2. Getting Started

2.0 Introduction

This Chapter discusses how to get started in Data Modelling.

It builds through a series of structured Steps.

You are invited to follow developments on our Web Site :-

- http://www.databaseanswers.org/index.htm

You can also join our Database Answers Community

- http://databaseanswers.ning.com/

Barry Williams

Principal Consultant

Database Answers Ltd.

barryw@databaseanswers.org

This Chapter covers similar material to Chapter 1 but it is more serious and more comprehensive.

We hope you like it and would be very pleased to have your comments at barryw@databaseanswers.org.

This material is also available as a Tutorial for Amazon and Starbuck on the Database Answers Website –

- http://www.databaseanswers.org/tutorial4_data_modelling/index.htm

We will cover these Basic Concepts :-

a. Creating Entities
b. Primary and Foreign Keys
c. One-to-Many and Many-to-Many Relationships
d. Hierarchies
e. Inheritance
f. Reference Data

At the end of this Tutorial, we will have produced a Data Model, which is commonly referred as an **Entity-Relationship Diagram**, or **'ERD'**

2.1 What is the Scope ?

Our Photo shows a typical Starbucks.
If we look closely, we can see people eating, drinking and placing orders.
What Starbucks sees are Customers, Products and Orders being met.

During the course of this Book we will see how Data Models can help to bridge this gap in perception and communication.

GETTING STARTED :
The area we have chosen for this Tutorial is a Data Model for a simple Order Processing System for Starbucks. We have done it this way because many people are familiar with Starbucks and it provides an application that is easy to relate to.
≠ We think about the area we are going to Model.
≠ We can see Customers ordering Products,(Food and Drinkg and so on).
≠ **My Approach** has three Steps :-

1. Establish the Scope of the Data Model
2. Identify the 'Things of Interest' that are within the Scope, These will be called **Entities**.
3. Determine the **Relationships** between them.

DECIDING THE SCOPE OF OUR DATA MODEL
≠ When we step inside, we see that Starbucks sells a wide range of Products, so our first task is to decide which of them should be included in our Data Model.
≠ Right now, we are interested only in something to eat and something to drink.
≠ Therefore, all the mugs and other items shown in this picture on the left, are outside the **Scope** of our Data Model, and are not **'Things of Interest'.**

2.2 What are the ' Things of Interest' ?

Our first step is to decide what Things are we interested in.

In other words, what is the Scope of our Data Model.

2.3 Creating Entities

This is how you create an Entity in Dezign

1. Right-click on a blank area in the diagram
2. From the drop-down list, choose Insert and Entity
2. Check the 'PK' box for the Primary Key attribute, which will usually be the first one on the Entity.
4. Click on Close to save the results.

2.4 Primary Keys

We decide that the Things we are interested in are Customers, Orders and Products.
You can buy a range of Products in Starbucks, including Cups, Coffee and Newspapers.

For the purpose of our first Model, we restrict our Products to Food and Drink.

This diagram shows the corresponding Entities with Primary Keys.

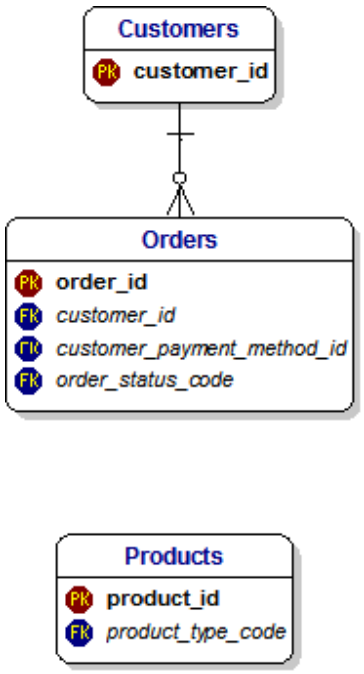
1. At this Stage, we show only the Entities with no Relationships and minimum Attributes. and specify only the Primary Key and one 'details' field that will be replaced later on..
2. The Primary Key field(s) should always be first.
3. You will notice that the first field in the Customers_version2 Table is the Customer_id.
4. It has a 'PK' symbol beside it, which indicates that it is the Primary Key for the Table.
5. The Primary Key is very important and is the way that we can recognise each individual record in the Table.

Creating a Primary Key in Dezign

1. Right-click on the Entity
2. Choose Attributes
2. Check the 'PK' box for the Primary Key attribute, which will usually be the first one on the Entity.
4. Click on Close to save the results.

2.5 Foreign Keys

This diagram shows Entities with Foreign Keys.
Customer_ID is a Foreign Key that links Orders to Customers.



Here we have added the **Relationships** between the **Entities**.

When this Primary Key is used in another Table, it is referred to as a 'Foreign Key'.

We can see a good example in this diagram, where the customer_id appears in the Customers_Payment_Methods Table as a Foreign Key.

This is shown with an 'FK' symbol beside it

Mandatory Key Fields

A Foreign Key is usually **mandatory**, in other words, a value for a customer_id in the Customers_Payment_Methods Table must correspond to an actual value of the customer_id in the Customers_Version_1 Table.

This is shown in the diagram by the short straight line at the end of the dotted line close to the Customers Table.

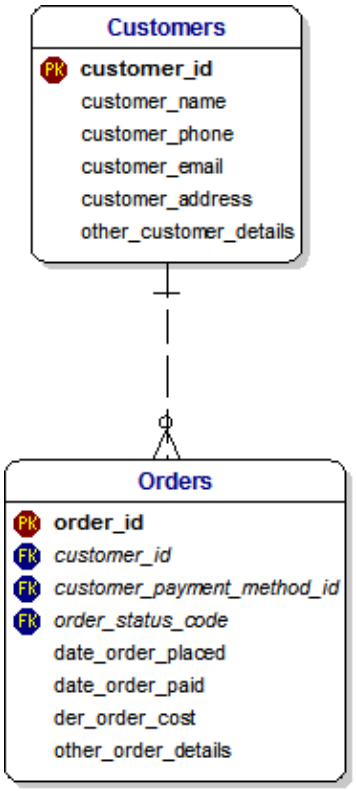
Foreign Keys in Dezign

Foreign Keys are created automatically when you make a Relationship between two Entities.

We recommend that you move the field up in the Entity so that it takes it place alphabetically among the Key fields.

To do this, right-click on the Entity choose the Attributes option, then click on the up or down arrow on the right-hand side.

2.6 One-to-Many Relationships



In this diagram, a Customer can place one or many Orders.

This is shown by the symbol that has three small lines at that end of the Relationship dotted line, which is referred to as **Crow's Feet**.

Optional Key Fields

Strictly speaking, a Customer does not have to place an Order.

He or she could change their mind and walk out without ordering anything.

In other words, we would say that the Relationship is **optional** at the Orders end.

This is shown by the little 'O' at that end of the Relationship dotted line.

A Customer can place many Orders.

This defines a One-to-Many Relationship.

A Data Modeller would say “For every Customer, there can be zero, one or many Orders”.

TERM	DEFINITION
Customer	Any Unit that can raise a Demand

Demand	A request for Assets to be supplied. The format of a request can be an electronic message, a paper Form and so on.

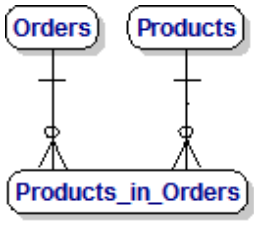
Business Rules : A Customer can raise zero, one or many Demands.
: A Demand must be associated with a valid Customer.

2.7 Many-to-Many Relationships

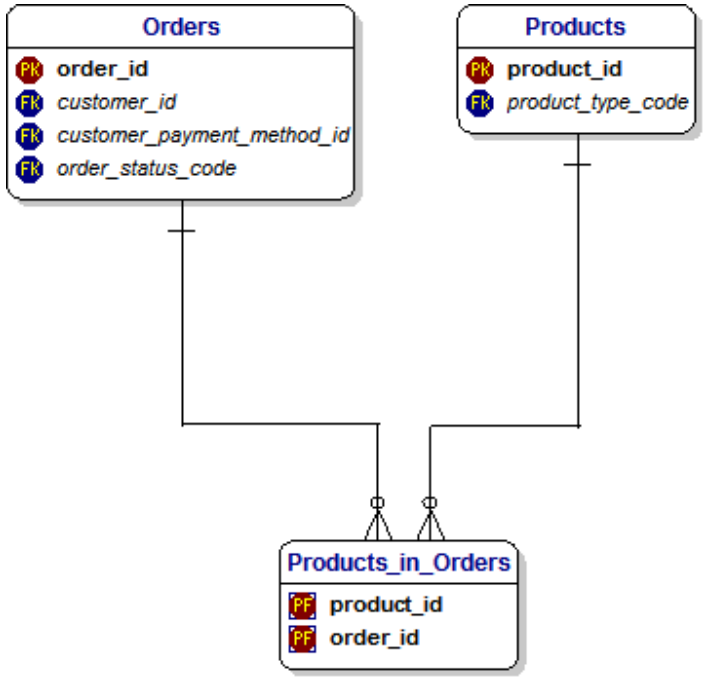
This diagram shows a many-to-many relationship between Orders and Products.
An Order can include many Products and a Product can appear on many Orders.

We can also say that a Demand can request many Products.
A Data Modeller would say “A Demand can request many Products, and each Product can be in many Demands”.
This defines a Many-to-Many Relationship and is shown in a Data Model as follows :-

Many-to-Many Relationship cannot be implemented in Relational Databases.
Therefore we resolve this many-to-many into two one-to-many Relationships, which we show in a Data Model as follows :-

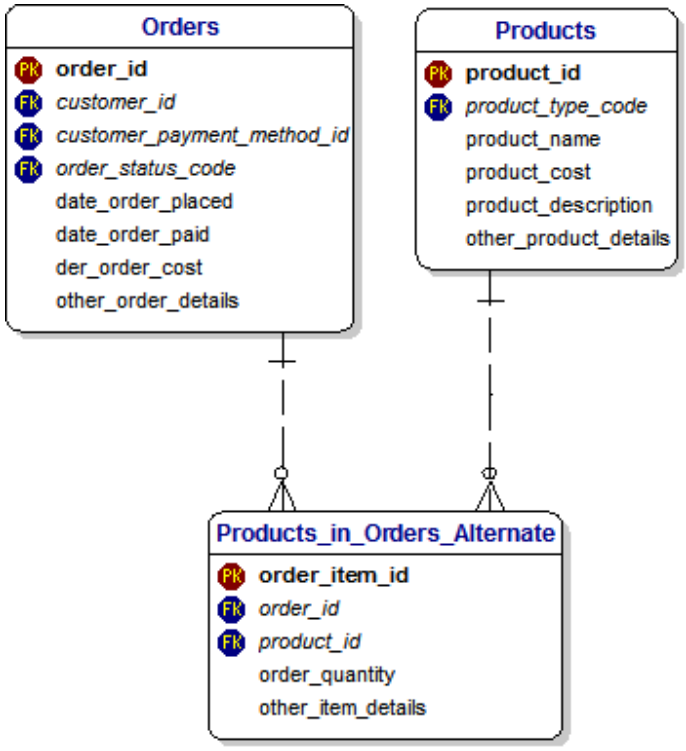


Sometimes it is good to see the Key fields to ensure that everything looks alright.



When we look closely at this Data Model, we can see that the Primary Key is composed of the Order_ID and Product_ID fields.
This reflects the underlying logic, which states that every combination of Order and Product is unique.
In the Database, this will define a new record.
When we see this situation in a Database, we can say that this reflects a many-to-many Relationship.
However, we can also show the same situation in a slightly different way, which reflects the standard design approach of using a surrogate key as the Primary Key and showing the Demand and Product IDs simply as Foreign Keys.

The benefit of this approach is that it avoids the occurrence of too many Primary Keys if more dependent Tables occur where they cascade downwards.
The benefit of the previous approach is that it avoids the possibility of 'orphan' records in the 'Products in a Demand' table.
In other words, invalid records that have invalid Demand ID and/or Product ID values.

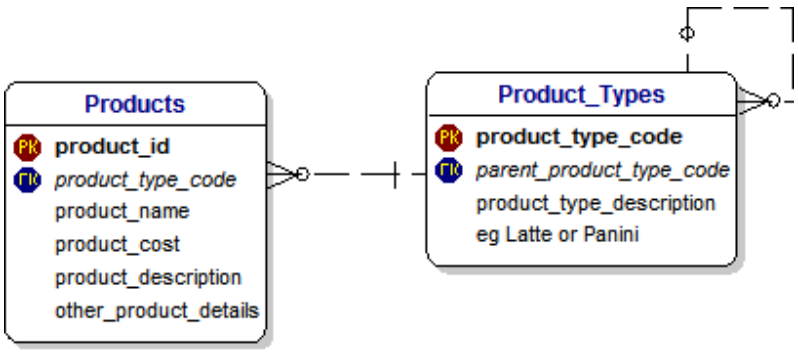


TERM	DEFINITION
Order	A request for Products to be supplied. The format of a request can be an electronic message or a paper Form.
Product	An Item that can be supplied on request. It can be something small, like a Muffin, or something that contains other Products, like a sandwich with multiple fillings.

Business Rules : A Order can refer to one or many Products.
: A Product can appear in zero, one or many Orders.
: In other words, there is a Many-to-Many Relationship between Demands and Products.

2.8 Hierarchies and Rabbit's Ears

Hierarchies are very common and we can see them all around us.
Fortunately, we can handle them every easily in Data Models.



This diagram shows how the hierarchies of Products and Product Types that we have just discussed are shown in our **Entity-Relationship Diagram**.

You will notice that the table called 'Product_Types_v1' has a dotted line coming out on the right-hand side and going back in again on the top-right corner.

Data Analysts call this a Reflexive Relationship, or informally, simply 'Rabbits Ears'.
In plain English, we would say that the Table is joined to itself and it means that a record in this Table can be related to another record in the Table.
This approach is how we handle the situation where each Product can be in a hierarchy and related to another Product.
For example, a Product called Panini could be in a Product Sub-Category called 'Miscellaneous Sandwiches' which could be a higher Product Category called 'Cold Food', which itself could be in a higher Product Super-Category called simply 'Food'.
Next time you go into Starbucks, take a look at the board behind the counter and try to decide how you would design the Products area of the Data Model.
You should **pay special attention** to the little 'zeros' at each end of the dotted line.
These are how we implement the fact that the 'Parent Product Type Code' is optional, because the highest level will not have a Parent.

This Tutorial is also available in the Database Answers Website :-

- http://www.databaseanswers.org/tutorial4_data_modelling/index.htm

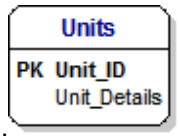
A number of Data Models show examples of Inheritance, including :-

- Charities
- City Tourist Guide

- CMDB - Configuration Mgt DB
- Customers Commercial and Personal
- Event Registrations
- Games Store
- Insurance Brokers
- Libraries for Lawyers
- National Trust (UK)
- New Egg
- Photo Catalogs
- School Management Systems
- Shrek 2 Movie
- Tracking Manufactured Items
- Travel & Tourism Worldwide
- Vehicle Imports

- In the Military

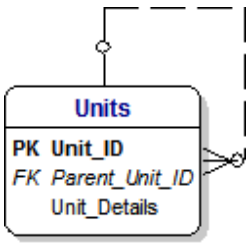
We start with the definition of a Unit, which at its simplest, looks like this :-
In this case, we use a meaningless ID for the Unit ID which is simply a unique number.



Then we think about the fact that every Unit is part of a larger organisation.
In other words, every Unit reports to a higher level within the overall organisation.

Fortunately, we can show this in a very simple and economical fashion by creating a relationship that adds a parent ID to every Unit.
This is accomplished by adding a relationship that joins the table to itself.

This is formally called a Reflexive relationship, and informally called 'Rabbits Ear's, and it looks like this :-



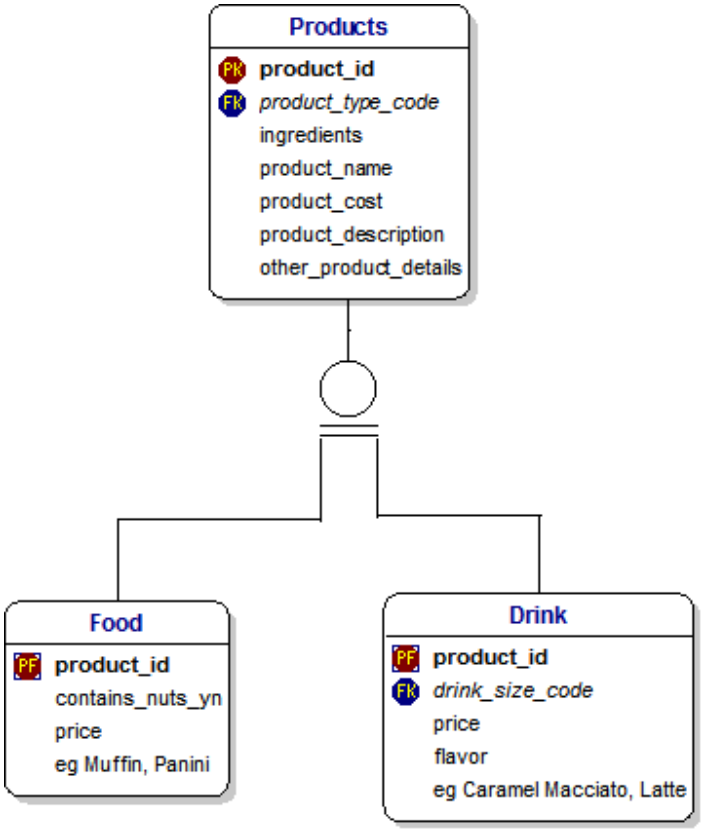
The Unit at the very top of organisation has no-one to report to, and a Unit at the lowest level does not have any other Unit reporting to it.

In other words, this relationship is Optional at the top and bottom levels.

We show this by the small letter ‘O’ at each end of the line which marks the relationship.

2.9 Inheritance

Inheritance is a very powerful technique.
It allows us to model complex situations in a manner and style that is very simple.



In this situation, we are thinking about Food and Drink.

Food and Drink are specific examples of the more general Thing called a Product.
They inherit common attributes from the Product, and also have some of their own.
For example, Food can contain Nuts but Drink may not contain nuts, but both have a Product Name.

The unusual symbol in the middle of the diagram, composed of a circle with two small lines underneath it is how Inheritance is shown with the Dezign Data Modelling Tool that I am using.

Inheritance is a very important topic when you are creating a Data Model.
In plain English, we would say that Inheritance occurs where a Parent-Child relationship exists between Things of Interest (or Entities).

You can ask the simple **'Is-a' question** - in this case, if we ask 'Is a Book a Product' then clearly the answer is 'Yes' so we think there is an Inheritance relationship between them.

In the example of Inheritance shown in this diagram, we can see that all Products have Names and Descriptions.

Therefore, Books, Food and Drink will inherit these characteristics from the parent Product.

However, each type of Product will have specific characteristics that it does not share with other types of Products. For example, Books have ISBNs and Authors, but Food and Drink do not.

One of the important things in your Data Model is to be sure you have identified all the Inheritance relationships.
However, from many years of experience as a DBA, I should point out that relationship is often blurred in a real physical Database because it can be clumsy to implement.

Inheritance can appear in a Logical Data Model but it disappears in the Physical Database, which is what ultimately becomes the Database.
The reason is that Relational Databases do not support Inheritance.

Inheritance is a very simple and very powerful concept.
We can see examples of Inheritance in practice when we look around us every day.

For example, when we think about ‘Houses’, we implicitly include Bungalows and Ski Lodges, and maybe even Apartments, Beach Huts and House Boats.

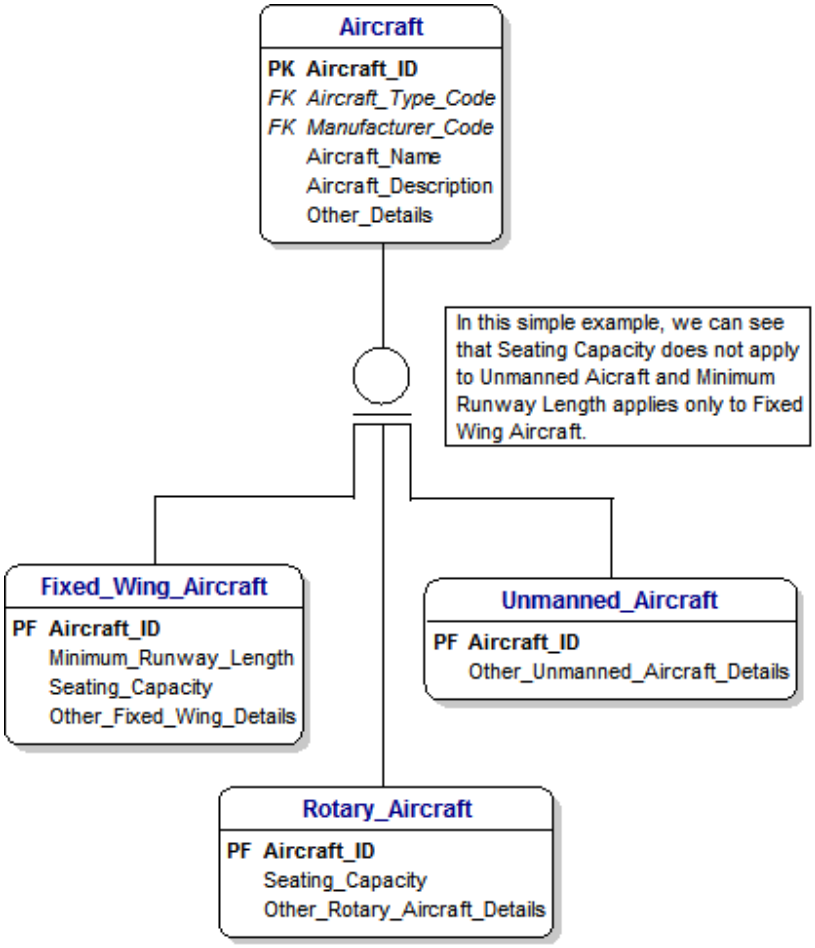
In a similar way, when we discuss Aircraft we might be talking about Rotary Aircraft, Fixed Wing Aircraft and Unmanned Aircraft.

However, when we want to design or review a Data Model that includes Aircraft, then we need to analyse how different kinds of Aircraft are shown in the design of the Data Model.

We use the concept of ‘Inheritance’ to achieve this.
Inheritance is exactly what it sounds like.
It means that at a high level, we identify the general name of the ‘Thing of Interest’ and the characteristics that all of these Things share.
For example, an Aircraft will have a name for the type of Aircraft, such as Tornado and it will be of a certain type, such as Fixed Wing or Rotary.

At the lower level of Fixed-Wing Aircraft, an Aircraft will have a minimum length for the runway that the Aircraft needs in order to take off.

This situation is shown in the following diagram :-


Aircraft
PK Aircraft_ID
FK Aircraft_Type_Code
FK Manufacturer_Code
Aircraft_Name
Aircraft_Description
Other_Details
In this simple example, we can see that Seating Capacity does not apply to Unmanned Aicraft and Minimum Runway Length applies only to Fixed Wing Aircraft.
Fixed_Wing_Aircraft
PF Aircraft_ID
Minimum_Runway_Length
Seating_Capacity
Other_Fixed_Wing_Details
Unmanned_Aircraft
PF Aircraft_ID
Other_Unmanned_Aircraft_Details
Rotary_Aircraft
PF Aircraft_ID
Seating_Capacity
Other_Rotary_Aircraft_Details

2.10 Reference Data

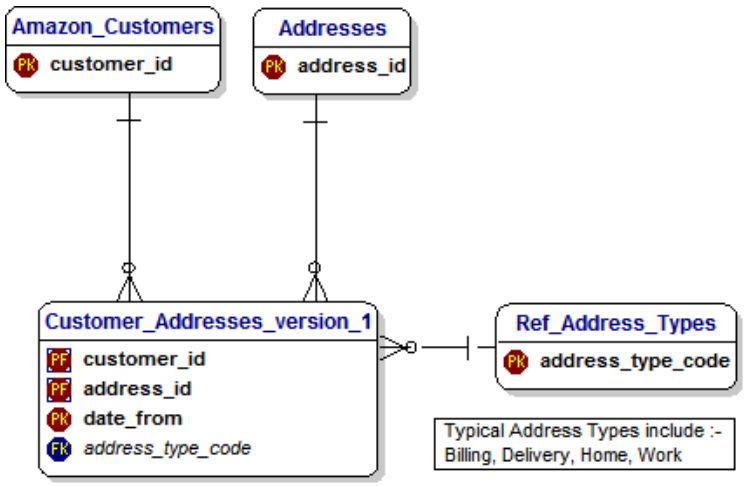
Reference Data is very important.
Wherever possible, it should conform to appropriate external standards, particularly national or international standards.
For example, the International Standards Organization ('ISO') publishes standards for Country Code, Currency Codes, Languages Codes and so on.

For Addresses, the UK Post Office Address File, 'PAF' File, is used to validate Addresses within the UK.

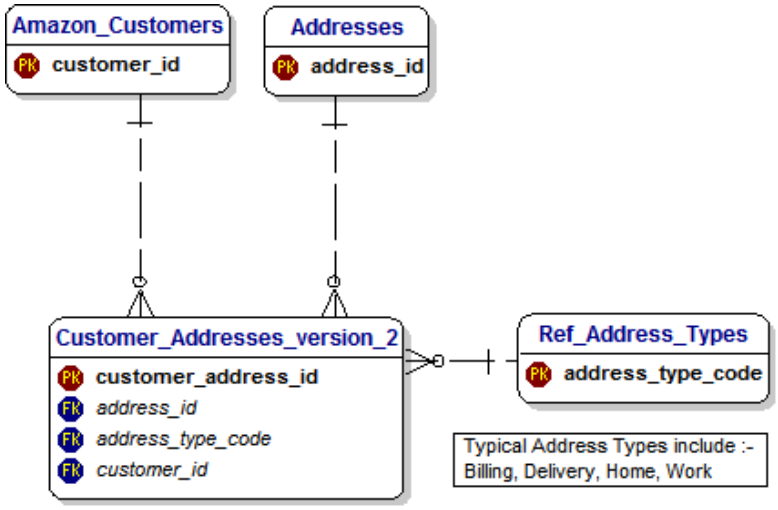
2.10.1 Address Types example

Another simple example of Reference Data is Address Types.

There are two design possibilities.
The first is good because it shows clearly the logical relationship where a Customer Address can be identified uniquely by a combination of the Customer ID, the Address ID and the Date From when the Address was valid for the Customer.



Of course, it is not always possible to determine the 'Date From' value, and it is not always something that it is appropriate to ask every Customer.
Therefore, a better general approach is to use a Surrogate Key for a record and leave the 'Date From' field optional.



Reference Data has the following characteristics :-

- it does not change very much
- it has a relatively small number of values, usually less than a few dozen and never more than a few hundred.
- Therefore we can show it with a Code as a Primary Key.
- Data in Reference Data Tables can be used to populate drop-down lists for Users.
- In this way, it is used to ensure that all new data is valid.

Standards

- In the Address Table, you will see a field called 'ISO_Country_Codes'.
- ISO stands for the 'International Standards Organisation'.
- It is always good to use national or international standards.

Customer Addresses

This is a general and flexible approach to handling Addresses in our Data Model.

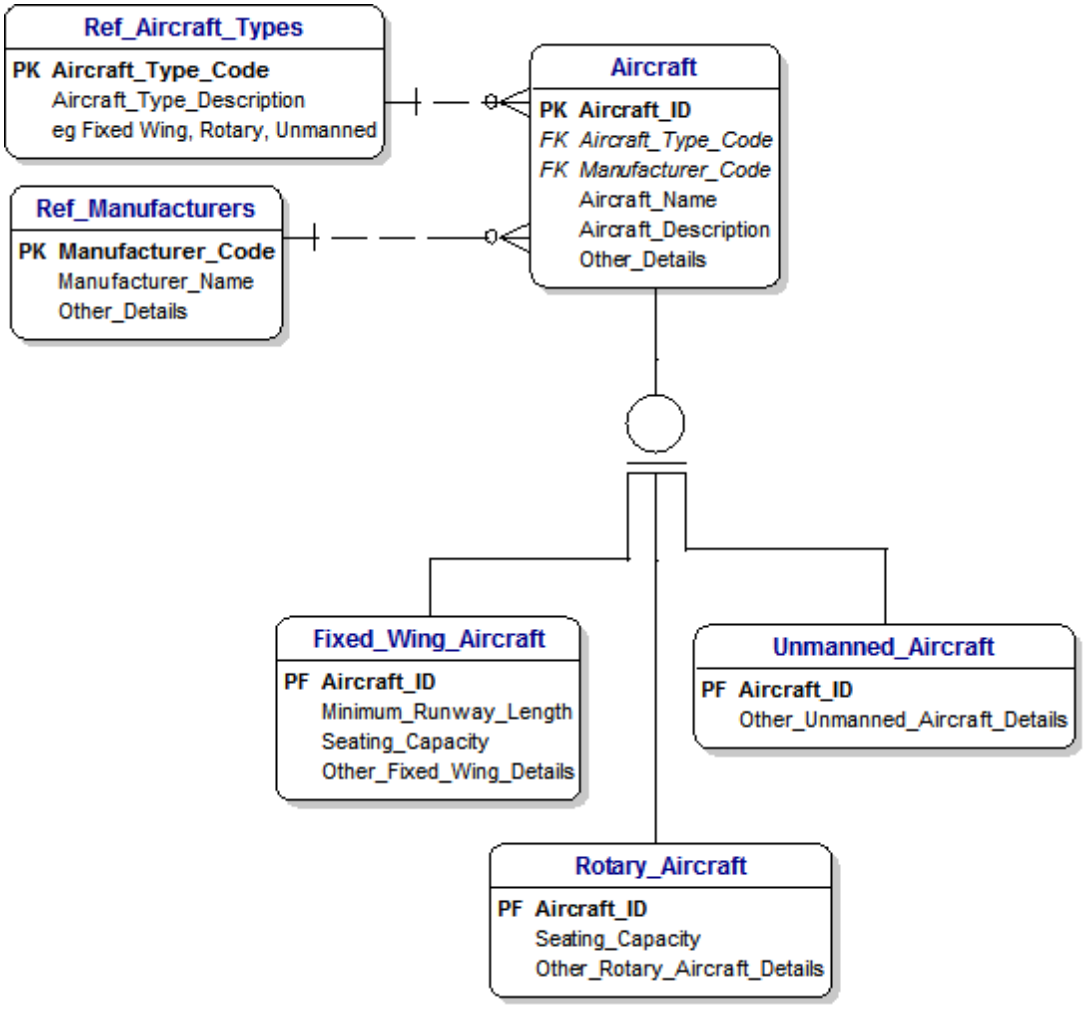
We have a separate Address Table, which allows us to have more than one Address for any Customer very easily.

This design also has other benefits :-

- We can accommodate more than one person at the same Address. We need to do this because different members of a family may sign-up separately with Amazon.
- With a separate table of Addresses, we can easily use commercial software to validate our Addresses.
- To find this kind of software, simply Google for "Address Validation Software".
- We have used QAS with great success in the past.
- With this approach, we can always be sure that we have 100% good Address data in our Database.

2.10.2 Aircraft example

This diagram shows two basic examples of Reference data that might apply to our simple Aircraft Data Model.



2.11 Topics we have covered

In this Chapter, we have covered the basic concepts in Data Modelling, which are :-

- Primary and Foreign Keys
- One-to-Many and Many-to-Many Relationships
- Rabbit's Ears or Reflexive Relationships
- Inheritance

We can see examples of each of these in this Data Model :-