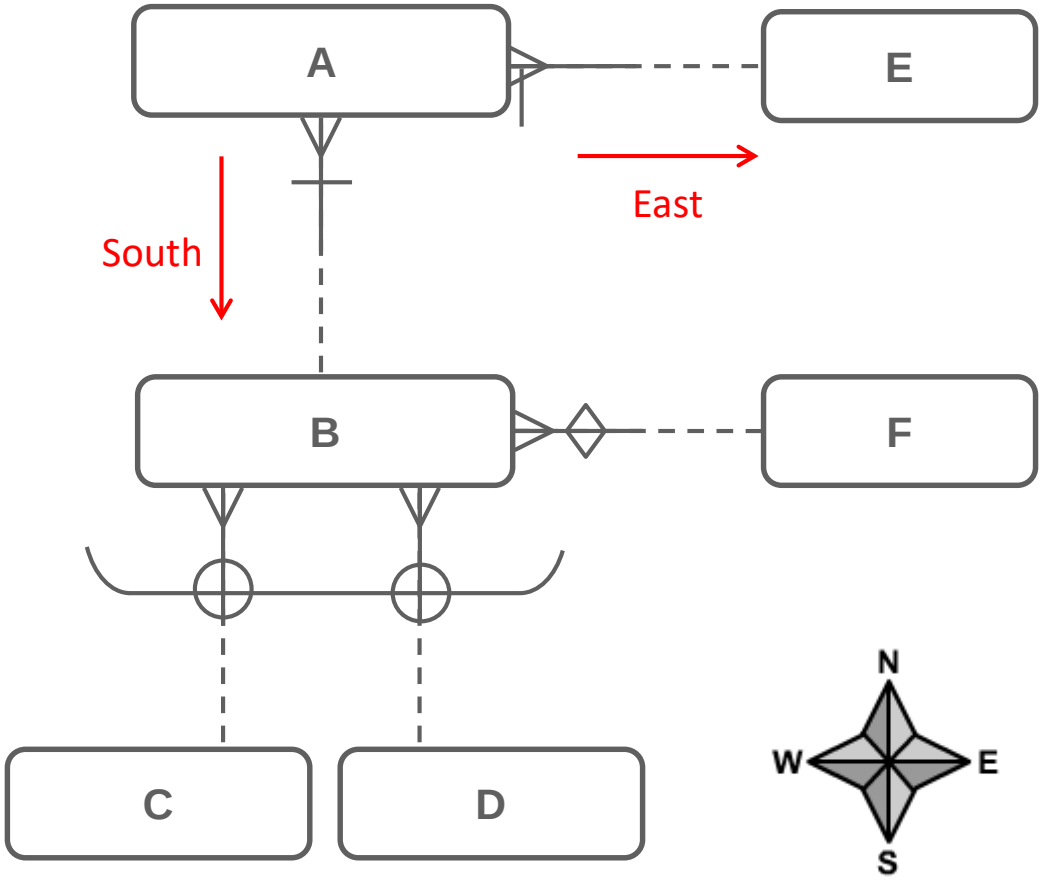
Large ERD Drawing Conventions

- High volume entities are often the “central” or more important entities in an ERD.
- They will have the highest number of relationships to other entities, and most of the business functions will affect the data stored in these entities.



Large ERD Drawing Conventions

- When high volume entities are on the upper left portion of the ERD, the crows feet will tend to point south and east.

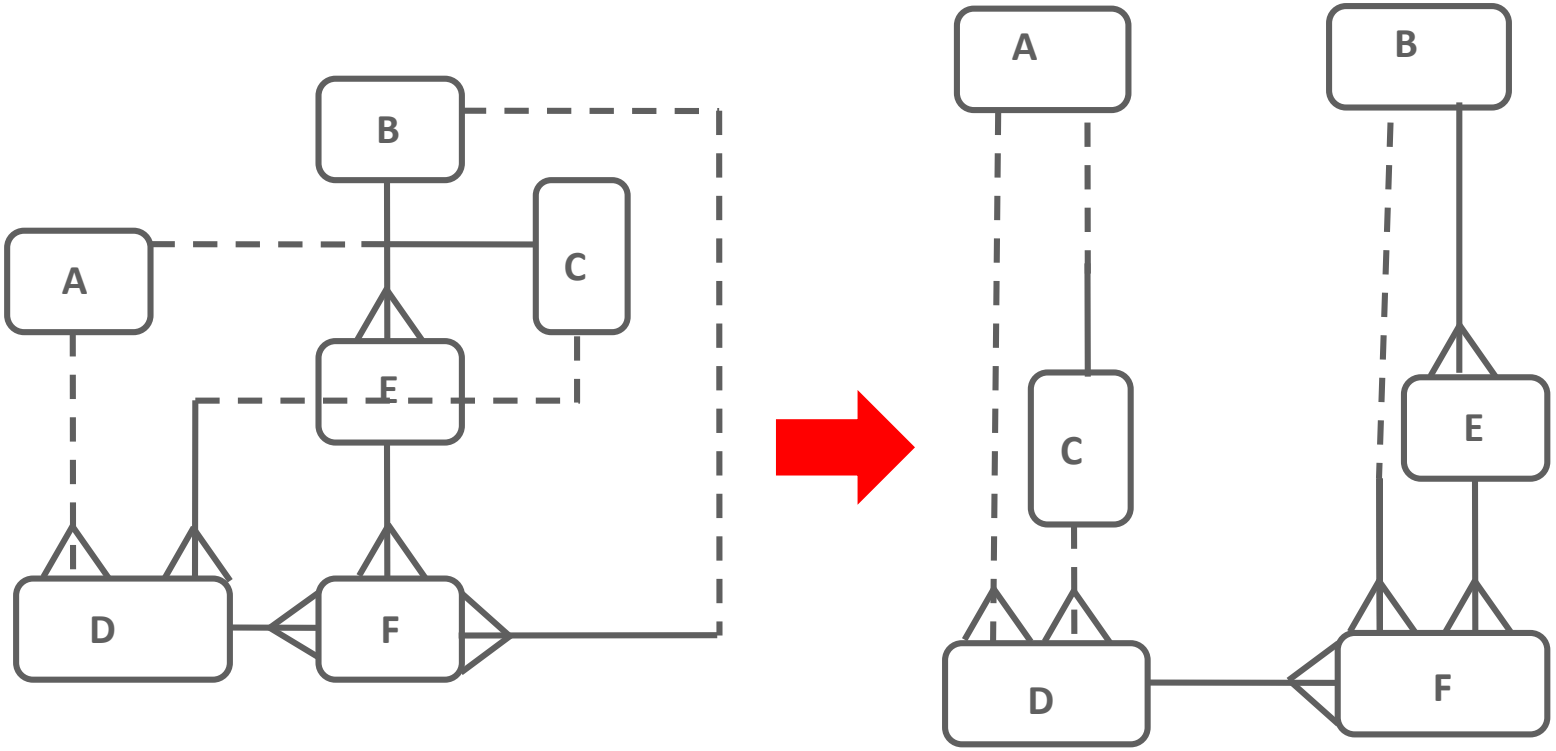


Clarity is Key

- For clarity and readability in an ERD:
- Avoid crossing relationship lines
- Avoid entities that overlap
- Avoid relationship lines that cross entities
- Use plenty of “white space”
- Split larger ERDs into smaller sub-diagrams if required

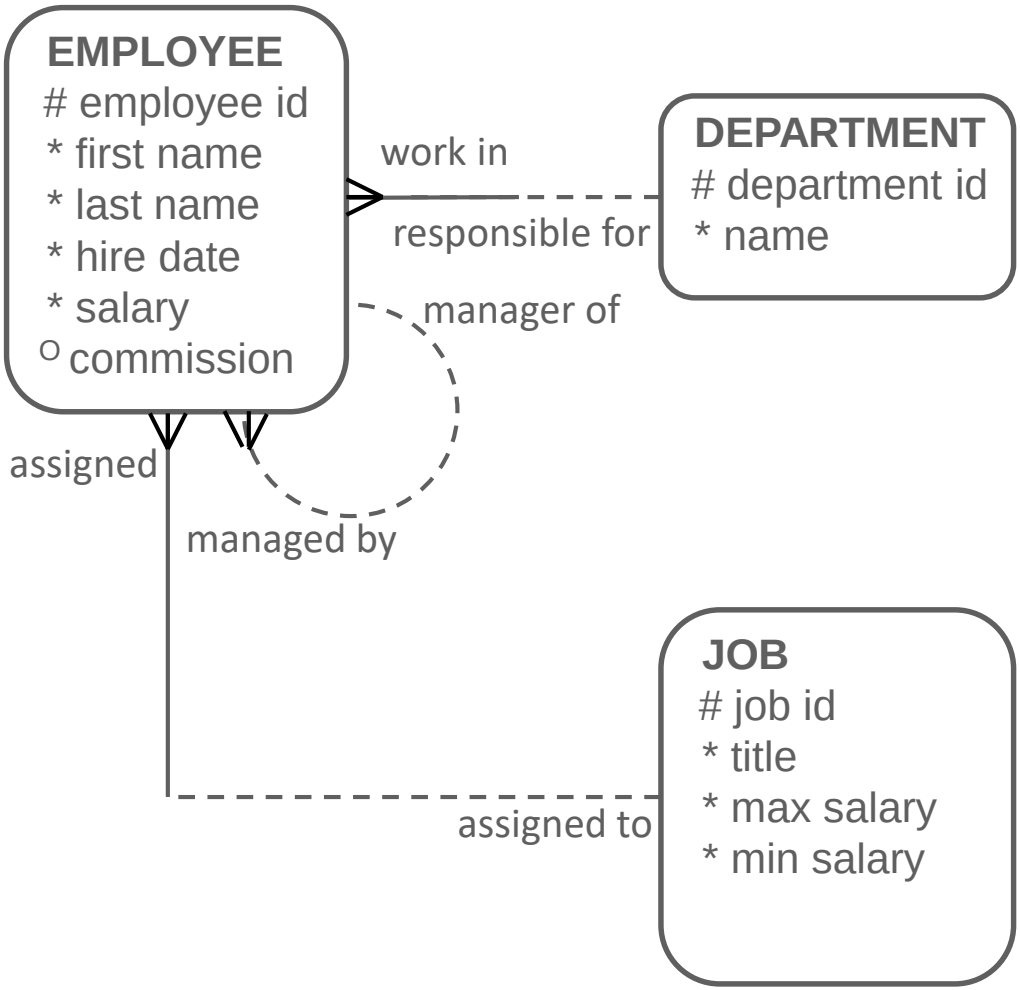
Space is Needed

- Readability takes space and is subject to taste. The use of white space helps clarify an ERD.



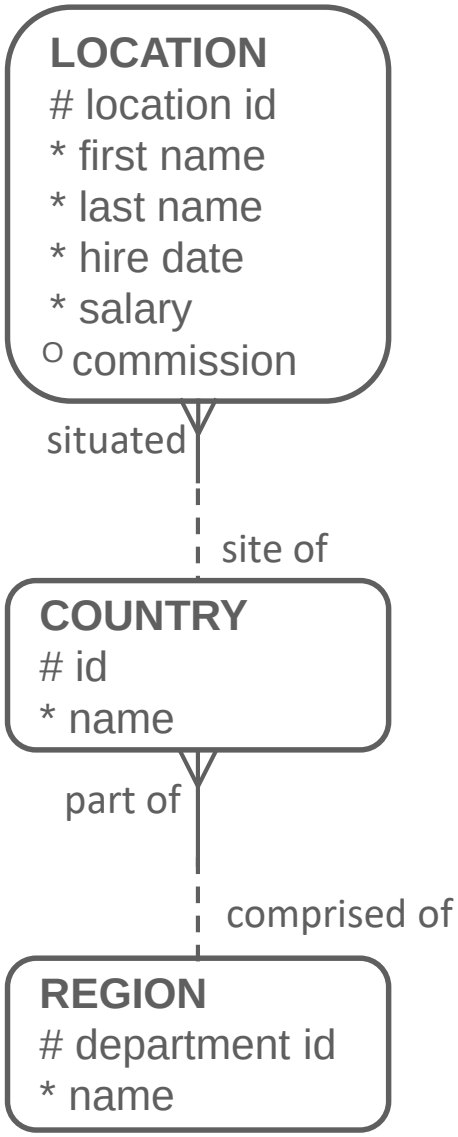
Use Sub-Diagrams

- When you have a very large diagram, it may also help to break it up into smaller diagrams of functionally related entities.



Use Sub-Diagrams

- You could use the smaller sub-diagrams when presenting to different groups within the customer's company.
- It is still important to have a big diagram that shows the whole picture (even if it has to be printed on a plotter or taped together from smaller pieces of paper).
- There may be relationships between entities in different sub-models, and these must be represented somewhere.




ORACLE
ACADEMY