

Design Patterns for Data Modelling

[1 Introduction](#)

[2. Concepts](#)

[One-to-Many Relationships](#)

[Many-to-Many Relationships](#)

[Rabbit's Ears](#)

[Inheritance](#)

[Reference Data](#)

[3 Data Warehouse](#)

[Design of an ERD](#)

[Design of a Data Warehouse](#)

[Reviewing the Design of a Data Warehouse](#)

[4. Applications](#)

[Customers and Demands](#)

[Units and Demands](#)

[Deliveries](#)

[Maintenance](#)

1 Introduction

This document is a starting-point for discussion.

The Design Patterns will be helpful for Quality Assurance of Data Models.

This applies particularly to Models produced by Third Parties.

2. Concepts

- **One-to-Many Relationships**

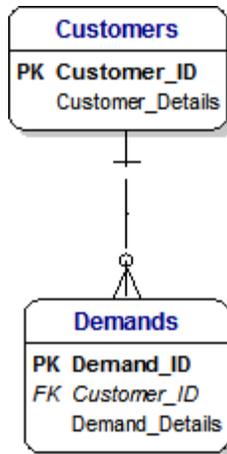
A Customer can place many Demands for Materiel.

This defines a One-to-Many Relationship.

A Data Modeller would say "For every Customer, there are many Demands".

This is shown in a Data Model as follows :-

Design Patterns for Data Modelling



- **Many-to-Many Relationships**

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”.

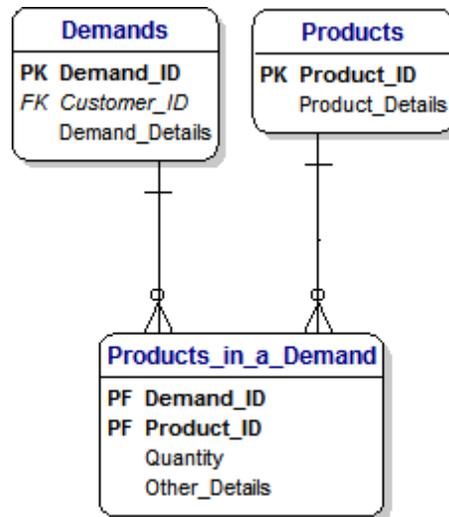
This defines a Many-to-Many Relationship and is shown in a Data Model as follows :-



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 :-

Design Patterns for Data Modelling



When we look closely at this Data Model, we can see that the Primary Key is composed of the Demand_ID and Product_ID fields.

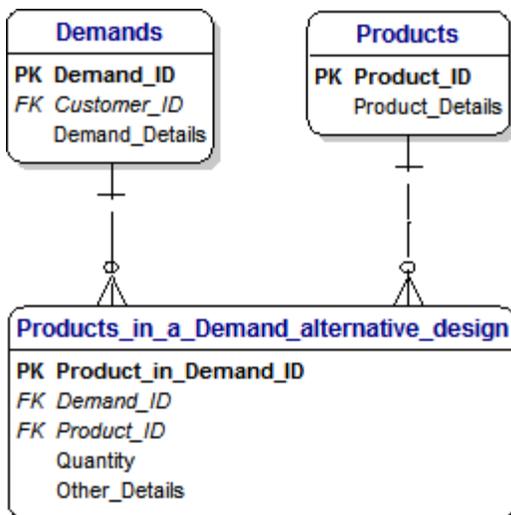
This reflects the underlying logic, which states that every combination of Demand and Product is unique. In the Database, this will define a new record.

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.

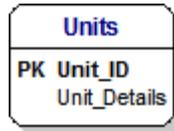
In other words, invalid records that have invalid Demand ID and/or Product ID values.



Design Patterns for Data Modelling

- **Rabbit's Ears**

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

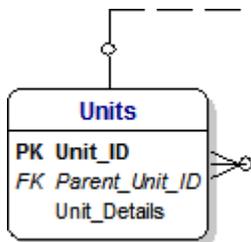


Then we think about the fact that every Unit is part of a larger organisation.
In other words, every Unit reports to a higher level within the overall organisation.

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.

- **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 Rotary and Fixed Wing 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.

Design Patterns for Data Modelling

We use the concept of 'Inheritance' to achieve this.

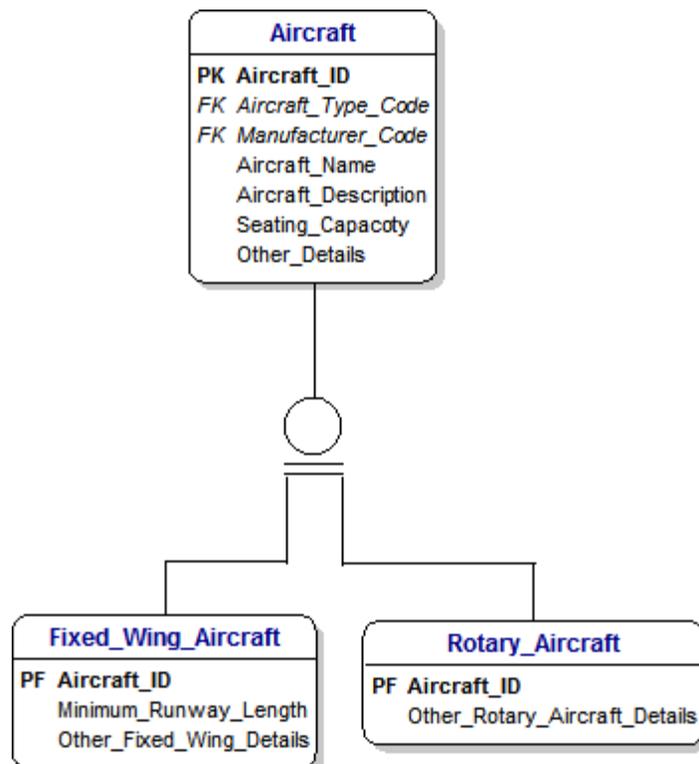
Inheritance is exactly what it sounds like.

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 :-



- **Reference Data**

Reference Data is very important.

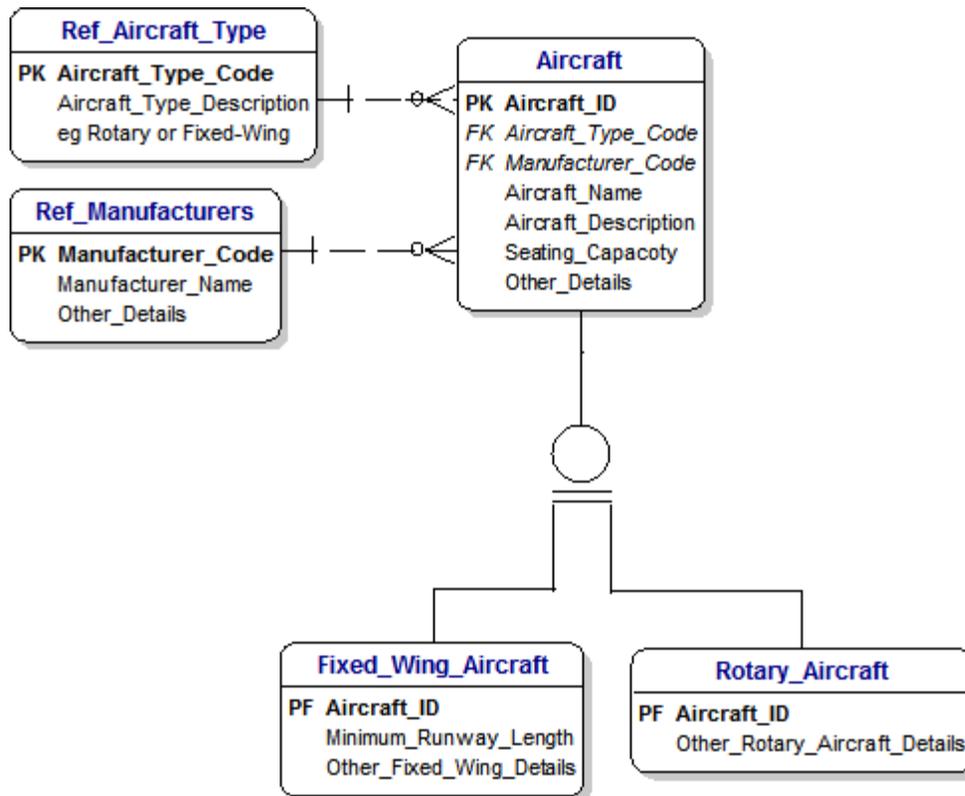
Wherever possible, it should conform to appropriate external standards, particularly national or international standards.

For example, the International Standards Organization ('ISO') publishes standards for Country Code, Currency Codes, Languages Codes and so on.

For Materiel and Products, NATO has published the National Codification Bureau (NCB) code. This is in use within the MOD and is administered from Kentigern House, Glasgow.

Design Patterns for Data Modelling

This diagram shows two examples of Reference data that might apply to Aircraft.

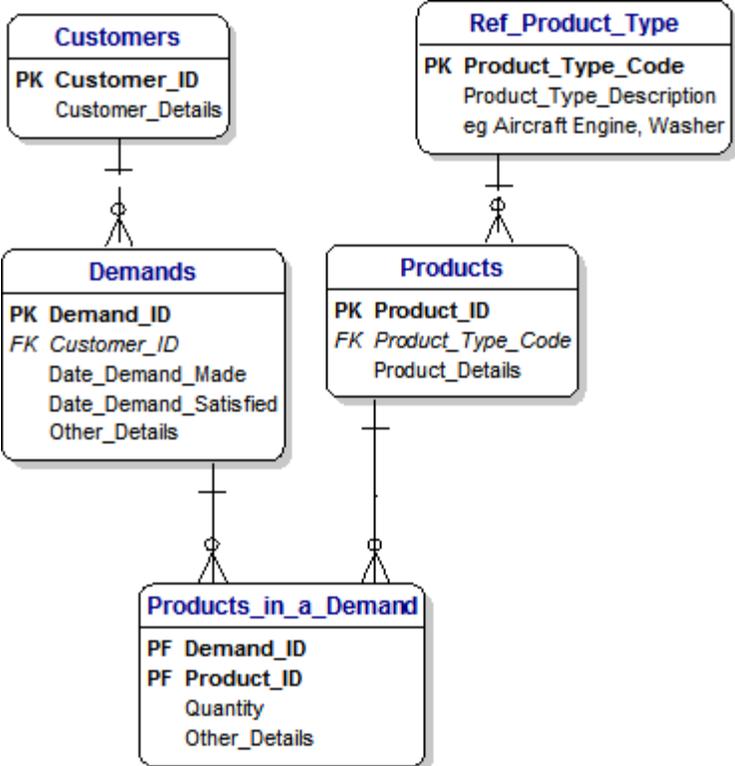


3 Data Warehouse

Design of an ERD

This Data Model is an Entity-Relationship-Diagram ('ERD') for Customers and Demands :-

Design Patterns for Data Modelling

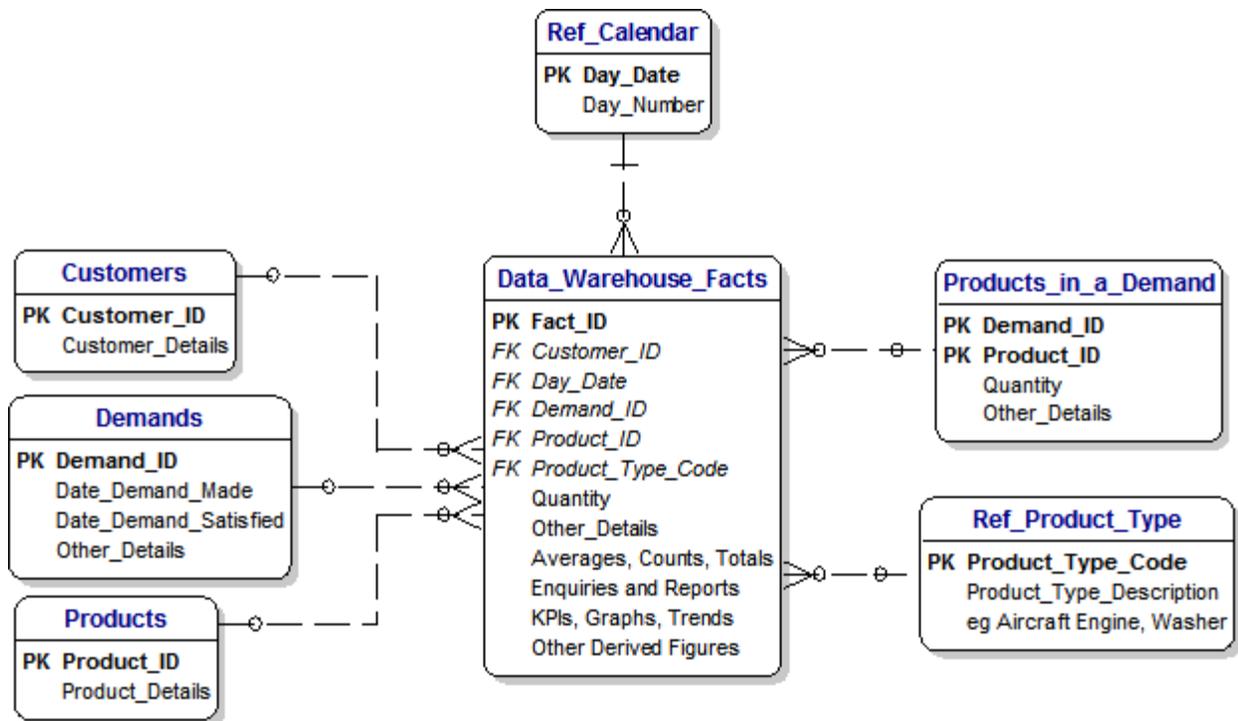


We could describe it in these terms :-
"Customers place Demands for Products of different Types."

Design of a Data Warehouse

This Data Model shows the corresponding Data Warehouse for Customers and Demands :-

Design Patterns for Data Modelling



The design of this Data Warehouse simply puts all data into a 'big basket'

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 demands for Aircraft Engines ?
- How many Demands did we receive in May ?

4. Applications

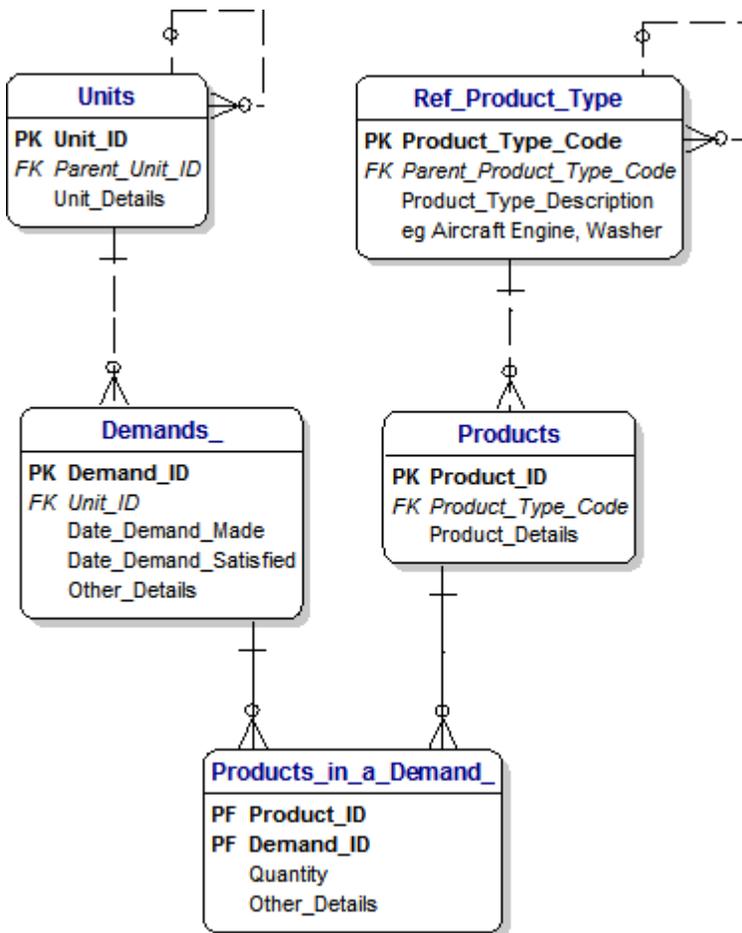
Design Patterns for Data Modelling

- **Customers and Demands**

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.

- **Units and Demands**

Here is a slightly different Model showing Units instead of Customers and highlighting the power of Rabbits Ears.



- **Deliveries**

A Simple Design Pattern

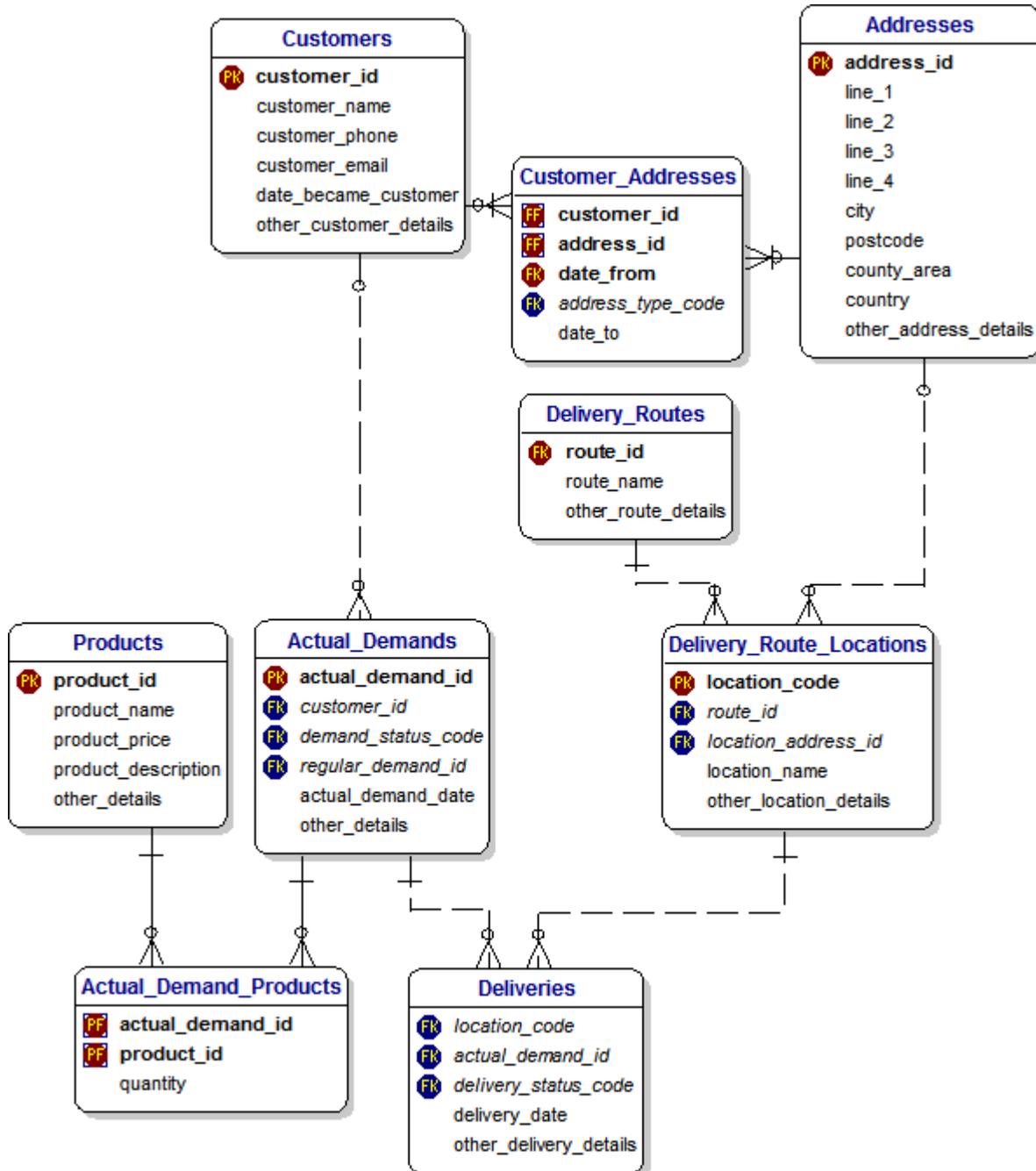
This Data Model is a simple Design Pattern that covers the activities of delivering items in a Demand to a designated address.

The process of reviewing a Data Model is to ask :-

Design Patterns for Data Modelling

“How do I describe the Business Rules behind this Model ?”
In this case, we could say :-

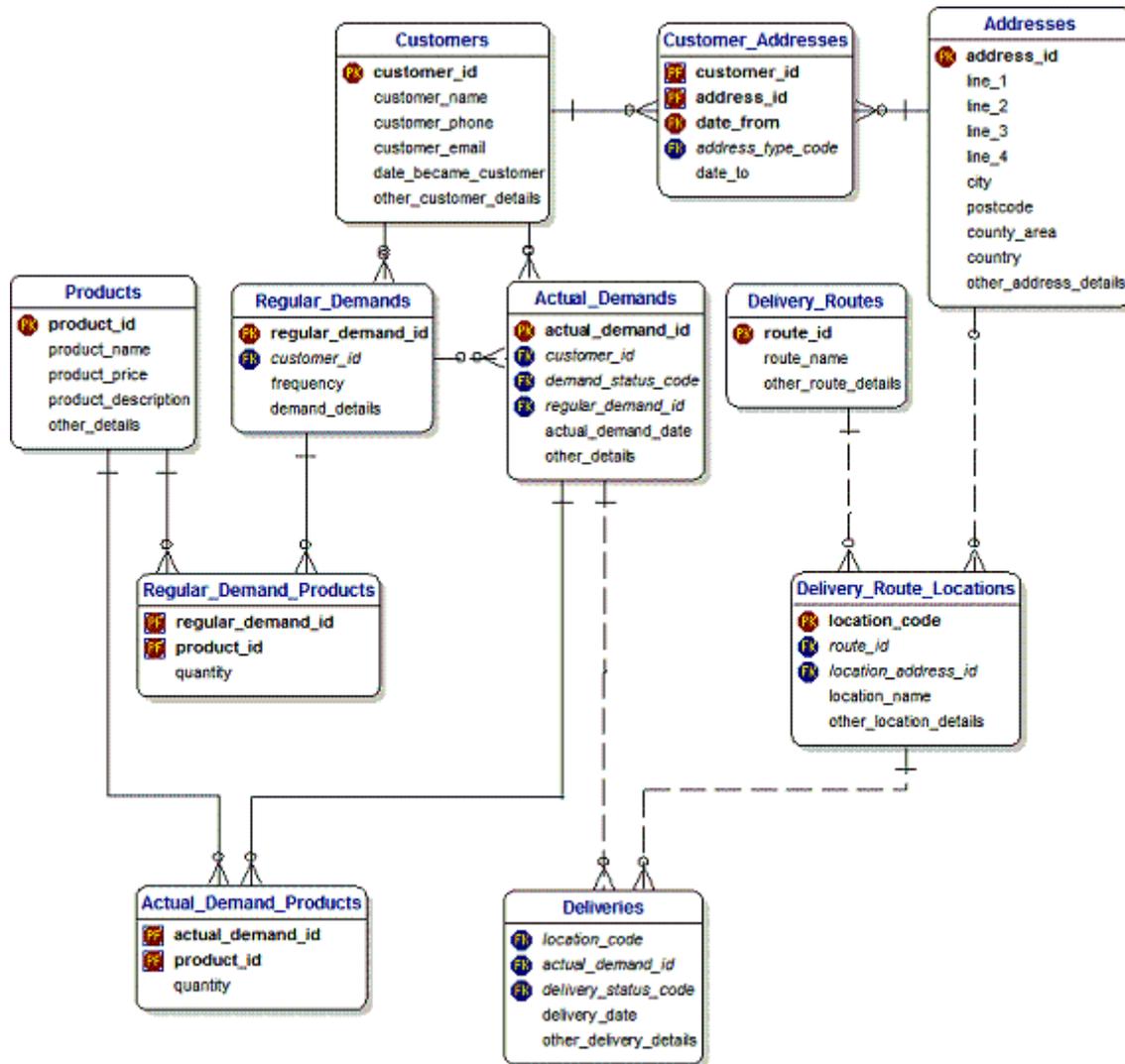
“A Customer can raise a Demand for Products to be delivered to a specified Address”.



- A Complex Design Pattern

Design Patterns for Data Modelling

This shows a complex Pattern which adds Regular Demands.



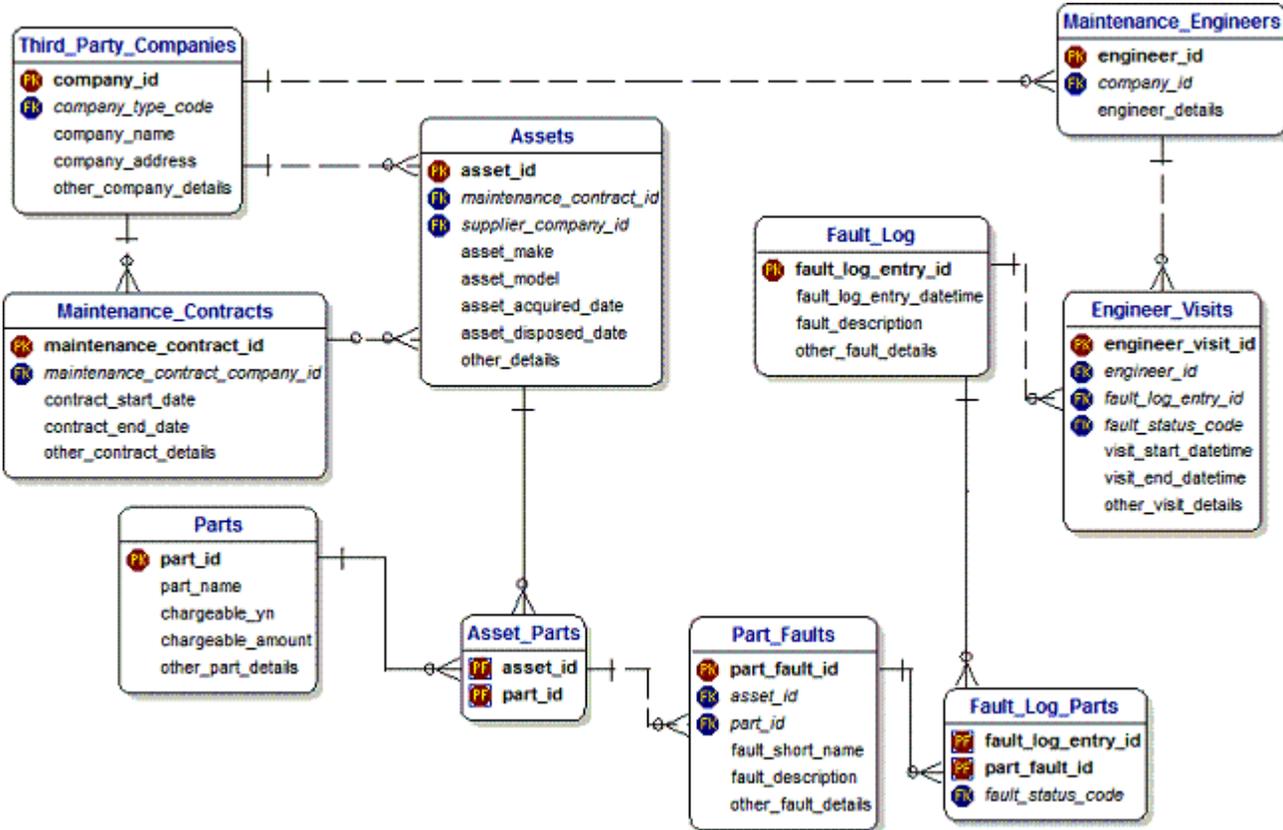
- **Maintenance**

The scope of this Data Model is the Maintenance of Assets by Third-Party Companies.

The Business Rules state :-

- * An Asset can have a Maintenance Contract.
- * An Asset consists of Asset Parts
- * Faults occur with these Parts from time to time.
- * Third Party Companies employ Maintenance Engineers to maintain these Assets.
- * Engineers pay Visits which are recorded in a Fault Log.
- * They correct the Faults and this is recorded in the Fault Log.

Design Patterns for Data Modelling



Barry Williams
Data Architect
Glue Ltd.