A Discussion Paper on Guidelines for Getting the Best out of Teradata

Chapter 1. Management Summary

The objective of this White Paper is to promote discussion of Best Practice and to explore common ground.

The Scope is the three Phases in the lifecycle of a typical Teradata Data Warehouse :-

1. Getting Started
2. Assessment of an operational Data Warehouse
3. Business As Usual (BAU) – changes and enhancements

Phase 1 – Getting Started

This incorporates a Blueprint and will incorporate any appropriate Teradata Industry Data Model.

It will then involve design of a Conceptual High-level Model and Subject Areas Models.

This Chapter provides material that is suitable for presentations, preparation of a business case, clarification of the requirements or to establish the scope of a Data Warehouse.

Presentations can be prepared for specific Industry Areas using this material, supplemented with 'Kick-Start' Data Model Templates in Appendix A for Banking, Communications, Retail, Transport and Web Visit Statistics.

Phase 2 - Assessment of an operational Data Warehouse

This offers a checklist framework that can be used to assess the status of an existing Warehouse Project.

We will establish what we expect to find and what we actually find.

Comparison of the actual against the expected will make it possible for us to define a migration path from the 'As-Is' of the actual to the 'To-Be' of the expected.

An important activity is a quality check of the Data Warehouse and all other Third-Normal Models.

This will combine relational theory with best practice in the design and usage of Data Models.

The result will be the identification of any Models that need modification to bring them up to the required standard. The Checklist is in Appendix B.

Phase 3 - Business As Usual (BAU)

This Phase is concerned with making changes and enhancements to an operational Data Warehouse.

It builds on Teradata's Best Practice in this area.

It requires a stable Data Architecture and an Agile approach.

The remainder of this document uses the Transport Area as an example.

Chapter 2. Getting Started

2.1 What is this ?

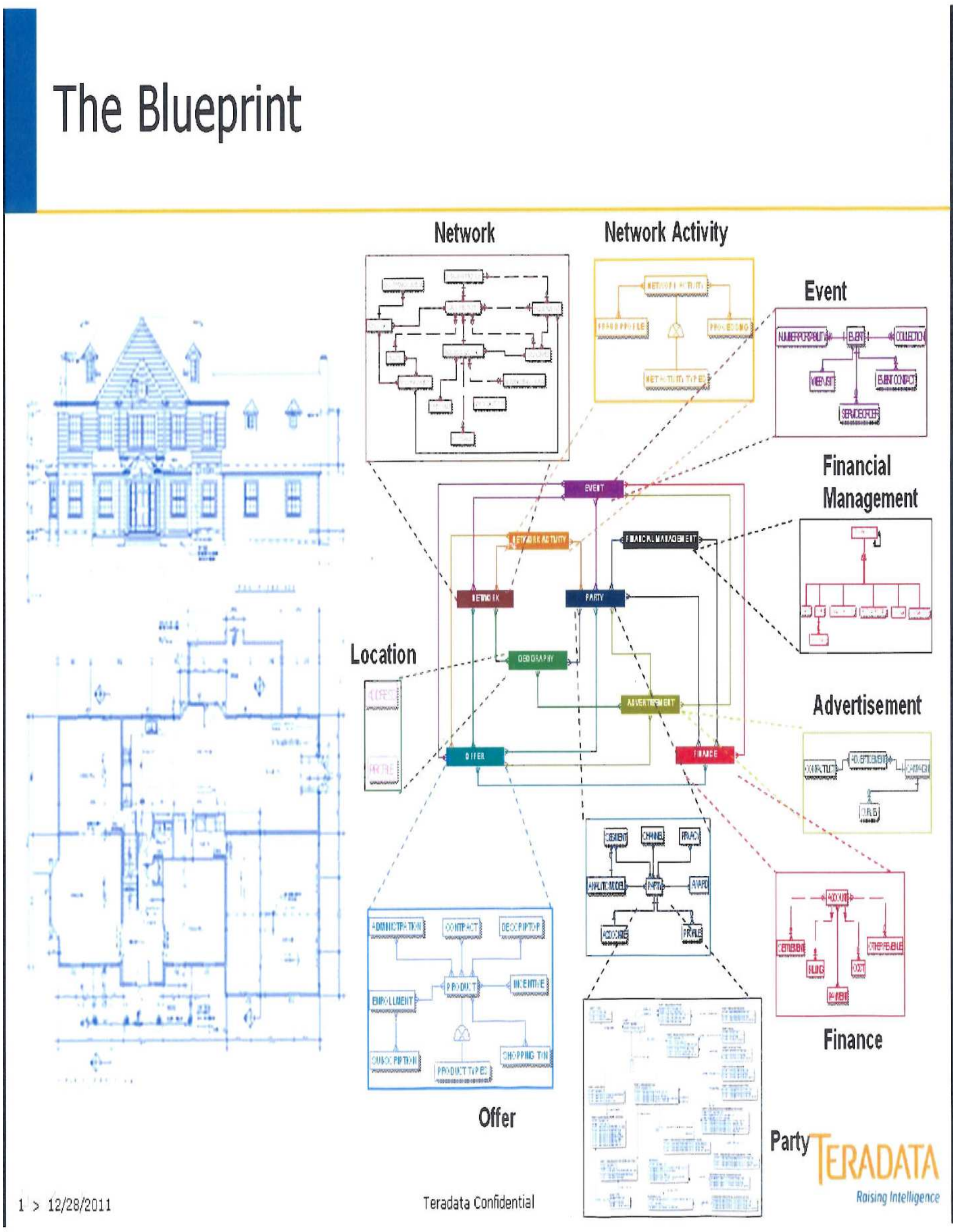
This Chapter describes how to get started with a Teradata Project.

2.2 Why is it important ?

It is important because it helps us to understand the overall process and concepts of a Data Warehouse.

2.3 A Blueprint

This is a Teradata Blueprint for an Architecture where rooms become Subject Areas in a Data Model.



2.4 A Transport Enterprise Data Model

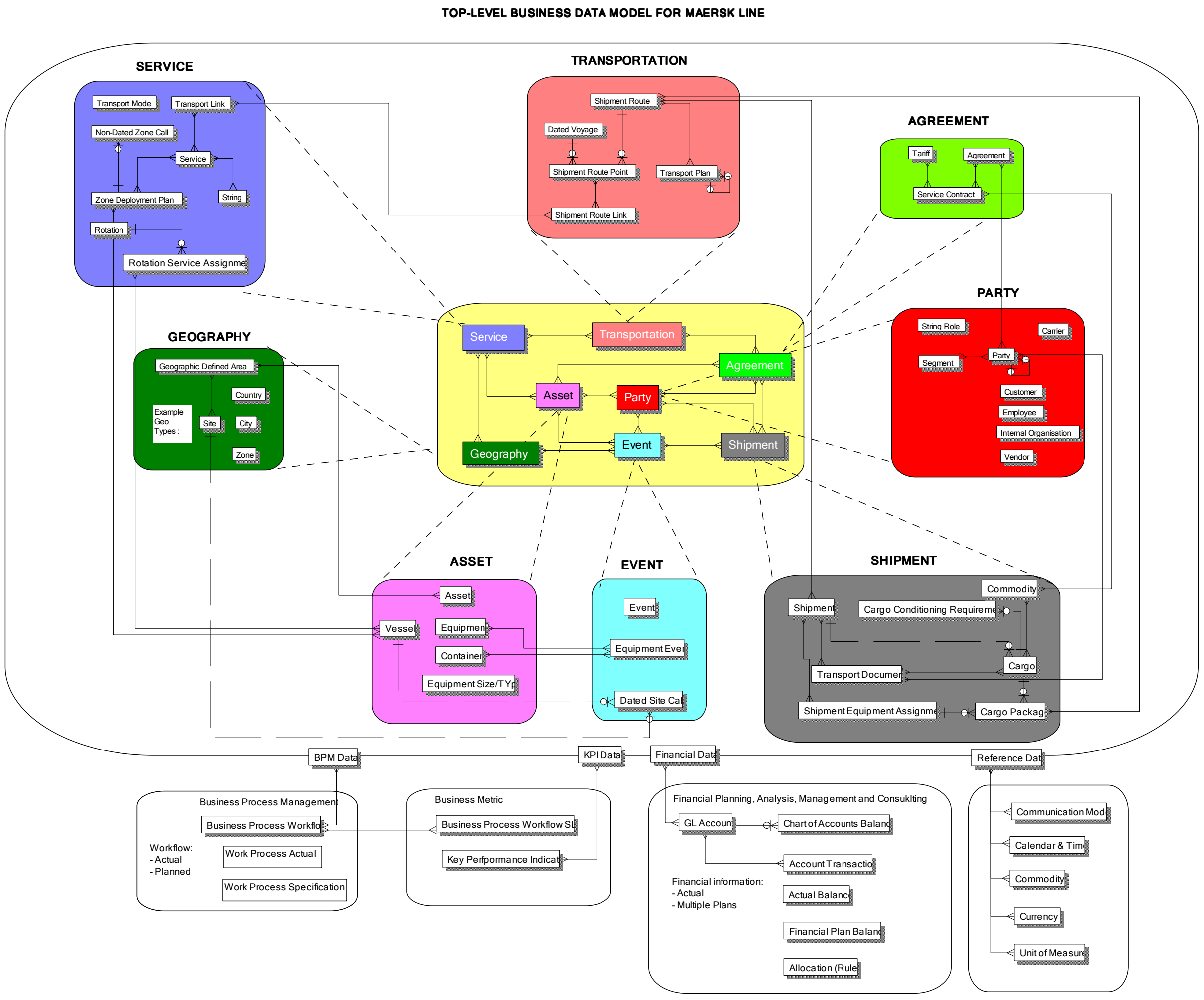
This shows an extract of the Teradata Transport and Logistics Data Model.

It acted as a starting-point for the creation of a version of the Model for Maersk, which was the Teradata Customer.

The yellow area in this diagram shows an EDM with separate Subject Areas :-

- Agreement, Asset, Event, Geography, Party, Service, Shipment and Transportation.

These are shown in larger sizes around the yellow rectangle.



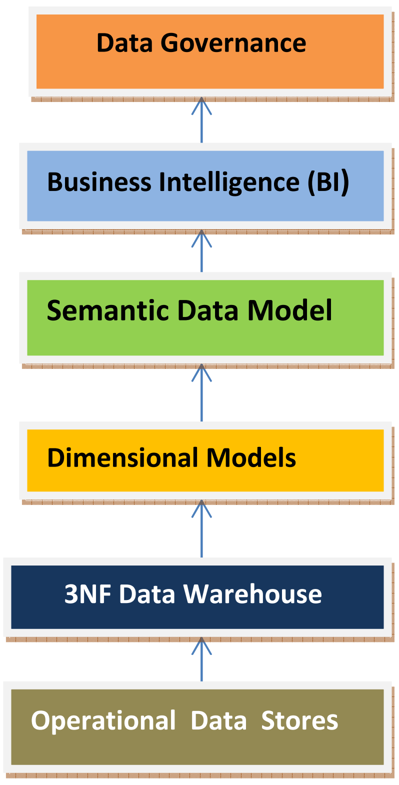
2.5 A Data Architecture and BIReady

This shows six Layers in a Data Architecture for Data Warehouses.

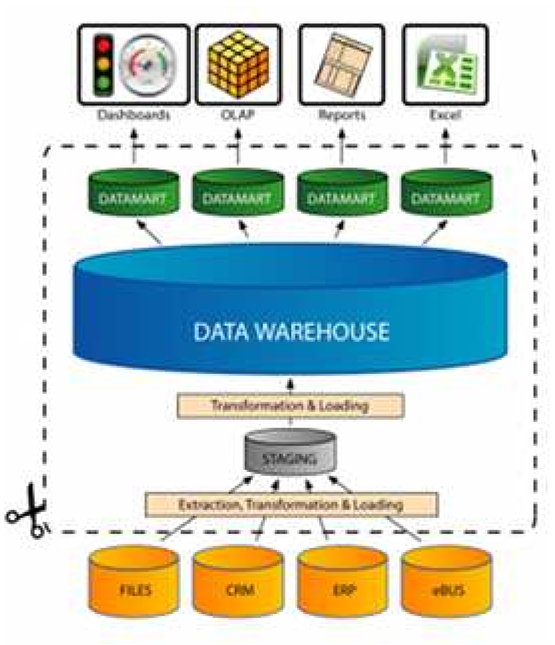
On the left-hand side we show the layers in a Teradata Data Architecture.

On the right-hand side, we use a product called BIReady that provides the functionality within the dotted lines which corresponds to the Data Integration and Data Marts Stages in the Road Map.

In other words, **BIReady** automates the implementation of ETL and ELT functions, the Data Warehouse and the Data Marts.



BIReady facilities are shown in the dotted square



2.6 Stages in loading Data into a Warehouse

There is a logical sequence to the Stages of loading data into a Warehouse.

The lowest levels of data provide a foundation. The lowest level is Reference Data, such as Currencies and Languages.

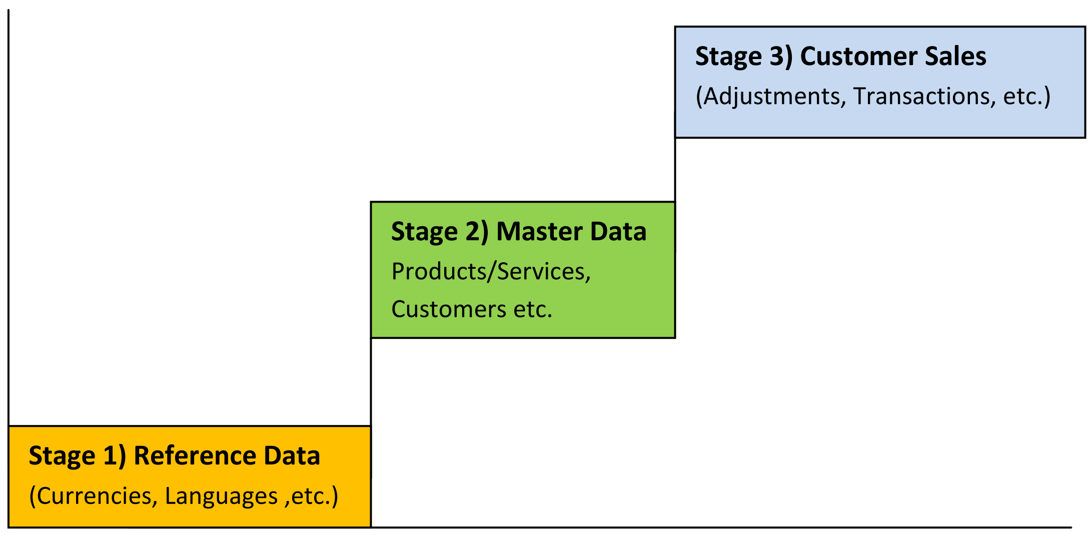
Then, after these have been identified, sources and loaded we can move to the Master Data.

This includes Products, Services, Customers and so on.

Then we come to the more dynamic data such as Sales, Transactions and so on.

Best Practice states that we should set up load procedures that can be run so that data at each Stage can be deleted and replaced repetitively.

This shows three Stages of data in a typical Data Warehouse.



2.7 Business Intelligence (Top-down)

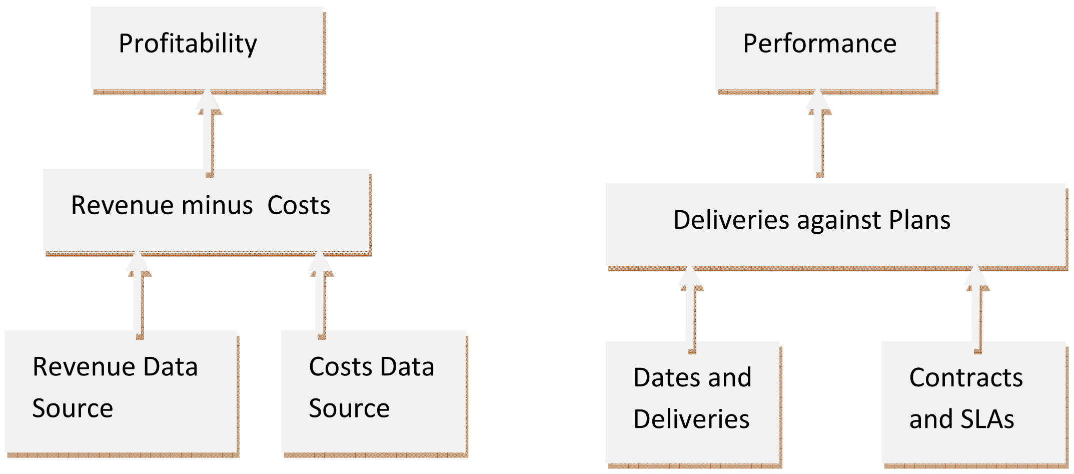
Key Performance Indicators ('KPIs') provide an excellent starting-point for Business Intelligence.

Basic KPIs can be defined for Profitability and Operational Performance.

These can be enhanced in collaboration with the business users to establish commitment.

Profitability can iniitially be defined as a basic difference between Profits and Revenue.

Performance can be measurement of Deliveries being 'On-Time' against plans agreed in Service Level Agreements (SLAs).



The initial Data Architecture will include this basic structure of KPI and Source Data.

This will provide a simple but flexible framework which can be enhanced in a controlled manner.

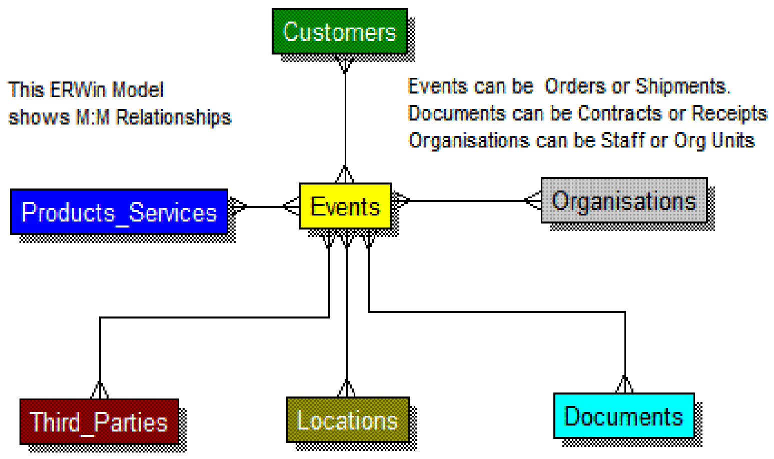
We will use this as a starting-point for discussions to establish the requirements with the business users.

2.8 Build the Data Warehouse Incrementally

A Common Data Model can be used to design a Data Warehouse in a series of incremental Steps that provide a validation with each Step.

2.8.1 Common Data Model

This shows a Common Data Model that provides an excellent Starting Point for building an Enterprise Data Warehouse.



2.8.2 Event Message

We can define Events with corresponding Messages.

For each Message, we can add to the design of our Data Warehouse.

The strength of this approach is that it validates the design of the Data Warehouse.

This Common Data Model defines a Message for each Event.

2.8.3 Example Event Message (1)

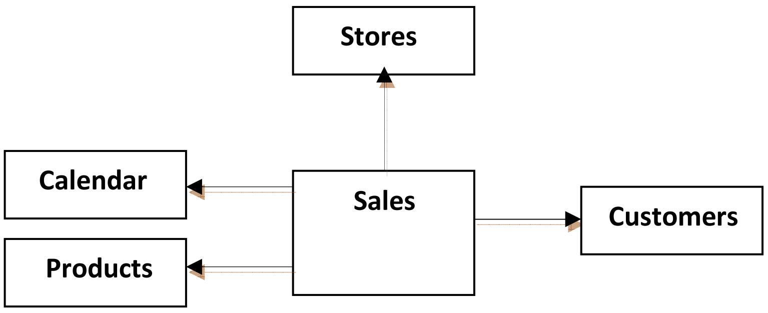
For example, a Customer purchases one Product at a Store :-

Date	Store ID	Customer Name	Product	Quantity	Amount

The Business Rules would say :-

1. Each Sale is made to one Customer
2. Each Sale can involve just one Product
3. Each Sale is made at a specific Store
4. Each Sale is made at a specific valid Date and Time which is defined in a Calendar

Therefore we can define this fragment of a normalised Data Model :-



2.8.4 Event Message (2)

In this Event, the Customer buys two Products. Therefore our initial Data Model must be enhanced to have a separate Table for ‘Products Sold’.

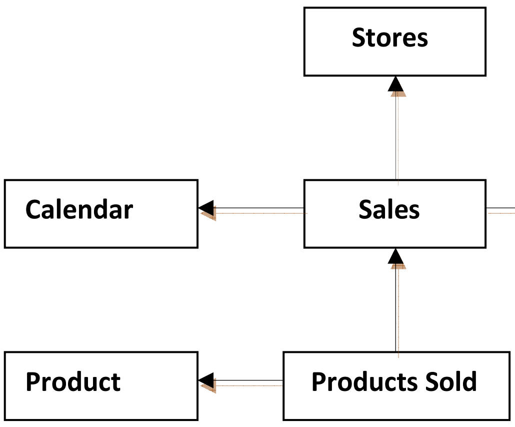
When we continue to include more Events, we enhance our Model until it is complete.

Date	Store ID	Customer Name	Product 1	Quantity 1	Amount 1	Product 2	Quantity 2	Amount 2	Total Amount

This Event changes one of the Business Rules :-

1. Each Sale can involve multiple Products

Therefore we add a separate entity called ’Products Sold’ to our Data Model :-

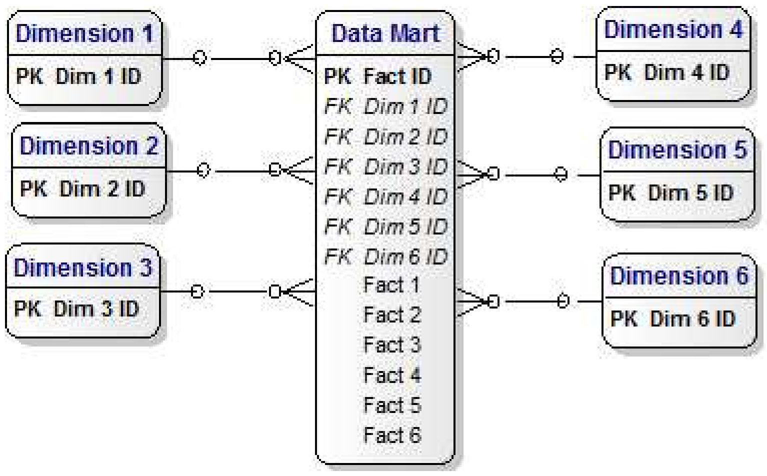


2.9 Generic Dimensional Model

The existence of Dimensions and Facts makes it possible to define a Generic Design for a range of Dimensional Models.

This example shows six Dimensions and six Facts.

Each of these Dimensions and Facts can be translated from the generic names to specifics, such as Customer, Product, Location, Time Period and so on.



2.10 Simple Data Architecture

A Data Architecture should be composed of separate Components with separate and clearly-defined functionality.
The 3NF Data Warehouse implements data integration and provides a ‘Single View of the Truth’ (SVOT).
The objective is to show the major Building Blocks in a simple clean Architecture.

$$\begin{tikzcd}
\text{ODS} \arrow[r] & \text{Staging Area} \arrow[r] & \text{3NF Data Warehouse)} \arrow[r] & \text{Dimensional Models (Data Marts)} \arrow[r] & \text{Semantic Layer}
\end{tikzcd}$$

A more detailed version is shown in the Data Warehouse Bus Architecture in Chapter 3.

2.11 Sample Kick-Start Data Warehouses

Some sample Data Warehouses are included in Appendix A for a starting-point in discussions.

Chapter 3. Checking Progress

The Approach in this Chapter is to define what we expect to see and then review whether the situation matches our expectations.

3.1 What is this ?

This is an approach to checking the Progress of a Teradata Project.

3.2 Why is it important ?

It is important because it makes it possible for us to establish where a Project is on the Road Map.

Then we can identify the long-term goal, the current situation and to establish a migration path from the 'As-Is' to the 'To-Be'.

3.3 Assessment Checklist

Our starting-point is that we have certain expectations, which are the occurrence of some specific Data Modelling

This table provides a summary checklist. We have shown Customers, Products/Services and Orders as templates and should be replaced as appropriate.

Data Model	**Exists**	**Quality**	**Action**
Conceptual High-Level Model			
Subject Area Models			
- Customers			
- Products/Svcs			
- Orders			
- Ref Data			
- Master Data			

3.4 Our Expectations

Our starting-point is that we have certain expectations, which are the occurrence of some specific Data Modelling Design Patterns.

This is based on the concept that Data Models are 'A Good Thing' and you can never have too much of 'A Good Thing'
Therefore, if the answer to everything is a Data Model then what exactly are the Questions ?

We propose that an End-to-End range of Data Models provides a robust and generalised foundation for any Data Migration activity.
Of course, we consider it is true that 'Data Migration' is a deceptively simple way of describing the activities that occupy most of our waking (or working) hours most of the time.

This is appropriate whether we are Data Architects, Data Modellers or Data Analysts, where each role has a different range of skills and responsibilities.

Our question then becomes :-
What kind of Data Models do we need to cover our End-to-End Scope ?

We propose a range, to include :-

- Semantic Models
  These provide a 'User-eye ' view of the data in Reports
- Enterprise Data Warehouse
  This provides a 'Single View of the Truth'
- Dimensional Models
  These support Data Marts and Data Views for specific Reports, KPIs and analyses.
- Operational Data Store - ODS
  This defines the Staging Area

This seems a modest ambition, so let's discuss it in detail to see if it holds up under
detailed analysis.

We start with Operational Data Stores and end with a Dimensional Model that feeds data for a BI Layer.

The question then becomes :-
What kind of Data Models do we need to cover our End-to-End Scope ?

We propose a range from ODS, Data Mapping, Data Warehouse, Dimensional Models and Semantic Models

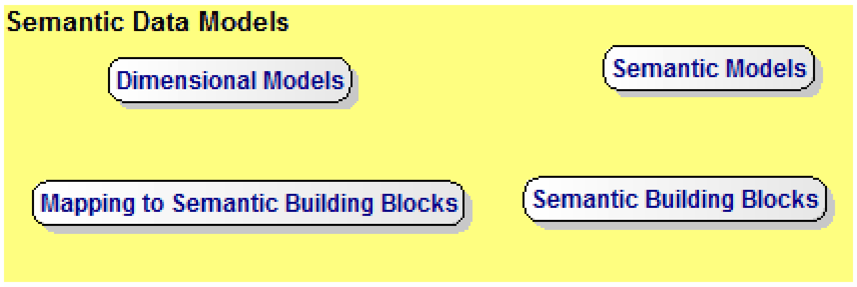
This seems a modest ambition, so let's discuss it in detail to see if it holds up under detailed analysis.

This page shows four Design Patterns.

3.5 Design Patterns

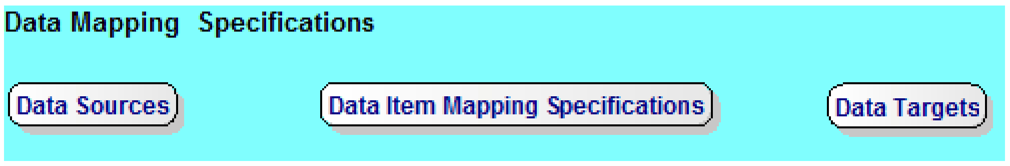
3.5.1 Semantic Models

The first Design Pattern shows Semantic Data Models which provide a 'User Friendly' front-end.
This helps Users to request KPIs, Reports and analyses using words and terminology that they normally use.
For example, talking about Customers, rather than Parties.
If Semantic Data Models have been created, then we can say that the work has been done to the established standards of Best Practice.



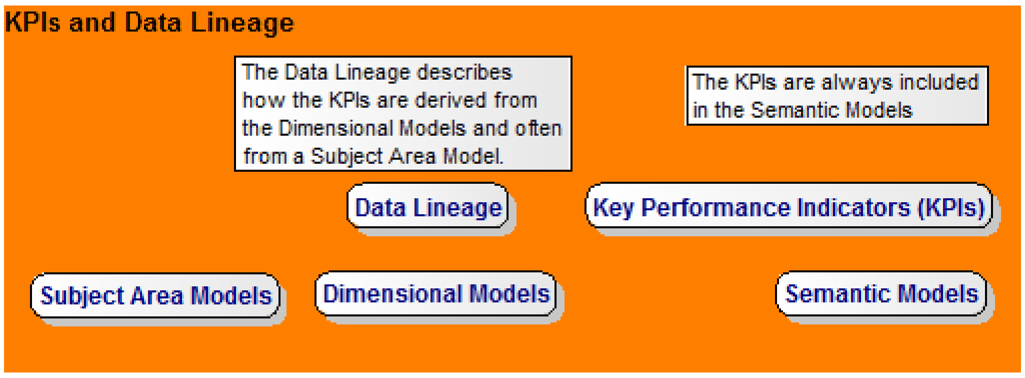
3.5.2 Data Mapping Specifications

The second shows Mapping Specifications which translate Source data to Target data.



3.5.3 KPIs and Data Lineage

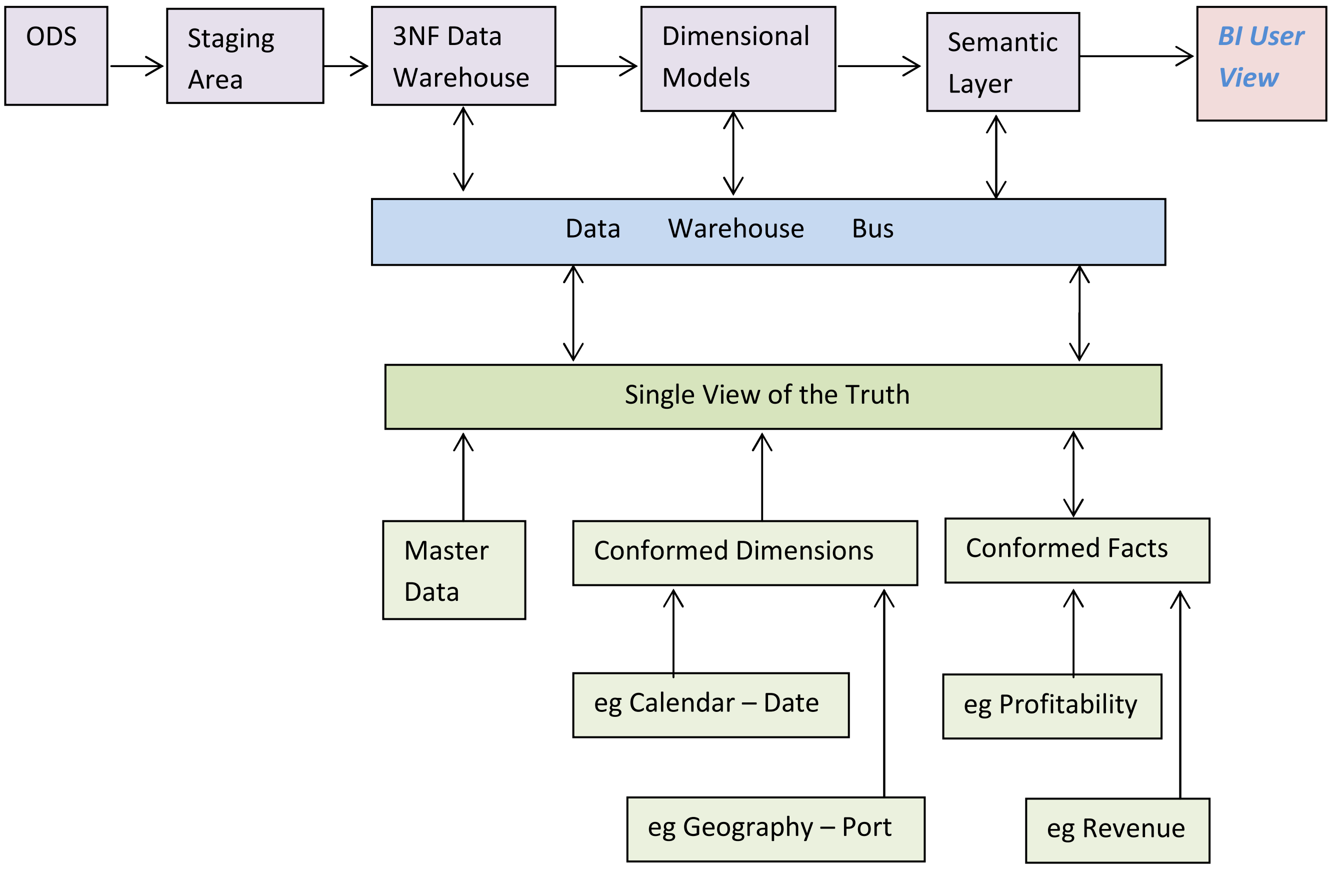
Then, we identify KPIs, and need to trace their derivation.
This helps to establish the Data Lineage for Sarbanes-Oxley and other Statutory Reporting requirements.


KPIs and Data Lineage
The Data Lineage describes how the KPIs are derived from the Dimensional Models and often from a Subject Area Model.
The KPIs are always included in the Semantic Models
Data Lineage
Key Performance Indicators (KPIs)
Subject Area Models
Dimensional Models
Semantic Models

3.5.4 Data Warehouse Bus Architecture

Now, we show the complete Data Warehouse Bus Architecture
It provides a common Framework for establishing consistent standards for data and structures.

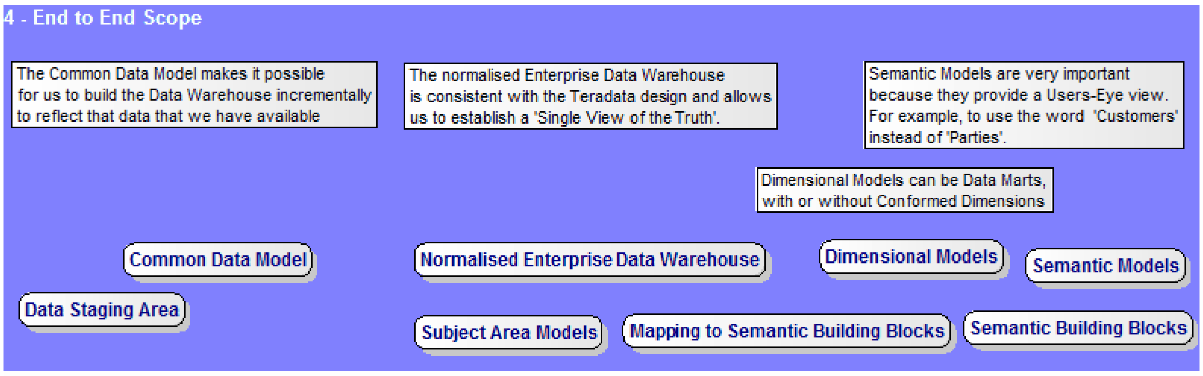
The top row shows the major Building Blocks in the migration of data from the Operational Data Stores to the Semantic Layer that defines the BI User View.



3.5.5 End-to-End Scope

Finally, we come to the complete End-to-End scope of Data Models.
The Common Data Model is used to standardise data that is loaded into the Data Warehouse (DWH).
We can also be sure that we have a 'Single View of the Truth' at any point in time.
It helps us to build the design of the DWH in a step-by-step fashion.

We identify that the Enterprise Data Warehouse is a top-level design which will have lower-level **Subject Area Models** such as Customers.


4 - End to End Scope
The Common Data Model makes it possible for us to build the Data Warehouse incrementally to reflect that data that we have available
The normalised Enterprise Data Warehouse is consistent with the Teradata design and allows us to establish a 'Single View of the Truth'.
Semantic Models are very important because they provide a Users-Eye view. For example, to use the word 'Customers' instead of 'Parties'.
Dimensional Models can be Data Marts, with or without Conformed Dimensions
Common Data Model
Normalised Enterprise Data Warehouse
Dimensional Models
Semantic Models
Data Staging Area
Subject Area Models
Mapping to Semantic Building Blocks
Semantic Building Blocks

Chapter 4. Business as Usual (BAU)

4.1 What is this ?

This describes a structured approach to adding new features and updates to an existing Data Warehouse.

4.2 Why is it important ?

It is important because changes are a fact of life and it is essential that they are carried out in a controlled manner.

At this stage, the Data Architecture will include a Data Warehouse, Dimensional Models and Semantic Models.

Therefore, our approach to BAU will be defined in terms of changes to these components.

There are two aspects to BAU :-

- Modifications to meet new and changed Requirements
- Enhancing the performance of the existing Data Warehouse

Our starting-point is that we expect to find :-

- A High-Level Conceptual Data Model
- A number of Subject Area Models

One of the most important activities is to check the Quality of a Data Model, such as the Enterprise Data Warehouse or a Subject Area Model.

The detailed Steps involved in doing this are shown in Appendix A.

Appendix A. Data Model Templates

A.1 What is this ?

These Templates provide introductory Data Models for some specific Applications.

A.2 Why is it important ?

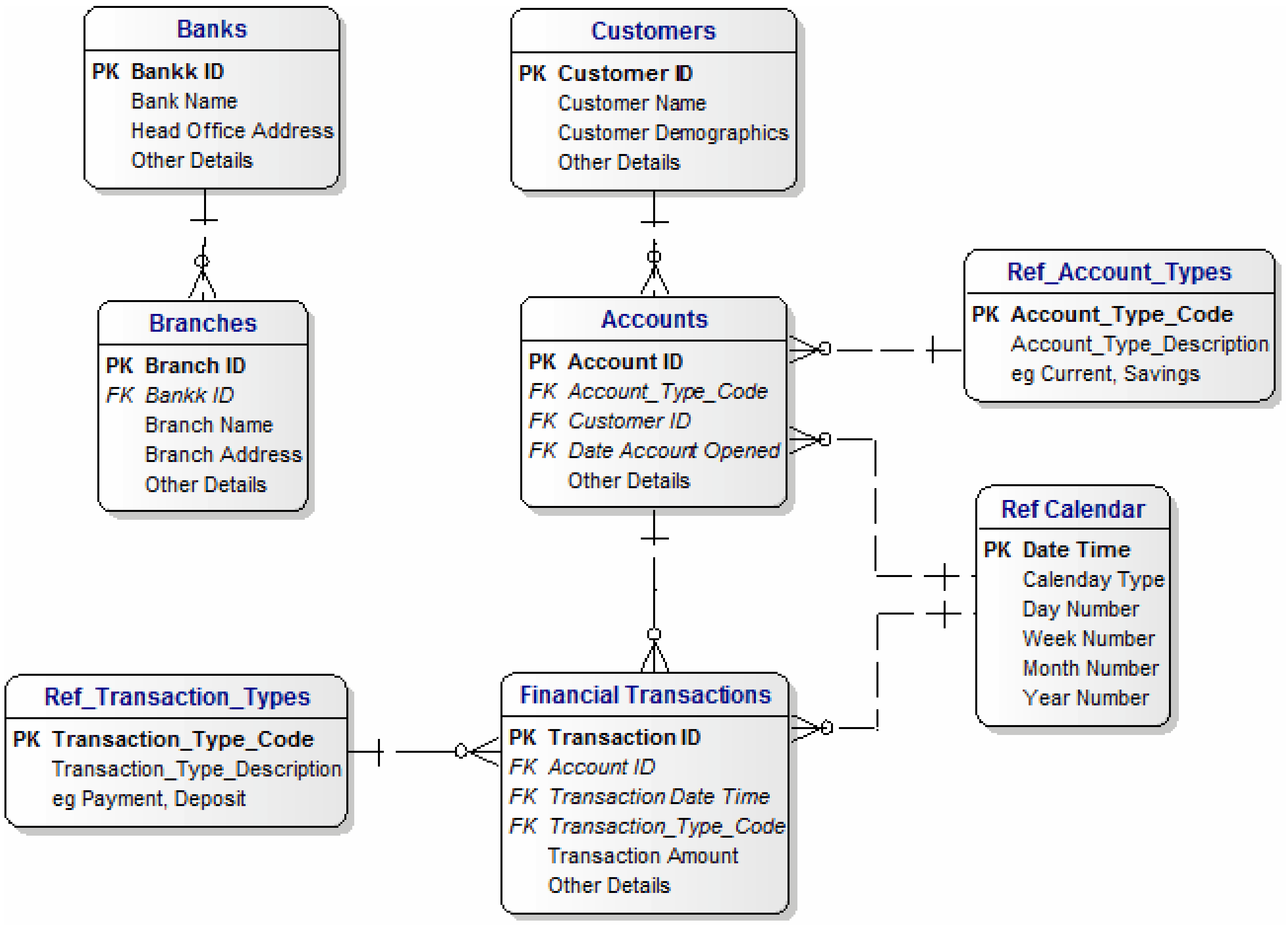
These Templates provide ’Kick-Start’ samples to get up and running quickly.

They also provide samples that can be used to assess existing Data Models for ‘fit for purpose’.

A.3 Banking

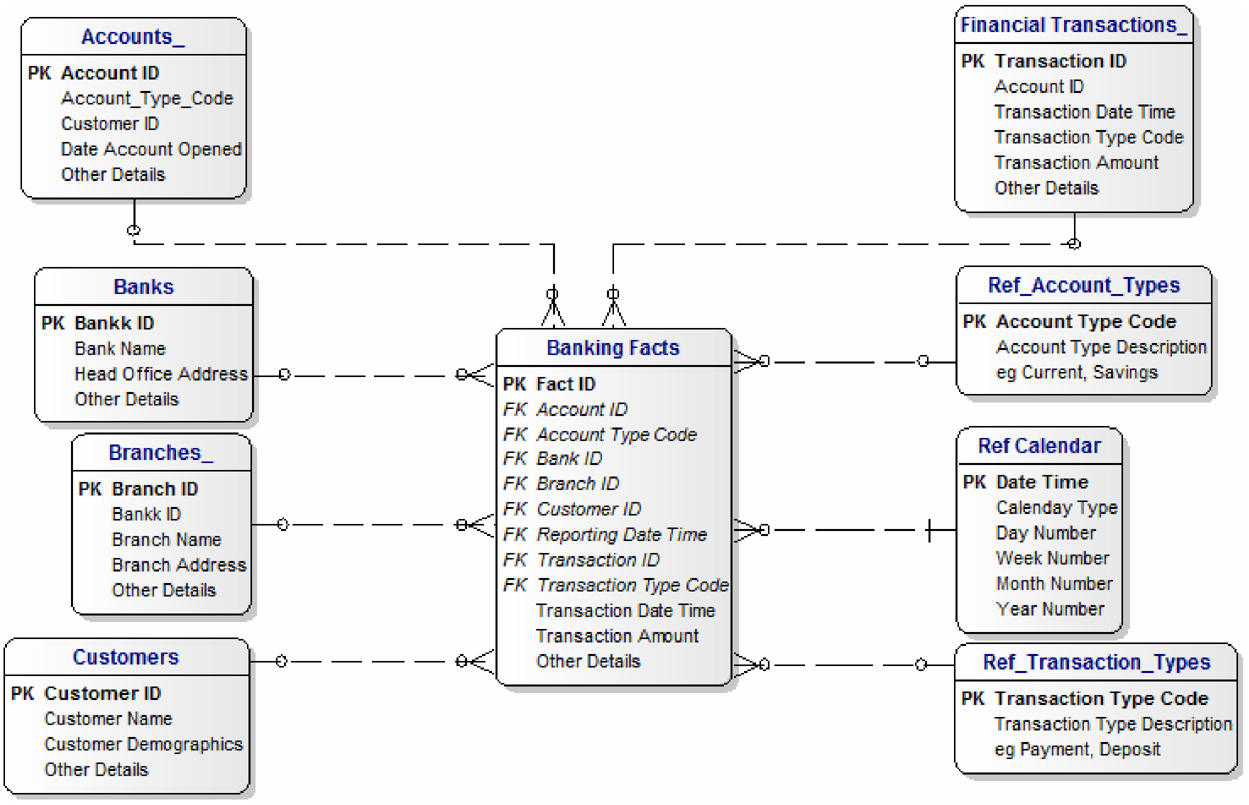
A.3.1 Data Warehouse

This shows theTemplate EDW for the analysis of retail Banking.



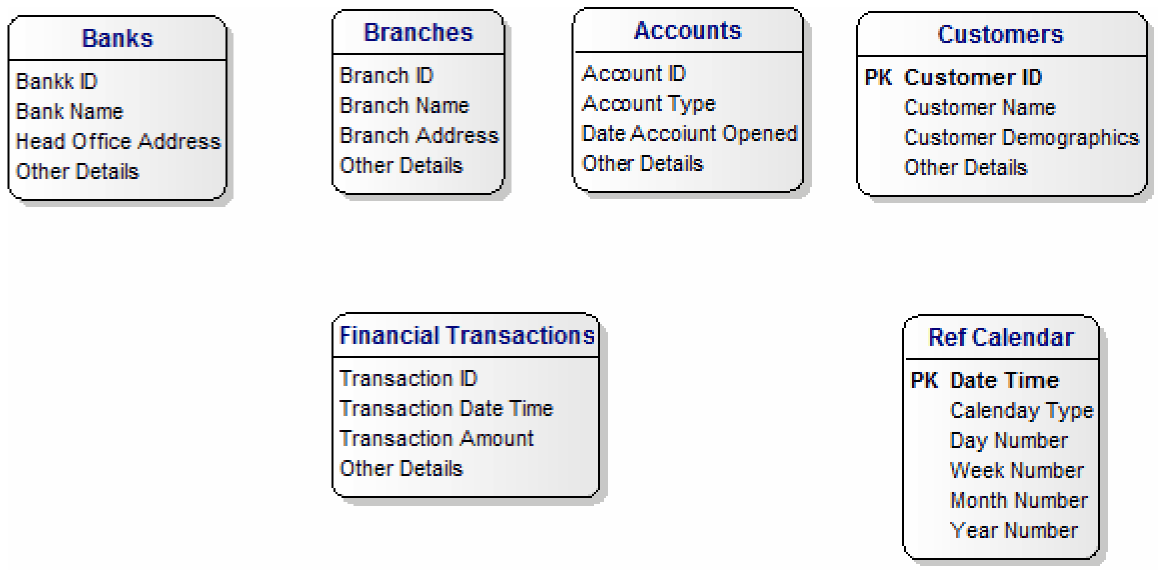
A.3.2 Dimensional Model

This shows theTemplate Dimensional Model for retail Banking.



A.3.3 Operational Data Stores (ODS)

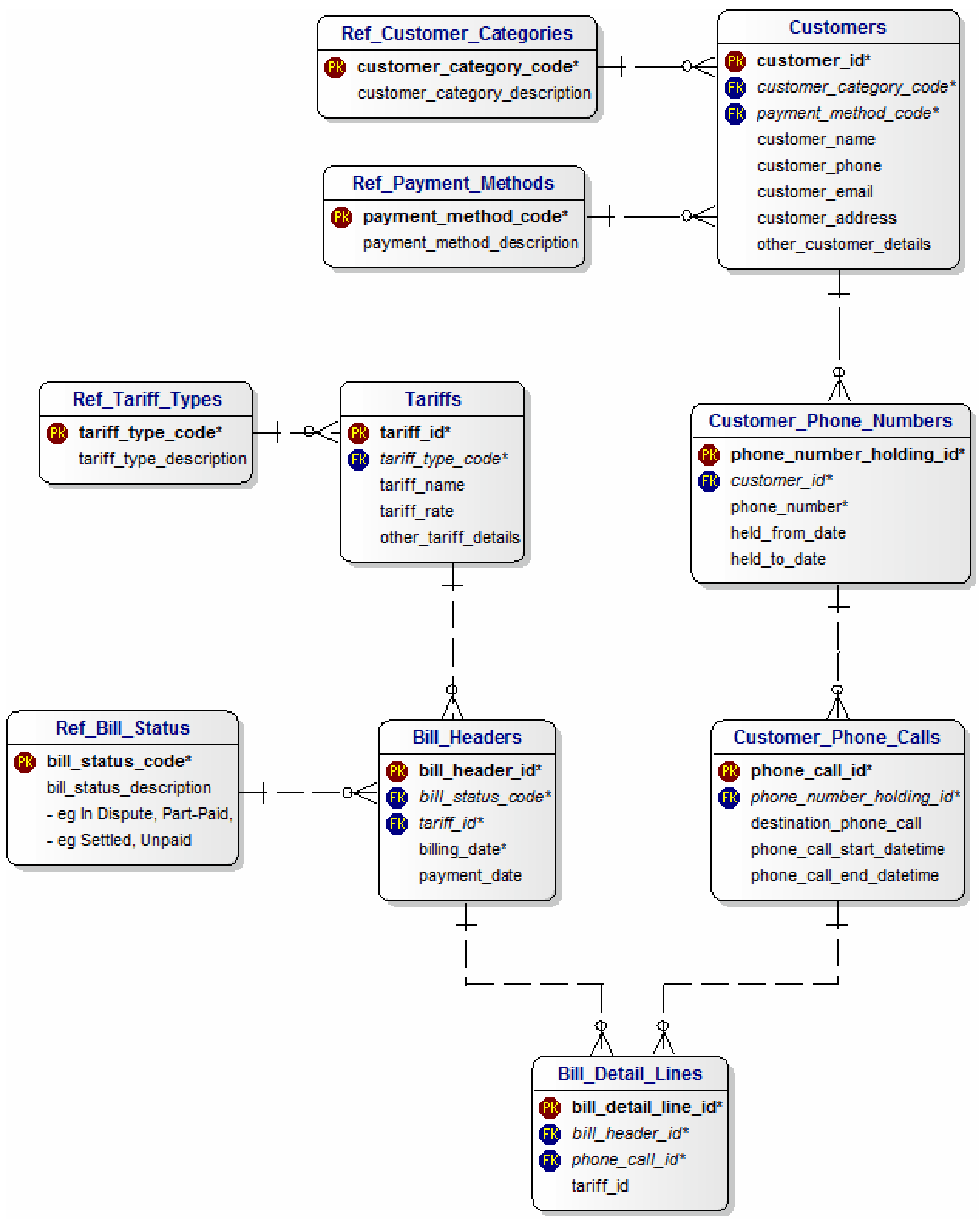
This shows the ODS for the analysis of retail Banking.



A.4 Communications

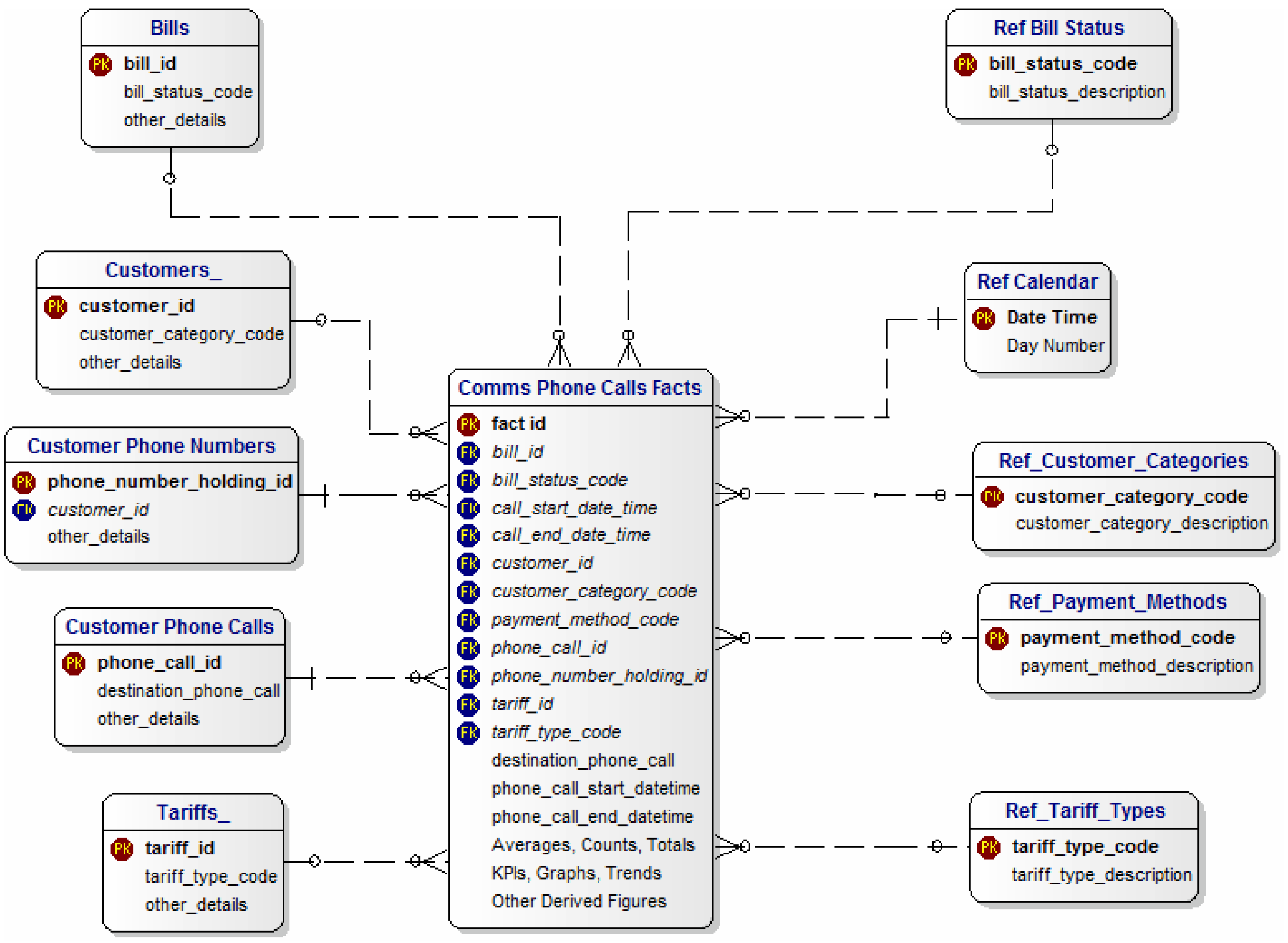
A.4.1 Data Warehouse

This shows theTemplate EDW for Communications and, in particular, Telephone Billing .Systems.



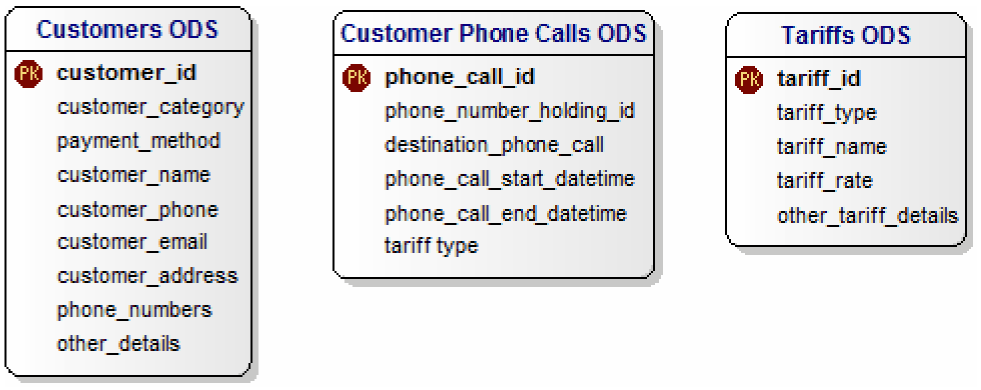
A.4.2 Dimensional Model

This shows theTemplate Dimensional Model for Communications.



A.4.3 Operational Data Stores (ODS)

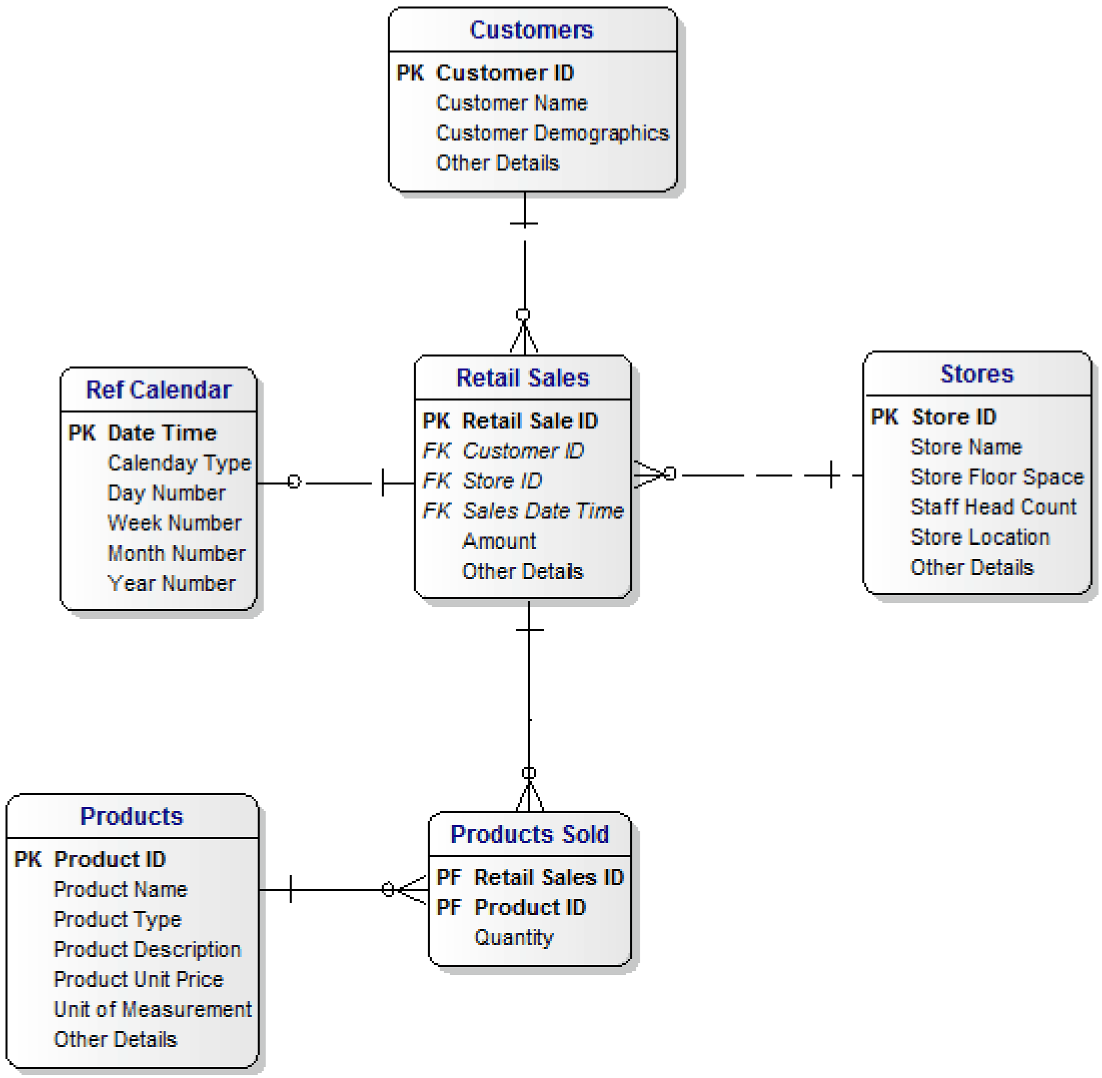
This shows the ODS for Communications.



A.5 Retail Sales

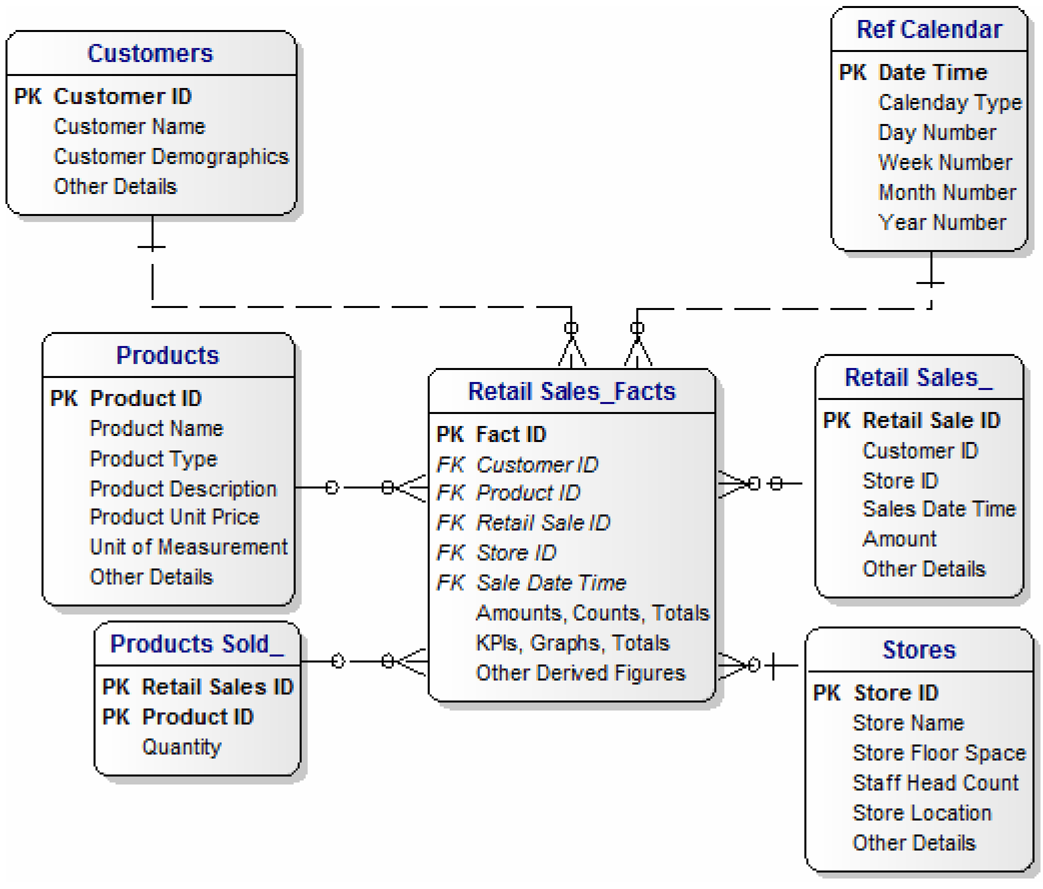
A.5.1 Data Warehouse

This shows the EDW for the analysis of Retail Sales.



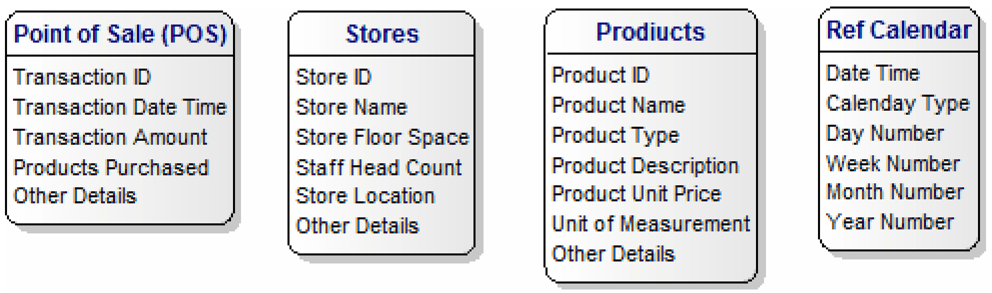
A.5.2 Dimensional Model

This shows the Dimensioanl Model for the Retail Sales.



A.5.3 Operational Data Stores (ODS)

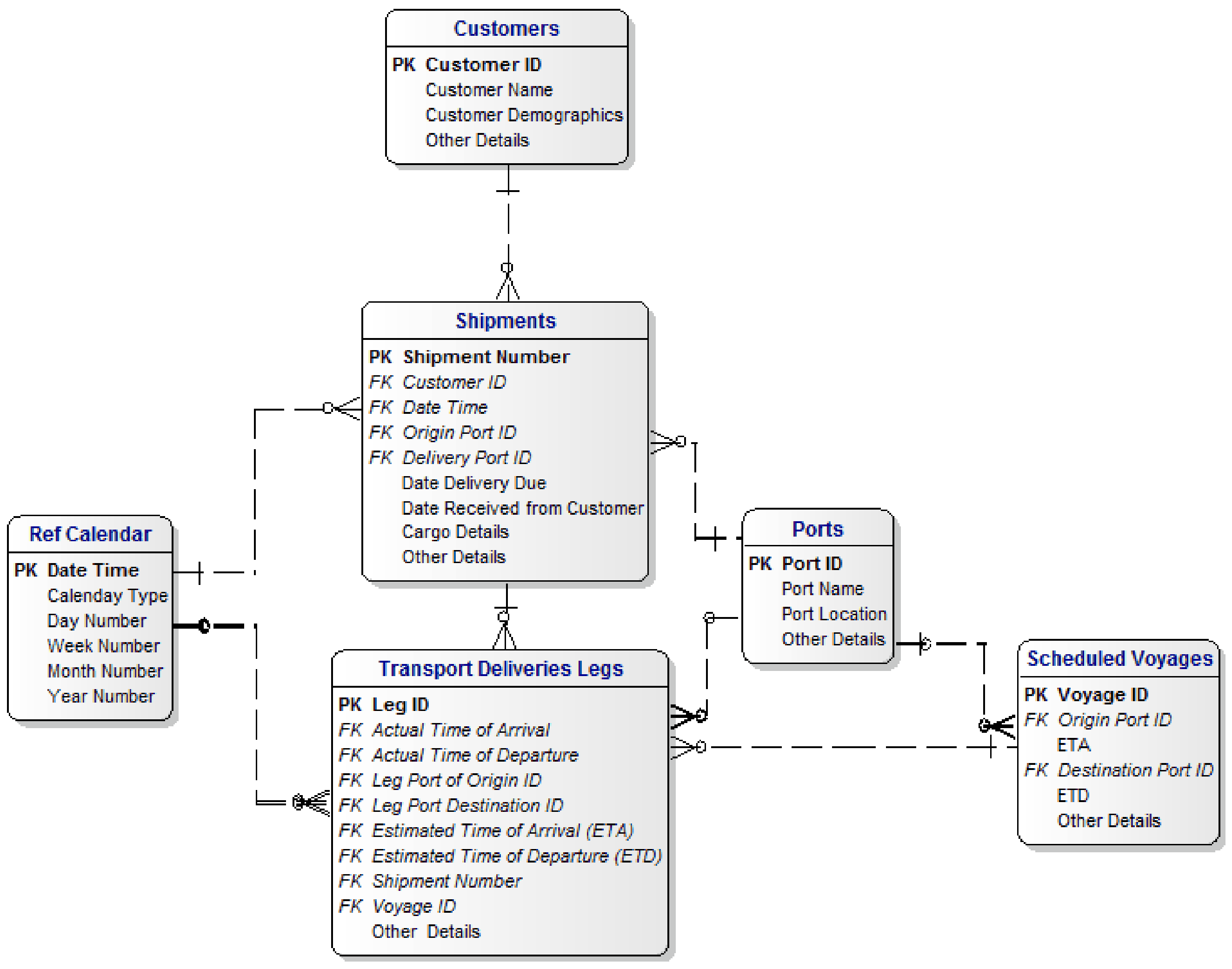
This diagram shows a sample of extract data for Retail Sales.



A.6 Transport

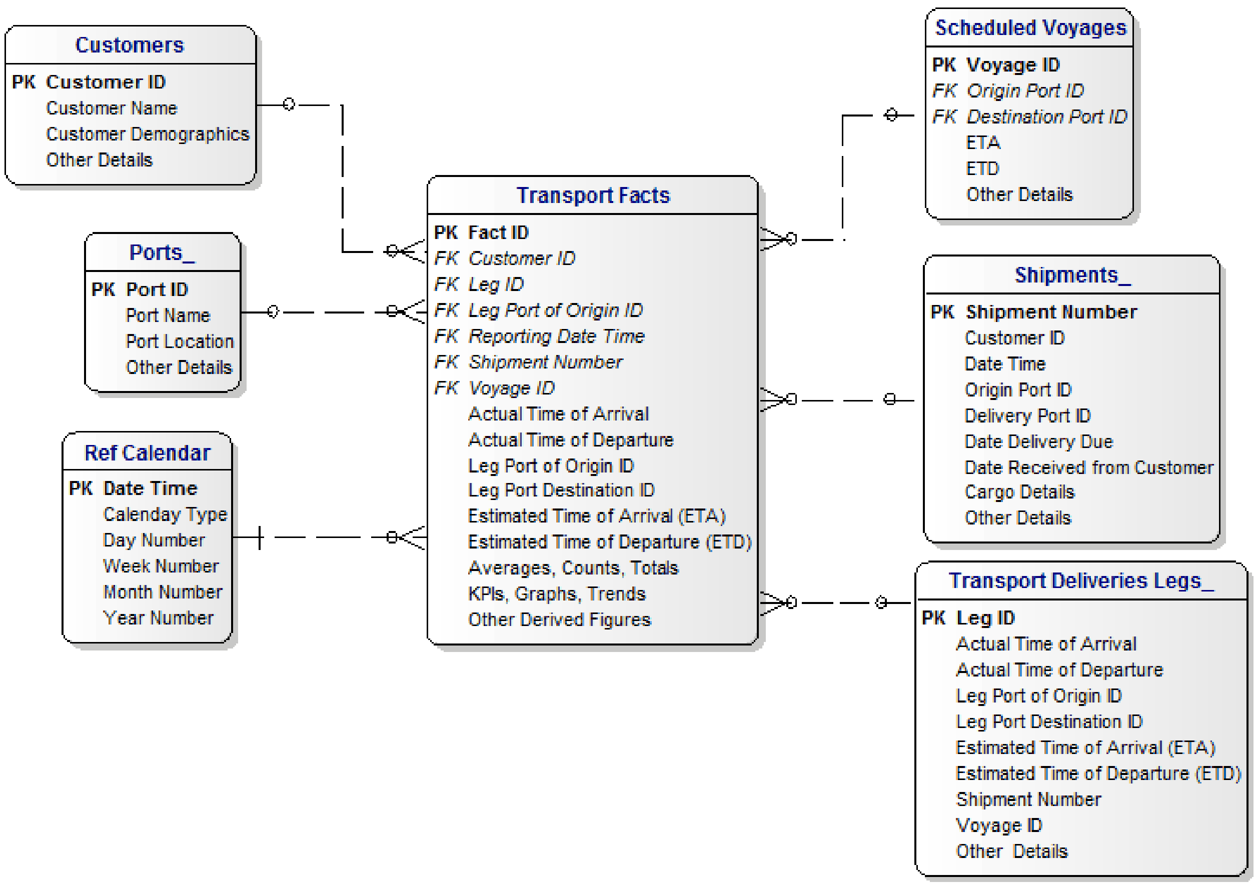
A.6.1 Data Warehouse

This shows the EDW for the analysis of Transport.



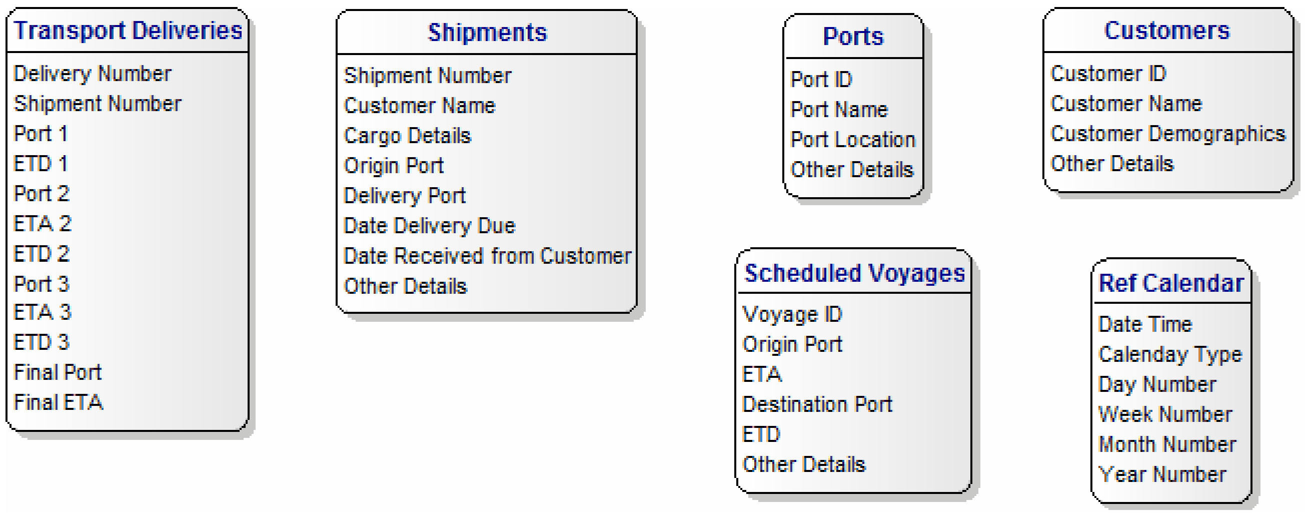
A.6.2 Dimensional Model

This shows the Dimensional Model for the Transport Subject Area.



A.6.3 Operational Data Stores (ODS)

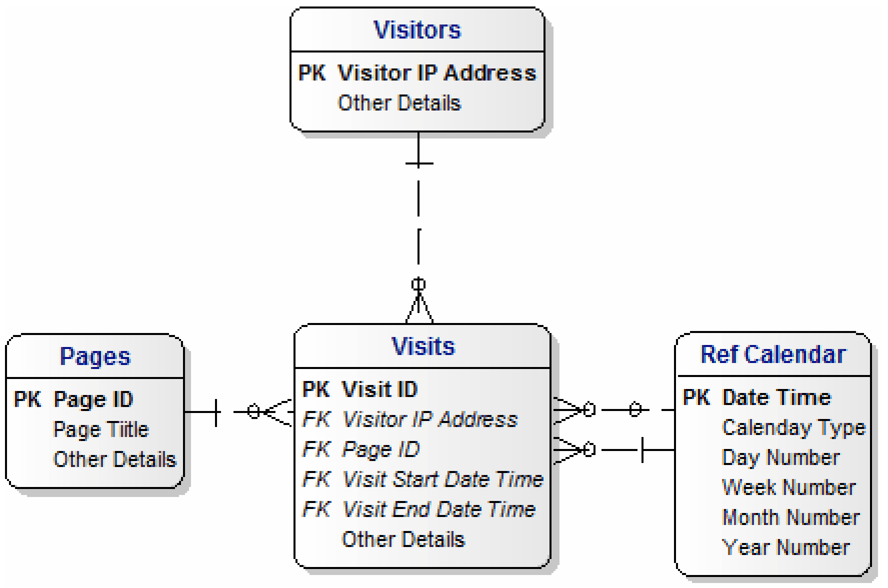
This diagram shows a sample of extract data for Transport.



A.7 Web Visit Statistics

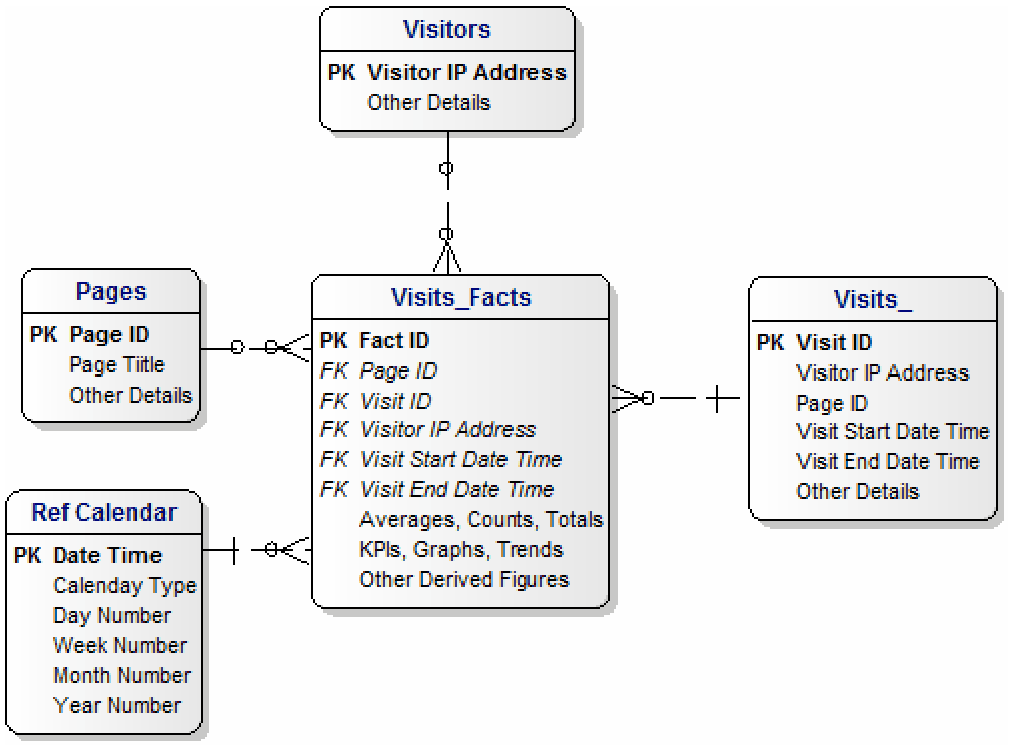
A.7.1 Data Warehouse

This shows the EDW for the analysis of Transport.



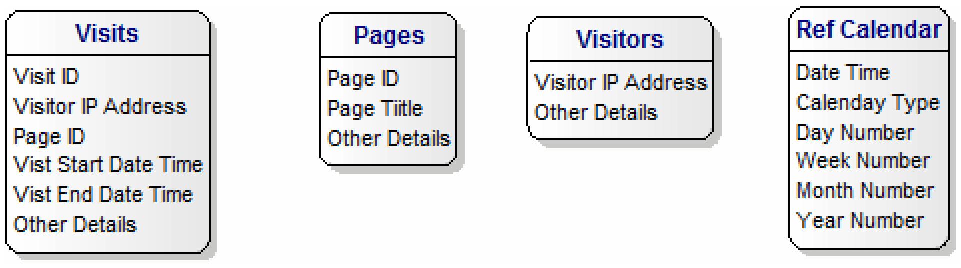
A.7.2 Dimensional Model

This diagram shows a Dimensional Model for Web Visit Stats.



A.7.3 Operational Data Stores (ODS)

This diagram shows a sample of extract data for Web Visit Stats.



Appendix B. Check the Quality of a Data Model

B.1 Introduction

B.1.1 What is This?

This chapter discusses how to check the quality of a data model.

It builds through a series of structured steps.

These steps reflect the theory that underpins relational data model.

It concludes with a checklist for assessment of an overall quality.

B.1.2 Why is it Important?

A data model often plays a fundamental part in the clarification of some key activities, such as the sources of data or the design of a database.

This makes it very valuable to be able to make an assessment of the quality of a data model.

B.1.3 What Will I Learn?

You will learn a series of steps that follow a structured path to the formal assessment of whether a data model is fit for purpose.

B.2 Create a Top-Level Business Data Model

B.2.1 Types of Data Models

All the data models that we will be discussing can be described as Entity-Relationship Diagrams, or 'ERDs'. They all show relationships between entities or tables.

At the conceptual level, the 'things of interest,' such as 'customers,' are called *entities* and at the logical or physical level they are called *tables*, because they often appear as tables in databases.

At the physical level, tables are given names in the plural, such as *Customers*, whereas at the conceptual level they often appear in the singular, that is *Customer*.

At the logical level they might be either singular or plural.

A top-level business data model can be created using Microsoft Word and is intended for business users and a non-technical audience.

The other models referred to in this document will always be created by a data modeling tool such as ERWin or IBM's Rational Rose.

They could be described as conceptual, logical or physical models.

Conceptual models show the 'things of interest' that are in scope, for example, customers and materiel. They may or may not include keys and will certainly not include physical data types, such as the length of character strings.

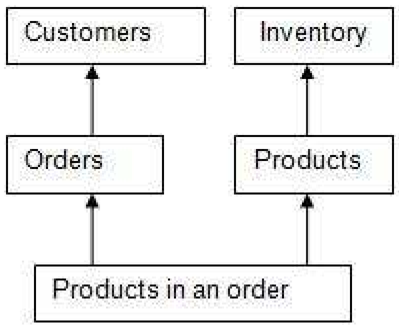
Logical models will include primary and foreign keys and often the modeling tool will provide a facility to generate a physical model from a logical one.

Physical models are often close to the actual design of an operational database. They will always show data types and field lengths.

B.2.2 Example of a Simple Business Data Model

This model was created in Word and shows customers, orders and products. The flow of logic in a data model should go from top-left to bottom-right. This means that the more fundamental things are on the top and to the left.

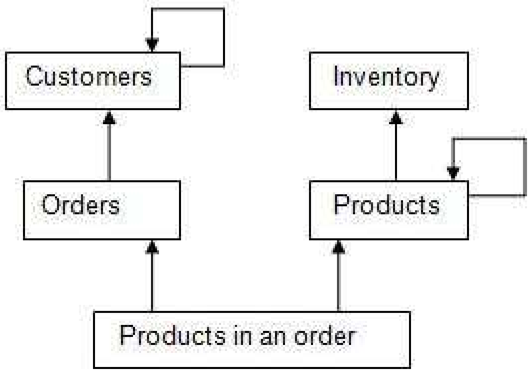
This diagram is a good example:



This version shows that customers and products each have a hierarchy so that a customer is part of a higher customer.

Similarly, a product can be part of a more complex product.

Which of these two you choose to use will depend on the audience. In general, it is better to choose the simple option.

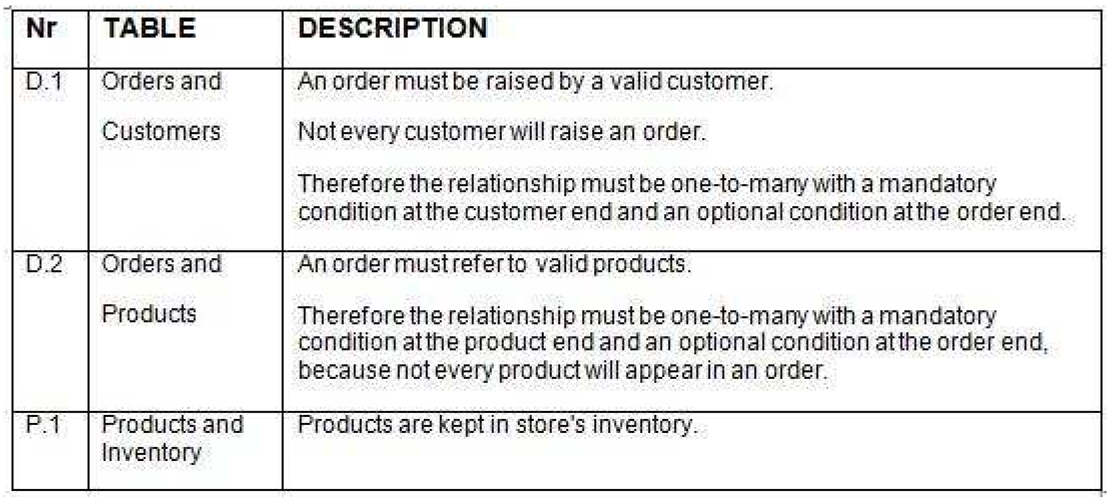


B.3 Draft the Business Rules

Business rules are valuable because they define in plain English with business terminology the underlying relationships between the terms that appear in a data model.

The user 'commCustomery' will then be able to agree and sign off the rules.

Here is a small example:

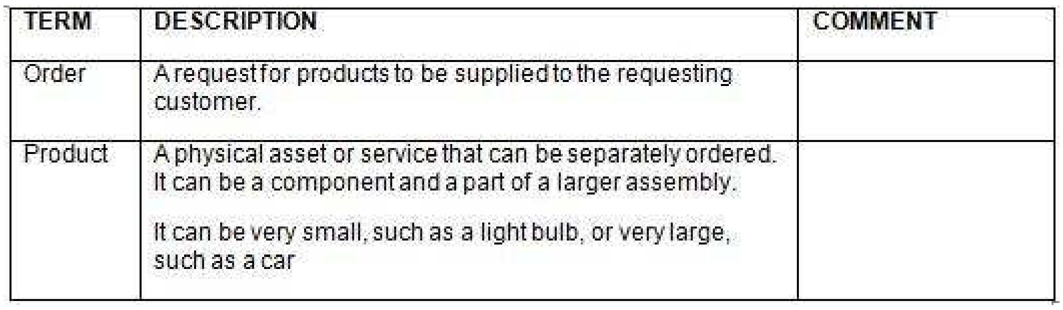


Nr	TABLE	DESCRIPTION
D.1	Orders and Customers	An order must be raised by a valid customer. Not every customer will raise an order. Therefore the relationship must be one-to-many with a mandatory condition at the customer end and an optional condition at the order end.
D.2	Orders and Products	An order must refer to valid products. Therefore the relationship must be one-to-many with a mandatory condition at the product end and an optional condition at the order end, because not every product will appear in an order.
P.1	Products and Inventory	Products are kept in store's inventory.

B.4 Draft a Glossary of Terms

It is very important to establish agreed definitions of terms and words in common use.

This is a small example:



TERM	DESCRIPTION	COMMENT
Order	A request for products to be supplied to the requesting customer.	
Product	A physical asset or service that can be separately ordered. It can be a component and a part of a larger assembly. It can be very small, such as a light bulb, or very large, such as a car	

B.5 Check that the Data Model is Correct

There may be errors that have a simple explanation. For example, the incorrect use of the modeling tool. Any errors should be discussed and resolved with the modeler and the users.

This is where the glossary and business rules are very valuable.

B.6 Review with Users

At this point, review the business rules and the glossary with users and aim to get sign off.

Make any necessary changes to format and contents.

B.7 Check Normalized Design

B.7.1 Normalized Design

This discussion applies to Entity-Relationship Diagrams (ERDs) and not to data warehouses.

We will start by defining the **rules for normalization** so that we can recognize cases where they have been broken.

Rules for Normalization

A little background is appropriate at this point.

The theory that provides the foundation for data models and ERDs was developed in 1970 by an Englishman called Ted Codd, who was a research scientist with IBM in California at the time.

Rule 1:

One of Codd's rules can be summarized as:

"The data in a table must belong to the key, the whole key and nothing but the key, so help me Codd".

This means, for example, that a record in a Customers Table must contain data only about

the customer, and nothing about people in the customer, or activities of the customer.

It might include things like the name of the customer and when the customer was founded.

Check 1: Can the values of every data item in a table be derived only from the primary key?

Rule 2:

Another of Codd's rules stated that derived data must not be included.

For example, the headcount for a customer would not be included in the Customers Table because it can be derived by counting the records of members in the customer.

Check 2: Can any data item be derived from other items?

Rule 3:

There must be no repeating groups in a table.

The one uncomfortable exception is addresses. They are very often stored as a number of repeated lines called 'Address_Line_1,' 'Address_Line_2,' and so on.

Check 3: Do any column names repeat in the same table?

Rule 4:

An item of data must only be in one table.

For example, the name of a customer would appear only in the Customers Table.

Check 4: Does the same item of data in appear in more than one table?

B.8 Reference Data

B.8.1 Background

A list should be made of the reference data referred to in a data model.

When the list is complete it should be analyzed for consistency. For example, there will not usually be any relationships between the reference data. However, if there are any, then they should be sensible and consistent. For example, a town might be in a county which would be in a country. These could all be classified a reference data that has relationships that should be validated.

Typical reference data could include ranks, and types of materiel or equipment. In passing, we should note that customers, ranks and materiel are all examples of hierarchical structures. Ranks will change only very, very rarely. However, when they are stored in a table that is joined to itself then the table will have a recursive relationship to itself. Therefore, wherever these occur, we would expect to find compact data models that include a great deal with compact and powerful structures.

B.8.2 Standards

Any appropriate national, international standards must be considered when values for reference data are decided. These include MOD, NATO and ISO standards. For example, NATO maintains standards for product classification and this is already in use within the MOD. Therefore any data model relating to products should consider this standard and where appropriate the necessary tables should be added to the model.

B.9 Slowly Changing Data

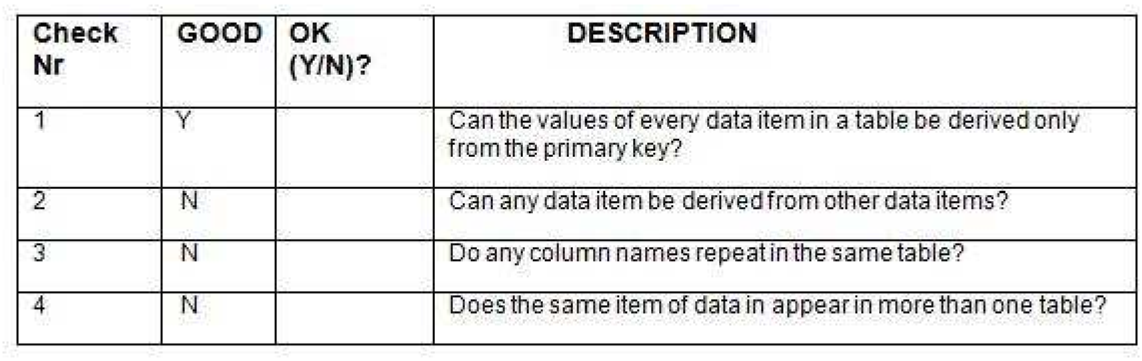
The classic example of reference data that never changes is a calendar. The values are predictable for hundreds of years ahead. There is a category in between that is usually called **slowly changing data**. This applies where the values of the data changes on roughly a six-monthly basis.

Data about categories and types is often fixed values but some can change infrequently.

For example, a new aircraft type was introduced with unmanned aircraft. The values then became fixed-wing, rotary and unmanned. This would be an example of slowly changing data.

This highlights the fact that what constitutes reference data can be subjective and may be defined differently in data models created by different people or organizations.

B.10 Check Normalization



Check Nr	GOOD	OK (Y/N)?	DESCRIPTION
1	Y		Can the values of every data item in a table be derived only from the primary key?
2	N		Can any data item be derived from other data items?
3	N		Do any column names repeat in the same table?
4	N		Does the same item of data in appear in more than one table?

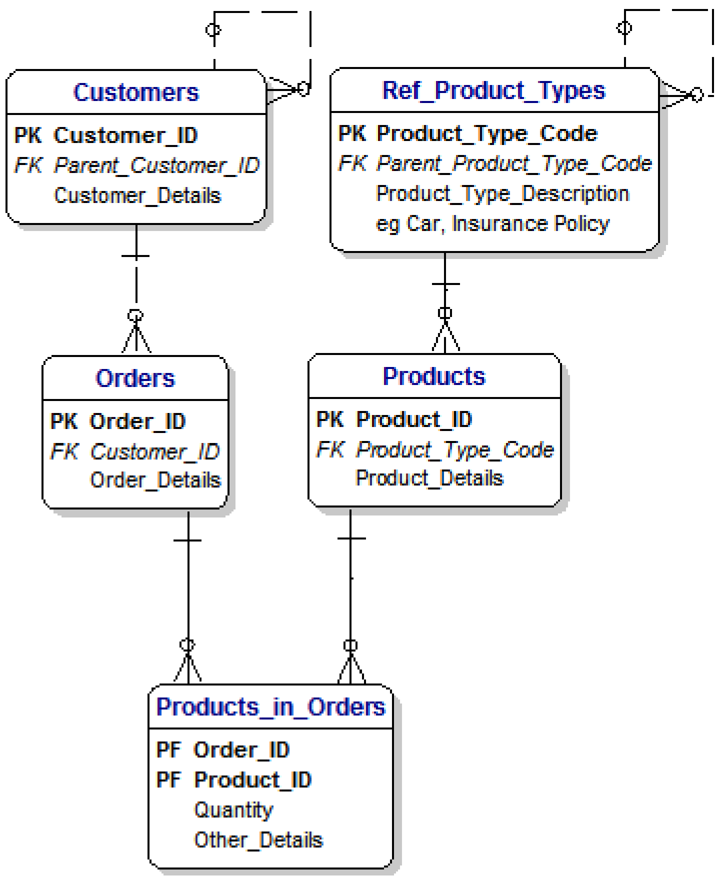
B.11 Look for Design Patterns

B.11.1 Some Examples

This data model shows examples of design patterns for one-to-many and many-to-many relationships, reflexive associations and reference data.

PK stands for 'Primary Key' and FK stands for 'Foreign Key'.

PF, which is shown in the Products_in_an_Order Table, stands for **P**rimary and **F**oreign key. This is a primary key in one table that is also a link to another table, where it is also a primary key.



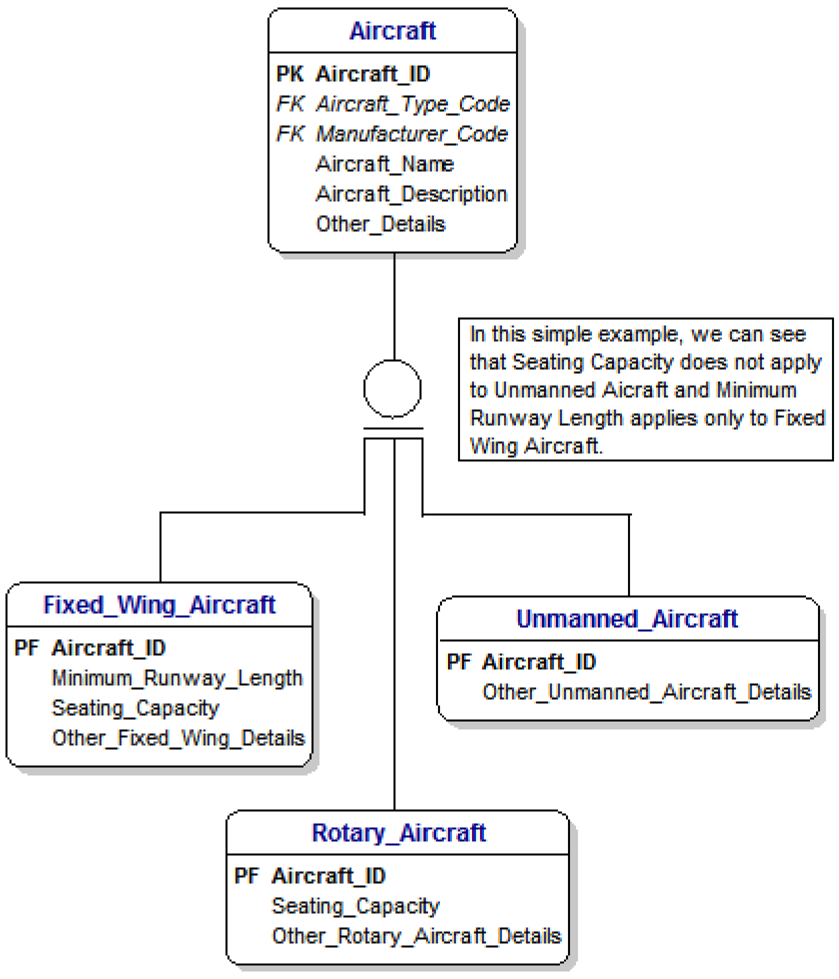
B.11.2 Inheritance

More details are provided in Chapter 3. Concepts in the document entitled "How to Understand a Data Model".

We use the concept of **inheritance** where have super-types and sub-types. Inheritance in data modeling is just the same as the general meaning of the word. 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:



B.11.3 One-to-One Relationships

We can remind ourselves that Rule 1 above states:

"The data in a table must belong to the key, the whole key and nothing but the key, so help me Codd".

One implication is that there should not be a one-to-one relationship between two tables in a model because the data can be combined into one table with the same primary key. However, there is an exception to this which is when a one-off event can occur which involves a substantial amount of data. In that case, it would not be good to create a large number of fields which will be blank in the large majority of cases.

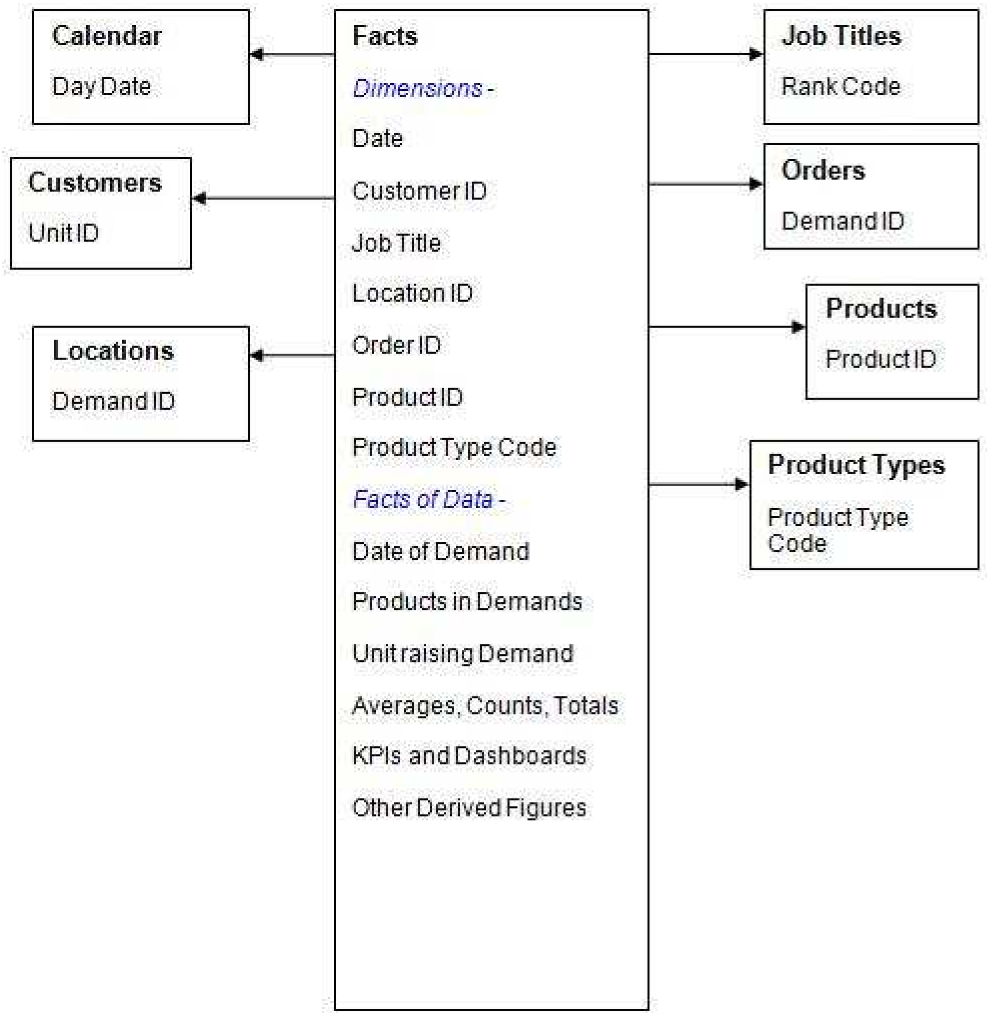
For example, when a soldier joins the army there might be data that is involved only with the joining details. The basic data for the soldier will be part of his or her basic records – such as date of birth and place of birth. If a separate table exists for 'Joining Details' then it would contain such things as date and place of joining. Then the Soldiers Table would have a one-to-one relationship with the Joining Details Table.

In other words, it can sometimes be acceptable to see a one-to-one in a data model. If that happens, it is necessary to establish the associated business rules to clarify the conditions.

B.12 Review Data Warehouses Designs

This section is relevant if the data model includes a data warehouse or data mart. A data warehouse can be a star or a snowflake design.

This diagram shows a typical data warehouse. It is a star structure with only one dimension for the related dimension tables. The arrows point from children to parents. This is a simple data warehouse for customers, orders and products.

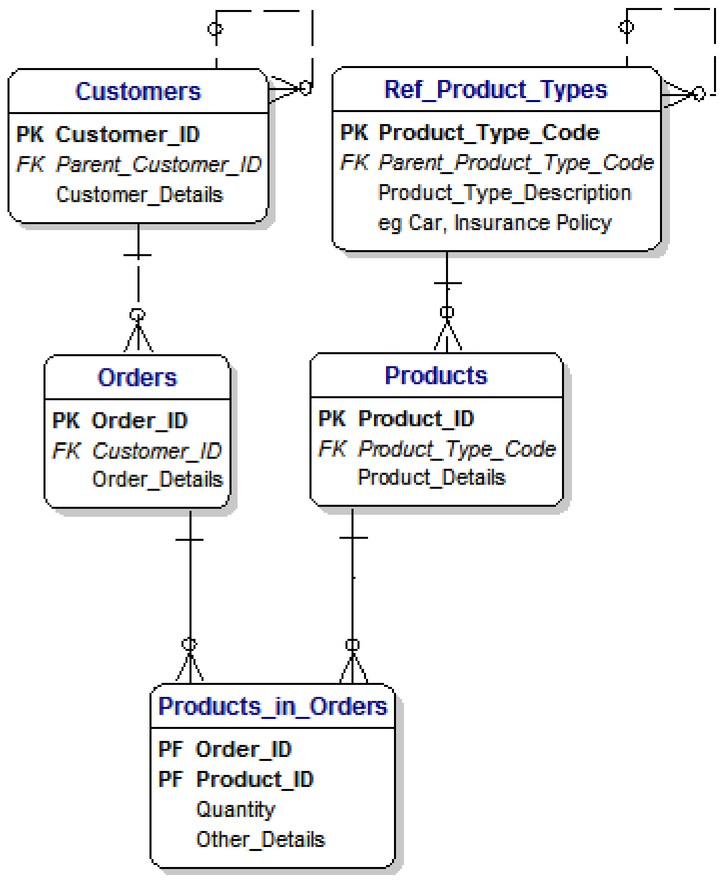


B.13 Check Naming Standards

At this step, we check for compliance with naming standards. For example, a typical standard might state that field names should be specified with underscores linking related words and first letters in capitals, such as Customer_ID. In the absence of any explicit standard, this should be the default.

This is shown in the model in Section 9.1 and also in this one.

We might say that naming standards are nice to have. In other words, they are not essential but they reflect best practice.



B.14 Check for Consistent Data Types

There are two reasons why it is important to check for consistent data types and lengths:

It avoids nasty surprises when a physical database is generated from the data model.

It is an indication of the professionalism of the manner in which the data model was produced, unless it has been reverse engineered from a database, in which case these design considerations do not apply.

For example, names should always be the same, or should be handled in a way that handles any differences in a way that ensures consistency.

Typically, a longer name should be explicitly truncated to a shorter value where appropriate.

In the absence of any explicit standard, the default for names or address lines should be VARCHAR(255) or VARCHAR2(255) for Oracle.

For other character strings they should default to *Memo* or *Text*.

B.15 Check for Defaults

We would like to see Default Values used wherever possible because they increase the discipline enforced by the model and they indicate that a thorough analysis was carried out during the creation of the data model.

For example, a ‘Start Date’ could default to the current day, or the ‘System Date’.

B.16 Determine the Assurance Level

The assurance level could be:

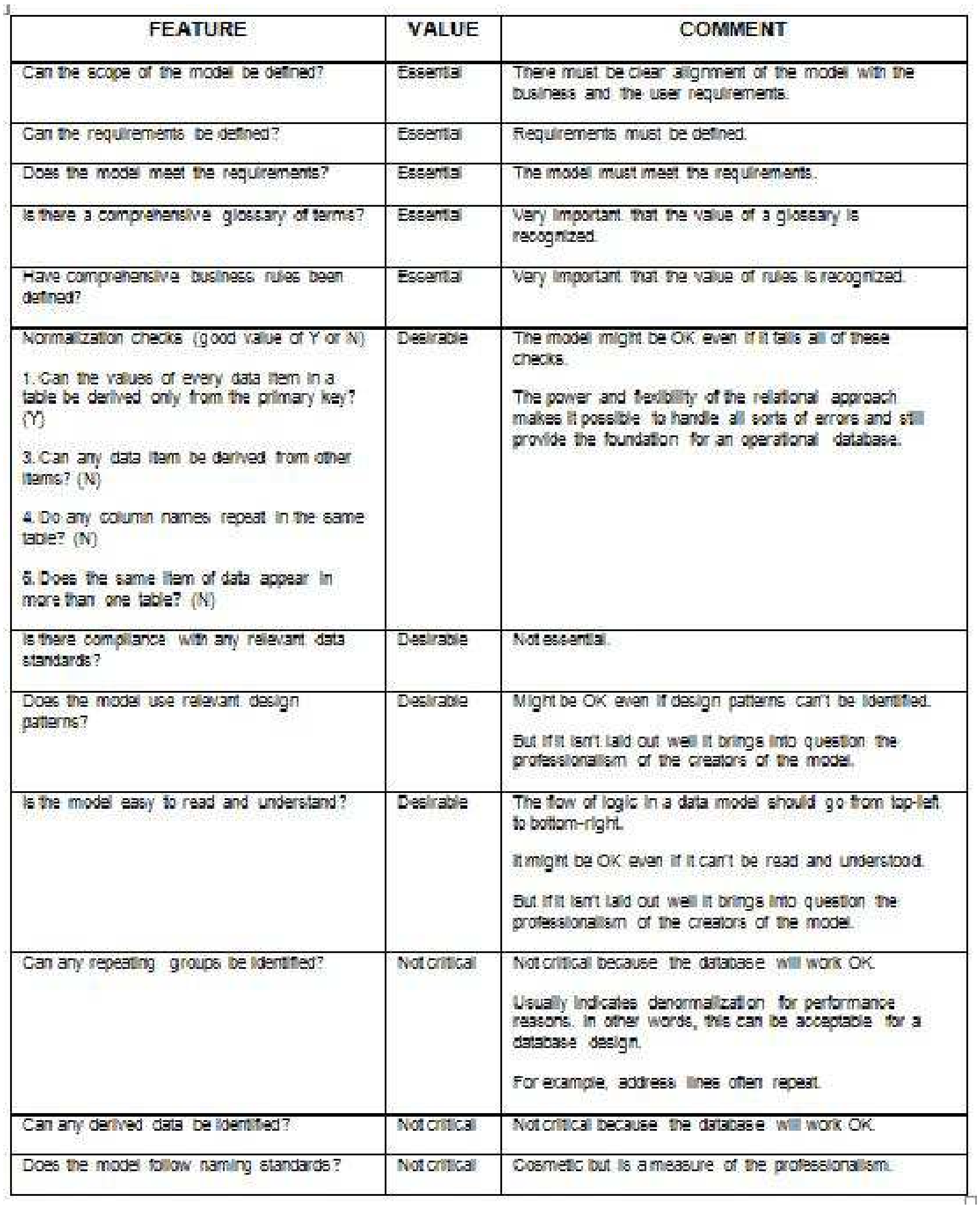
Acceptable

Acceptable with reservations

Not acceptable

B.17. Checklist for Quality Assurance

This Checklist extends the basic concept of the data model scorecard that was originated by Steve Hoberman.

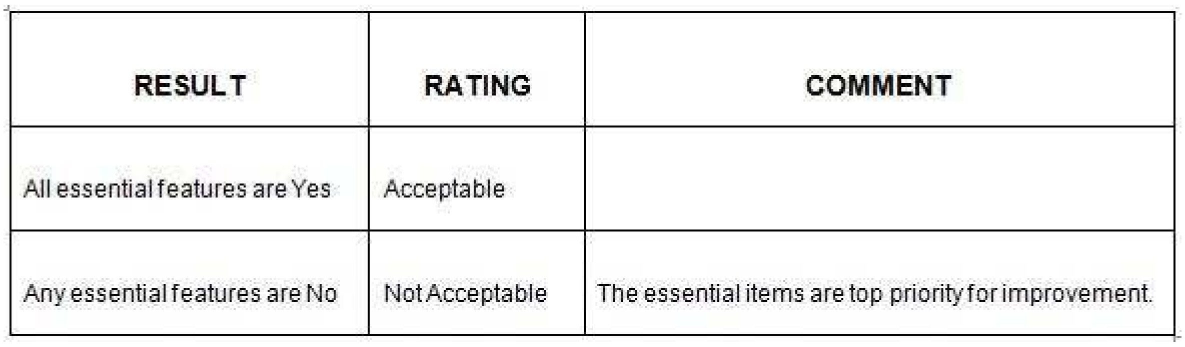


FEATURE	VALUE	COMMENT
Can the scope of the model be defined?	Essential	There must be clear alignment of the model with the business and the user requirements.
Can the requirements be defined?	Essential	Requirements must be defined.
Does the model meet the requirements?	Essential	The model must meet the requirements.
Is there a comprehensive glossary of terms?	Essential	Very important that the value of a glossary is recognized.
Have comprehensive business rules been defined?	Essential	Very important that the value of rules is recognized.
Normalization checks (good value of Y or N) 1. Can the values of every data item in a table be derived only from the primary key? (Y) 3. Can any data item be derived from other items? (N) 4. Do any column names repeat in the same table? (N) 5. Does the same item of data appear in more than one table? (N)	Desirable	The model might be OK even if it fails all of these checks. The power and flexibility of the relational approach makes it possible to handle all sorts of errors and still provide the foundation for an operational database.
Is there compliance with any relevant data standards?	Desirable	Not essential.
Does the model use relevant design patterns?	Desirable	Might be OK even if design patterns can't be identified. But if it isn't laid out well it brings into question the professionalism of the creators of the model.
Is the model easy to read and understand?	Desirable	The flow of logic in a data model should go from top-left to bottom-right. It might be OK even if it can't be read and understood. But if it isn't laid out well it brings into question the professionalism of the creators of the model.
Can any repeating groups be identified?	Not critical	Not critical because the database will work OK. Usually indicates denormalization for performance reasons. In other words, this can be acceptable for a database design. For example, address lines often repeat.
Can any derived data be identified?	Not critical	Not critical because the database will work OK.
Does the model follow naming standards?	Not critical	Cosmetic but is a measure of the professionalism.

If the answer to all the essential features is 'Yes' then the model is acceptable.

If any of the essential questions have a ‘No’ answer, the model is not acceptable.

Any ‘No’ answers to ‘Desirable’ or ‘Not critical’ questions do not affect the acceptability of the model but mean that it could be improved.



RESULT	RATING	COMMENT
All essential features are Yes	Acceptable	
Any essential features are No	Not Acceptable	The essential items are top priority for improvement.

B.17.1 Typical Summary

The results of a typical model might result in this summary:

“Reservations are that the documentation does not demonstrate that the data model meets the user requirements. The data model shows some weaknesses that the supplier has agreed to address.”

B.17.2 Follow-Up Remedial Action

A reasonable result of a QA analysis would be the identification of some problems that could be rectified fairly easily and quickly. This applies to things like documentation and naming standards. The appropriate remedial action will depend on the context and scope of the data model.

B.17.3 For a Health Check:

No action is required beyond the presentation of a report because the QA is simply to establish the ‘as-is’ situation.

B.17.4 For a Proposed Application:

It is essential that the model accurately meets the user requirements. If it does not, then it must be corrected in discussion with the users and the modeler.

B.17.5 For Data Migration:

It is essential that the model is correct at the detailed level of tables, fields and data types.

B.18 What Have We Learned?

In this chapter we have learned a structured approach to checking the quality of a data model produced by somebody else.

We have learned to determine whether it is fit for purpose.

Our approach is based on the sound theoretical foundation that is a very important element of data models for relational databases.

Appendix C. A QA Case Study

C.1 Background

This case study provides an example of the tutorial in action.

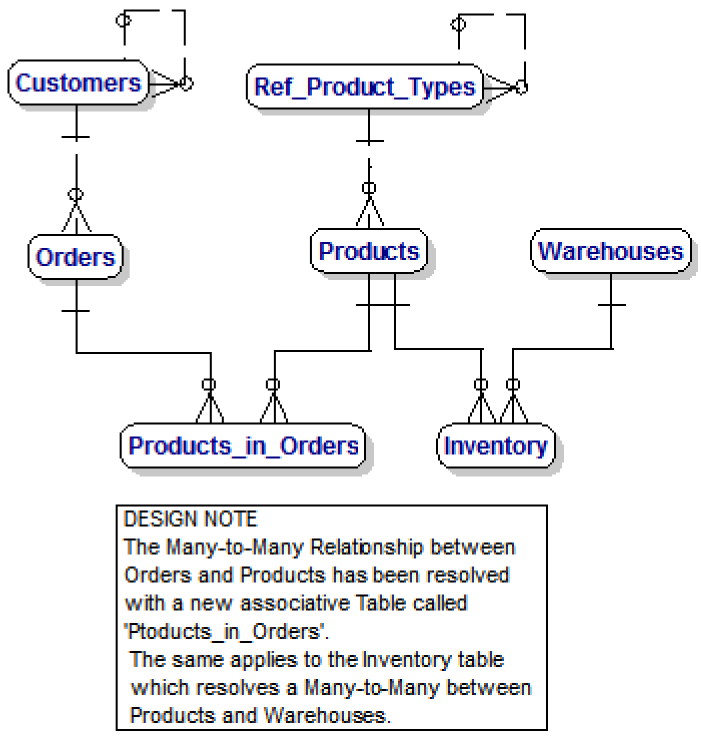
It includes blank templates and sample templates that are guidelines.

Step 1. Create a Top-Level Business Data Model

Let's assume that a data model has been provided by a third party.

The first step is to understand the data model by creating a top-level business data model.

Here is the data model that we will use as an example:


Customers
Ref_Product_Types
Orders
Products
Warehouses
Products_in_Orders
Inventory
DESIGN NOTE
The Many-to-Many Relationship between Orders and Products has been resolved with a new associative Table called 'Ptoducts_in_Orders'.
The same applies to the Inventory table which resolves a Many-to-Many between Products and Warehouses.

C.2 Our Conclusions

Our conclusions are that this is not a good data model.

Reasons include:

- It contains reference data that is not appropriate at the top level.
- There is no description of the functional area that the model supports.

Our first activity therefore is to produce an equivalent business model that we like that we can use as the basis for discussion.

Corrective actions include:

- Create a simple business data model. This should be a model in a Word document that does not include reference data.
- Produce a short description.
- Create a glossary of terms.
- Define the representative business rules.
- Identify the intended users and the owners of the model.

C.3 Functional Description

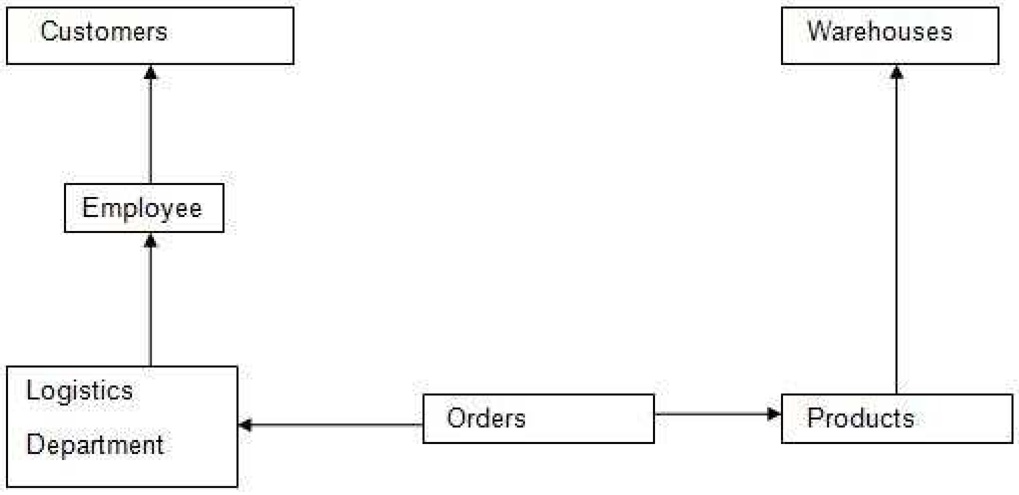
In this diagram, arrows point from children to parents. The scope of the data model includes orders for products from customers.

The Functional description is a simple one-liner:

“Customers issue orders for products that are stored in warehouses”.

C.4 A Specific Model in Word

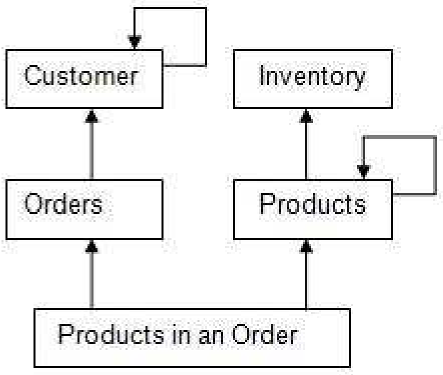
The specific version is consistent with the generic version and looks like this:



C.5 A Generic Model in Word

In this diagram, arrows point from children to parents.

Our generic data mode looks like this:

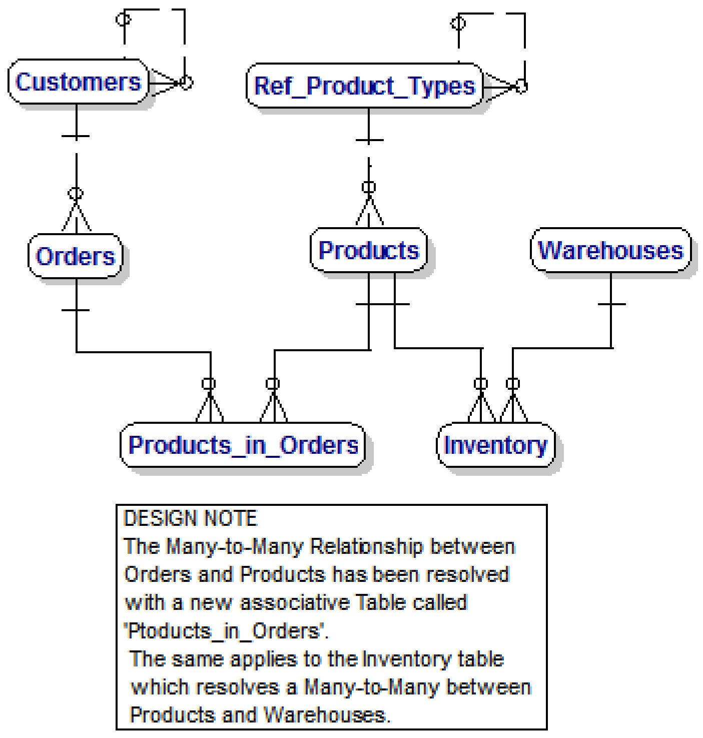


C.6 A Top-Level Generic Data Model

This is a top-level model that was created using a data modeling tool. It shows useful detail, such as details of the relationships. It also replaced a many-to-many with two one-to-many relationships.

This diagram is a generic version of the one below and is useful to help in providing a higher level context for lower-level, more specific models.

An additional level of detail shows the rabbit ears relationship that implements hierarchical relationships for customers and for products.



DESIGN NOTE
The Many-to-Many Relationship between Orders and Products has been resolved with a new associative Table called 'Ptoducts_in_Orders'.
The same applies to the Inventory table which resolves a Many-to-Many between Products and Warehouses.

This model corrects an error in the original model that we were given.

Step 2. Draft the Business Rules

These Rules must be phrased in unambiguous English. Where possible, the English should make it possible to implement a rule in a data model. For example, Rule 1 makes it clear that there is a one-to-many relationship between a ship and an officer.

A customer is staffed with many employees.

Purchasing departments raise orders.

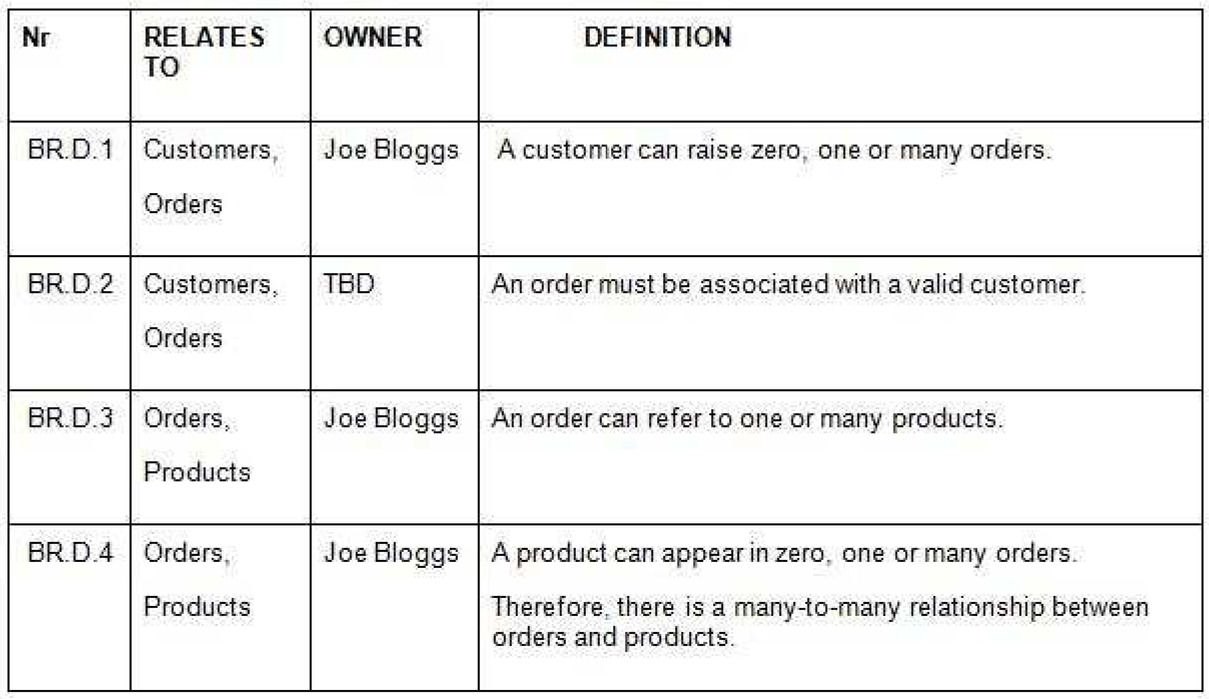
An order must be authorized by an employee.

An employee is assigned to one department at any point in time.

An employee is assigned to one or many departments during the course of their career.

Templates

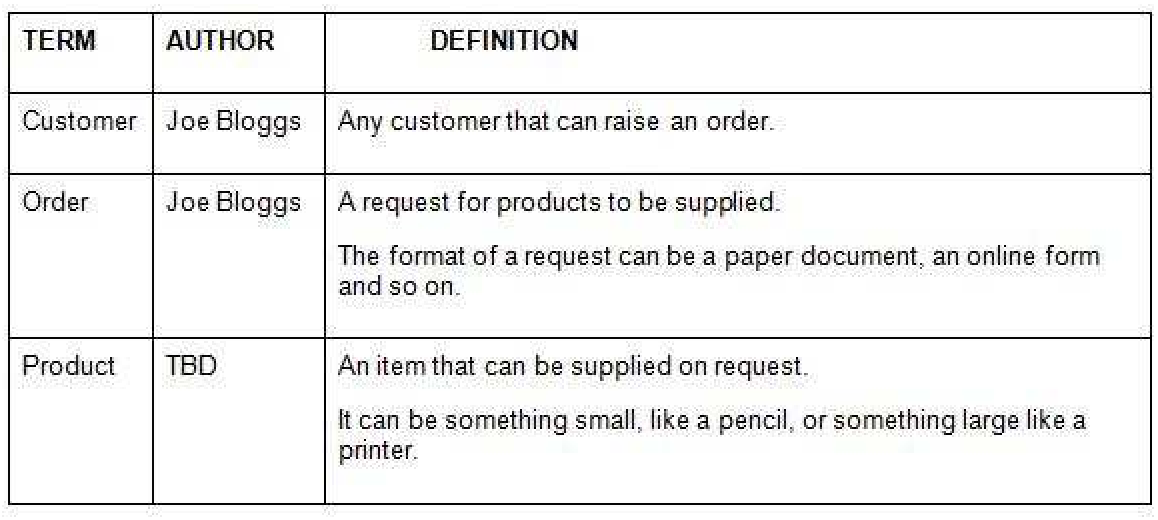
Here is a sample template for business rules:



Nr	RELATES TO	OWNER	DEFINITION
BR.D.1	Customers, Orders	Joe Bloggs	A customer can raise zero, one or many orders.
BR.D.2	Customers, Orders	TBD	An order must be associated with a valid customer.
BR.D.3	Orders, Products	Joe Bloggs	An order can refer to one or many products.
BR.D.4	Orders, Products	Joe Bloggs	A product can appear in zero, one or many orders. Therefore, there is a many-to-many relationship between orders and products.

Step 3. Draft a Glossary of Terms

This is a sample template.



TERM	AUTHOR	DEFINITION
Customer	Joe Bloggs	Any customer that can raise an order.
Order	Joe Bloggs	A request for products to be supplied. The format of a request can be a paper document, an online form and so on.
Product	TBD	An item that can be supplied on request. It can be something small, like a pencil, or something large like a printer.

Step 4. Check that the Data Model is Correct

The rules will help in determining whether the model is correct.

In this case, there is an error in that ships are shown coming between ships and departments. The reality is that departments exists without officers. This is corrected in the top-level data model in Section 1.9.

Step 5. Review with Users

Review and revise as necessary.

Step 6. Check Normalized Design

The design looks normalized and therefore is acceptable. The reference data looks appropriate and is not related and therefore is acceptable.

Step 7. Look for Design Patterns

This business model shows these examples of design patterns:

A one-to-many relationship between ship and office.

A many-to-many relationship between requisition and product.

It does not show inheritance but in general we would not expect to find it.

There are a number of reasons why inheritance does not appear.

For example:

Inheritance is not appropriate in this case.

Inheritance does not show in a data model for a physical database.

Step 8. Review any Data Warehouses

In this case study, this step is not necessary because we do not have a data warehouse or data mart.

Step 11. Check Naming Standards

Standards that are common include:

Initial capitals with lower case elsewhere – for example, Customer_id

All capitals – for example, CUSTOMER_ID

Lower case everywhere – for example customer_id

Any of these standards is acceptable.

If no standard has been established, then number 1 is recommended.

Step 9. Check for Consistent Data Types

This check requires a physical data model or some other document that includes this level of detail. The procedure then is to visually scan the documents or use the domain feature of the modeling tool or perhaps SQL to look for discrepancies.

The domain feature allows you to define standard data types for any data item that occurs frequently and then use this domain for every occurrence of the data item.

For example, a name could be defined as a variable-length character string with a length of 258.

Then whenever a name appears in a model, the modeler can select this domain as convenient shorthand and also a simple way to enforce consistency.

It will be necessary to analyze any discrepancies and decide on a standard to resolve them.

Step 10. Check for Defaults

Default values are a powerful technique for adding values in a data model. They can be used to enforce consistency.

Probably the most common example is to specify that the date of entry and creation of a new record should be the current system date.

This applies to new customers, orders and the date of any payment or adjustment and so on.

Step 11. Determine the Assurance Level

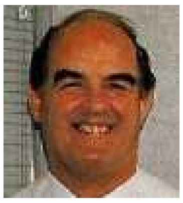
Appendix A defines the process to be followed and discussed appropriate remedial follow-up action.

C.7 What Have We Learned?

We have learned some very important rules that will help us to assess the quality of a data model created by someone else.

They are a combination of theory based on Dr. Codd's original work at IBM (see http://www.databaseanswers.org/codds_page.htm) and some best practice guidelines based on our experience.

(This is the back cover.)



Barry Williams is the founder and principal consultant with Database Answers.

His company has been providing advice and assistance to a wide range of blue-chip clients for over 20 years.

His particular interest is in advancing the role of data models as a way of improving communication between the business user community and data management professionals.

As part of this role he publishes best practice on his Database Answers Web site at

http://www.databaseanswers.org/