Database Modelling Notes

by Frankie Inguanez

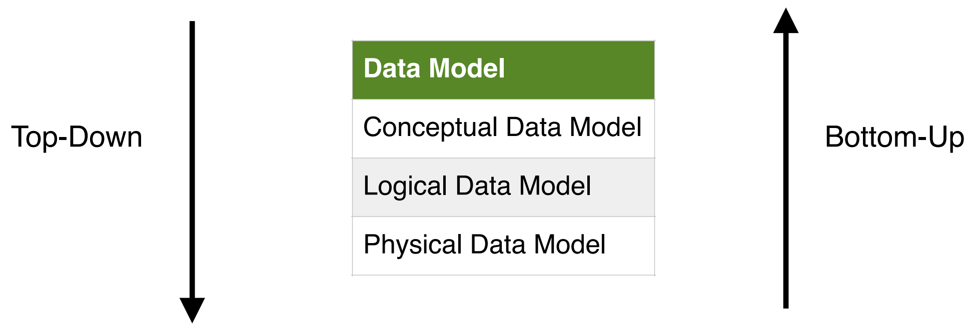
Disclaimer

Feedback
Whilst no profit is done from this work, your feedback is greatly appreciated. This can be done using the following media:

Email: frankieinguanez@gmail.com
LinkedIn: frankieinguanez

General Information

A very central part of every database project is the database design phase. There are numerous data diagrams, we shall be focusing on the Entity Relationship Diagram, generally associated with Relational Database Projects. There are three different versions of an ERD and the sequence of creation is generally based on whether the project adopts a Top-Down Approach or a Botton-Up Approach.



Data Model
Conceptual Data Model
Logical Data Model
Physical Data Model

In most cases a Top-Down approach is adopted reflecting the progression in information and details added to the diagram.

Data Model	Detail
Conceptual Data Model	Just entities and relationships. No attributes, data types or keys.
Logical Data Model	Adds attributes informations and keys
Physical Data Model	Add data types and constraints

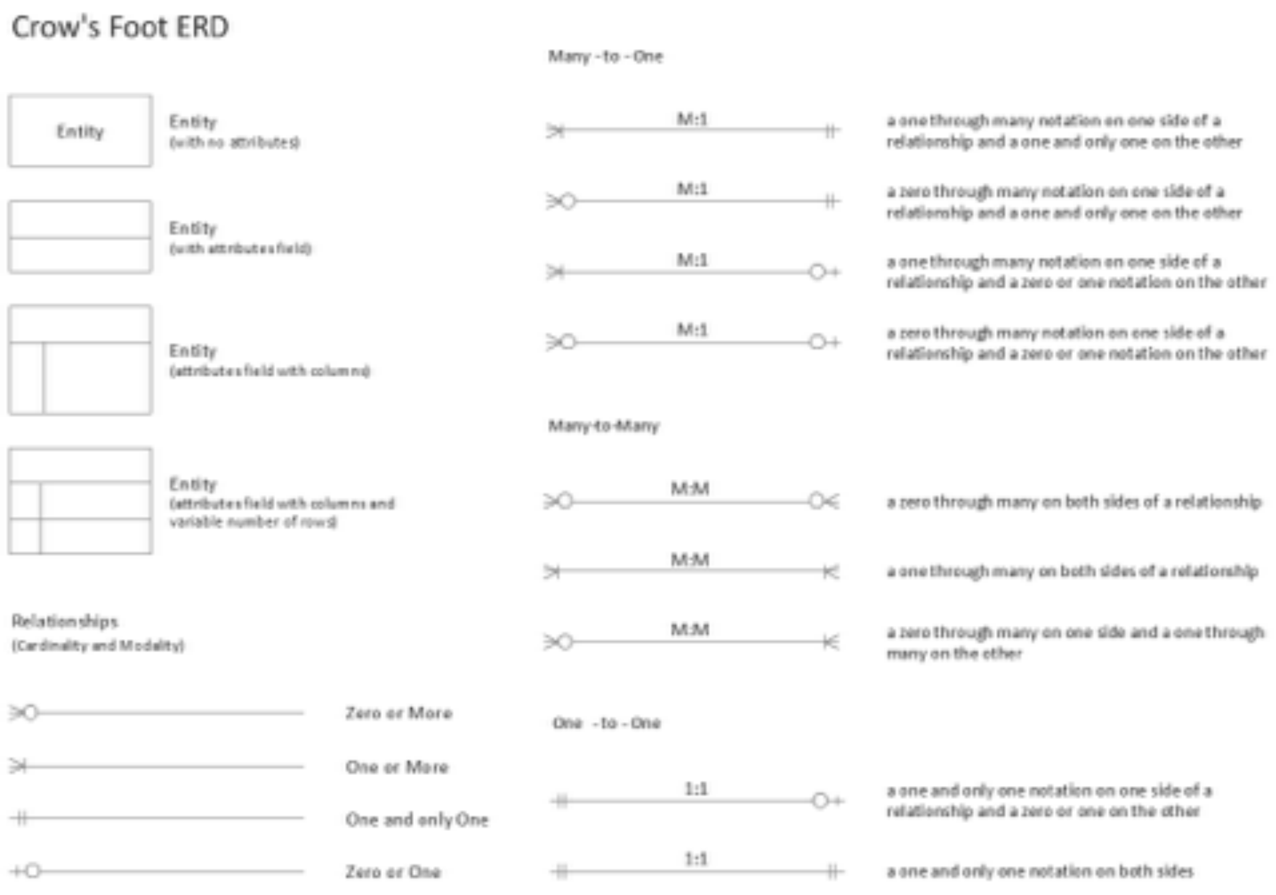
ERD Concepts

There are three basic concepts in ERDs with which a relational database can be fully designed.

Concept	Implementation
Entity	A table representing an object of importance to the project
Attribute	A column for which data will be stored for every row/tuple/record
Relationship	A Foreign Key Constraint representing a link between one or two entities

ERD Notation

Numerous ERD notations exist we shall be focusing on the Crow's Foot Notation. Also, many software applications support this notation yet, sometimes some variations are presented to the notation within a specific software tool. For the purpose of this course we shall be adopting the Oracle Data Modeler tool.

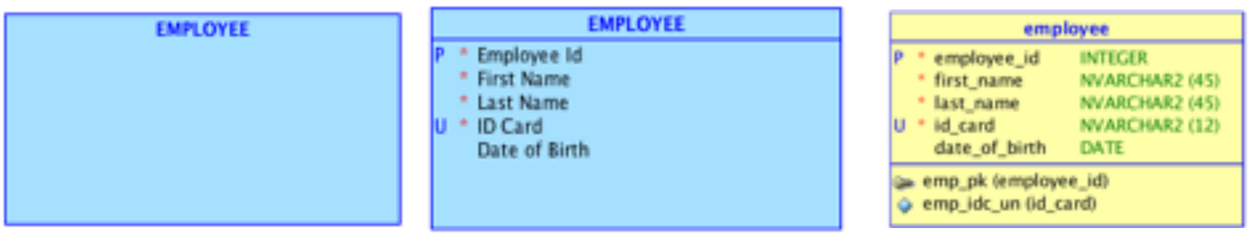


Crow's Foot ERD
Entity
Entity (with no attributes)
Entity (with attributes field)
Entity (attributes field with columns)
Entity (attributes field with columns and variable number of rows)
Relationships (Cardinality and Modality)
Zero or More
One or More
One and only One
Zero or One
Many - to - One
M:1
a one through many notation on one side of a relationship and a one and only one on the other
M:1
a zero through many notation on one side of a relationship and a one and only one on the other
M:1
a one through many notation on one side of a relationship and a zero or one notation on the other
M:1
a zero through many notation on one side of a relationship and a zero or one notation on the other
Many-to-Many
M:M
a zero through many on both sides of a relationship
M:M
a one through many on both sides of a relationship
M:M
a zero through many on one side and a one through many on the other
One - to - One
1:1
a one and only one notation on one side of a relationship and a zero or one on the other
1:1
a one and only one notation on both sides

ERD Concepts - Entity

An entity represents an object of importance for the project. In a different project the same entity can be represented as a mere attribute and thus its up to the database designer to determine the importance of the object.

An entity is represented in a rectangle with the entity name written in the centre upper section. The notation as to whether write the name in singular or plural varies across companies. For the purpose of this course we shall write all entities in singular and upper case. Note that the notation used when implementing the entity as a table might differ.



ERD Concepts - Attributes

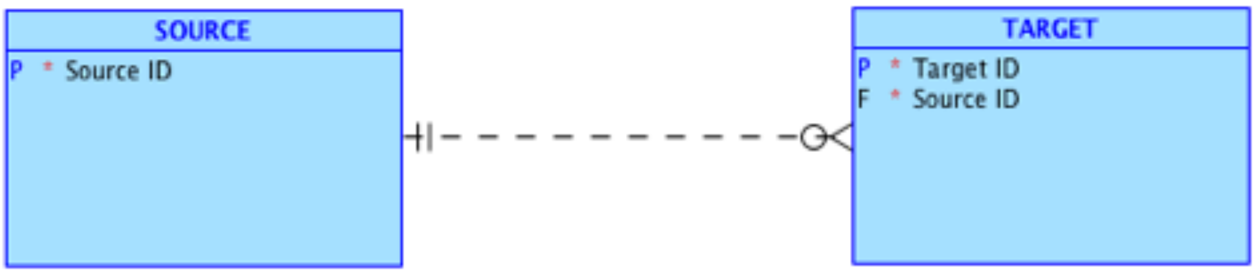
Depending on the data model being presented then the level of details varies. Following are the interpretations of the various attribute prefixes.

Prefix	Meaning
P	Primary Key (PK)
F	Foreign Key (FK)
U	Unique Key
*****	Mandatory attribute

Note that Primary Key and Unique Constraints are specified at the bottom section of the entity.

ERD Concepts - Relationships

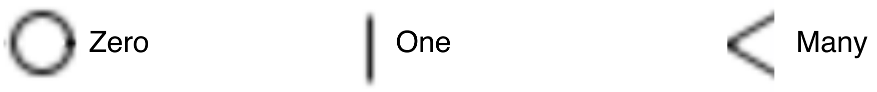
The link between two entities is a relationship and this is implemented by a PK and FK pair. The entity from which a relationship originates is called the source entity and this should contain the PK of the relationship. The entity at which a relationship ends is called the destination entity and this should contain the FK of the relationship.



There are three different types of relationships which are:
1. **One-to-One (1:1)** Every entity instance/row/record/tuple in the source entity is related to zero or exactly one entity instance/row/record/tuple in the target entity.
2. **One-to-Many (1:M)** Every entity instance/row/record/tuple in the source entity is related to zero one or more entity instances/rows/records/tuples in the target entity.
3. **Many-to-Many (M:N)** Every entity instance/row/record/tuple in the source entity is related to zero one or more entity instances/rows/records/tuples in the target entity and the same applies in inverse order.

Relationship Cardinality

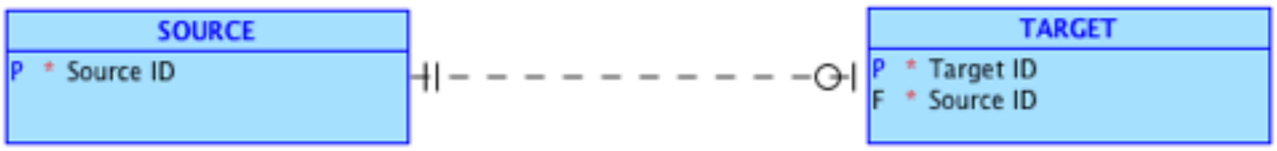
For each relationship type there are different cardinality options with the following symbol interpretations:



One-to-One Relationship

In a 1:1 relationship every entity instance/row/record/tuple in the source entity is related to exactly one entity instance/row/record/tuple in the target entity.

Scenario 01

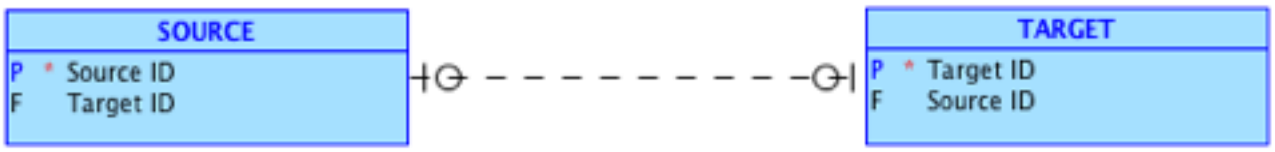


In this scenario the Source entity is the dominant role due to the cardinality. The cardinality is read on the opposite side of the relationship from the entity. So the 0I belong to the Source, which mean that: Each SOURCE may be related to a minimum of zero and a maximum of one TARGET. The II belong to the Target entity such that: Each TARGET must be related to a minimum of 1 and a maximum of 1 SOURCE.

Note that when reading the cardinality from left to right (from source to target) the left symbol (in this case 0) is the minimum, whilst the right symbol is the maximum. When reading from right to left (from target to source) the right symbol is the minimum and the left symbol is the maximum. A simpler manner of remembering it is, the first symbol encountered is the minimum then the next symbol is the maximum.

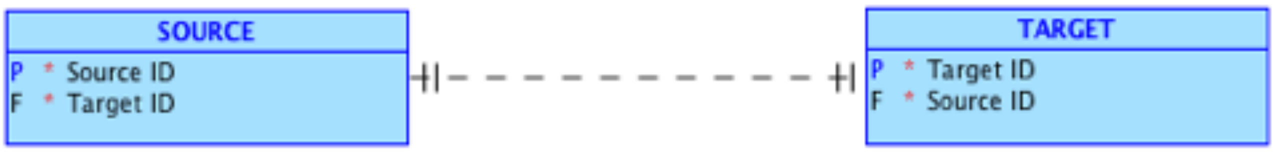
Since the Source entity is the dominant role then the FK is placed in the Target entity.

Scenario 02



In this scenario, there is no dominant entity role since in both cases the minimum is 0 and the maximum is 1. Therefore the FK attribute can be listed in both entities. Both FK are optional

Scenario 03

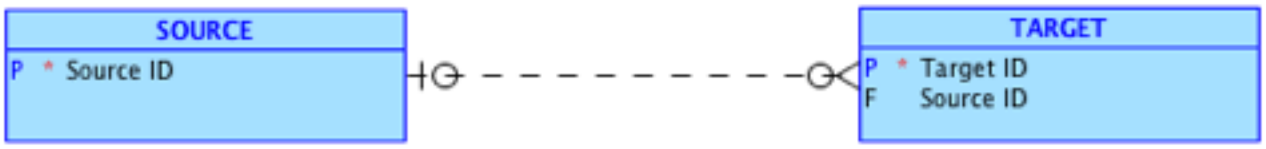


Again in this scenario, there is no dominant role and therefore the FK can be listed in both entities. This time both FK are mandatory.

One-to-Many Relationship

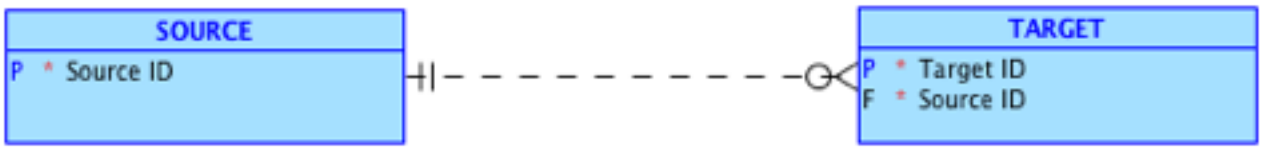
In a 1:M relationship every entity instance/row/record/tuple in the source entity is related to zero, one or many entity instances/row/records/tuples of the target entity. In this case the FK is always located in the target entity.

Scenario 01



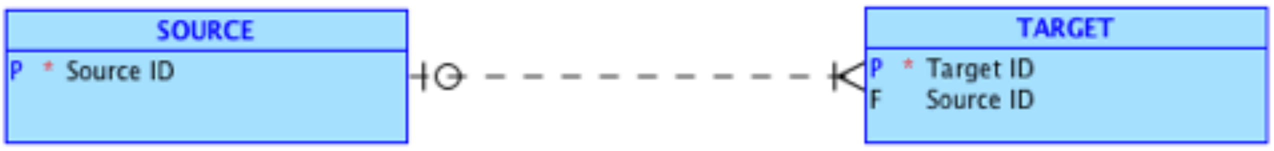
In this situation the minimum on both sides is zero. This means that the FK in the target entity is an optional attribute and can be null. It also means that certain rows in source may not be related to a target row.

Scenario 02



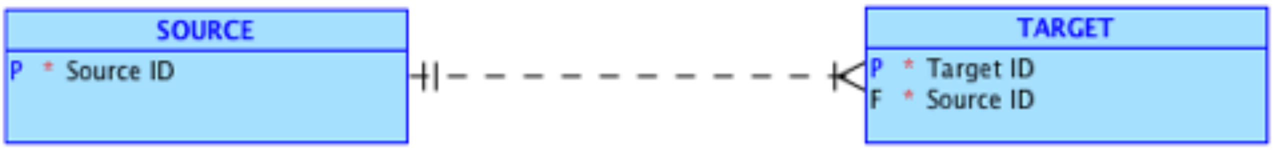
In this scenario the minimum cardinality for the target entity is 1 therefore the FK is mandatory and cannot be null. What this scenario means is that some rows in source might not be related to a target, but every target row must be related to 1 source row.

Scenario 03



This scenario sets the minimum of the target entity to 0 therefore the FK is optional again and thus can be null. Yet every row in source must be related to at least 1 row in target.

Scenario 04

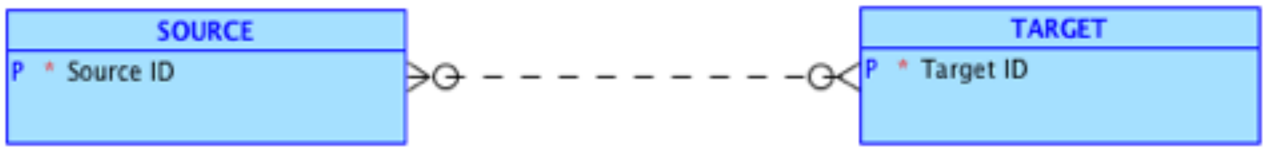


In this scenario the minimum on both perspectives of the relationship is 1 and therefore the FK is mandatory meaning that every row in target must be related to 1 row in source and every row in source can be related to one or more rows in target.

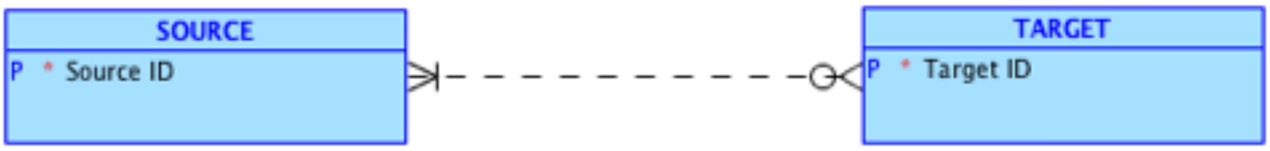
Many-to-Many Relationship

In a M:N relationship every entity instance/row/record/tuple in the source entity may be related to zero, one or more entity instances/rows/records/tuples in the target entity. We do not implement this kind of relationship but resolve it by following a number of steps. First let us see the different combinations.

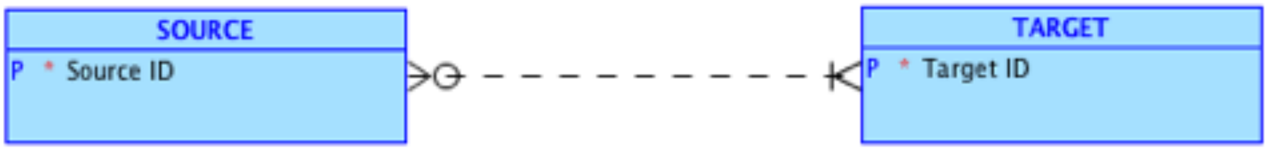
Scenario 01



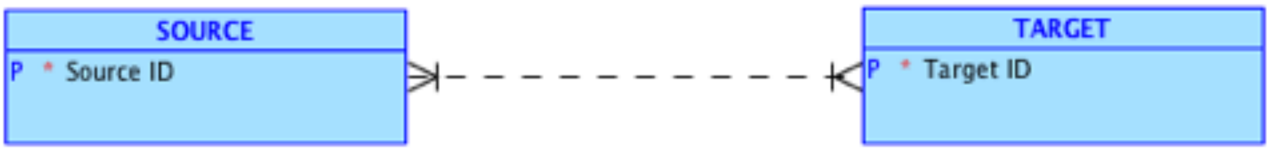
Scenario 02



Scenario 03



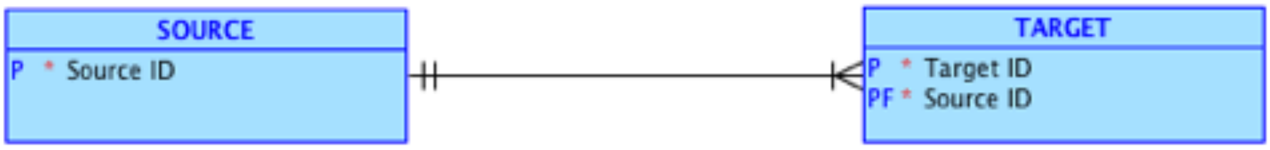
Scenario 04



To solve a M:N we need to introduce the concept of an identifying relationship. What we presented so far are all non-identifying relationships.

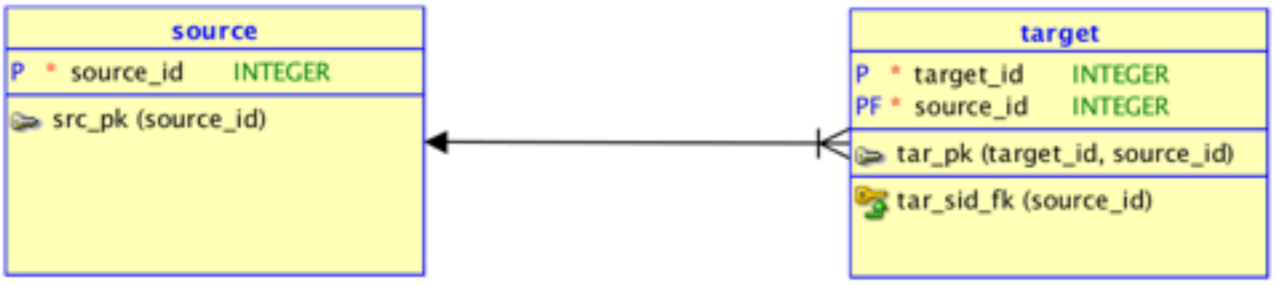
Identifying relationship

An identifying relationship is the situation when the FK of a relationship forms part of the PK in the target entity as shown below. Note that an identifying relationship is represented by a solid line. This concept is also known as a Composite Primary Key in this case.



What you should also notice is that the Source ID attribute in the target entity is prefixed by a PF *, what the following interpretation is needed: P indicating that the Source ID is part of the Primary Key, F indicating the the source ID is the Foreign Key, * indicating that it is a mandatory attribute.

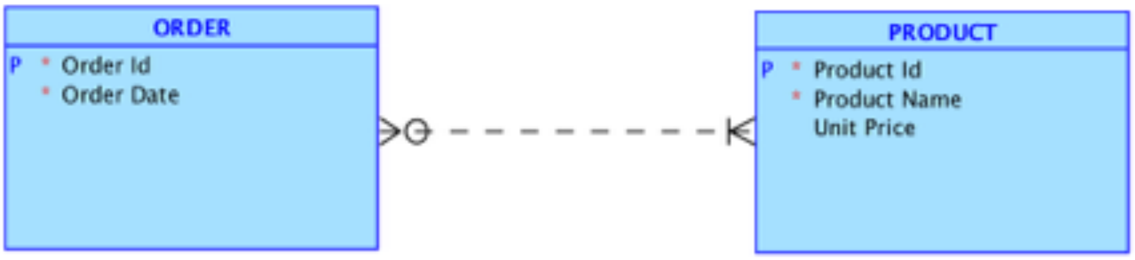
Whenever there is an identifying relationship, the minimum cardinality of the target entity is 1 since not PK attribute can be left null.



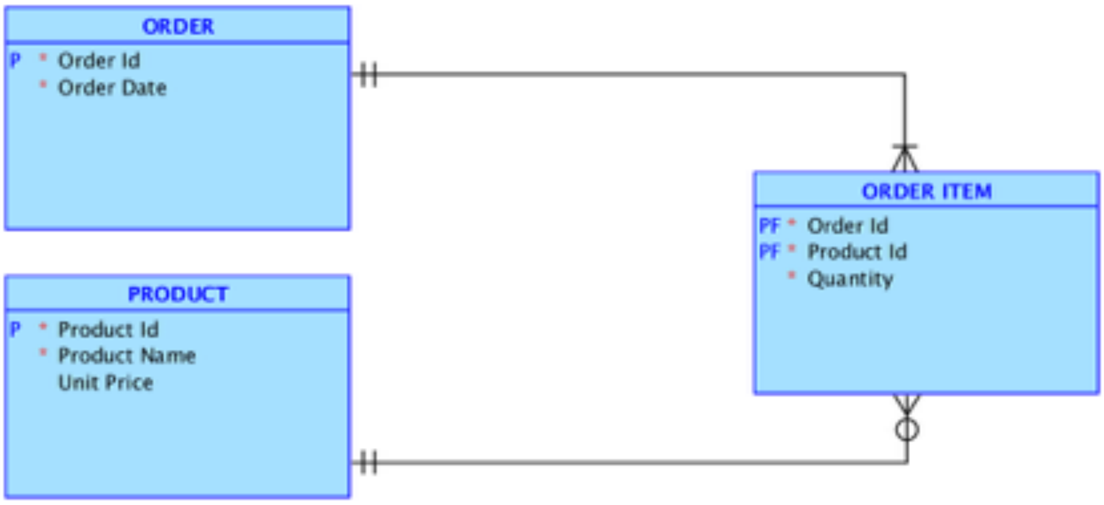
M:N Solution

Consider the following scenario:

> A table of products stores the product name and unit price. A table of orders stores the order table. Every order must have one or more products. Every product may be listed in zero, one or more orders.



The problem here is that there is missing information. What is the quantity of each product in each order? We cannot add a FK in any entity.



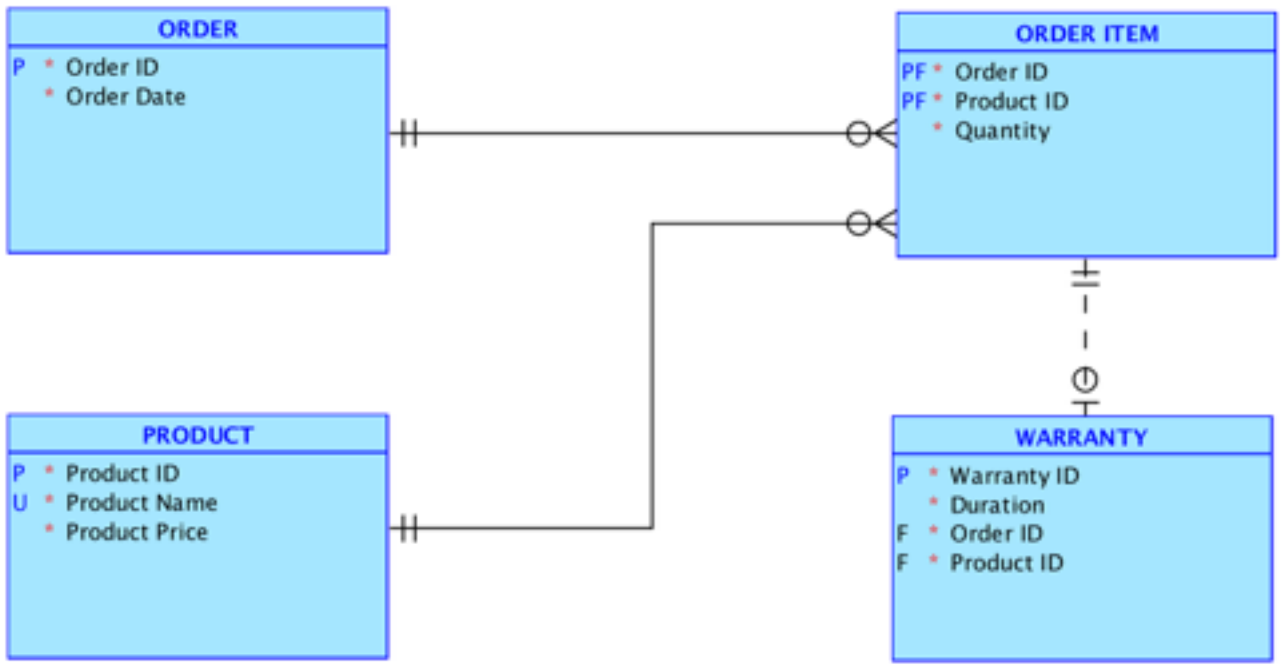
Some
points need to be mentioned:

1. A new transactional/intermediary table is created
2. The M:N relationship is replaced by two 1:M relationships
3. The cardinality of the intermediary table always has a minimum and maximum of 1.
4. The minimum cardinality of the other tables is taken from the original M:N relationship (1 for order, 0 for product)
5. Additional attributes may be added in the intermediary table if needed.

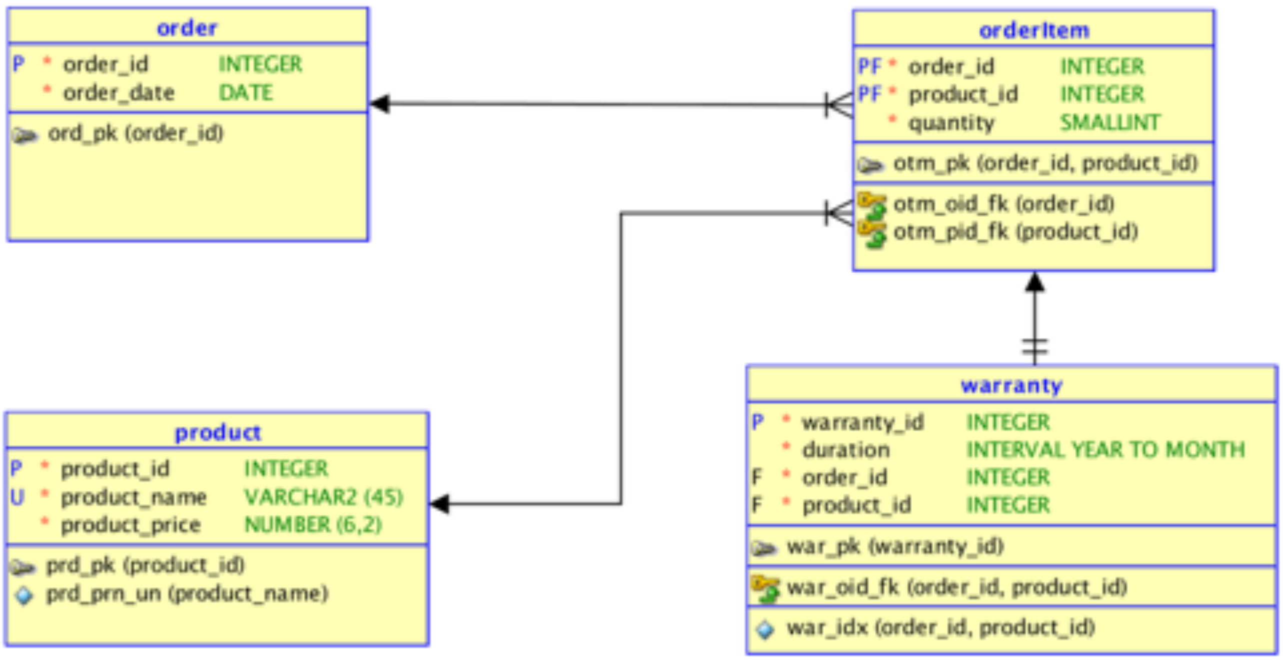
Note that a composite primary key is a singular constraint applied on multiple attributes. This is generally implemented as a table-level constraints.

Composite Foreign Keys

When creating a Composite Primary Key (CPK) one should note that any relationship from the entity with a CPK would result in a Composite Foreign Key (CFK).



Note that a composite foreign key constraint would be one singular constraint applied on multiple attributes. This is generally implemented as a table-level constraint.

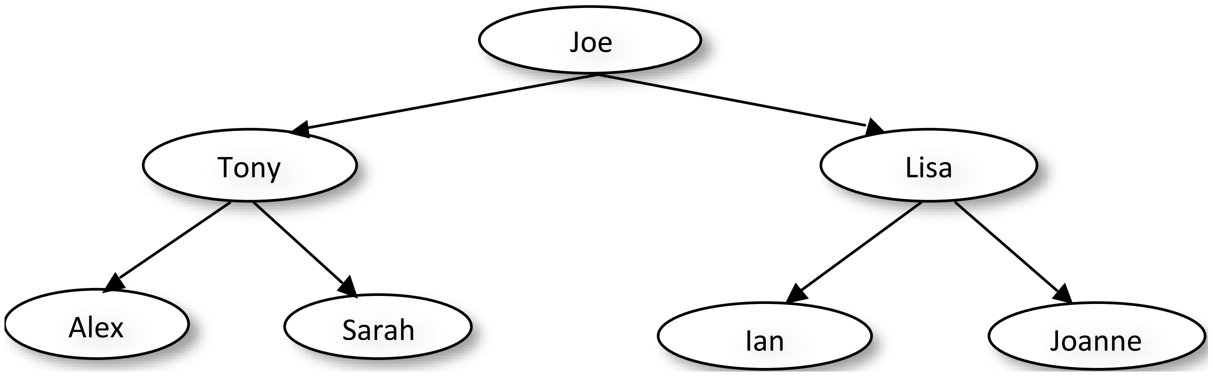


Recursive Relationship

Whenever a hierarchical scenario needs to be represented a recursive relationship is needed. What this means is that a relationship would originate and terminate in the same entity. Common applications are comment replies, product categories with sub-categories and company job roles.

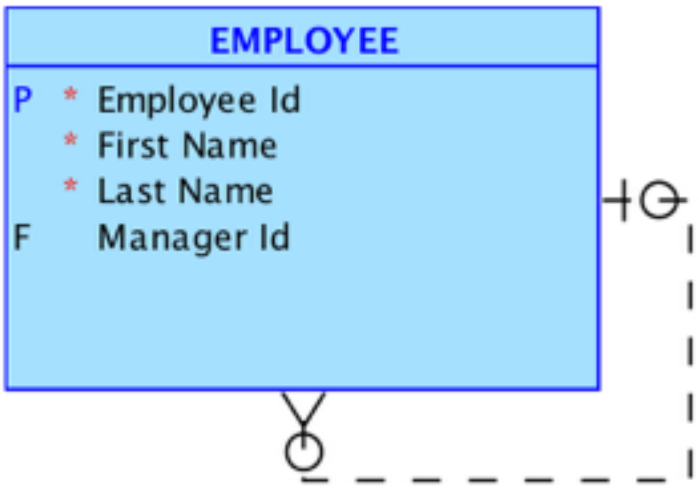
Consider the following scenario:

> Joe is the manager of Tony and Lisa. Tony is the manager of Alex and Sarah. Lisa is the manager of Ian and Joanne.



Rather than creating an entity manager, we can consider just one entity Employee and state that:

> Each employee may be the manager of zero, one or more employees.
> Each employe may be managed by zero or exactly one employee.

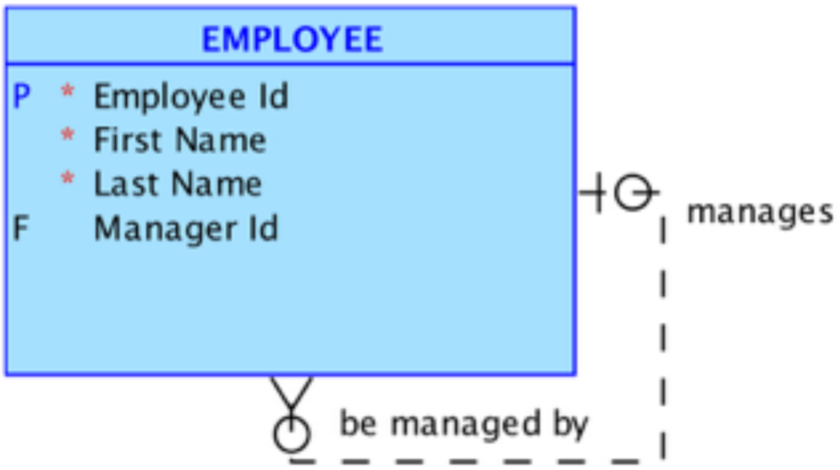


So what we have here is a relationship where the source and target entity are the same, the Employee entity. You will recall that no two attributes may have the same name within an entity and therefore we renamed the FK to Manager Id to reflect the logic of the relationship.

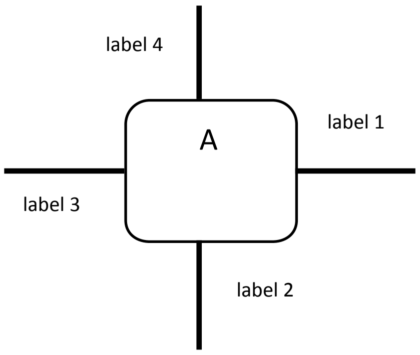
N.B. It is very important to note that the minimum cardinality of a recursive relationship should always be 0 otherwise the tree would grow infinitely.

Relationship Labels

Most Crow's Foot notation tools allow for just one label per relationship which in general is acceptable. Yet Oracle Data Modeler allows for two labels one per relationship perspective. This allows for a more descriptive display of the ERD.



ERDs for larger systems might get very clutters and thus agreeing on a protocol where to position labels, such as the one shown below, would greatly help readability.

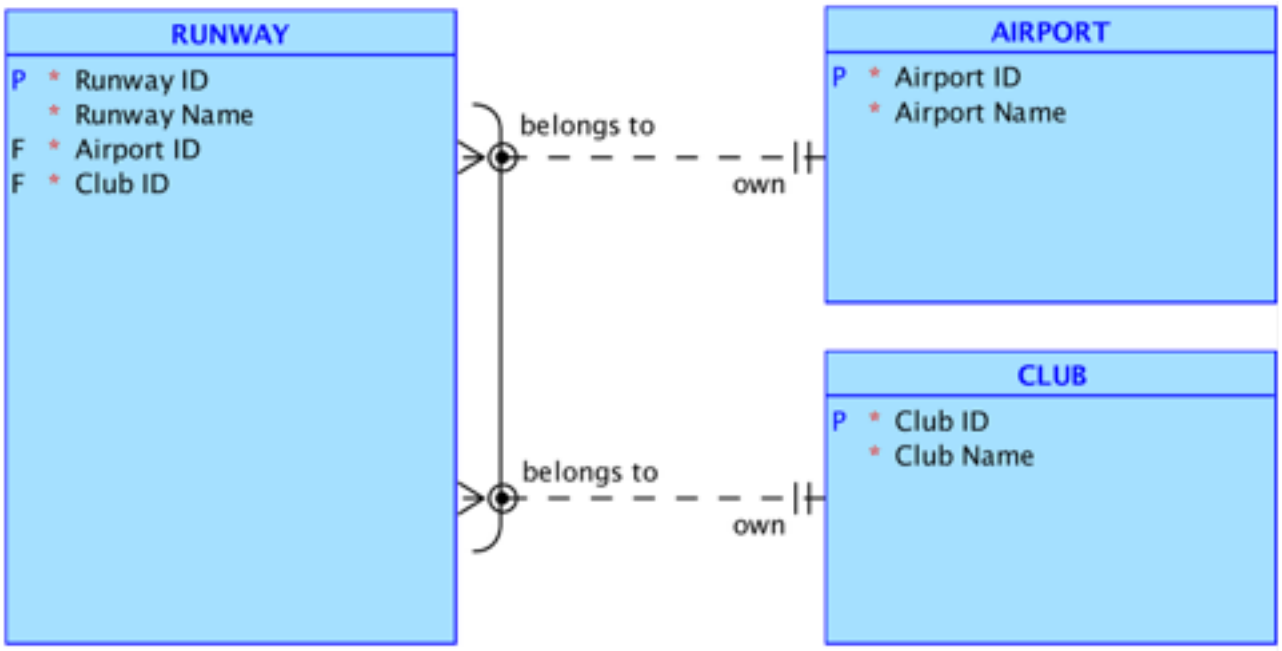


Relationship Exclusivity

There are situations where an entity is either related to one entity or the other but not both. This is called a relationship exclusivity and is generally implemented via a check constraint on the foreign key attributes within the target entity.

Consider the following scenario:

> A runway is either managed by an airport or else by a club such as the Motor Racing club or RC Planes Club.



The rules for an exclusive relationship arc are:

1. The minimum number of relationships is of 2
2. There can be more than one arc on the same relationship as long as it belongs to a different group
3. The arc acts upon the target entity only.

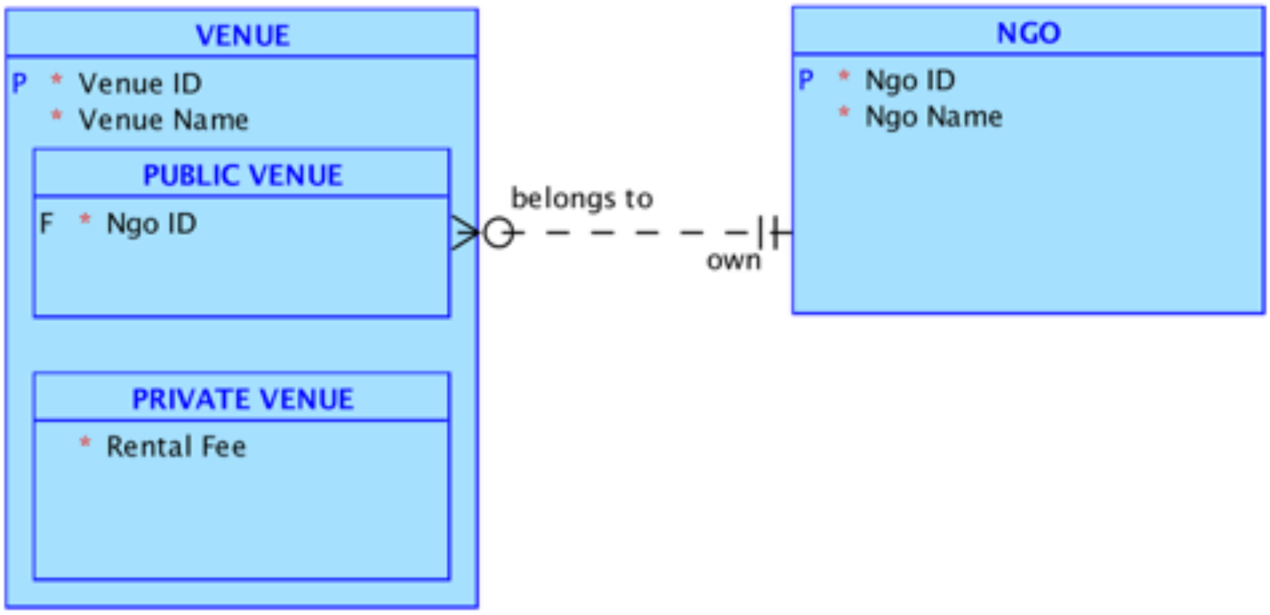
The logic follows an Exclusive Or Logic Gate (XOR) implying that either one and only one of the FK affected by the arc can have a value, the rest must all be null values.

Entity Subtypes

Entity generalisation refers to when two or more entities share common attributes and or relationships. The generic entity is referred to as the Entity Super type, whilst the specific entities are called the Entity Subtypes. The subtypes inherit all concepts found within the super type and must have a structure difference from the remaining subtypes. By structural difference we refer to an attribute and/or relationship.

Consider the following scenario:

> A table of venues stores the name of the location. If the venue is privately owned then a rental fee is specified. If the venue is a public location then it must be managed by a non-government organisation.



The concept here is that every venue is either a public venue or a private venue, not both and never none. So both a public venue and a private venue have a Venue Id and a Venue Name. Yet only a Public Venue is linked to an NGO and only a Private Venue has a rental fee.

There are different manners of implementing subtypes, with one of the most common being to:

1. implement just the super type entity as a table
2. implement all subtype attributes in the super type entity table as optional attributes
3. set a check constraint whereby only the attributes of one subtype can have a value, the rest must be all null.

Relationship Transferability

All relationships set so far were transferable, meaning that the target entity instance can be updated such that it is related to a different source entity instance. This is similar to a car being sold to a different person, so the owner of the vehicle changed. Yet there are certain situations where the relationship is non-transferable, meaning that it cannot change at all.

Consider the following scenario:

> A person is the author of zero, one or many books. A book is written by exactly one person (ok in real life a book can be co-authored, just consider this as a small example) and that is a fact that can never change.

There is no transferability symbol for Crow's Foot notation but other notations, such as Barker's notation do have one. This is implemented by not allowing any update operations on a FK.