3. How to Understand a Data Model

3.0 Welcome

Barry Williams

Principal Consultant

Database Answers Ltd.

barryw@databaseanswers.org

3.1 Introduction

3.1.1 What is this ?

This Chapter is a Tutorial to help you in looking at a Data Model, understanding it and determining whether it is of an acceptable quality.

3.1.2 Why is it important ?

It is important because it helps you to understand a Data Model, even if it is not one of your principal concerns.

3.1.3 What will I Learn ?

You will be learn how to read a Data Model, so that you will be comfortable looking at any Model, regardless of the notation and style and you will be able to understand the underlying logic.

The approach is largely based on the concept of Design Patterns which are general solutions to common problems which occur on a regular basis.

This document starts with some simple Concepts and then discusses common Design Patterns based on these Concepts.

The document applies in two situations :-
i) Data Models created by Reverse Engineering existing Databases.
ii) Other Data Models.

This document will help in the Quality Assurance ('QA') of these Data Models, which might be produced internally or externally, by Partners, for activities such as Data Migration.
.
i) For the first situation, it is not appropriate to try to do a Quality Assurance of the Model.
This is primarily because Databases in operational systems have usually gone through a series of changes and usually the impact on design has not been thought through and there has not been time to redesign the Database.
The objective is primarily to understand the Database.

The many-to-many Pattern will not occur because this cannot be implemented directly in a Relational Database.
This applies also to Inheritance (SEE Section 2.4) which can only be identified by implication when the Model for the Database is examined..

It is often useful to create a general Business Data Model that renames Tables as appropriate to replace the physical Table names with corresponding Business Terms.
This is different from a Logical Model and can usually be implemented in Microsoft Word, rather than a Data Modelling Tool.

For complex Databases, it is usually valuable to create a Top-Level Data Model with lower-level Subject Area Models.

It is important to try to establish a Glossary of Terms, covering descriptions of the most important Tables and Attributes and Reference Data.
Another important activity is to establish the Business Rules that define the logic underlying any Database.
Some simple examples that can be used as Templates have been shown in this document.
ii) For the second situation, it is appropriate to do a Quality Assurance of the Model
This would include a number of tasks, such as :-

- looking for examples of the Design Patterns being used where appropriate.
- review of the Reference Data

3.2 Types of Data Models

There are three different types :-
1. Business Data Model.
This can also be called a Conceptual Model because it focuses on the important 'Things of Interest' and how they are related. It can be created in Microsoft Word and is very useful for discussion with business users.
2. Logical. This usually shows Primary and Foreign Keys. It is invariably produced in a Data Modelling Tool like Dezign or ERWin.
3. Physical. This is usually close to the design of the Database.

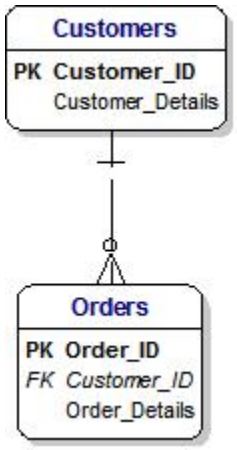
Conceptual Models are often Business Data Models, intended to be understood by non-technical Users.
Logical Models add Primary and Foreign Keys.
Physical Models are often used to generate SQL to create Database Tables.
They can also be created by reverse engineering from an existing operational Database.

3.3 Concepts

3.3.1 One-to-Many Relationships

A Customer can place many Orders for Products.
This defines a One-to-Many Relationship.

A Data Modeller would say "For every Customer, there are many Orders".
This is shown in a Data Model as follows :-



Sample Template :

TERM	AUTHOR	DEFINITION
Customer	Joe Bloggs	Any Person or Organisation that can raise an Order
Order	Joe Bloggs	A request for Products to be supplied. The format of a request can be an Online Form, a paper Document and so on.

Business Rules : A Customer can raise zero, one or many Orders.
: An Order must be associated with a valid Customer.

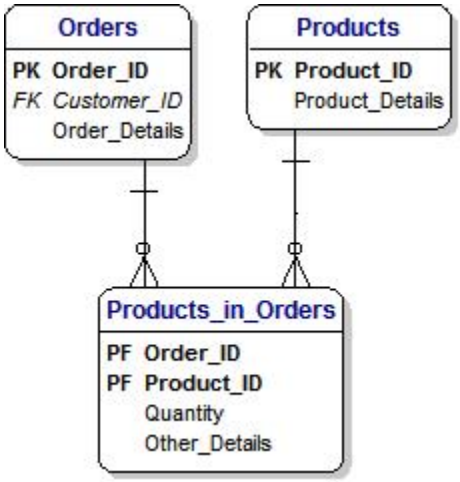
Blank Template :

TERM	AUTHOR	DEFINITION

3.3.2 Many-to-Many Relationships

We can also say that an Order can request many Products.
A Data Modeller would say "An Order can request many Products, and each Product can be in many Orders".
This defines a Many-to-Many Relationship and is shown in a Data Model as follows :-

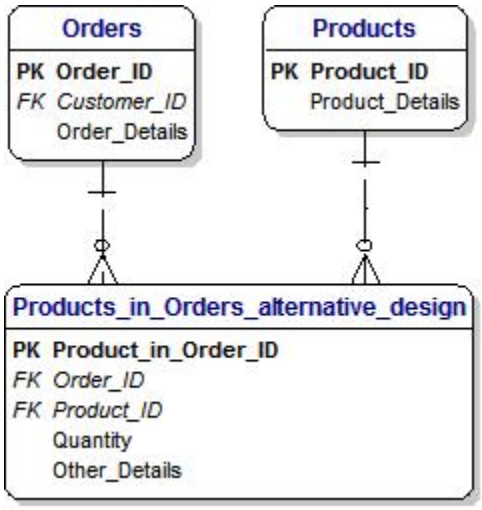
Many-to-Many Relationship cannot be implemented in Relational Databases.
Therefore we resolve this many-to-many into two one-to-many Relationships, which we show in a Data Model as follows :-



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 Order 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 an Order' table.
In other words, invalid records that have invalid Order ID and/or Product ID values.



TERM	AUTHOR	DEFINITION
Order		A request for Products to be supplied. The format of a request can be an electronic message, a paper Form and so on.
Product		An Item that can be supplied on request. It can be something tangible, like a Car, or something intangible, like an Insurance Policy.

Business Rules : An 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 Orders and Products.

3.3.3 Rabbit's Ears

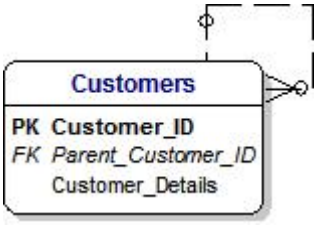
We start with the definition of a Customer, which at its simplest, looks like this :-
In this case, we use a meaningless ID for the Customer ID, which is simply an automatically generated unique number.

.

Then we think about the fact that every business Customer is part of a larger organisation.
In other words, every Customer 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 Customer.
.This is accomplished by adding a relationship that joins the table to itself.

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



The Customer at the very top of organisation has no-one to report to, and a Customer at the lowest level does not have any other Customer 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.
Note that we have positioned the Parent_Customer_ID field immediately below the Customer_ID field.
This is in line with our Best Practice policy of putting all Key fields at the beginning of the list, with Primary (PK) fields first.

3.3.4 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.

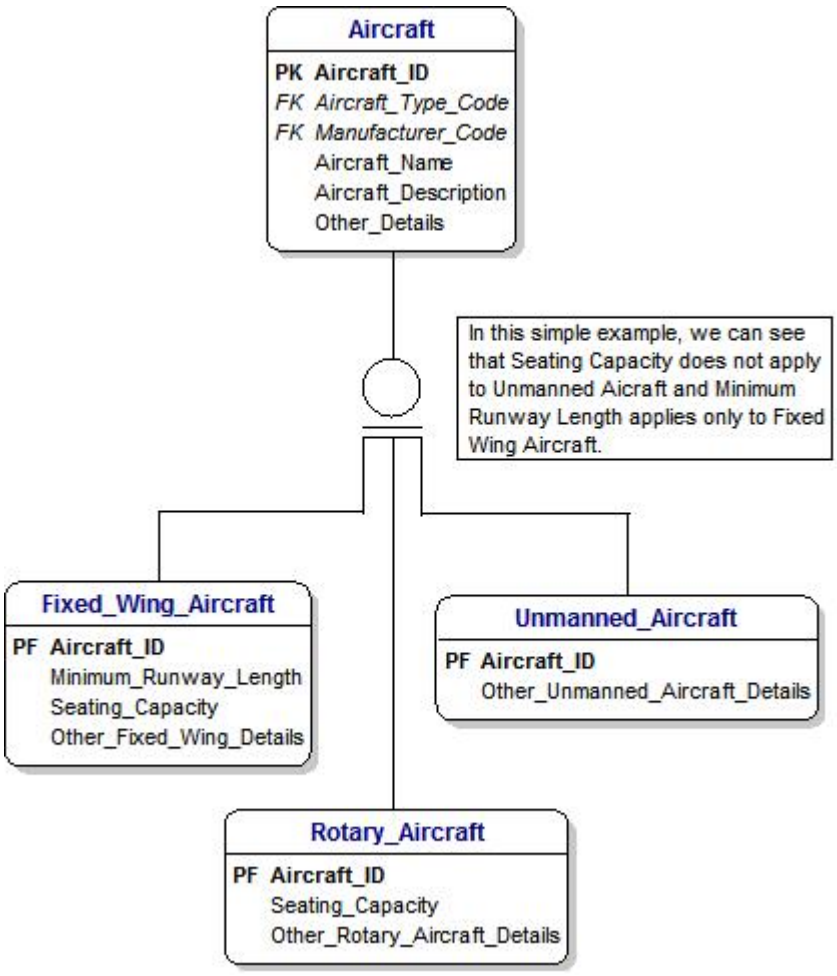
In a similar way, when we discuss Aircraft we might be talking about Rotary Aircraft, Fixed Wing Aircraft and Unmanned Aircraft.

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 always 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 :-

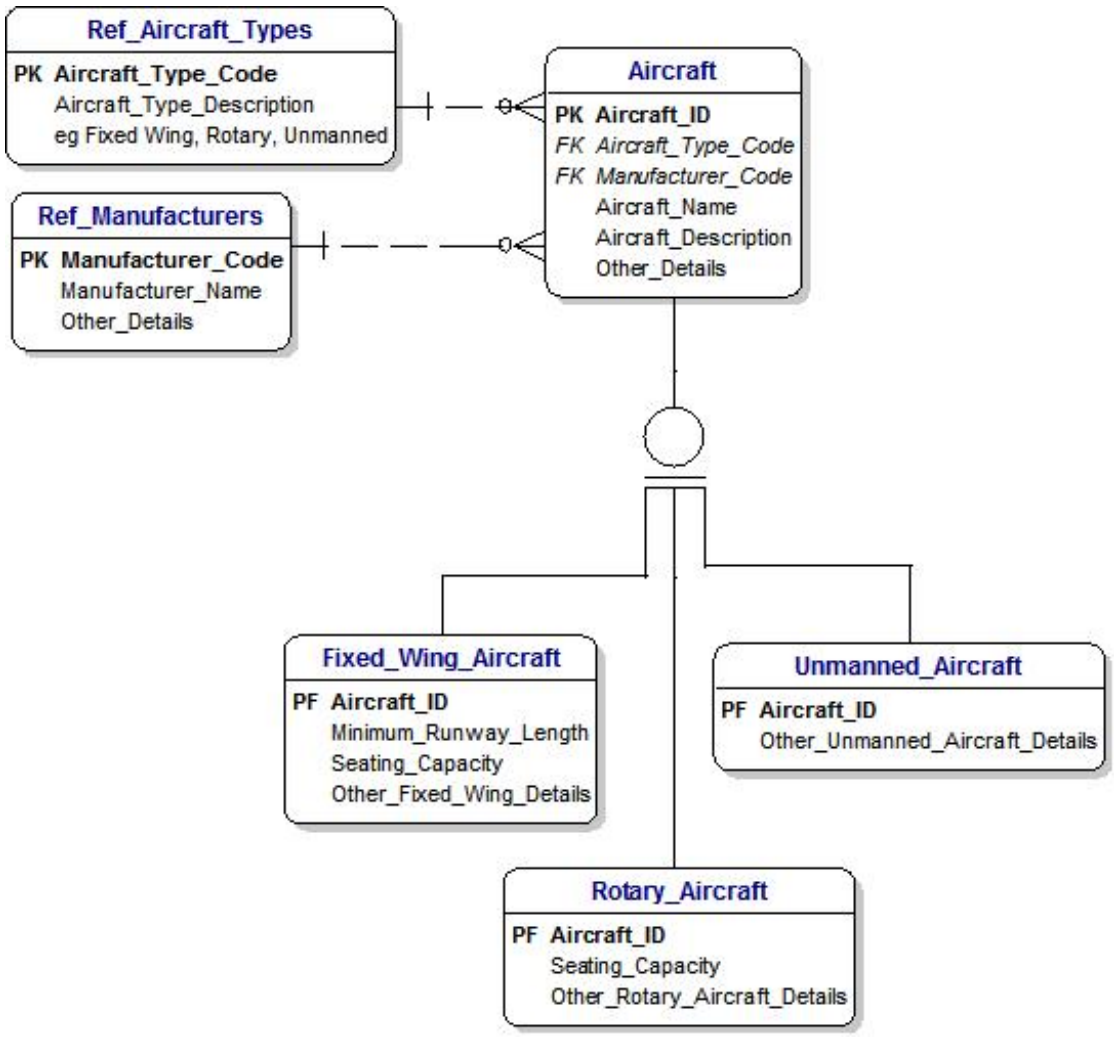


3.3.5 Reference Data

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.
For Customers, the overall structure is classified.

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

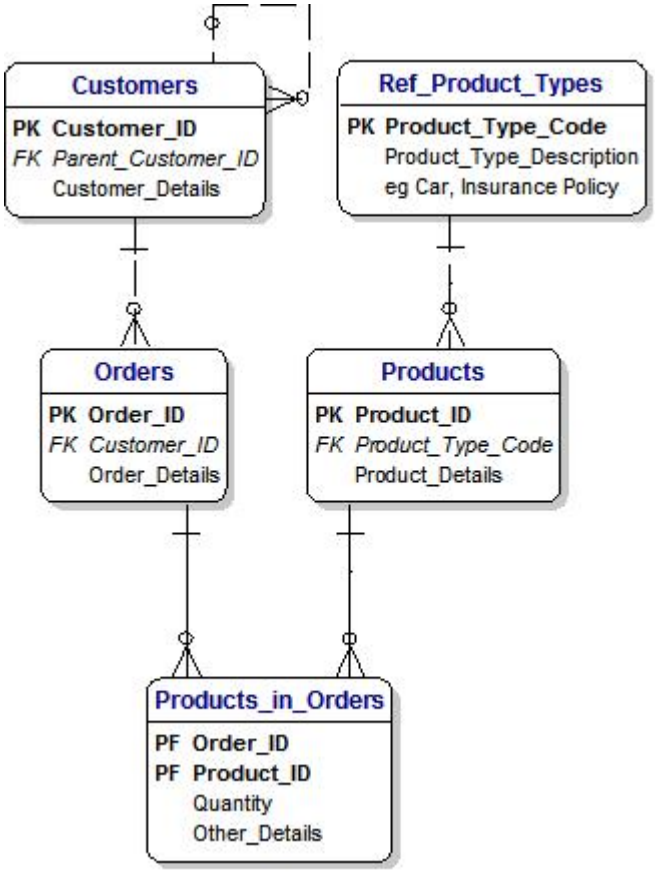


3.4 Data Warehouses in contrast to ERDs

Different considerations apply to Data Warehouses than apply to ERDs.
For the purpose of this discussion, we include Data Marts with Data Warehouses.

3.4.1 Design of an ERD

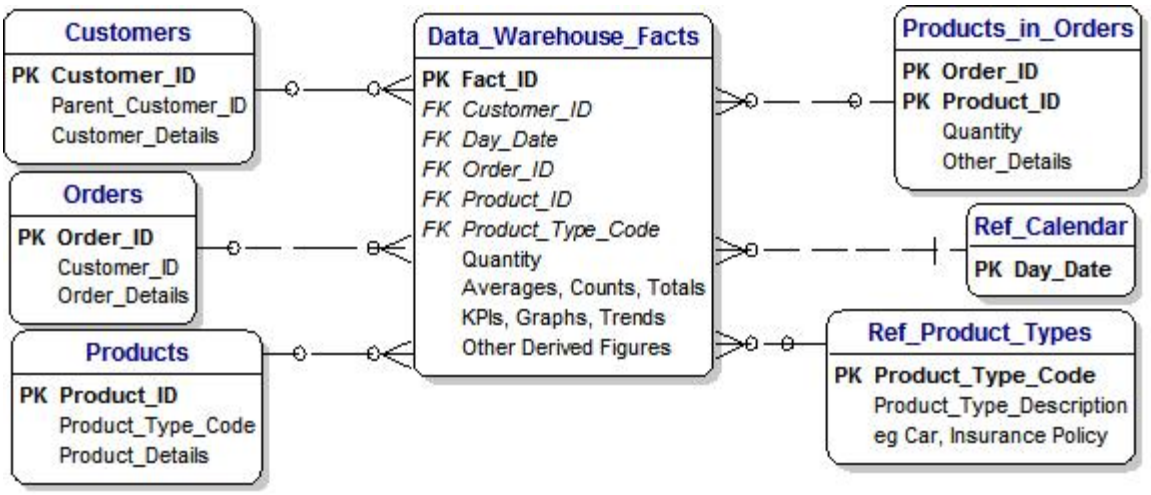
This Data Model is an Entity-Relationship-Diagram ('ERD') for Customers and Orders.
In passing, we observe that we use the prefix 'Ref_' to indicate Reference Data.



We could describe it in these terms :-
"Customers place Orders for Products of different Types."

3.4.2 Design of a Data Warehouse

This Data Model shows the corresponding Data Warehouse for Customers and Orders.
The design of this Data Warehouse simply puts all data into a 'big basket' to satisfy any request for information from management and the business community.



3.4.3 Reviewing the Design of a Data Warehouse

The design of any Data Warehouse will conform to this pattern with Dimensions and Facts.
Dimensions correspond to Primary Keys in all the associated Tables (ie the Entities in the ERD) and the Facts are the derived values that are available.

Therefore, reviewing the Design of a Data Warehouse involves looking for this Design Pattern.

With one exception, the Relationships are optional because the Enquiries need not involve any particular Dimension.
The one exception to this rule is that the Relationship to the Calendar is mandatory because an Enquiry will always include a Date.
Of course, an Enquiry might include all data since the first records, but the principle still applies.

The purpose of the Data Warehouse is to make it easy to retrieve data in any combination in order to answer questions like this :-

- Which Customers ordered the most Products ?
- Which were the most popular Products in the first week of April ?
- What was the average time it took to respond to Orders for Washing Machines ?
- How many Orders did we receive in May ?

3.5 Design Patterns

3.5.1 Addresses

Addresses can be a problematic area to handle correctly in Databases.
The usual approach is to simply store :-

- Three lines for the Address, called simply Line_1, Line_2 and Line_3.
- The Town or City
- The Postcode
- The County
- The Country

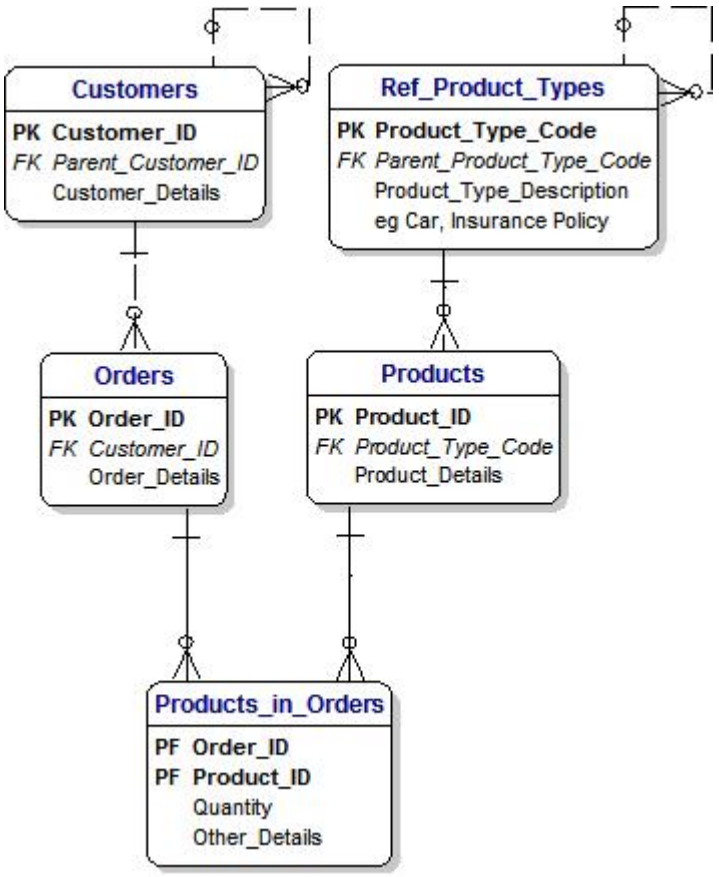
A more professional approach is to store Addresses in a dedicated Address File.
This allows the use of commercial software to validate the Addresses.
For UK Addresses, the most professional approach is to use the Post Office PAF File for validation of Addresses and to support a standard format for storing Addresses.
A variety of vendors, such as QAS, offer software to use the PAF File for validation.

3.5.2 Customers and Orders

The design of the ERD in the Chapter on Data Warehouses shows a typical Customers and Orders Data Model which represent a widespread kind of application.

3.5.3 Customers, Products and Orders

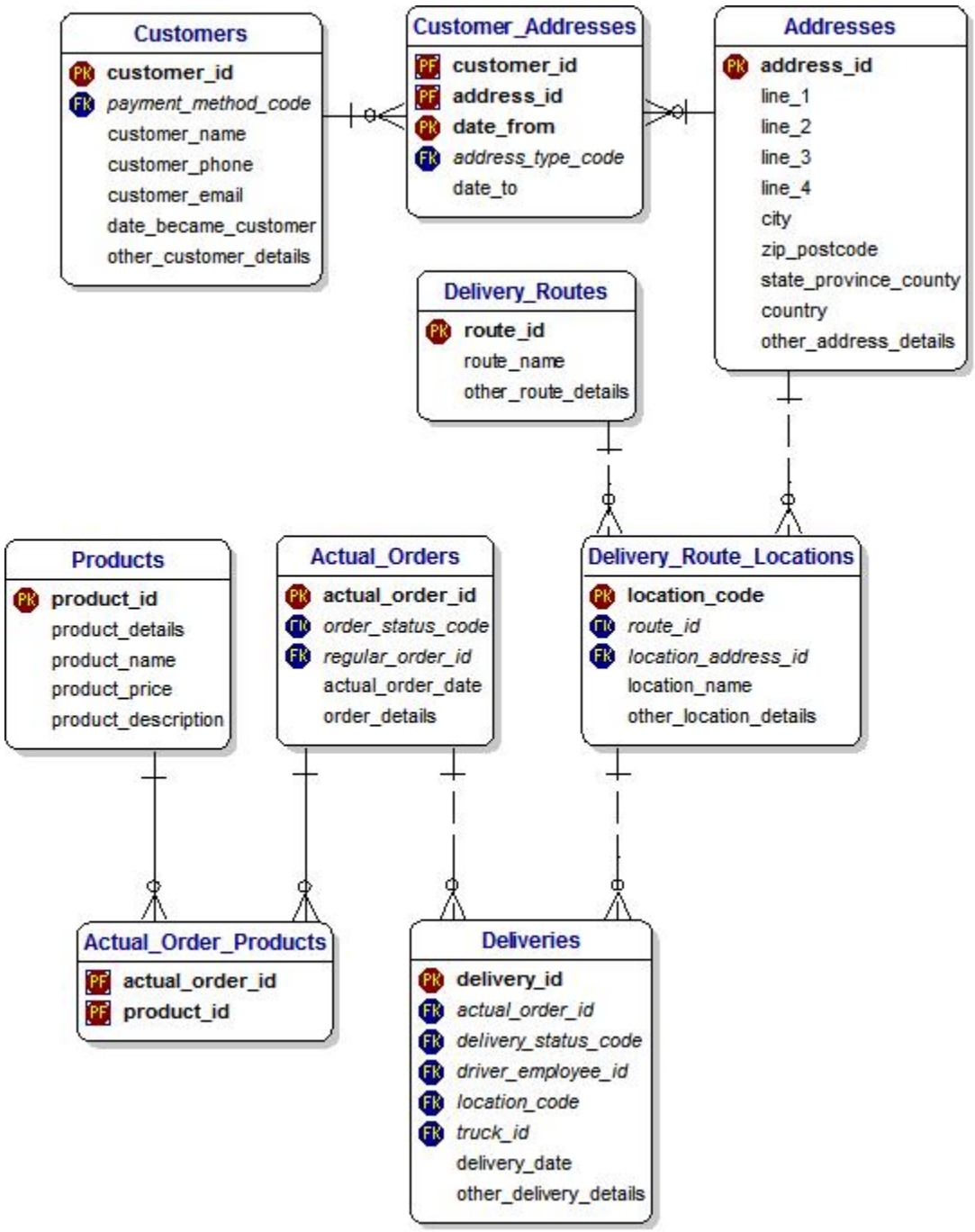
This Data Model demonstrates the power of 'Rabbit's Ears'.
It shows Customers in an organisational hierarchy and Product Types in a hierarchical Catalogue.



3.5.4 Deliveries

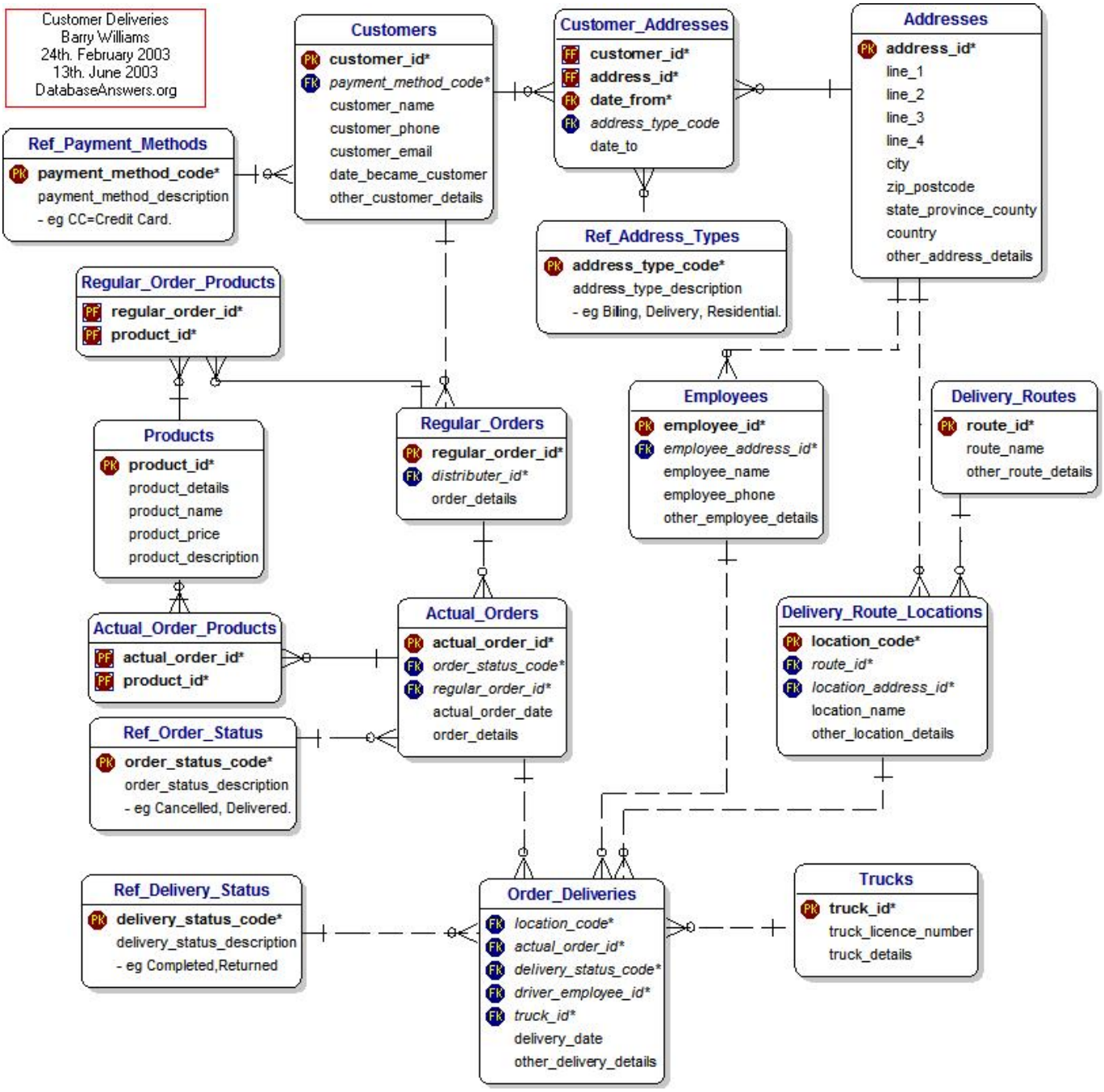
3.5.4.1 A Simple Design Pattern

This Data Model covers the activities of delivering an Order to a Customer at a designated address.
The process of reviewing a Data Model is to ask :-
"How do I describe the Business Rules behind this Model ?"
In this case, we could say "A Customer can raise an Order for Products to be delivered to a specified Address".



3.5.4.2 A Complex Design Pattern

This shows a complex Pattern which adds Regular Orders to the Simple Model shown above.

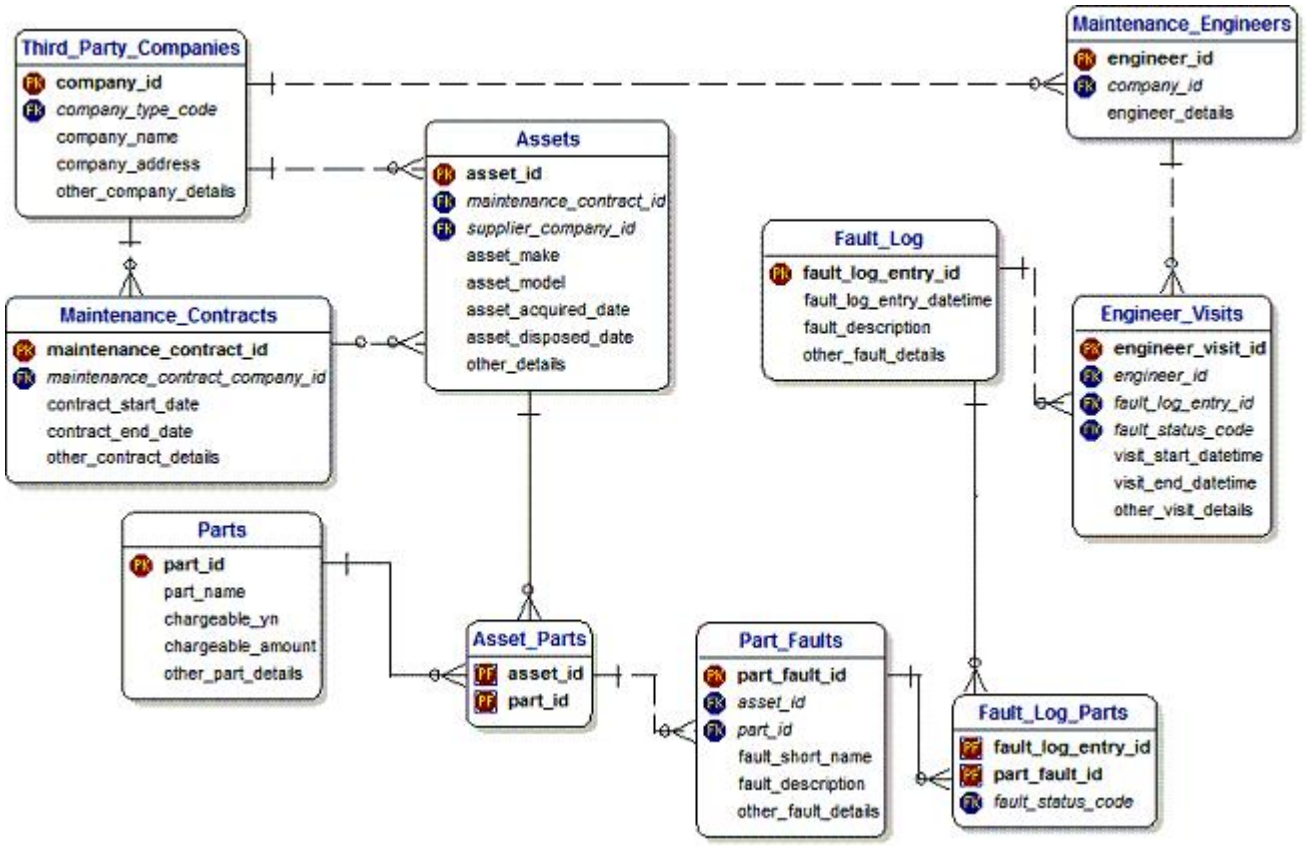


3.5.5 Maintenance

3.5.5.1 Scope

The scope of this Data Model is the Maintenance of Products by Third-Party Companies.
The Business Rules state :-

- An Product such as a Washing Machine can have a Maintenance Contract.
- An Product can consist of Product Parts :-
  - Faults occur with these Parts from time to time.
  - Third Party Companies employ Maintenance Engineers to maintain these Products.
  - Engineers pay Visits which are recorded in a Fault Log.
  - They correct the Faults and this is recorded in the Fault Log.



3.5.6 Subject Areas

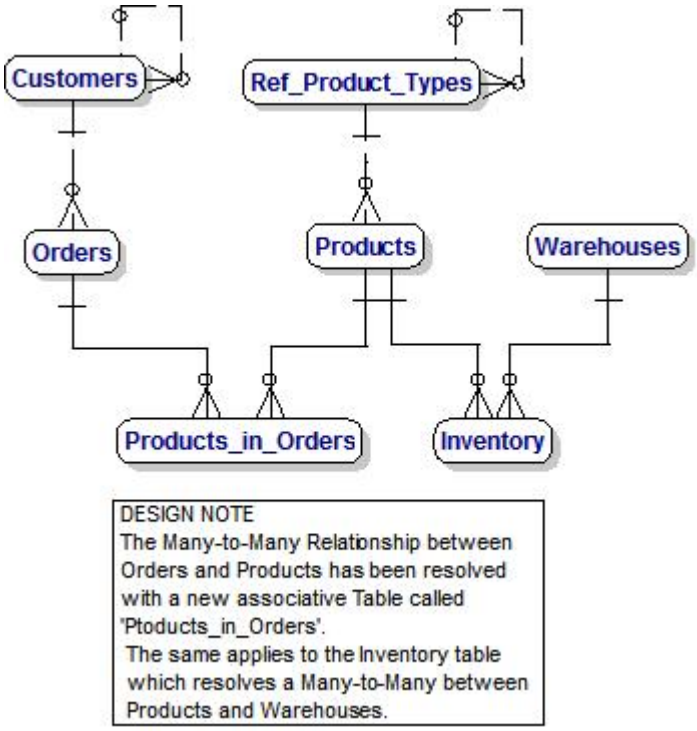
Complex Data Models which are common in large organisations can best be understood when they are broken down into a Top-Level Model and Lower-Level Subject Areas.
Typical Subject Area Models are Deliveries and Maintenance.

These are shown in earlier Sections of this document.

3.5.6.1 Top-Level Model

*** DOES THIS ADD VALUE ??? ***
This is a top-level Model showing the Entities that are important at the top level.
It provides a suitable form of communication with a wide range of stakeholders.

A lower-level Model has been created for each specific Subject Area.



3.6 What have we learned ?

In this Chapter we have learned about a range of complex concepts and how they are achieved in Data Models.

When we have mastered this understanding we can truly consider ourselves advanced in the art and science of interpreting sophisticated Data Models.

At this stage, it would be very interesting and educational to look at a range of Models and consider how we could redesign them.

On many occasions, there is only one really good design but on others, there might be a choice.

Trying to decide which category a particular Model falls into would be a very challenging and valuable exercise.