Database Answers

Learn Data Modelling

by Example

Barry Williams

Part 2 - Table of Contents

First Edition: London, 2011

ISBN-13: 978-1466237414

Welcome

This is the second Part of our three-part Book On Data Modelling.

This book has been produced in response to a number of requests from visitors to our Database Answers Web Site.
It incorporates a selection from our Library of about 950 data models that are featured on our Web site:

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

I hope you enjoy this Book and would be very pleased to have your comments at barryw@databaseanswers.org.

Barry Williams
Principal Consultant
Database Answers Ltd.
London, England

6. How to Understand a Data Model

6.1 Introduction

6.1.1 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.

6.1.2 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.

6.1.3 What Will I Learn?

You will 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 that occur on a regular basis.

This tutorial starts with some simple concepts and then discusses common design patterns based on these concepts.

The tutorial applies in two situations:
i) Data models created by **reverse engineering** existing databases.
ii) Other data models.

This tutorial 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 attempt 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 3.4), 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 modeling 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, 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 book.

ii) For the second situation it is appropriate to perform 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.

6.2 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 modeling 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.

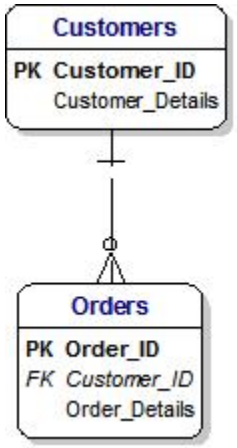
Physical models are often used to generate SQL to create database tables. They can also be created by reverse engineering from an existing operational database.

6.3 Concepts

6.4.1 One-to-Many Relationships

A customer can place many orders for products. This defines a one-to-many Relationship.

A data modeler 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 organization 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.

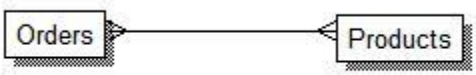
Business Rules: A customer can raise zero, one or many orders.
: An order must be associated with a valid customer.

Blank Template:

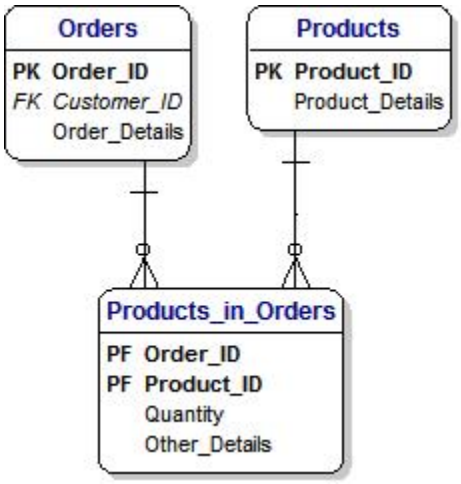
TERM	AUTHOR	DEFINITION

6.4.2 Many-to-Many Relationships

We can also say that an order can request many products. A data modeler 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:



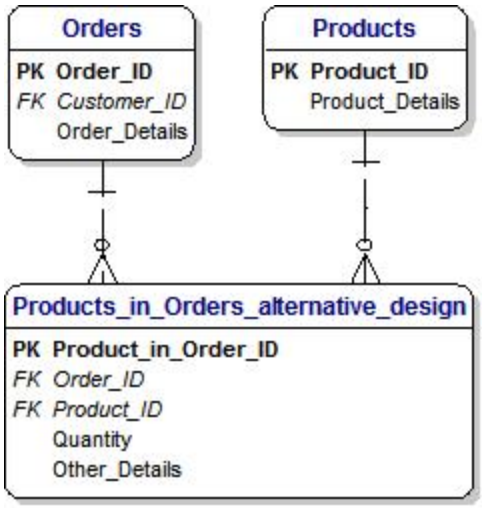
Many-to-many relationship cannot be implemented in relational databases. Therefore we resolve this many-to-many into two one-to-many relationships, which we show in a data model as follows:



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, to reflect the standard design approach of using a surrogate key as the primary key and show the order and product IDs simply as foreign keys. We learnt in Section 2 that a surrogate key is simply a key that stands for something else. It is a meaningless integer that is generated automatically by the database management software, such as Oracle or SQL Server. The values are usually consecutive integers, starting with 1,2,3,4 and so on.

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.

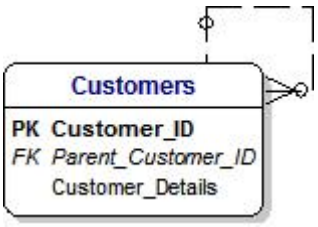
6.4.3 Rabbit Ears

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 organization. In other words, every customer reports to a higher level within the overall organization.

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 *rabbit ears*, and looks like this:



The customer at the very top of organization 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 that 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.

6.4.4 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.

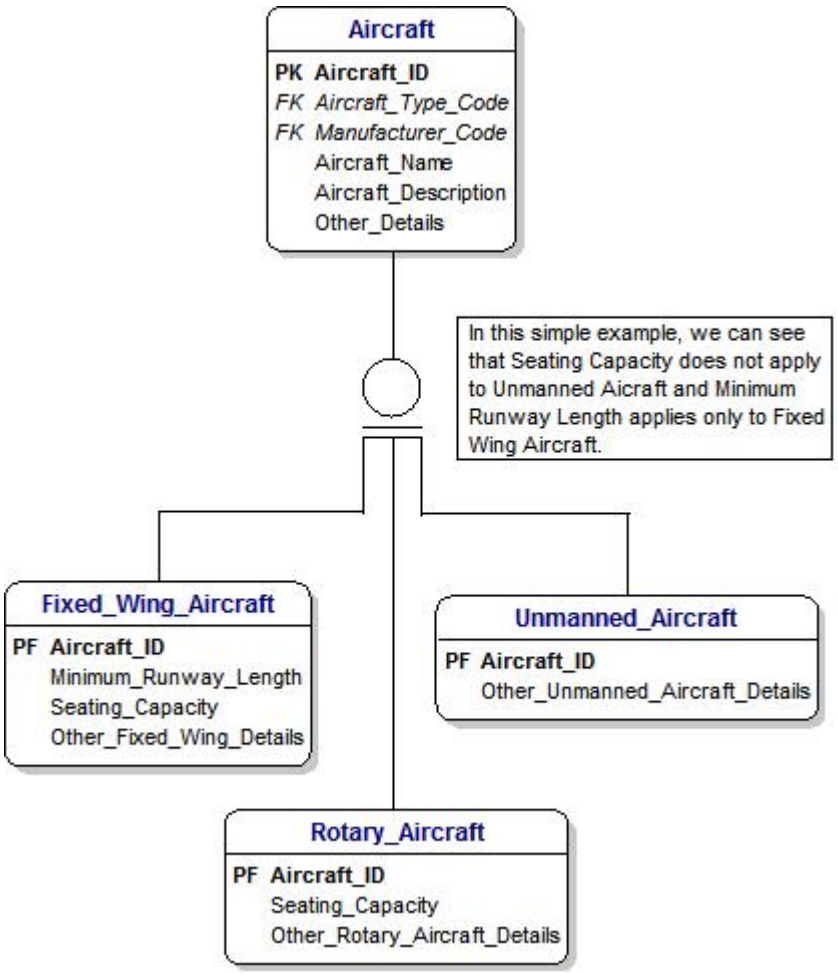
In a similar way, when we discuss aircraft we might be talking about rotary aircraft, fixed-wing aircraft and unmanned aircraft.

However, when we want to design or review a data model that includes aircraft, then we need to analyze how different kinds of aircraft are shown in the design of the data model.

We use the concept of **Inheritance** to achieve this. 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 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:



6.4.5 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.

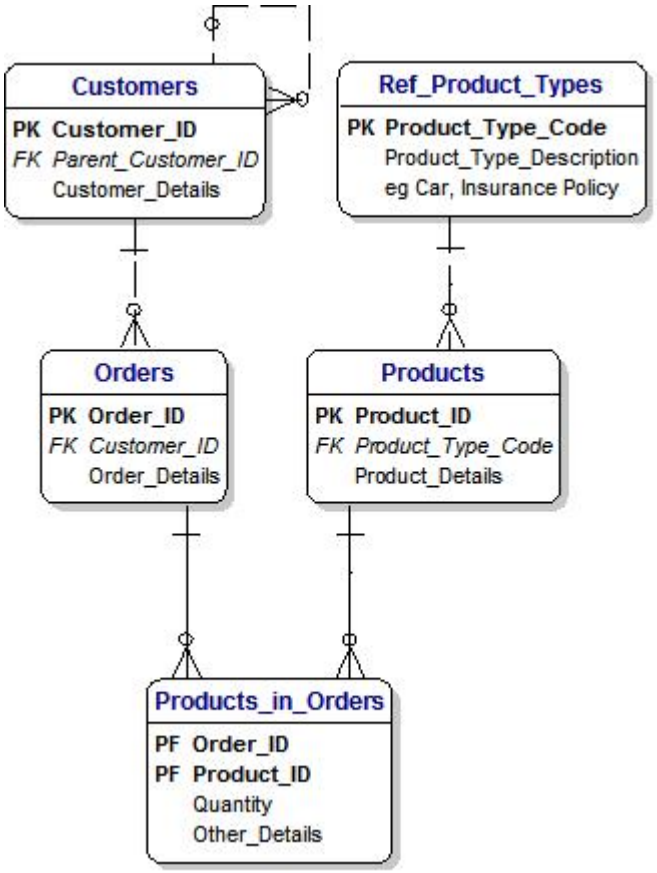
This diagram shows two basic examples of reference data that might apply to our simple aircraft data model.

6.4 Data Warehouses in Contrast to ERDs

Different considerations apply to data warehouses than apply to **Entity-Relationship-Diagrams** (ERDs). For the purpose of this discussion, we include data marts with data warehouses.

6.6.1 Design of an ERD

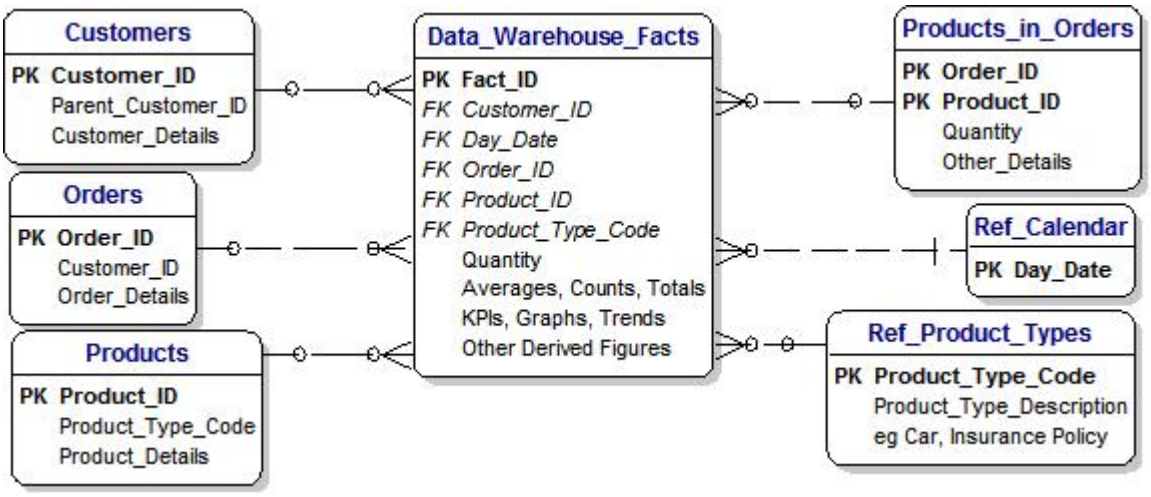
This data model is an Entity-Relationship-Diagram (ERD) for customers and orders. In passing, we observe that we use the prefix 'Ref_' to indicate reference data.



We could describe it in these terms:
"Customers place orders for products of different types."

6.6.2 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.



6.6.3 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 (i.e. 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 inquiries need not involve any particular dimension. The one exception to this rule is that the relationship to the calendar is mandatory because an inquiry will always include a date. Of course, an inquiry 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?

6.5 Design Patterns

6.7.1 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_6.
- The Town or City
- The Postcode
- The County
- The Country

However, in the United States, the US Postal Service has established a standard of two lines for addresses

A professional approach is to store addresses in a dedicated address file. One great benefit of this approach is that this allows the use of commercial software to validate the addresses.

In the UK, the Post Office PAF file is used for validation of addresses and to support a standard format for storing addresses.

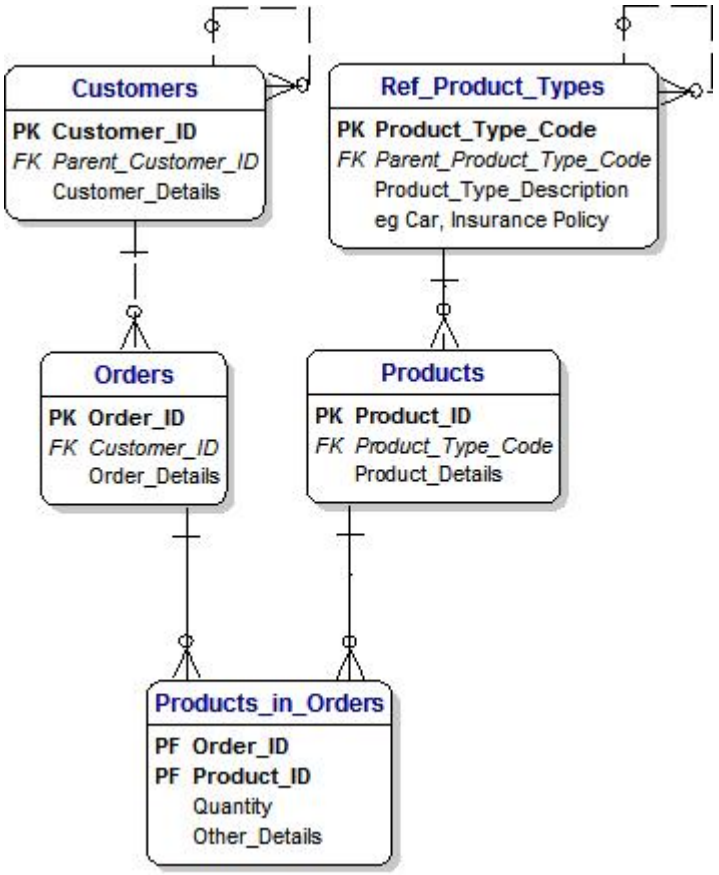
A variety of vendors, such as QAS, offer software to use the PAF file for validation.

6.7.2 Customers and Orders

The design of the ERD in the chapter on data warehouses shows a typical customers and orders data model, which represents a widespread kind of application.

6.7.3 Customers, Products and Orders

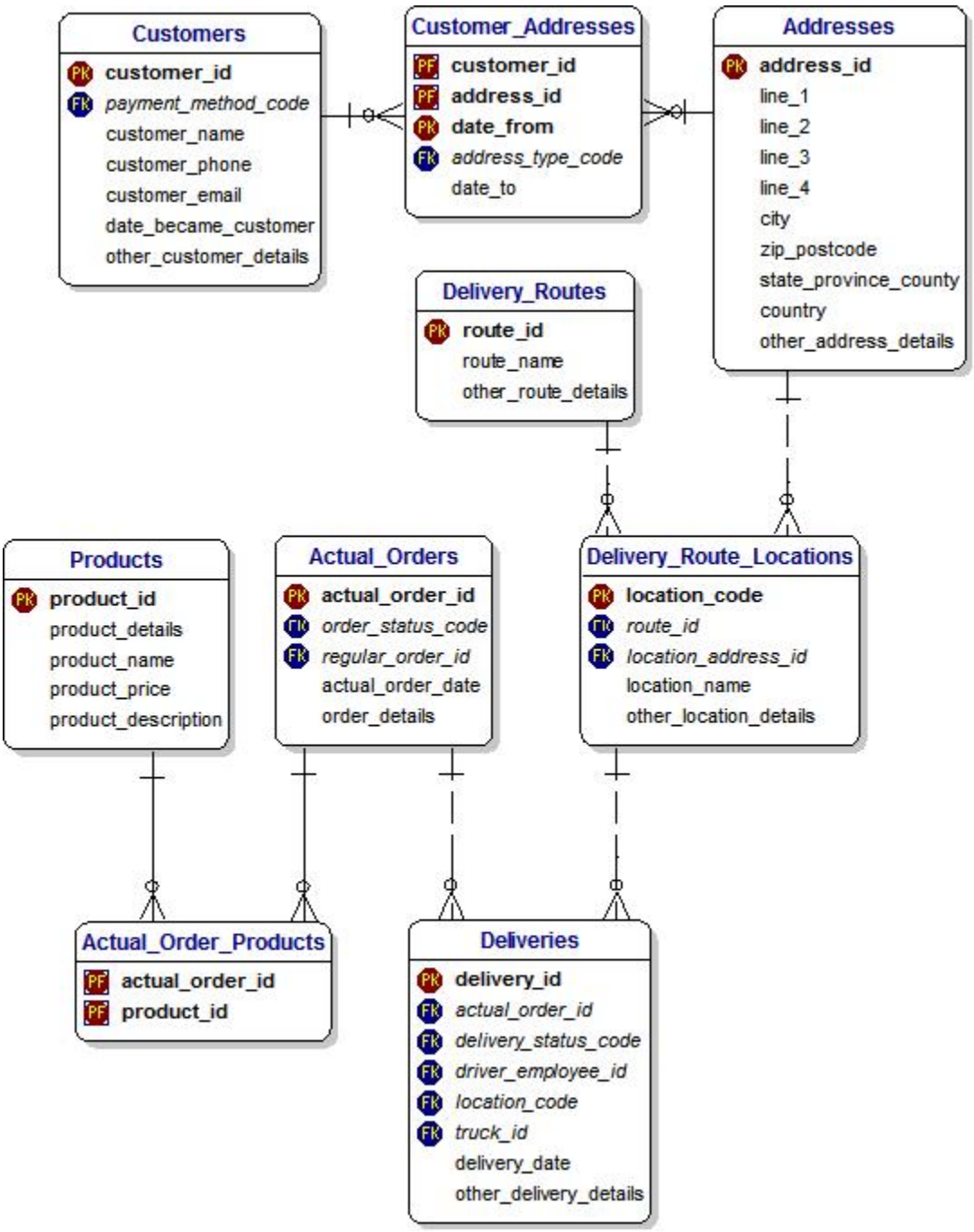
This data model demonstrates the power of rabbit ears. It shows customers in an organizational hierarchy and product types in a hierarchical catalogue.



6.7.4 Deliveries

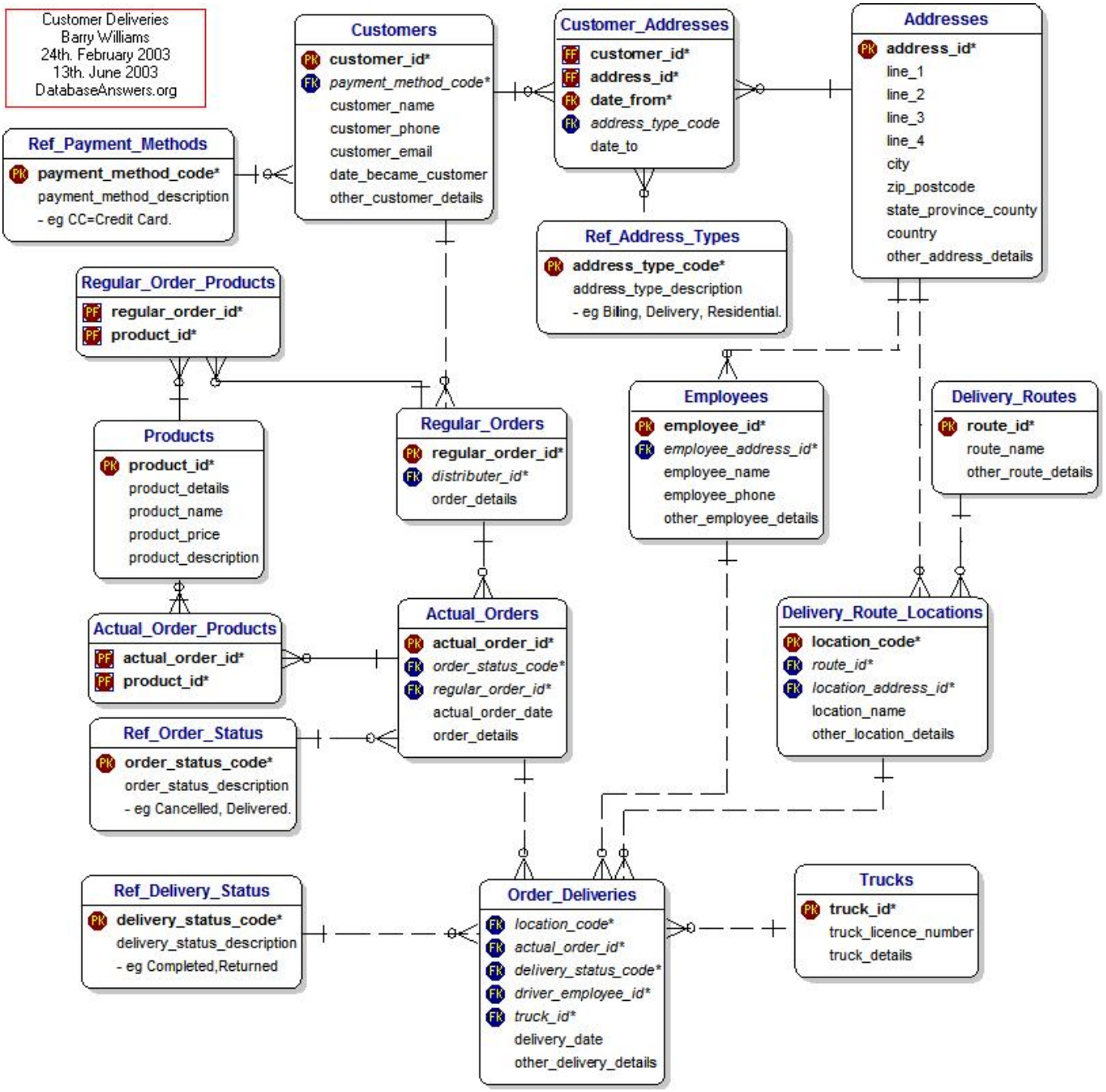
6.7.6.1 A Simple Design Pattern

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".



6.7.6.2 A Complex Design Pattern

This shows a complex pattern that adds regular orders to the simple model shown above.

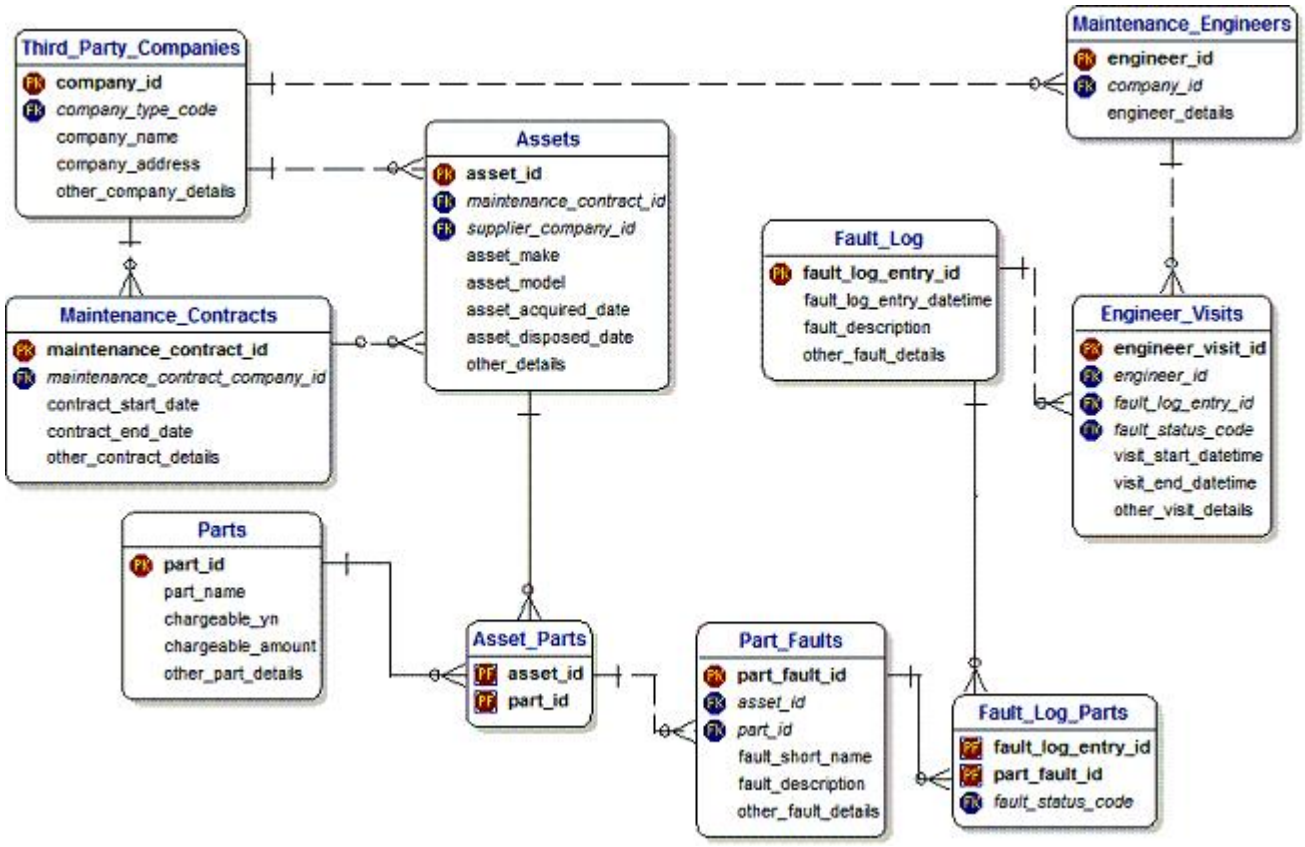

Customer Deliveries
Barry Williams
24th. February 2003
13th. June 2003
DatabaseAnswers.org
Customers
customer_id*
payment_method_code*
customer_name
customer_phone
customer_email
date_became_customer
other_customer_details
Customer_Addresses
customer_id*
address_id*
date_from*
address_type_code
date_to
Addresses
address_id*
line_1
line_2
line_3
line_4
city
zip_postcode
state_province_county
country
other_address_details
Ref_Payment_Methods
payment_method_code*
payment_method_description
- eg CC=Credit Card.
Ref_Address_Types
address_type_code*
address_type_description
- eg Billing, Delivery, Residential.
Regular_Order_Products
regular_order_id*
product_id*
Products
product_id*
product_details
product_name
product_price
product_description
Regular_Orders
regular_order_id*
distributer_id*
order_details
Employees
employee_id*
employee_address_id*
employee_name
employee_phone
other_employee_details
Delivery_Routes
route_id*
route_name
other_route_details
Actual_Order_Products
actual_order_id*
product_id*
Actual_Orders
actual_order_id*
order_status_code*
regular_order_id*
actual_order_date
order_details
Delivery_Route_Locations
location_code*
route_id*
location_address_id*
location_name
other_location_details
Ref_Order_Status
order_status_code*
order_status_description
- eg Cancelled, Delivered.
Ref_Delivery_Status
delivery_status_code*
delivery_status_description
- eg Completed,Returned
Order_Deliveries
location_code*
actual_order_id*
delivery_status_code*
driver_employee_id*
truck_id*
delivery_date
other_delivery_details
Trucks
truck_id*
truck_licence_number
truck_details

6.7.5 Maintenance

6.7.7.1 Scope

The scope of this data model is the maintenance of products by third-party companies.
The business rules state:

- A product such as a washing machine can have a maintenance contract.
- A 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 that are recorded in a fault log.
  - They correct the faults and this is recorded in the fault log.



6.7.6 Subject Areas

Complex data models that are common in large organizations 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'.

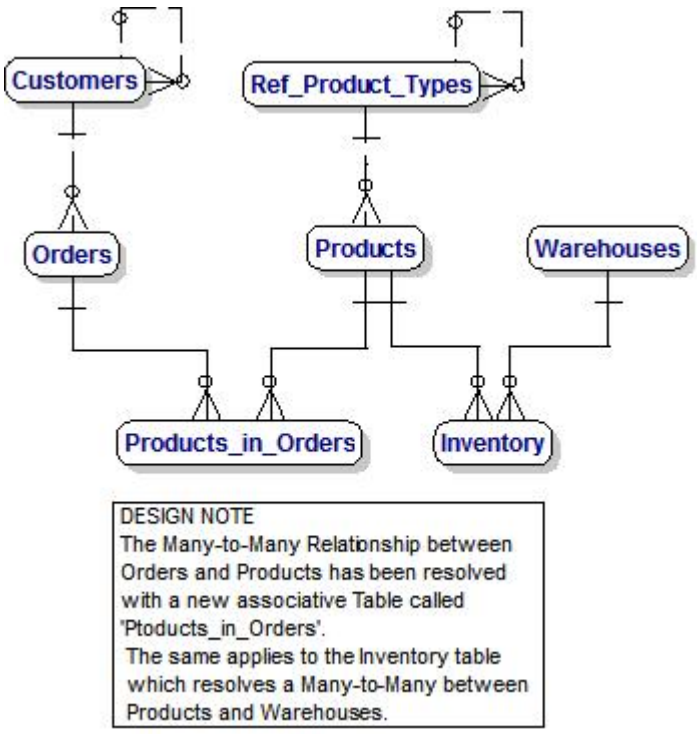
These are shown in earlier Sections of this document.

6.7.8.1 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.

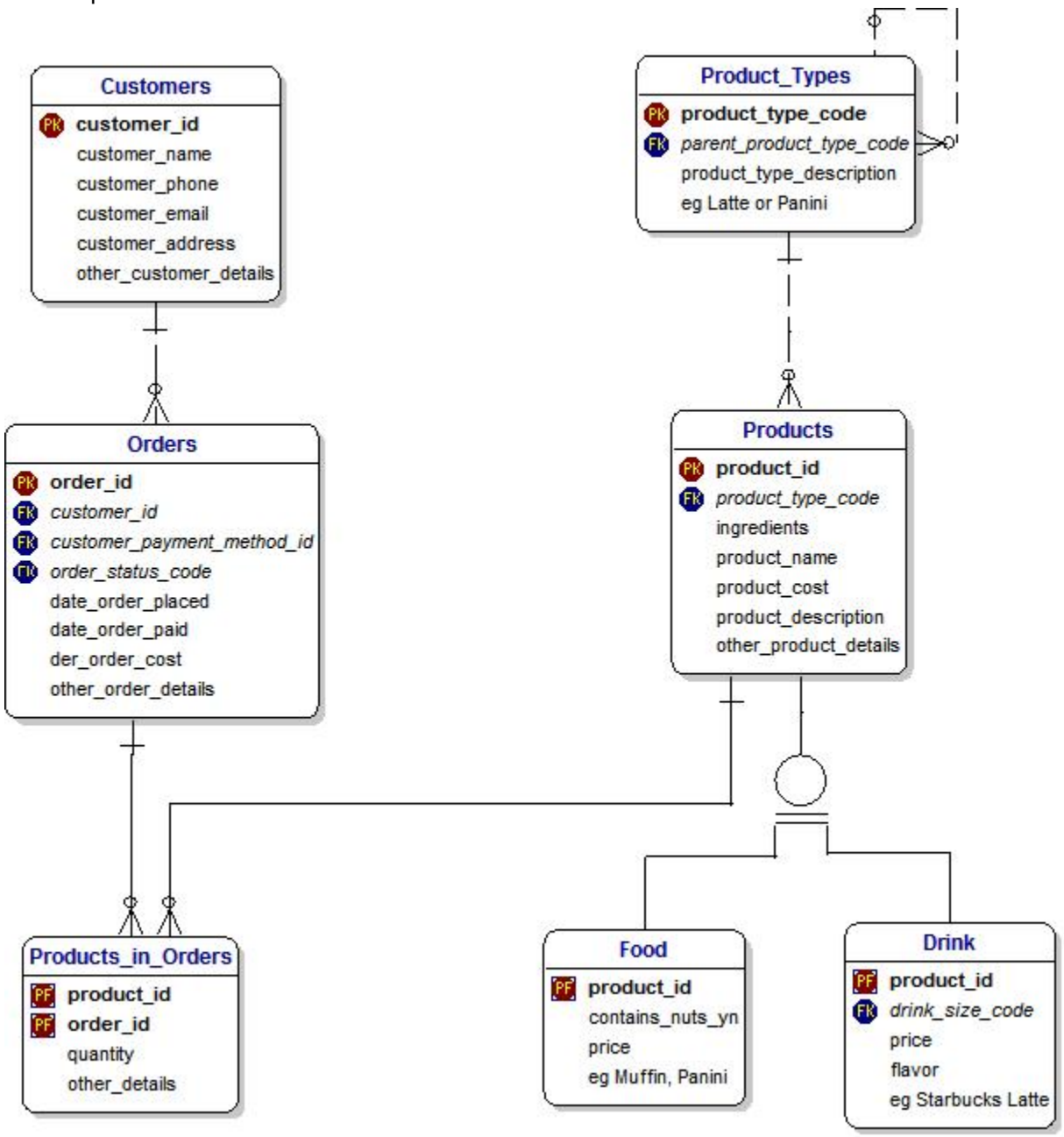


6.6 What Have We Learned?

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 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 is a challenging and valuable exercise.

It would be useful practice to look at this data model where you can see examples of each of these concepts and describe them out loud:



7. Design Patterns

7.1 Introduction

This chapter will discuss the importance of design patterns and give some examples.

7.1.1 What is this?

Design patterns are solutions to common problems.

7.1.2 Why is it important?

They are important because they occur all around us.

Design patterns exist because they reflect the existence of common problems and common solutions in the world of data modeling. Because of this, when we want to understand a data model, an obvious starting-point is to look for common design patterns.

This tutorial can help you to look at a data model and understand it. It is based on the concept of design patterns that are general solutions to common problems that occur on a regular basis.

This tutorial starts with some simple concepts and then discusses common applications that use these concepts.

This applies in two situations:

- Data models created by **reverse engineering** existing databases.
- 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 attempt a quality assurance of the model. This is primarily because databases in operational systems have usually gone through a series of changes and the impact on design has not been thought through. As a result there has not been the knowledge, commitment or 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, which can only be identified by implication.

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.

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.

7.1.3 What Will I Learn?

You will be able to recognize design patterns when they occur and find it much easier to cope with complex situations because you will be able to break them down into simple components.

7.2 Addresses

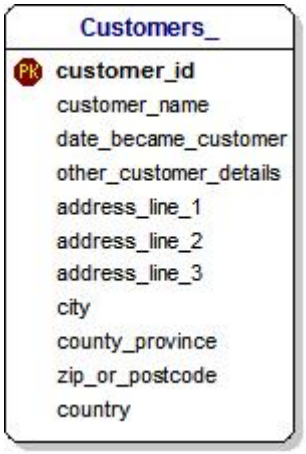
This is taken from the data model shown on this page:

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

There are several options for modeling addresses.

Option 1: The Simplest Design

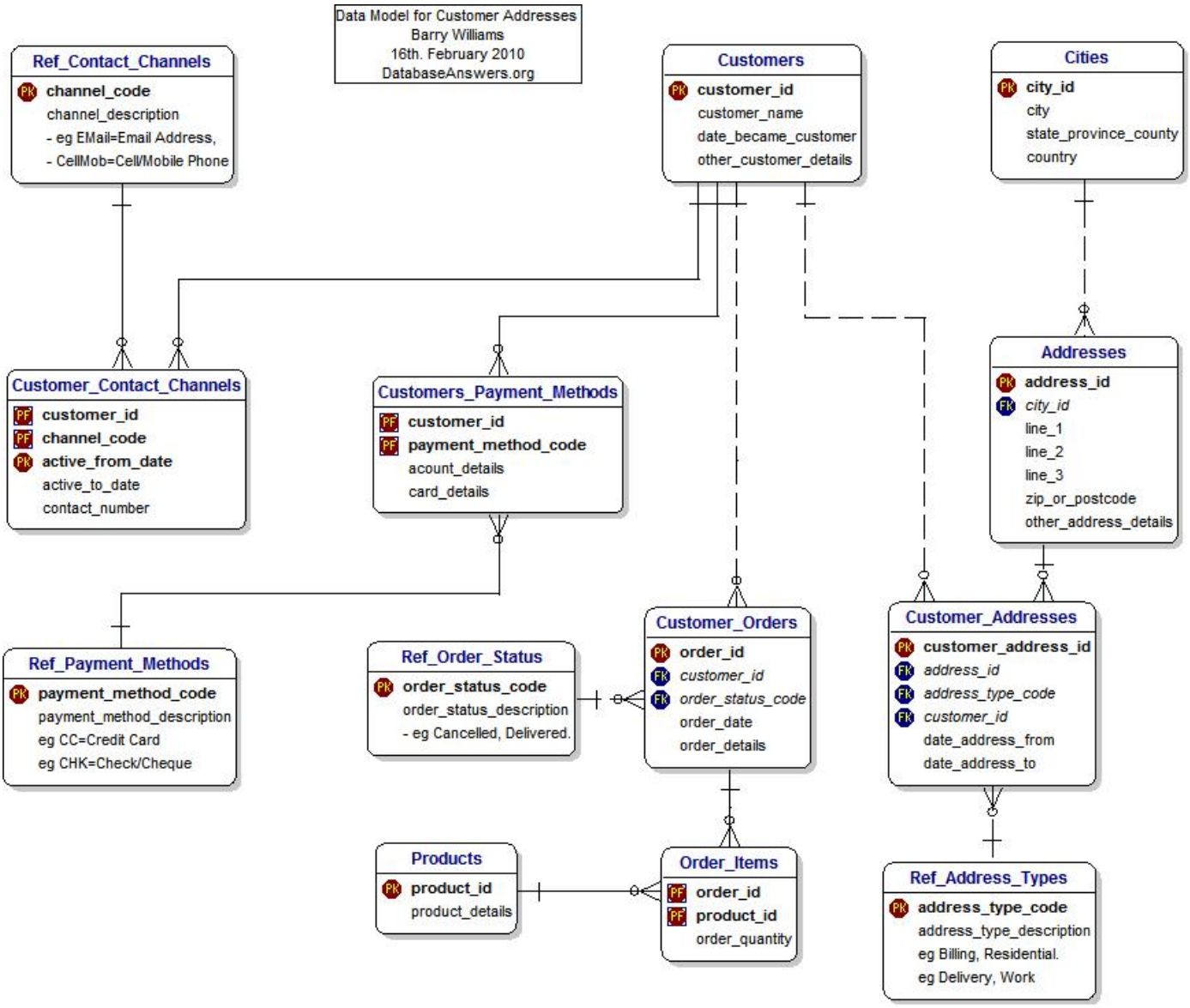
This is the most simple and most basic design. It stores the address in the Customer Table:



Option 2: A Separate Address Table

This design stores the address in a separate table, as shown in this example.

The Customer_Addresses Table makes it possible for a customer to have many addresses of different types.



Option 3: A More General Standards-Based Design

This design also stores the address in a separate table which then makes it possible to validate addresses using bespoke or commercial software.

This is a very valuable option because it does a lot in a short time and at a relatively low cost.

An example is QAS software:

- In the US - http://www.qas.com/
- In the UK - http://www.qas.co.uk/

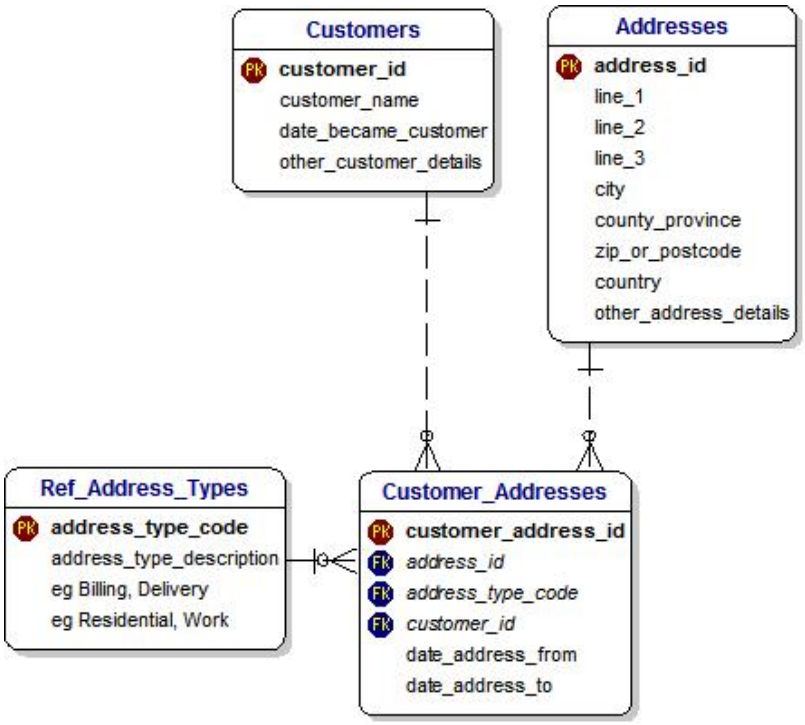
There is a two-line standard recommended by the US Postal Service:

- http://pe.usps.com/text/pub28/PUB28C3.html
- http://pe.usps.com/

You can find a discussion of the Universal Postal Union UPU S42 International Address Standard, published in 2003 by following this link:

- http://xml.coverpages.org/Lubenow-UPUS43.html

The UK standard is called the PAF file ('Post Office Address File'), which favors four lines. This design is compatible with the US and the UK standard:



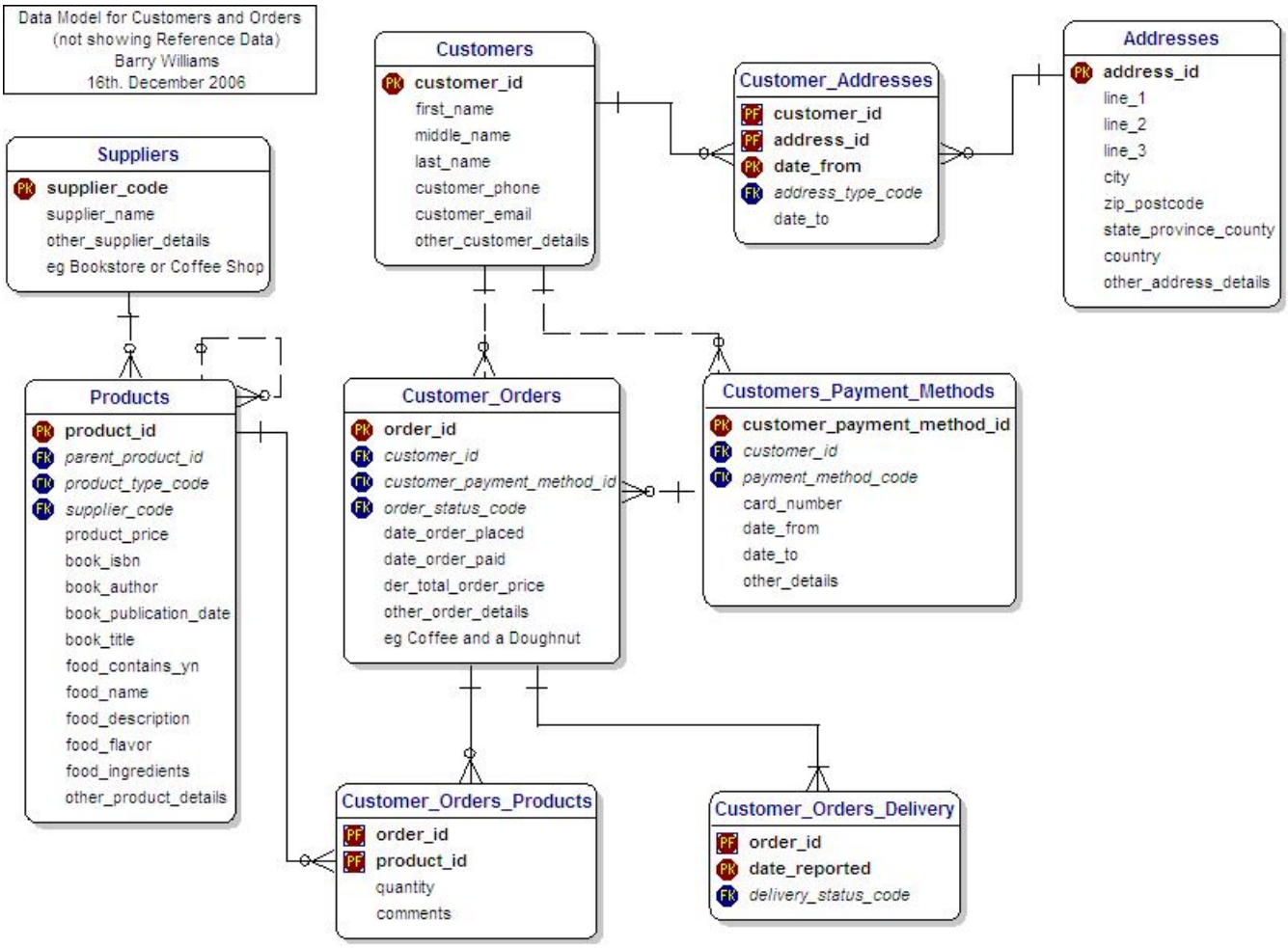
7.3 Master-Detail

Master-detail is a very common design and is described on this Wikipedia page:

- http://en.wikipedia.org/wiki/Master-detail

A typical example master-detail is customers and orders:

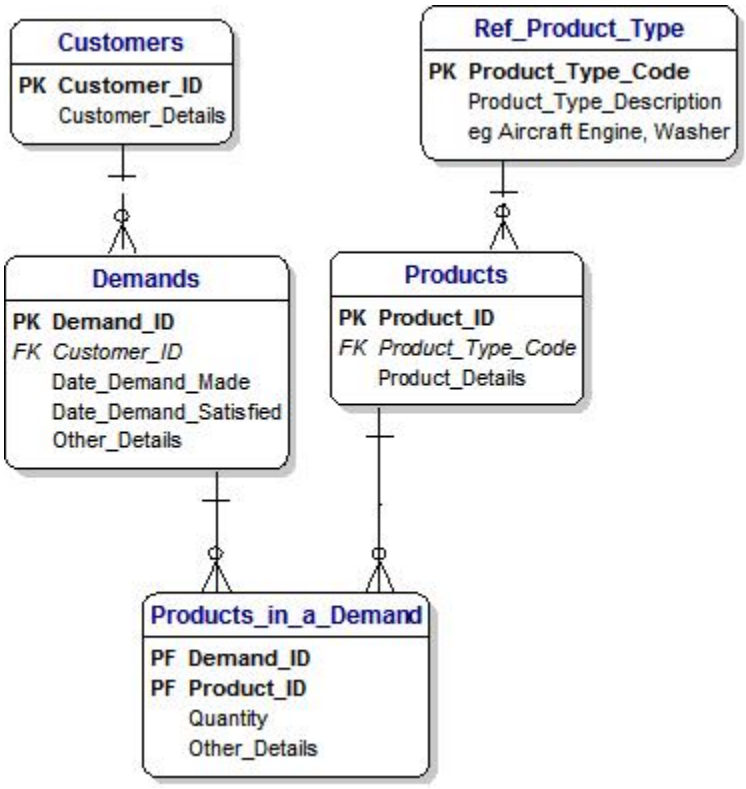
- http://www.databaseanswers.org/data_models/customers_and_orders/index.htm



7.4 Data Warehouses

7.6.1 Design of an ERD

This data model is an **Entity-Relationship-Diagram** (ERD) for customers and demands:



We could describe it in these terms:

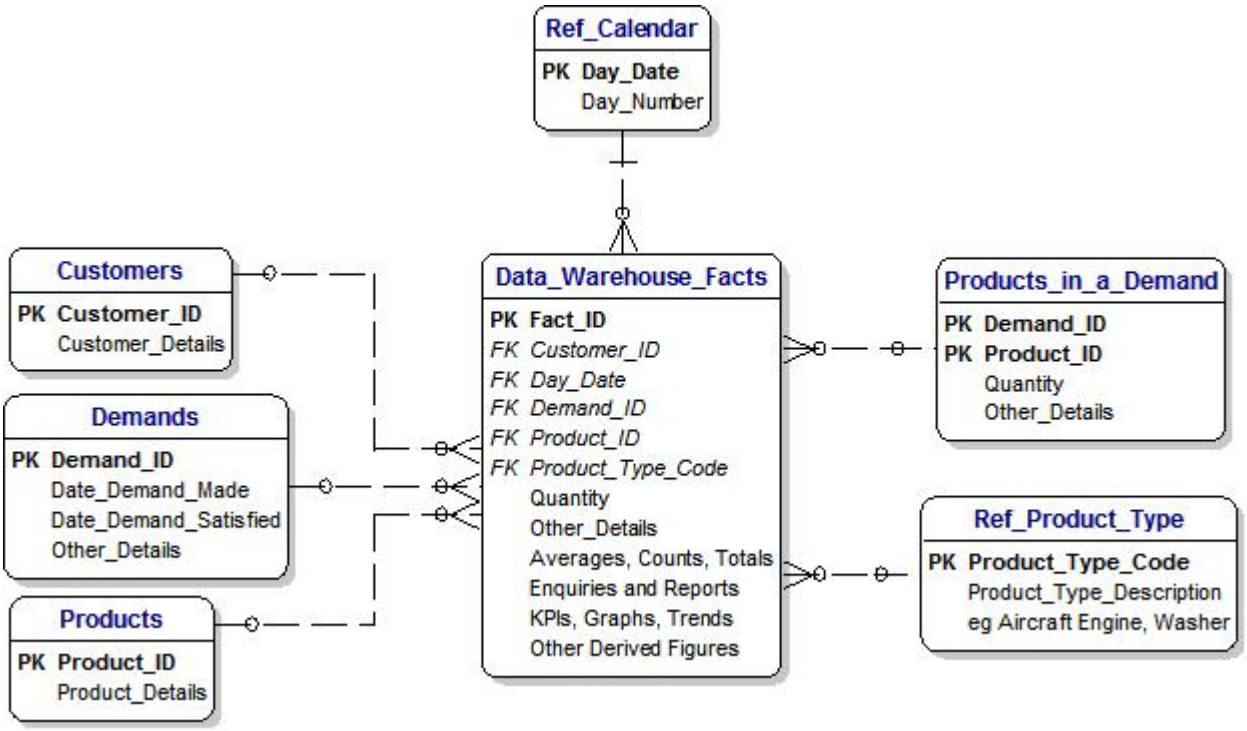
"Customers place demands for products of different types."

7.6.2 Design of a Data Warehouse

For our purposes, we can consider a data warehouse to be the same as a data mart.

In our professional experience, we have designed data marts that had a specific scope and timescale and defined users. A data warehouse, on the other hand, simply puts all data into a big basket and says "Here it is, come and get it".

This data model shows the corresponding data warehouse for customers and demands:



7.6.3 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 (i.e. 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 inquiries need not involve any particular dimension. The one exception to this rule is that the relationship to the calendar is mandatory because an inquiry will always include a date. Of course, an inquiry 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?
- What was the most popular product in the first week of April?
- What was the average time to respond to demands for the most popular product?
- How many demands did we receive in May?

7.5 Applications

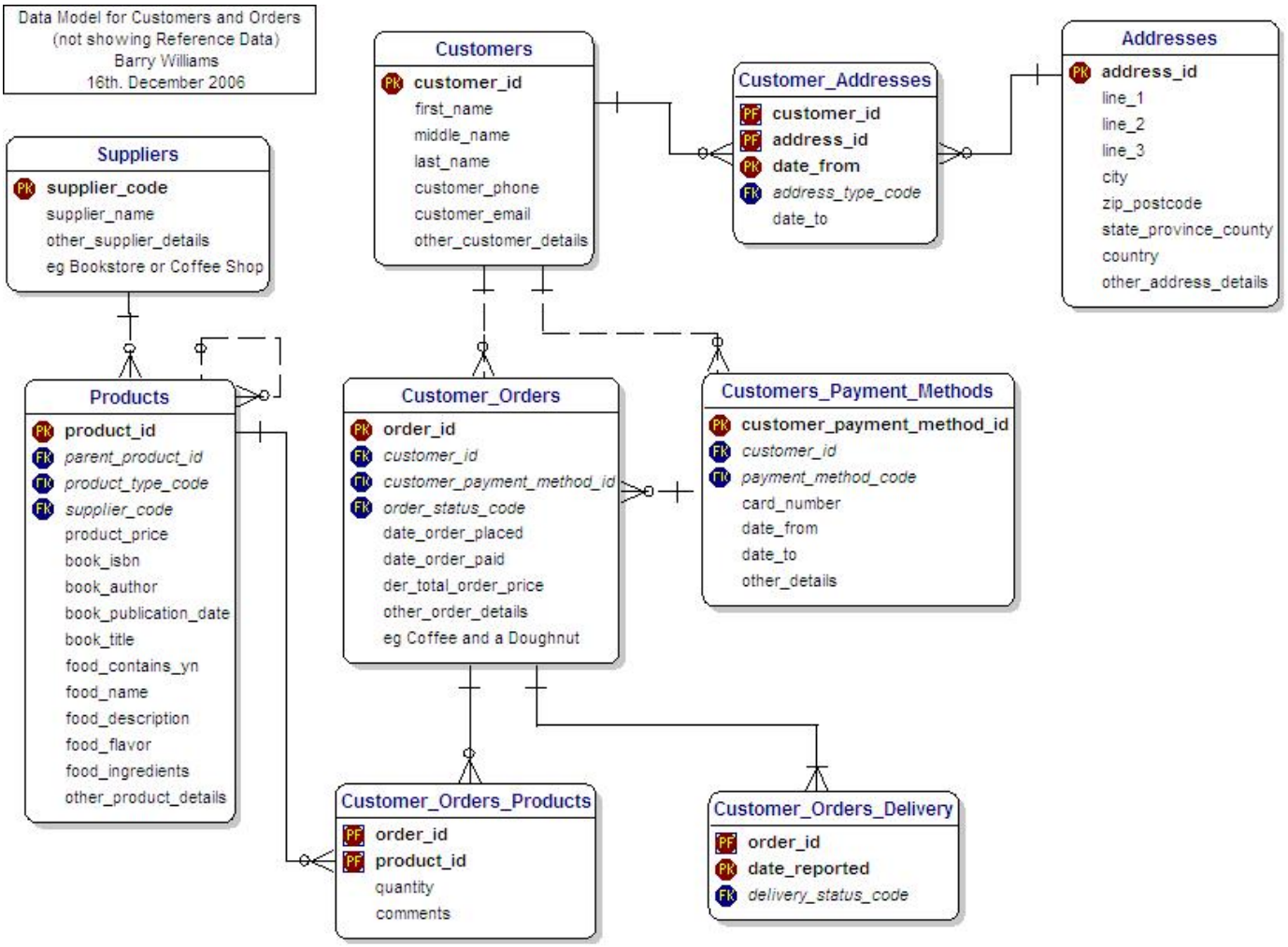
This shows some examples of simple **applications** that occur frequently.

7.7.1 Customers and Orders

This typical customers and orders data model represents a widespread kind of application:

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

It is an example of how a many-to-many relationship between customer orders and products is broken down into two one-to-many relationships. One of these is Customers-_Orders to Customer-Order-Products and the other is products to Customers_Order_Products.



7.7.2 Deliveries – Simple

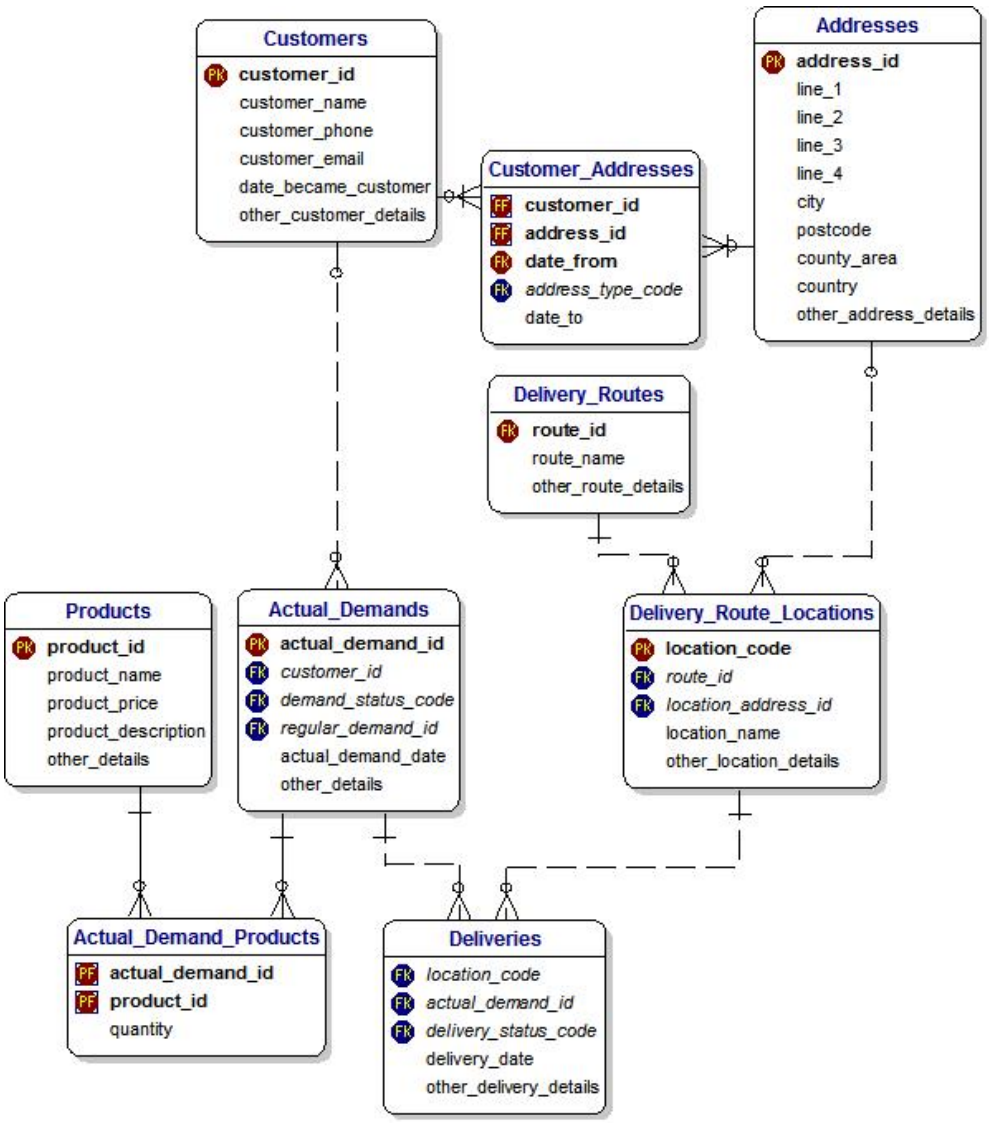
This data model is a **simple** design pattern that covers the activities of **one-off deliveries** to a designated address.

The process of reviewing a data model is to ask:

"How do I describe the business rules behind this model?"

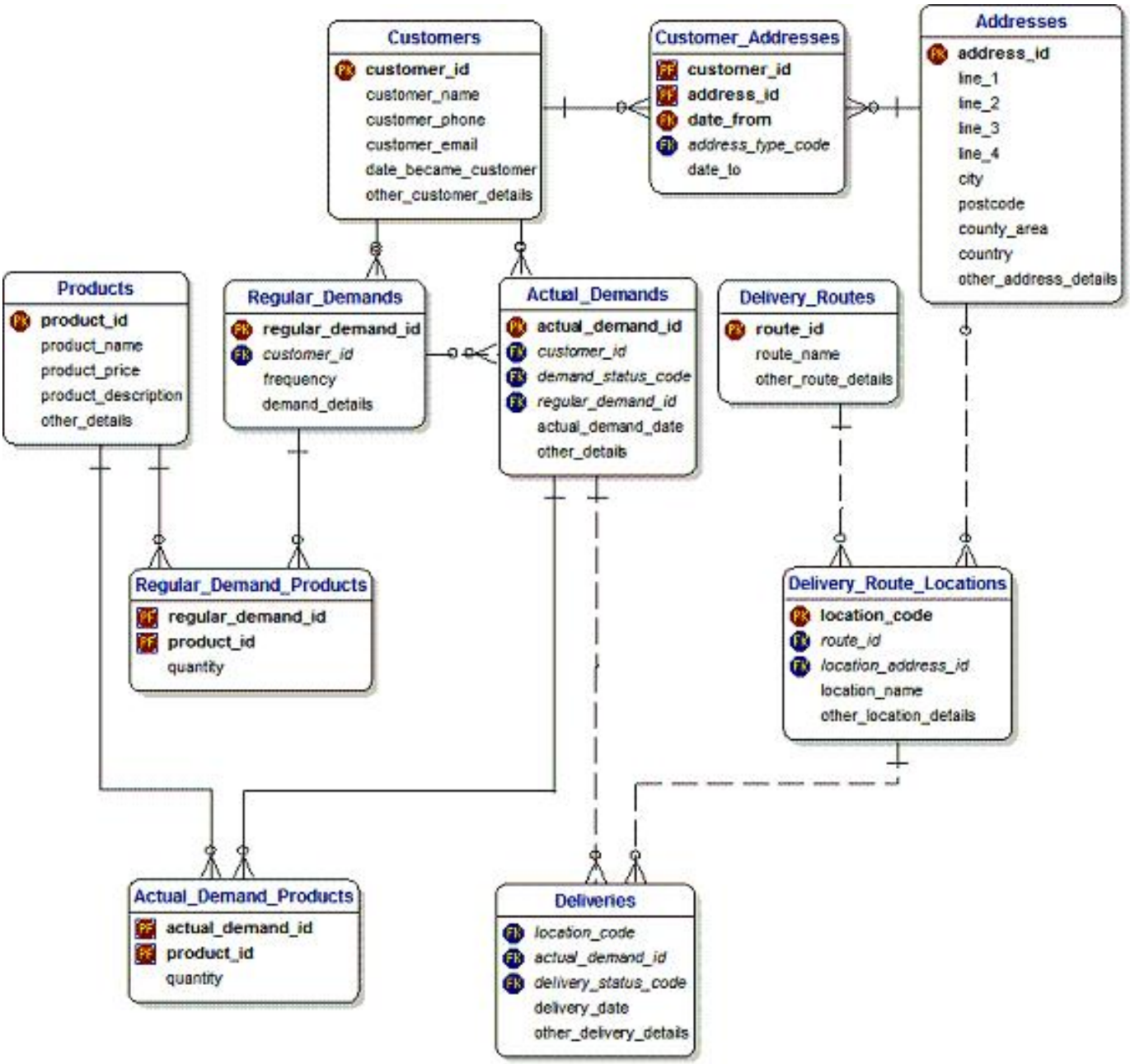
In this case, we could say:

"A customer can raise a demand for products to be delivered to a specified address"



7.7.3 Deliveries – Complex

This shows a more **complex** pattern that adds **regular demands.**

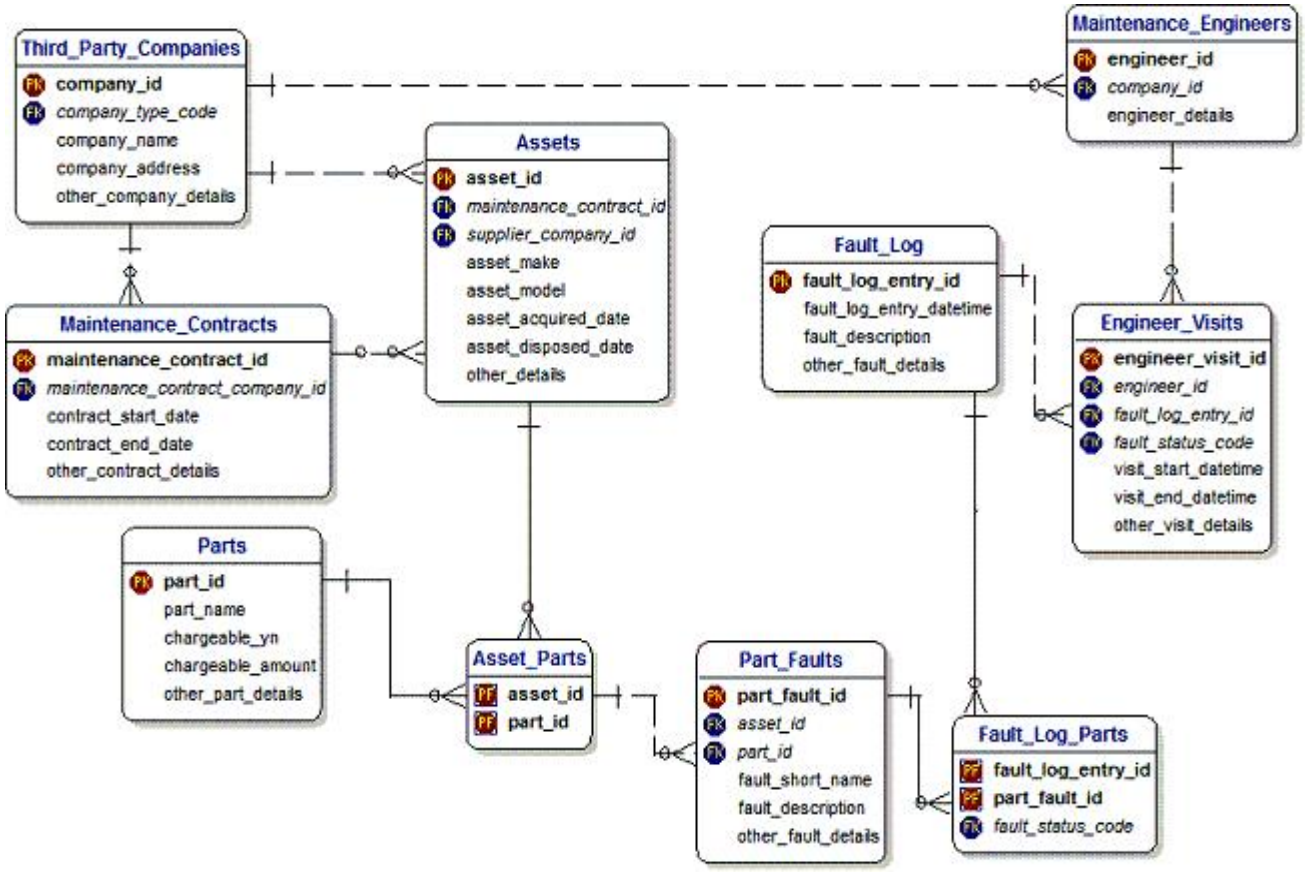


7.7.4 Maintenance of Equipment

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 that are recorded in a fault log.
They correct the faults and this is recorded in the fault log.



7.7.5 Top-Level and Subject Areas

Complex data models are common in large organizations. They can best be understood when they are broken down into a top-level model and lower-level subject areas.

Typical subject area models might include Finance, HR, Deliveries and Maintenance.

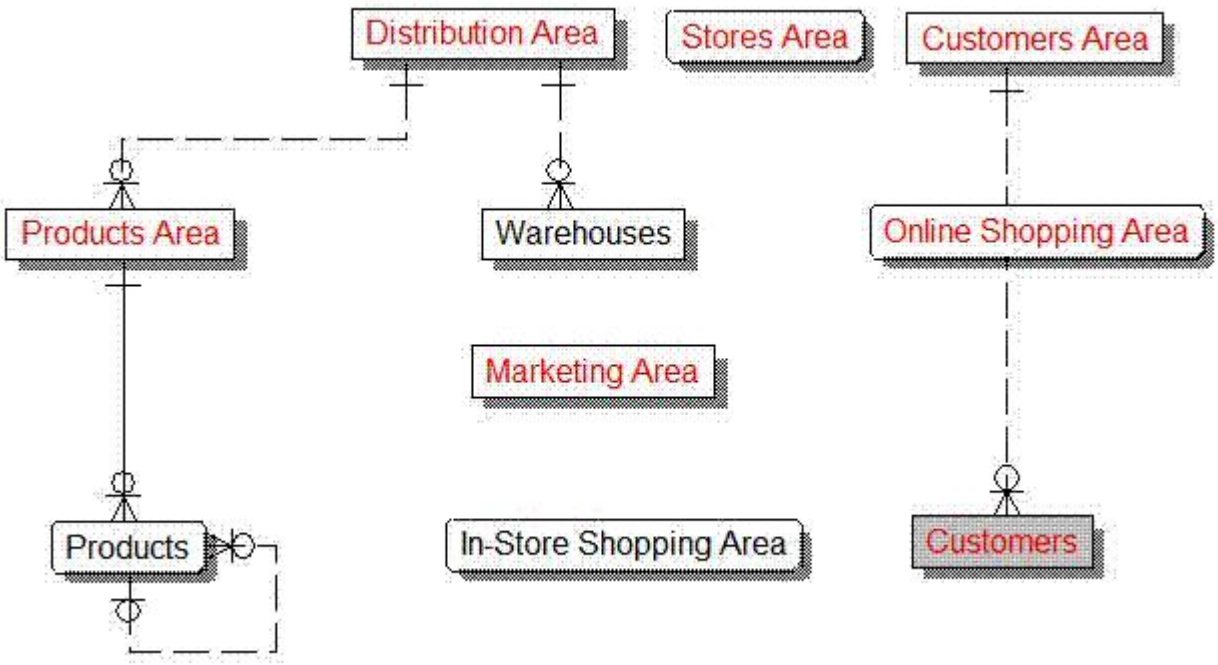
These are shown in earlier sections of this document.

Top-Level Model
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.

Here is an example for a retail organization that you can find on this page:

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



7.6 What have we learned?

This chapter has discussed design patterns that are generic solutions to problems that occur frequently. After we become familiar with these design patterns we can see them frequently when we look around us. Therefore, an understanding of these patterns can be a great help when we want to assemble a data model quickly and we can recognize that specific patterns are relevant.

8. 'Bang for the Buck' Data Models

8.1 Introduction

8.1.1 What is this?

This chapter discusses some examples of small, economical data models.

What they all have in common is that they offer a rich functionality with just a limited number of entities, which is what we mean by 'Bang for the Buck'.
The data models in this chapter are listed on this page on the Database Answers Web site:

- http://www.databaseanswers.org/data_models/bang_for_the_buck_models.htm

8.1.2 Why is it important?

Small, economical data models are important because they demonstrate the power of the underlying principles and concepts of Ted Codd's original Relational Theory. They illustrate how common situations occur frequently all around us. For example, situations involving people and events occur every day all over the world.

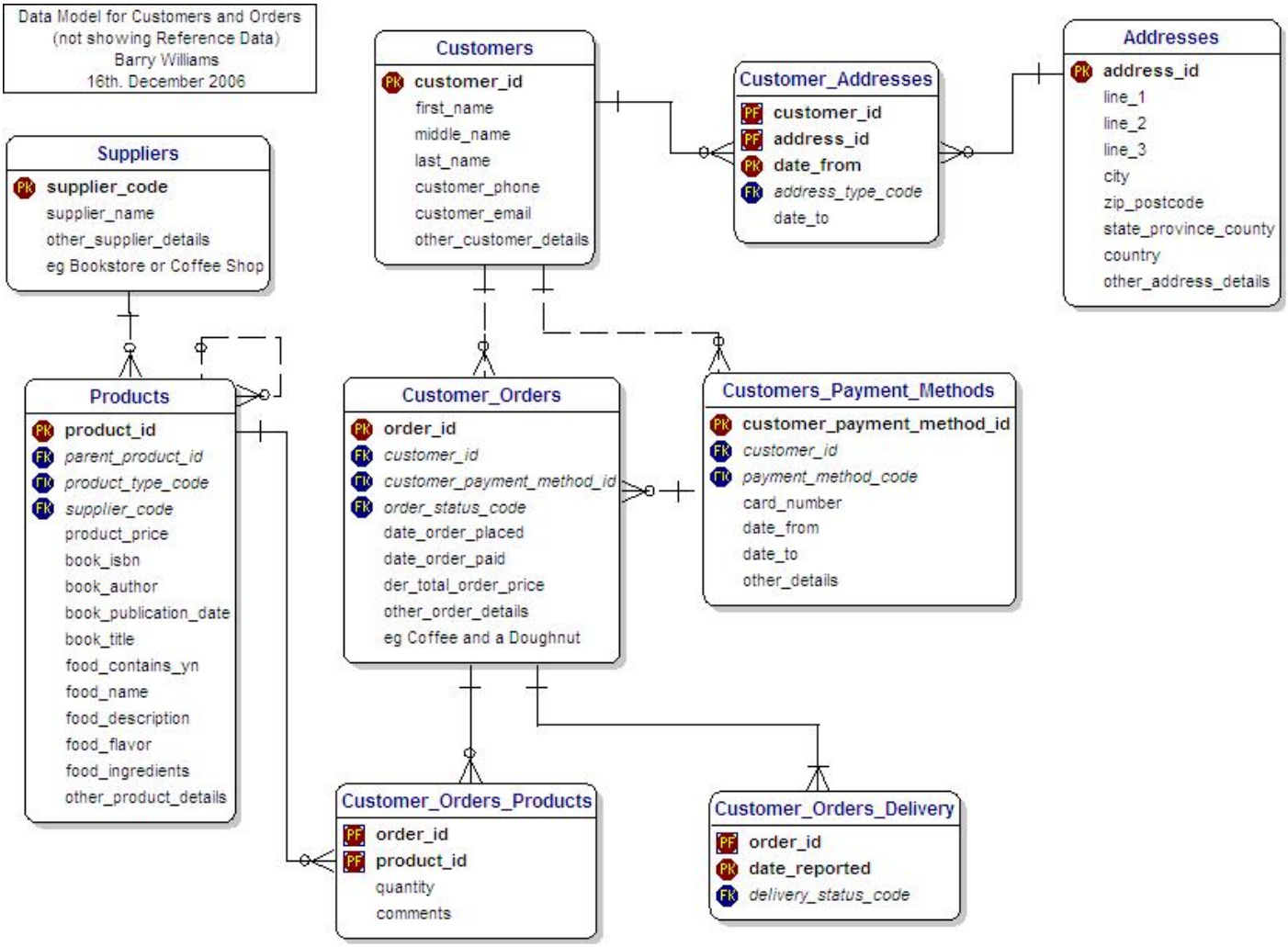
8.1.3 What Will I Learn?

You will learn how to identify and understand design patterns. This will make it very easy for you to interpret complex models.

8.2 Customers and Orders

This model appears on this page on the Database Answers Web site:

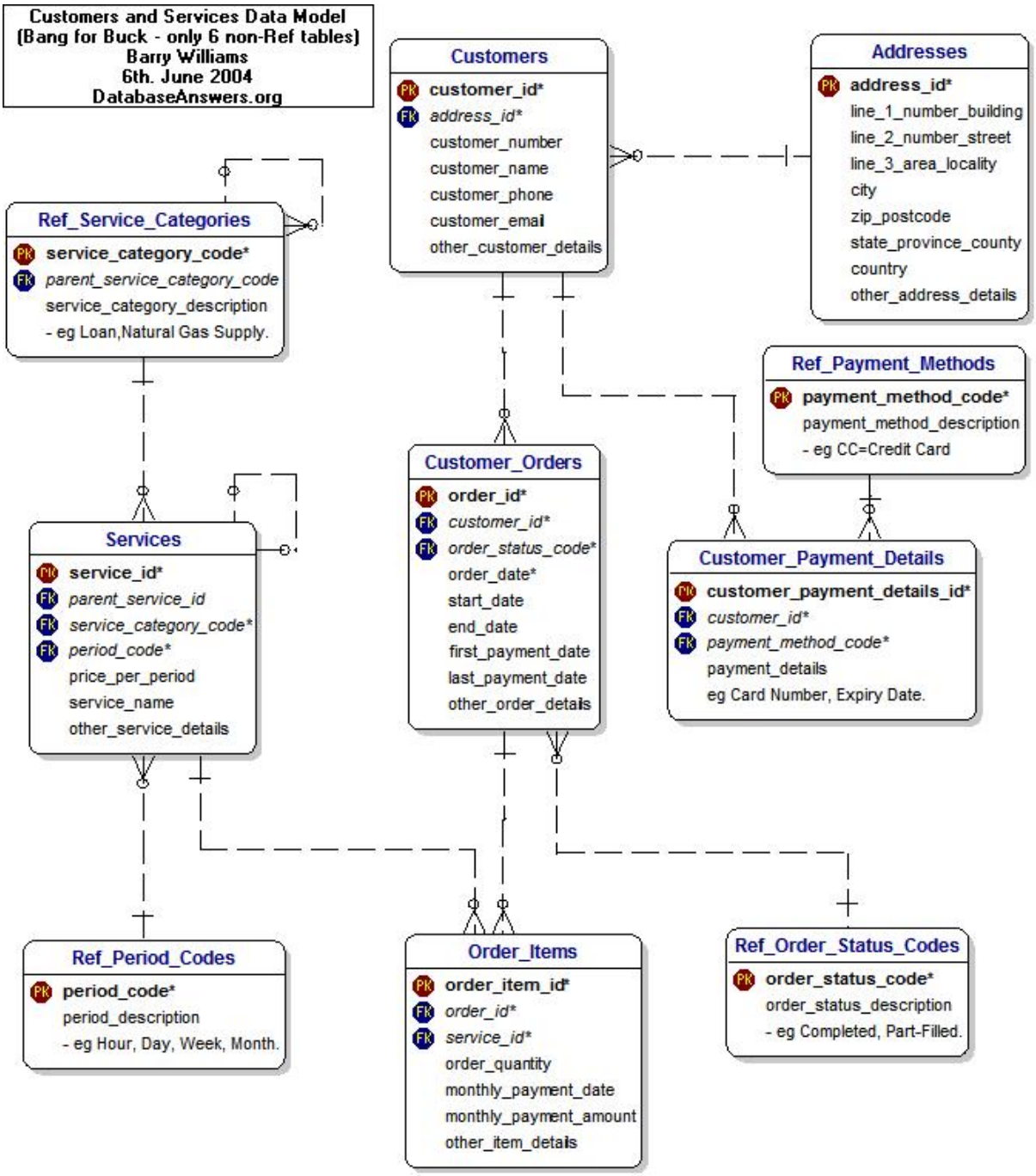
- http://www.databaseanswers.org/data_models/customers_and_orders/index.htm



8.3 Customers and Services

This model appears on this page on the Database Answers Web site:

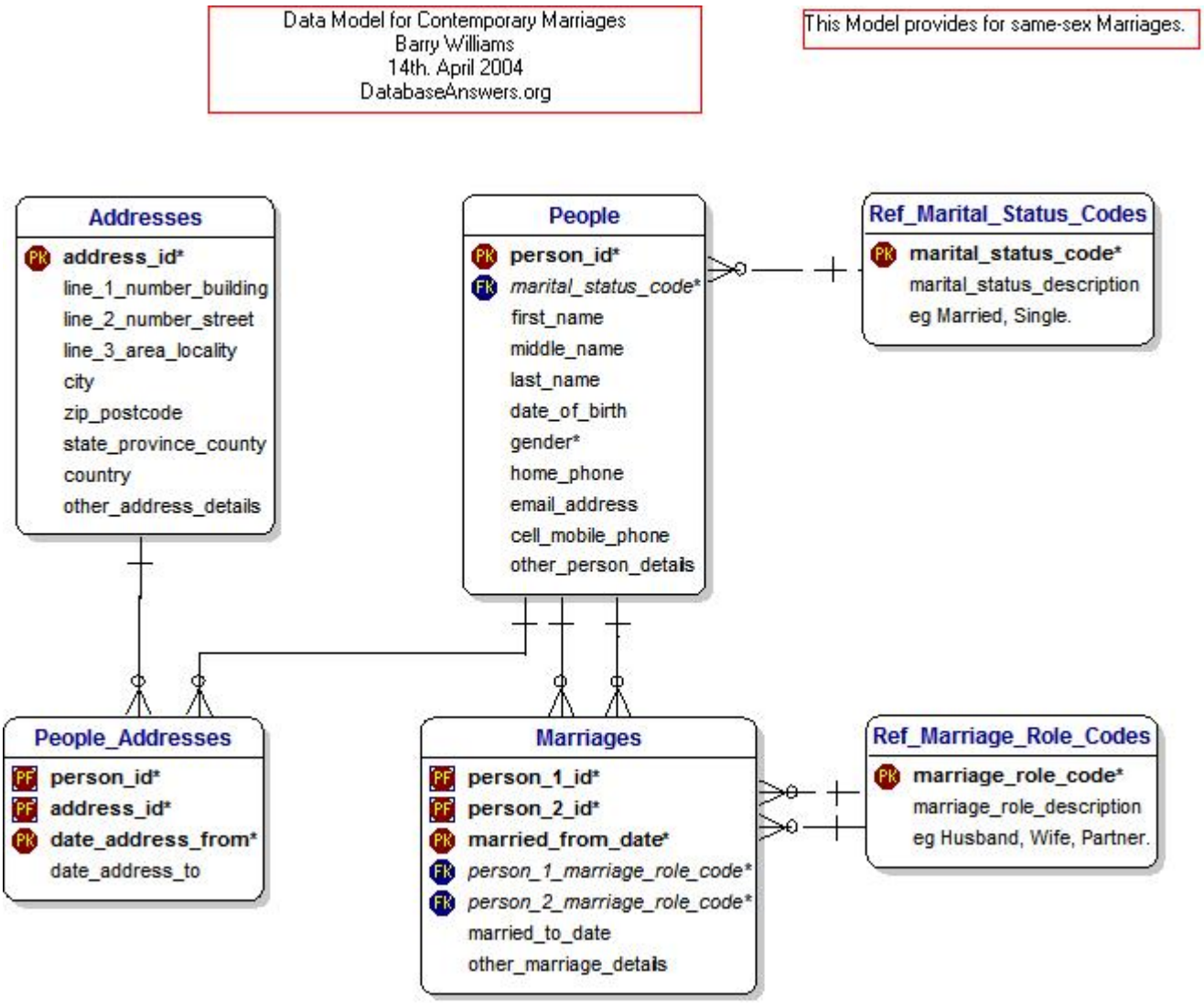
- http://www.databaseanswers.org/data_models/customers_and_services/index.htm



8.4 Marriages (Contemporary)

This model appears on this page on the Database Answers Web site:

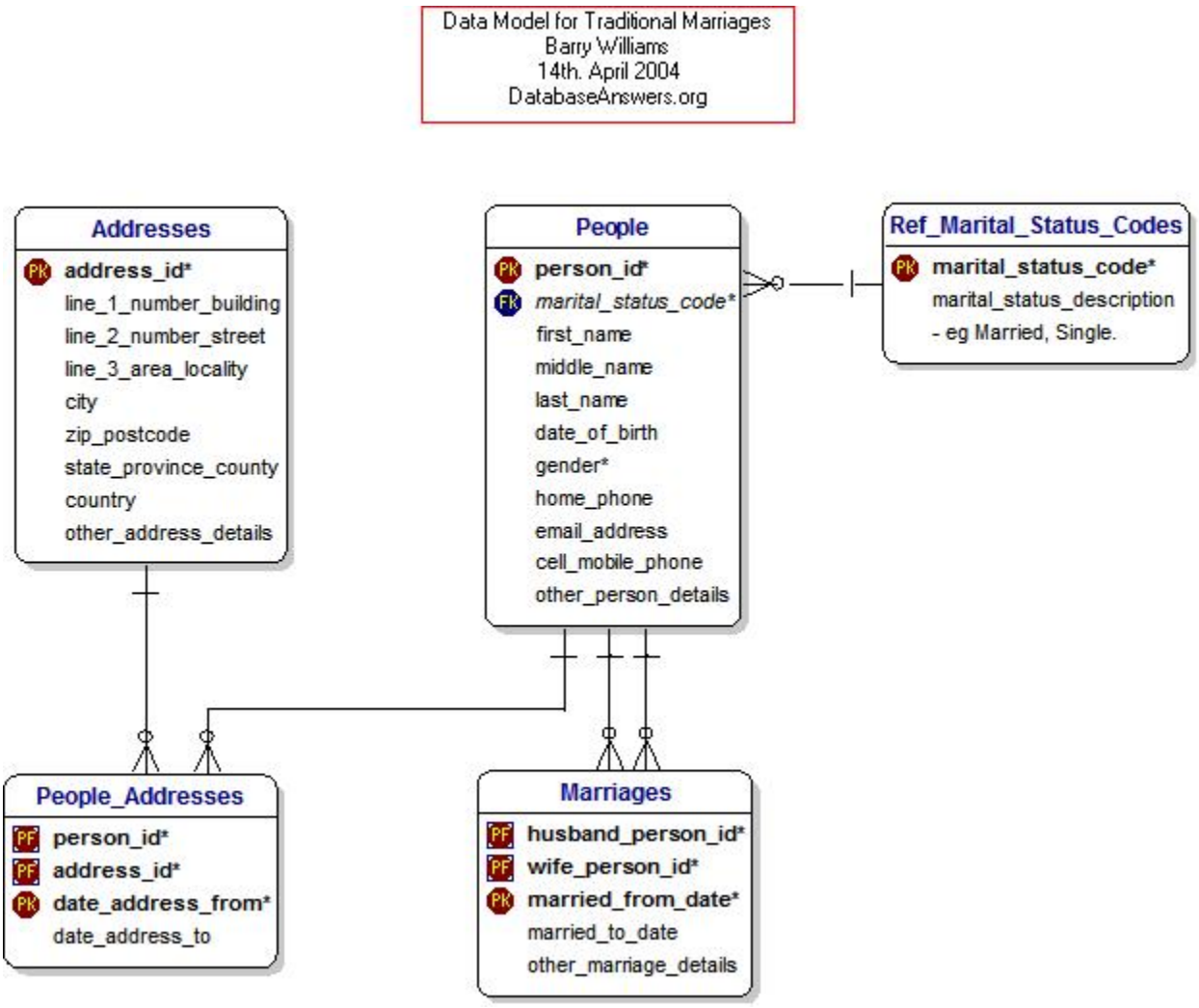
- http://www.databaseanswers.org/data_models/marriages_contemporary/index.htm



8.5 Marriages (Traditional)

This model appears on this page on the Database Answers Web site:

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



8.6 Organizations, People and Events

This model appears on this page on the Database Answers Web site:

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

and a generic foundation model:

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

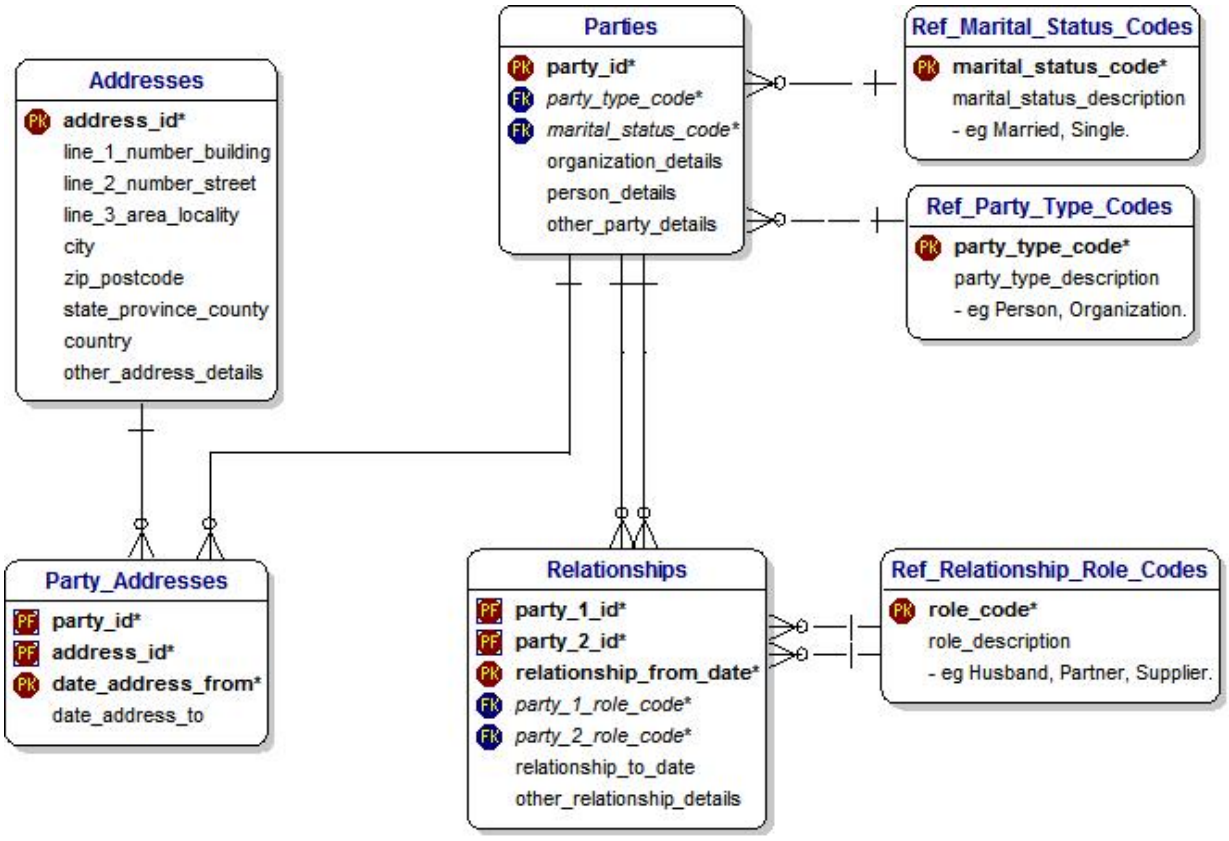
8.7 Partnerships and Relationships

Here is the page on the Database Answers Web site:

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

The concept of partnerships and relationships includes:

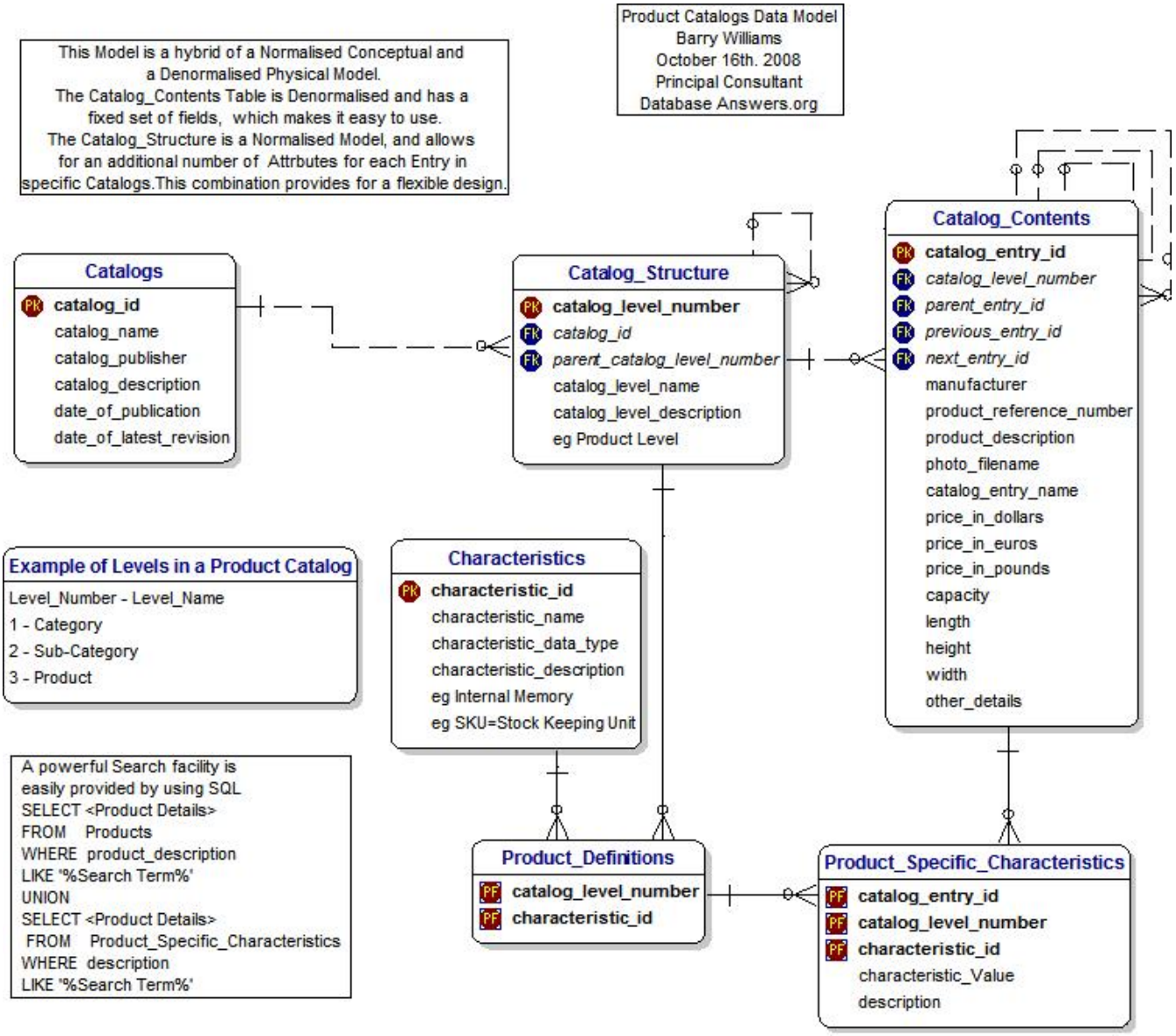
- Civil partnerships
- Employers and employees
- Friends and associates
- Marriages
- Participants in a law suit



8.8 Product Catalogues

Here is the page on the Database Answers Web site:

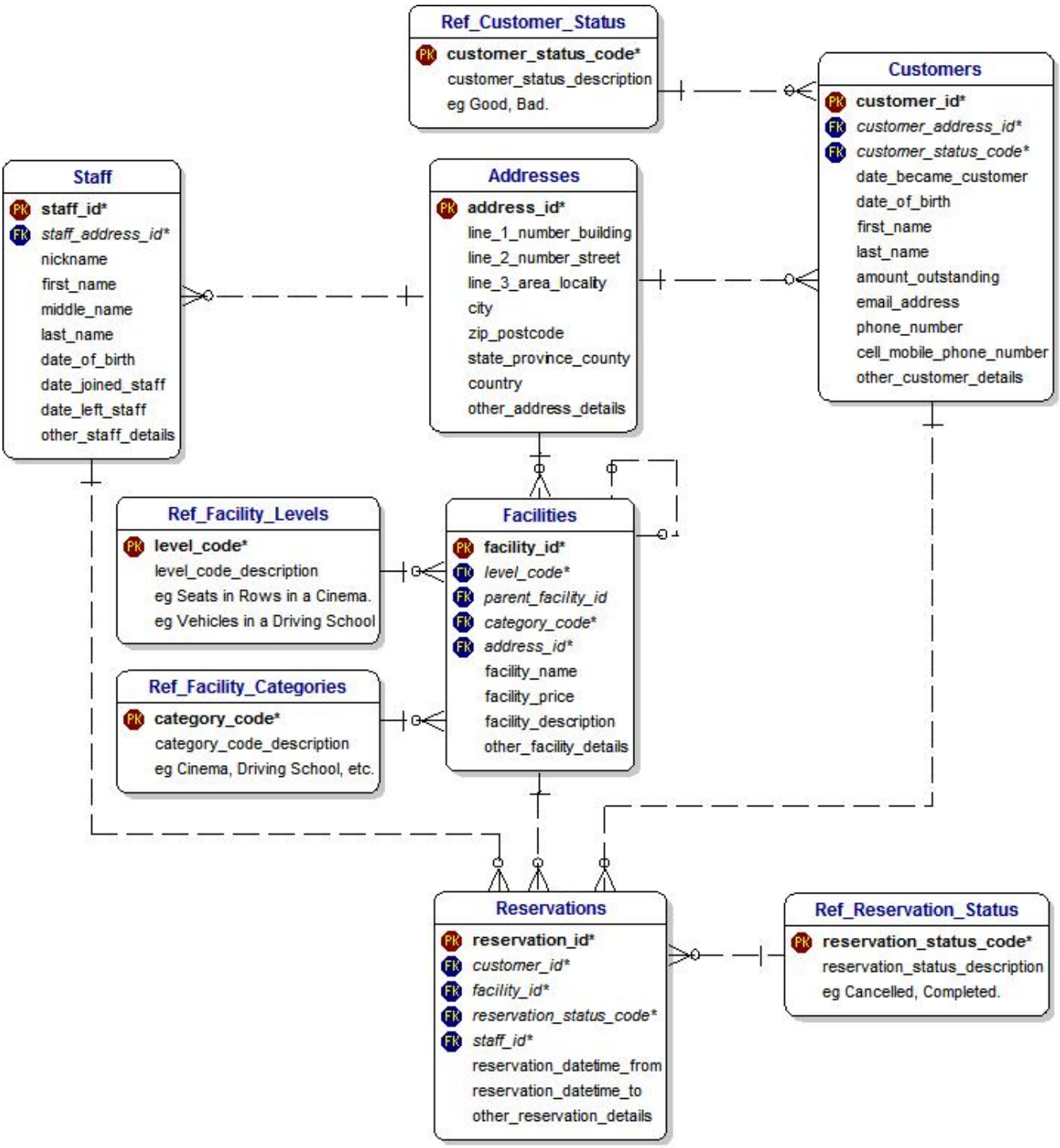
- http://www.databaseanswers.org/data_models/product_catalogs/index.htm



8.9 Reservations

Here is the page on the Database Answers Web site:

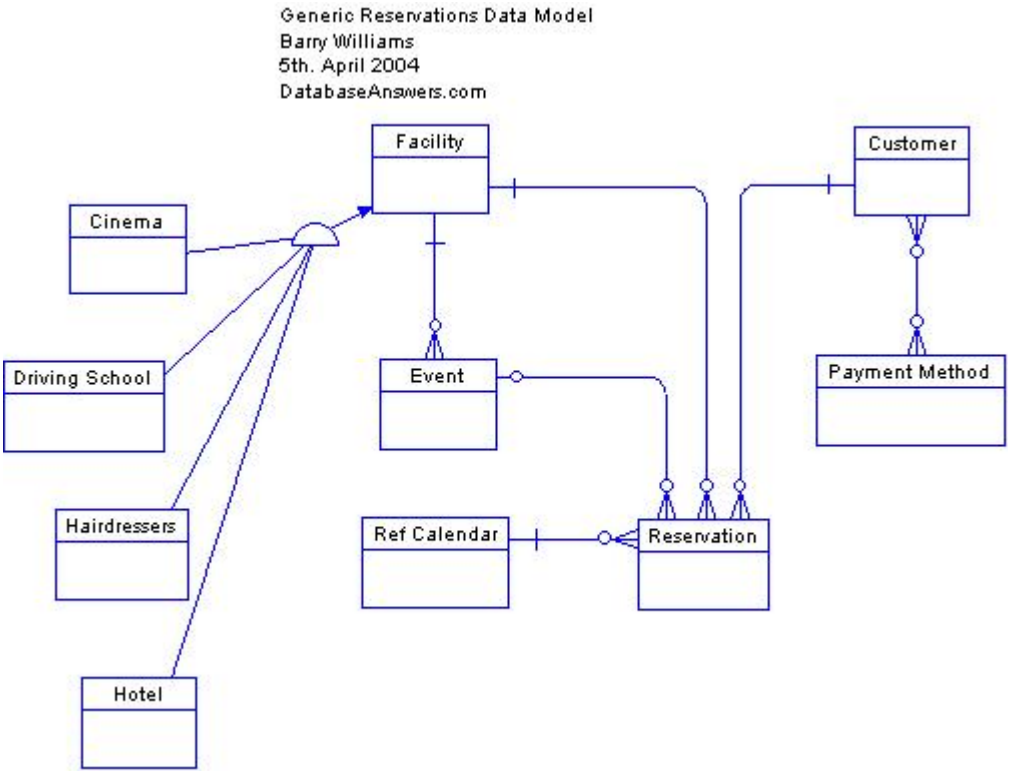
- http://www.databaseanswers.org/data_models/generic_reservations/index.htm



8.10 Reservations with Inheritance

Here is the page on the Database Answers Web site:

- http://www.databaseanswers.org/data_models/generic_reservations/generic_reservations_inheritance.htm



8.11 What have we learned?

In this chapter we have learned how it is possible to design data models that are simple and compact but can at the same time be very powerful in what they represent.

Inheritance and recursive relationships are two techniques that help us to achieve this.

9. Generic Data Models

9.1 Introduction

This chapter will discuss a number of different generic data models.

9.1.1 What is this?

Generic data models are generalized solutions to problems that occur many times and perhaps in slightly different forms.

When we look around us, we can find many examples of the same situation occurring repeatedly. A simple example is making a reservation or buying something, like a newspaper or a coffee.

9.1.2 Why is it important?

Generic data models are very powerful because they cover a wide area of applications. They expose the underlying structure of a specific area of application. This can be used to identify the common occurrences of design patterns and provide insight to a variety of solutions.

This chapter includes a selection taken from this list on the Database Answers Web site:

a. Circus (Events & Players)
b. Customers and Services
c. Document Management
d. Father of All Models
e. Generic Foundation
f. Generic Me and My Life
g. Organizations and People
h. Organizations, People and Transactions
i. Organizations, Members & Meetings
j. Patient Care
k. Reservations
l. Shrek 2 Movie (Events & Places)
m. Transport
n. User-Defined Hierarchies

9.1.3 What will I Learn?

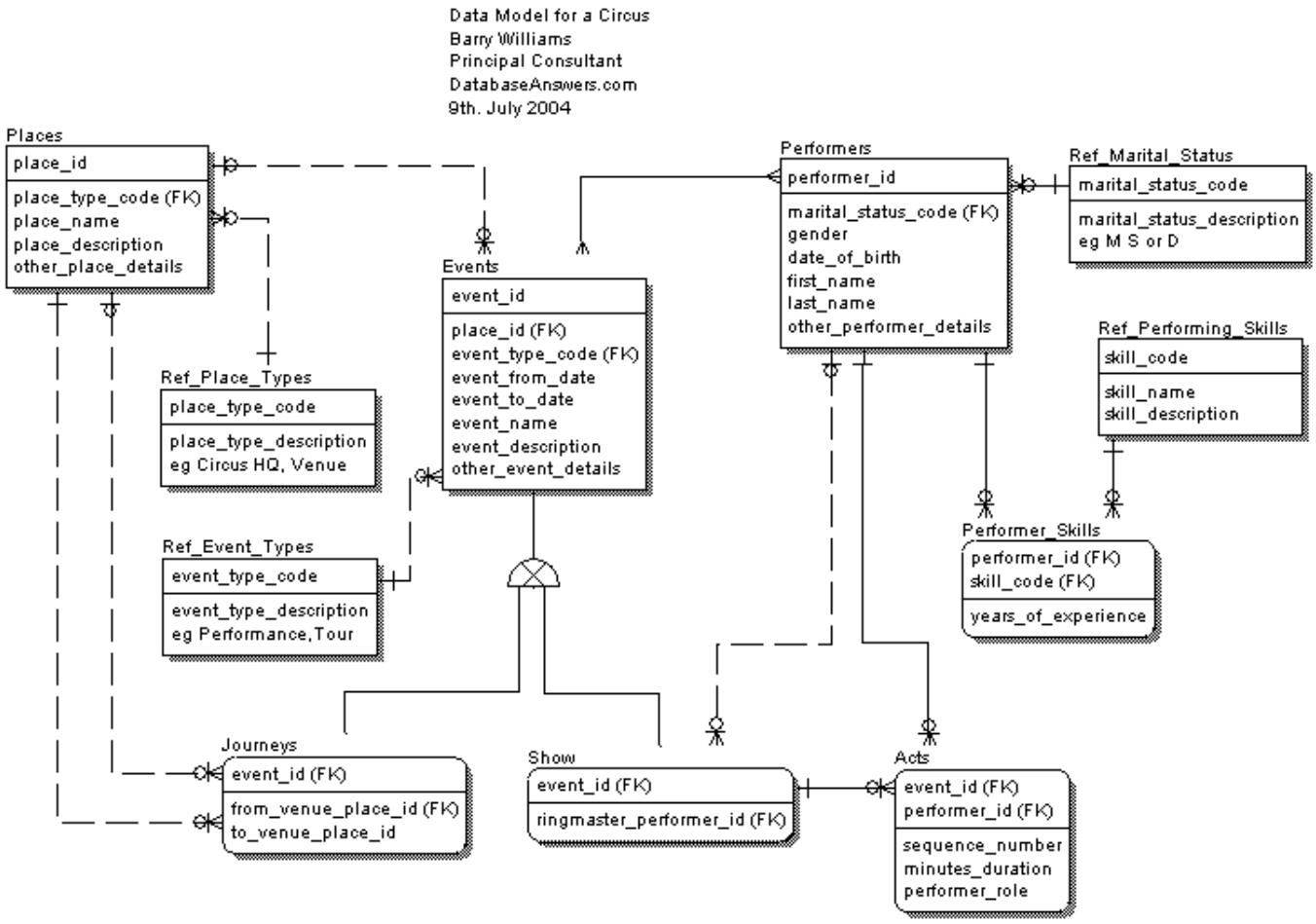
You will learn how to quickly recognize and understand situations similar to those you have already encountered and for which you have a solution in your head – in other words, a generic data model.

9.2 Circus

This model is a very interesting example of a generic model involving customers, events and people. In this case the people are performers. The model centers on events and performers. There is a many-to-many relationship between them.

Other relationship rules that we can see include:

- A performer can have one or many skills
- There is a defined list of skills
- An event takes place at a specified or unspecified place.
- An event is a super-type and has journeys and a show as sub-types.

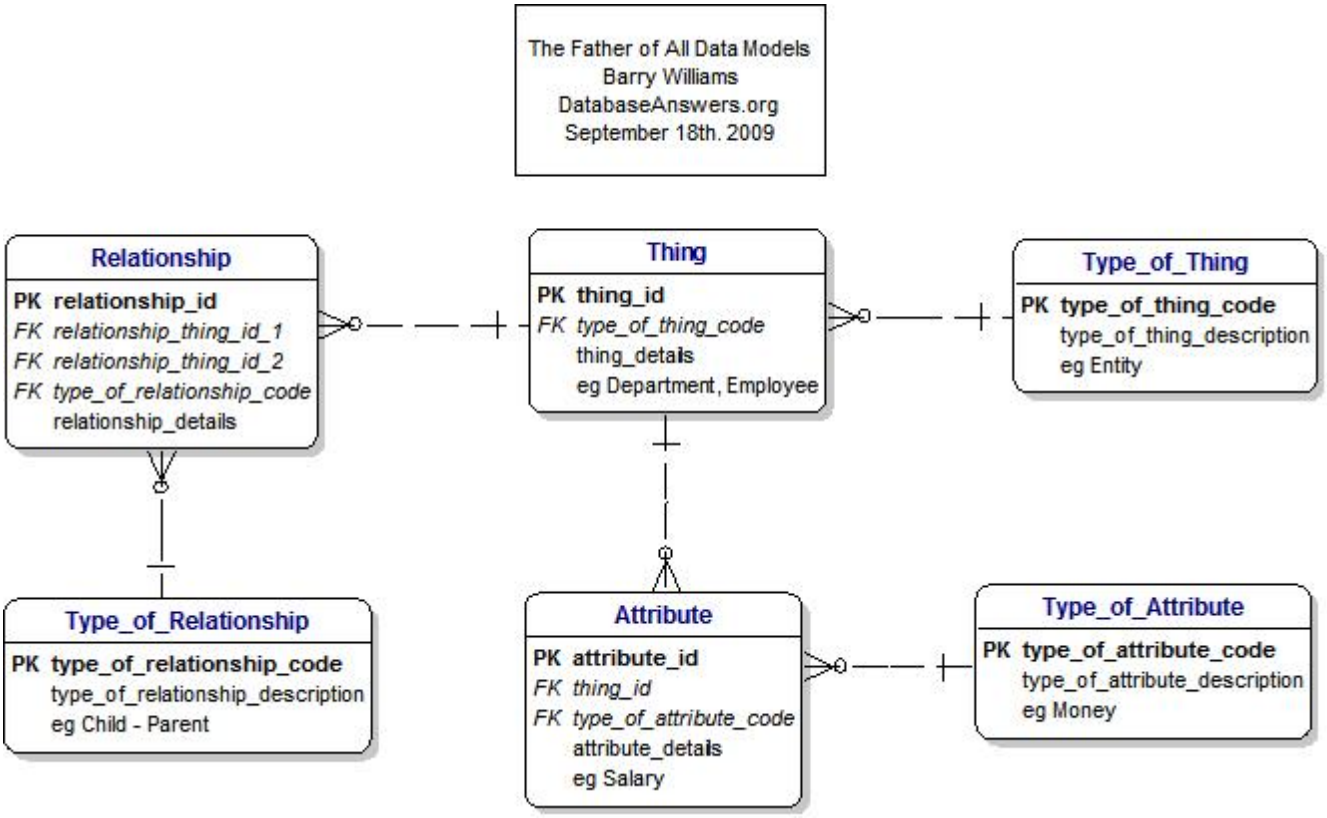


9.3 Father of all Data Models

This is a wonderful data model that shows a design for a foundation for all other data models that can possibly be created.

This could be considered to be a variation of the 'Entity-Attribute-Value' approach, about which Wikipedia has a useful entry:

- http://en.wikipedia.org/wiki/Entity-attribute-value_model

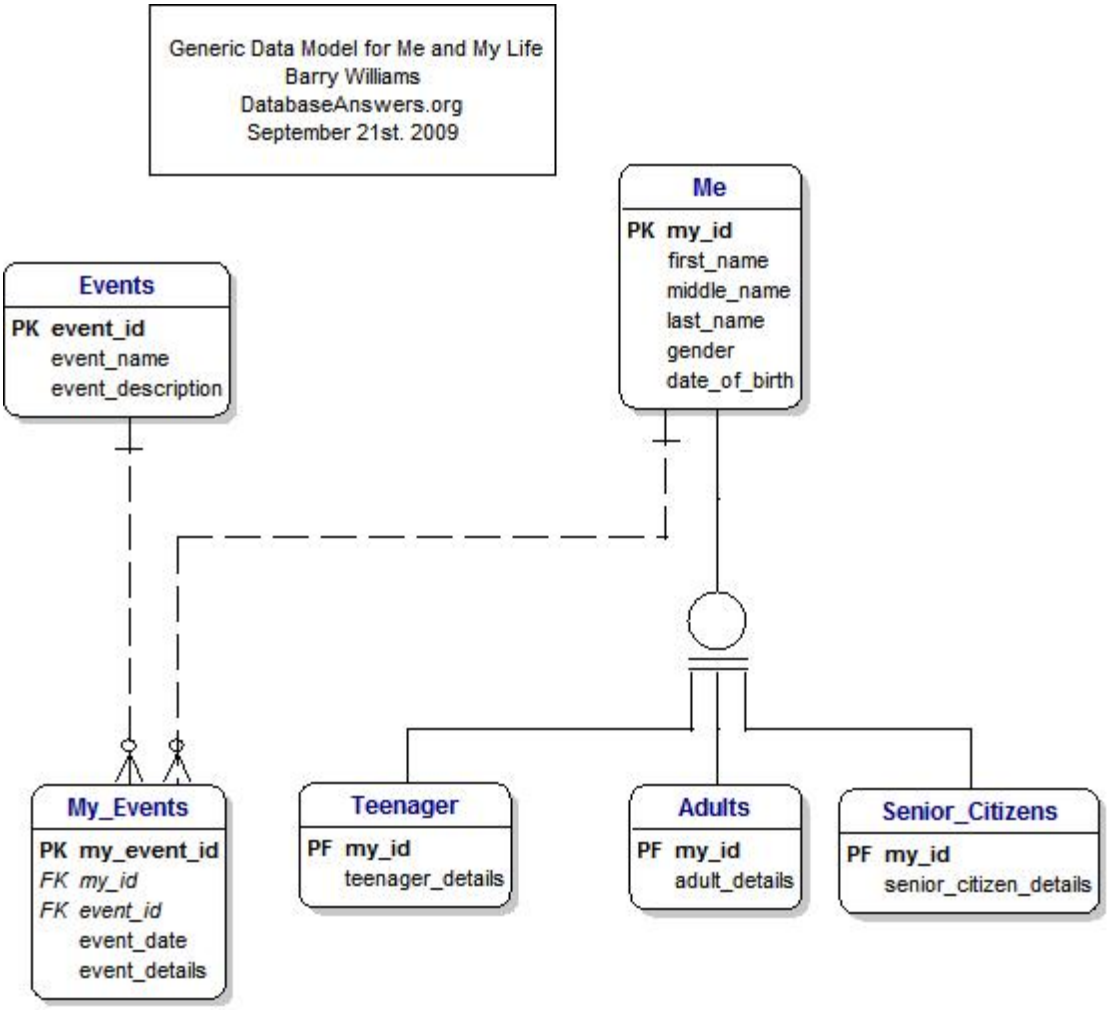


9.4 Me and Events in my Life

Generic and specific data models are very important because they illustrate how a number of specific data models can be incorporated into one generic model. In this case, the concept of a life-cycle from the cradle to the grave is used to generate scenarios:

1. Baby (Me and Mommy)
2. Teenager (Traffic Cops and Tickets
3. Student (Behavior Monitoring)
4. Adult (Partnerships and Relationships)
5. Adult (Golf Club Tournament)
6. Senior Citizen (Health Centers)

Here is the related Web page:
http://www.databaseanswers.org/data_models/generic_and_specific_models/index.htm



9.5 Retail Customers

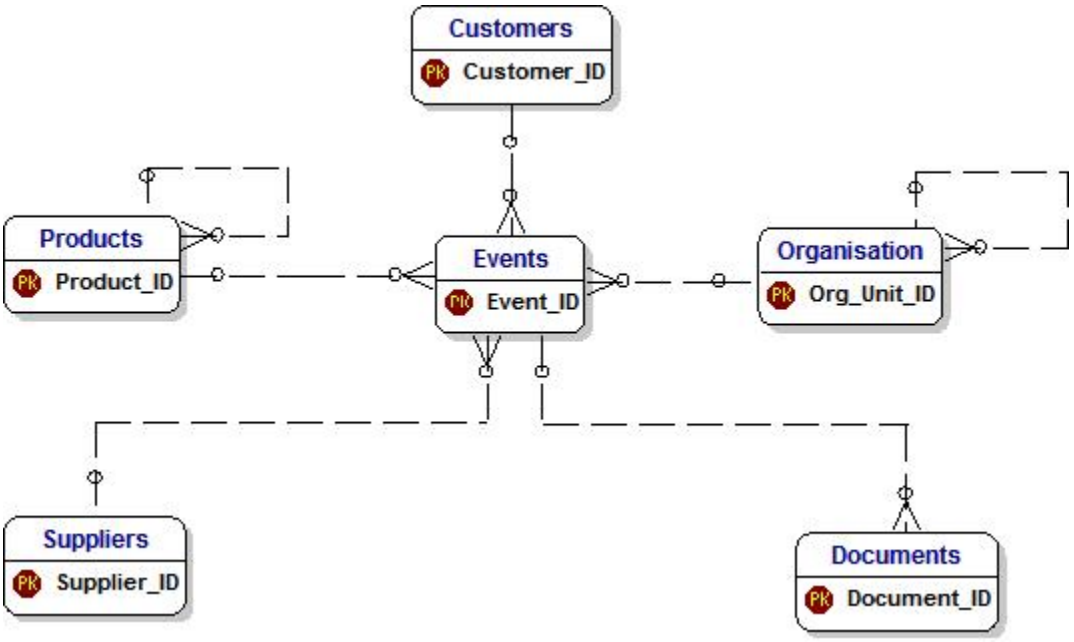
This section provides a very powerful and broad set of models that can be used as a foundation for a wide range of generic applications.

It consists of a top-level **overview** data model, with subject area models for the entities that appear in the top-level model.

9.7.1 Top-Level Model

This is the **top-level model** and is shown on this page on the Database Answers Web site:

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



The following sections contain a data model for each of the entities in the top-level model shown above.

You will see that the subject area models all follow a similar pattern:

- Generic entities
- ID fields for primary keys
- Codes for reference data

Following this simple pattern allows us to extend each model and still maintain a basic common architecture.

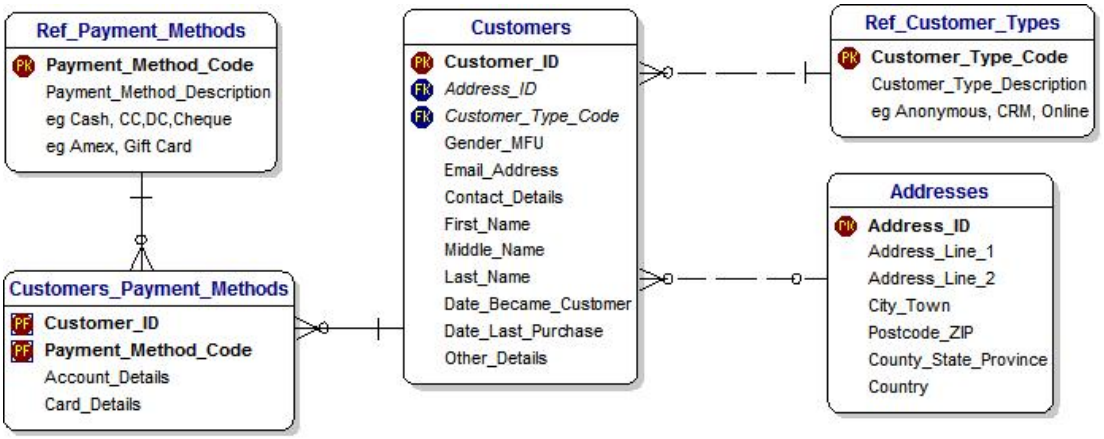
Here are the subject area models, listed alphabetically:

9.7.2 Customers

You can see that the dominant entity is **customers**, as you would expect. Customers have only one address so there is a foreign key for the address ID in the customers entity.

Each customer can have multiple payment methods, such as cash, check and so on.

This gives us a many-to-many relationship between customers and payment methods and we resolve this with an **associative table** called Customer Payment Methods.

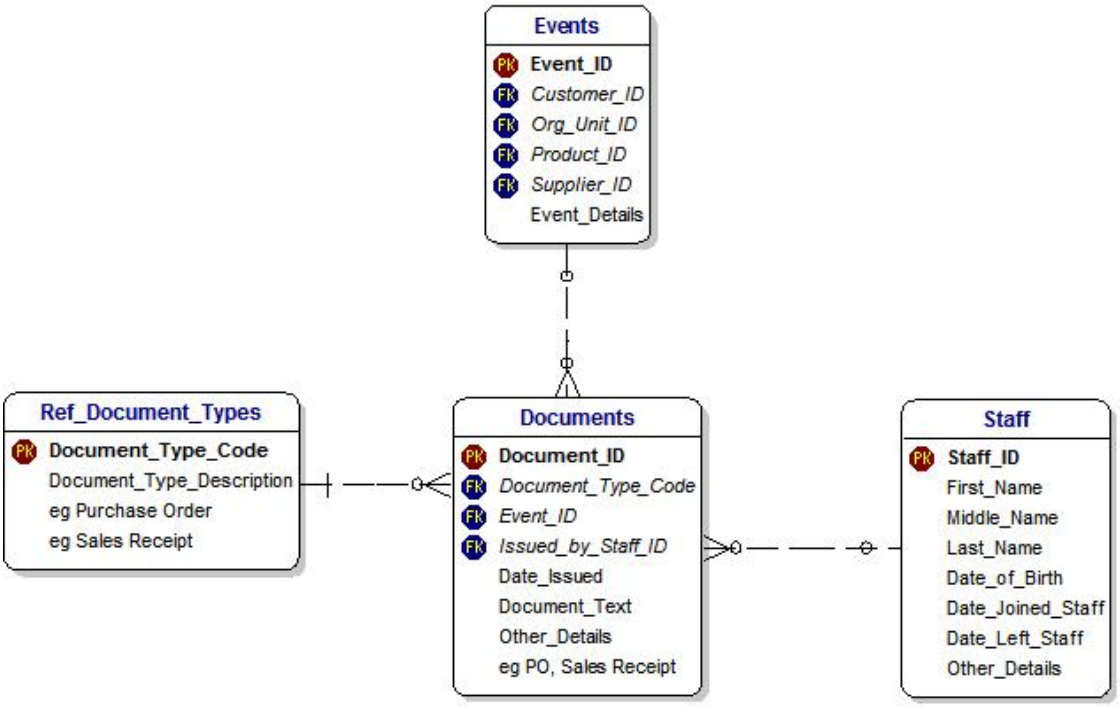


9.7.3 Documents

You can see that the dominant entity is **documents**, as you would expect. This model assumes that each event creates at most one document, and maybe not any.

It also provides for a member of staff to optionally issue a document.

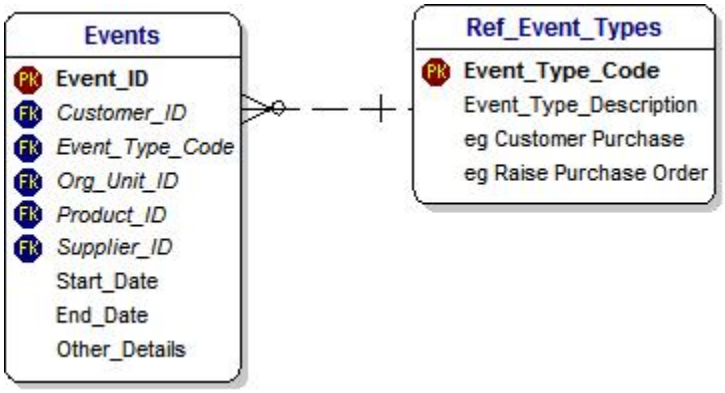
Documents can be a predefined number of different types, such as purchase orders and sales receipts.



9.7.4 Events

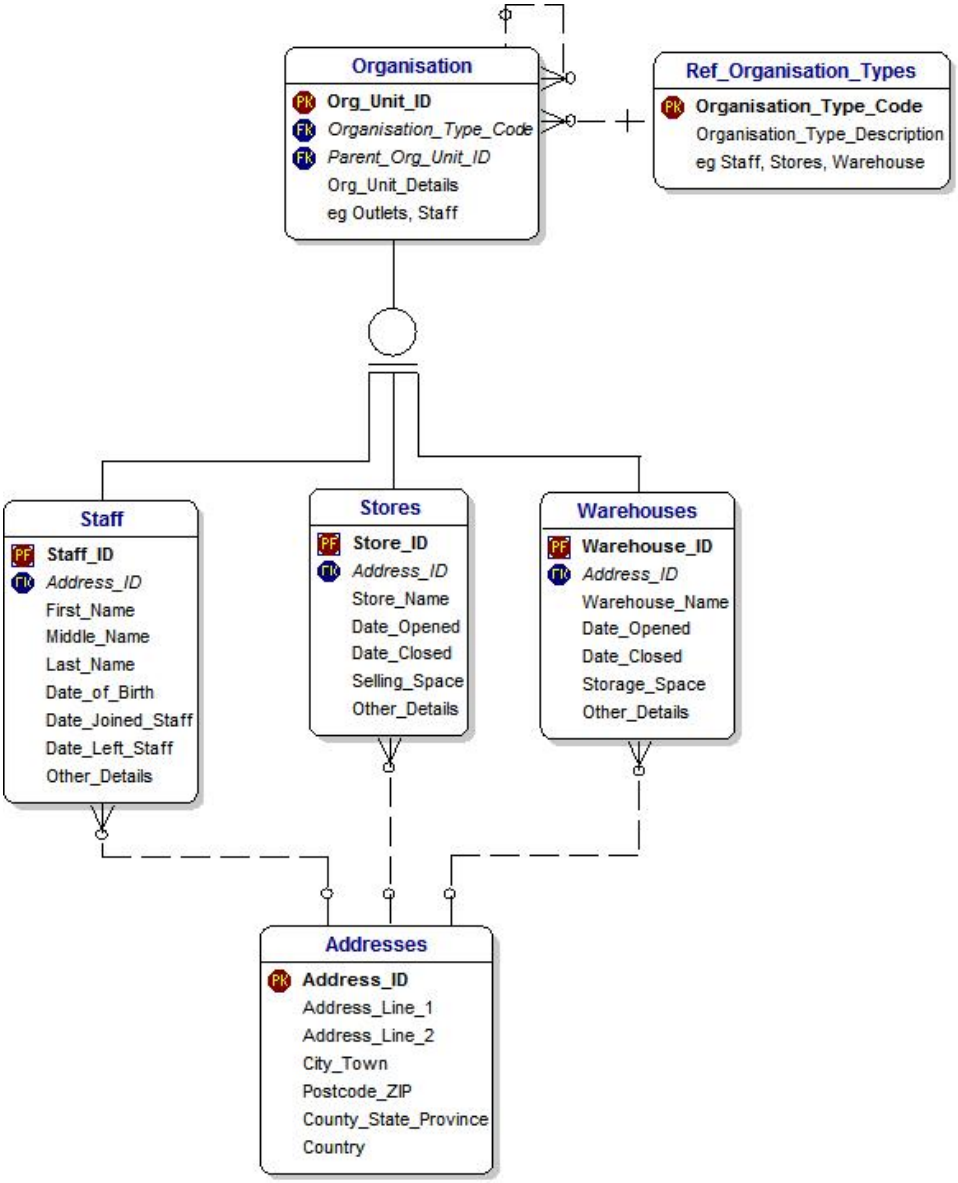
You can see that the dominant entity is **events**, as you would expect. This model is very simple because we have not repeated the entities that are shown in the top-level model. Therefore, the only additional entity that we show is for event types.

Typical events would be customer purchases and raising purchase orders.



9.7.5 Organization

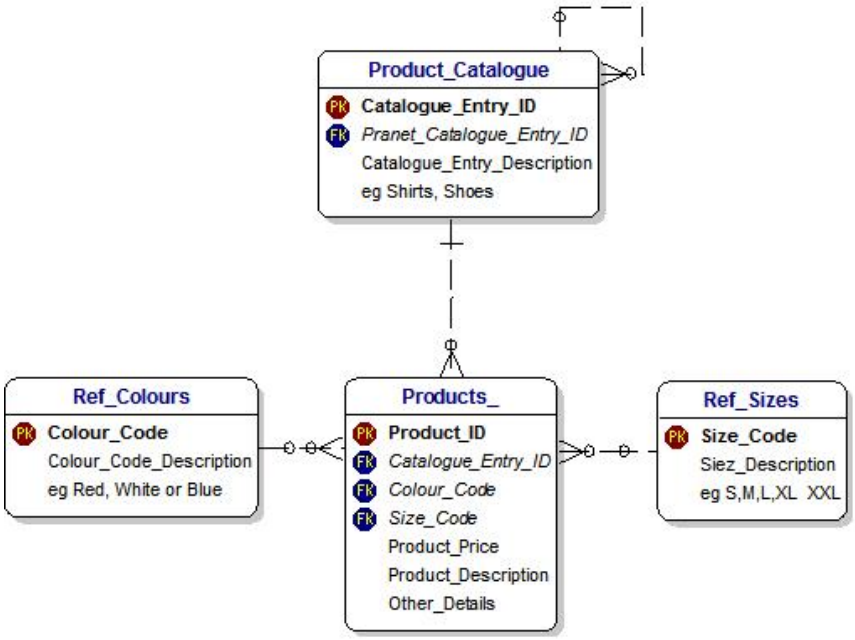
First, we see a familiar **pattern of entity** and **entity data types**. In this case, of course, it is organizations and organization types. Then we introduce the concept of inheritance because staff, stores and warehouses are all different types of organization.



9.7.6 Products

You can see that the dominant entity is **products**, as you would expect. It is interesting to see the rabbit ear or recursive relationship on the product catalogue entity. This is a very powerful technique, which says "An entry in the product catalogue can be related to a parent entity".

By using this simple technique, we are able to implement a hierarchy.



9.7.7 Suppliers

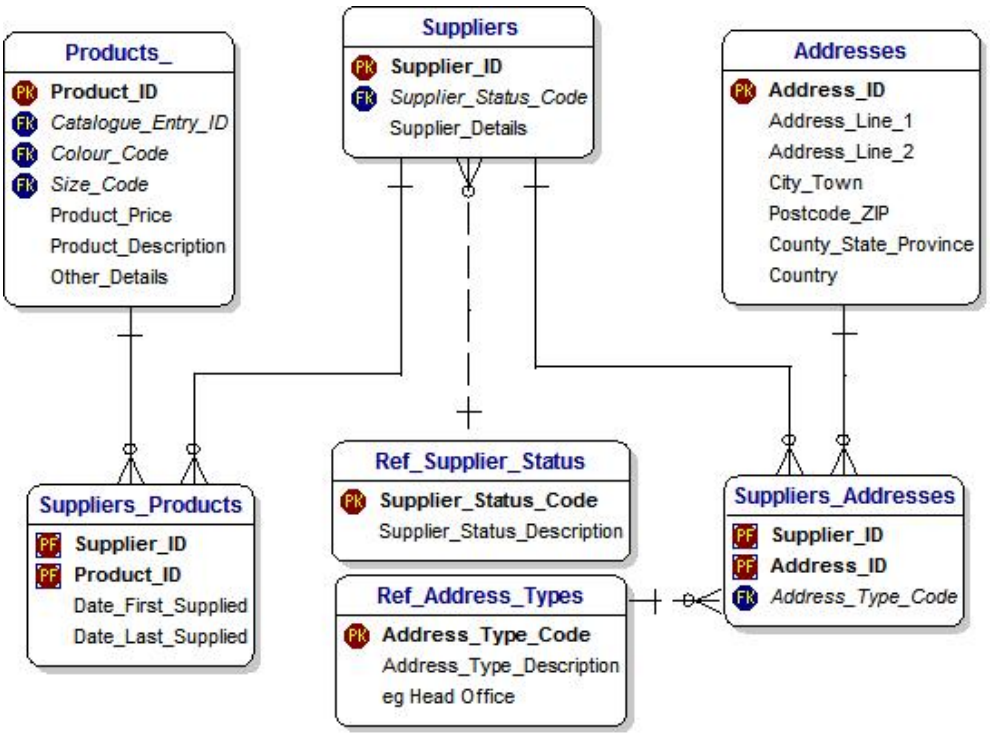
You can see that the dominant entity is **suppliers**, as you would expect. We can see a familiar pattern with a suppliers entity and associated reference data. We can also see two many-to-many relationships broken down into two one-to-many relationships:

- A supplier can have many addresses, such as billing, warehouses, regional offices, head office and so on.
- Many suppliers can also share the same address.

There we show this many-to-many broken down with an associate entity that we call suppliers address. For each record, we have an address type code field that tells us what type of address this is for each supplier.

The same argument applies to suppliers and products.

We recommend this as an exercise for the student.

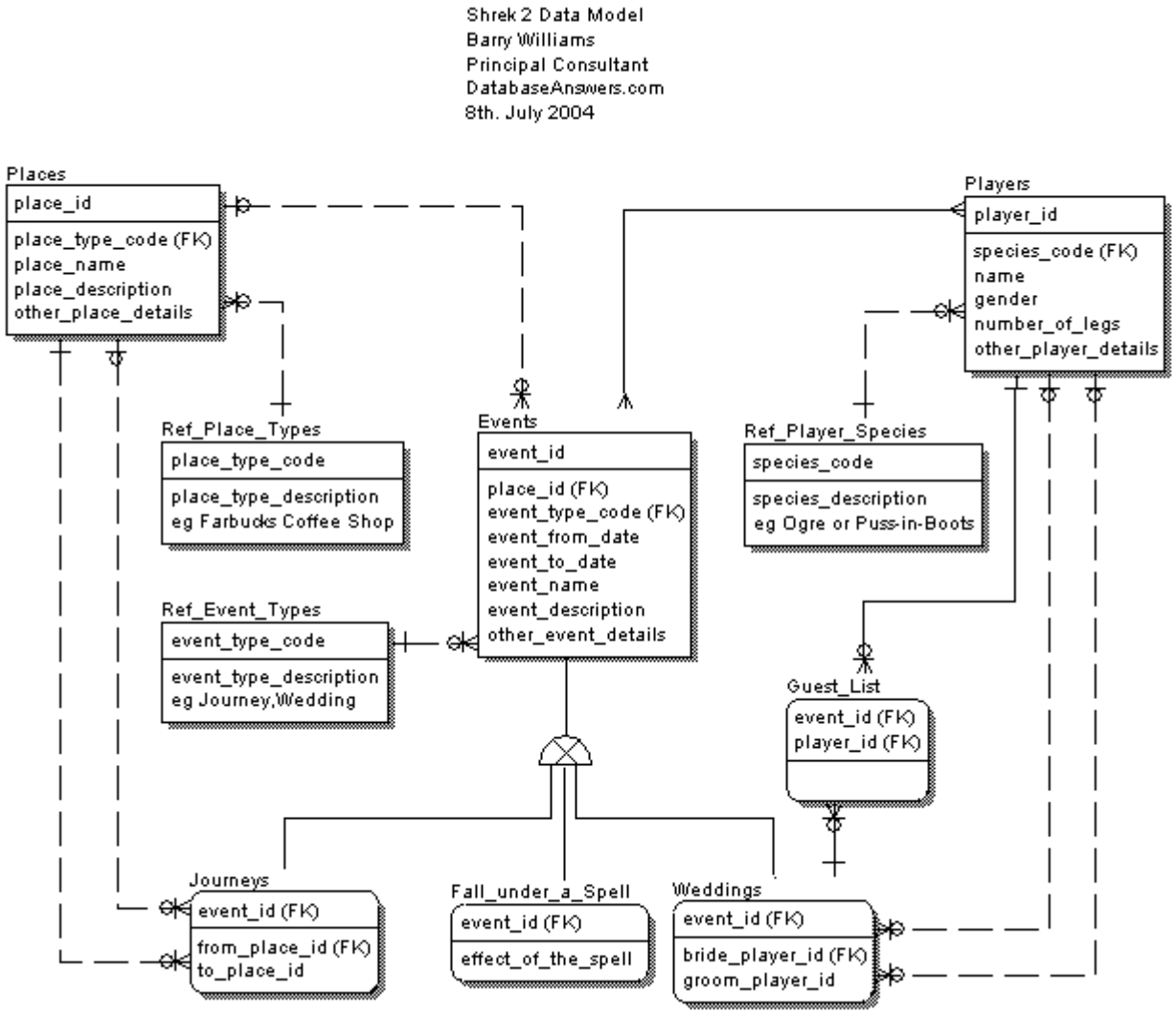


9.6 Shrek Movies

The author's wife enjoyed the *Shrek 2* movie so this model was created. It turned out to be a very interesting exercise in generality.

You can see that events are the dominant entity that are the super-type, and examples of sub-types are journeys, weddings and people falling under a spell.

A need for inheritance was also found, so creating the data model turned out to be a worthwhile task.



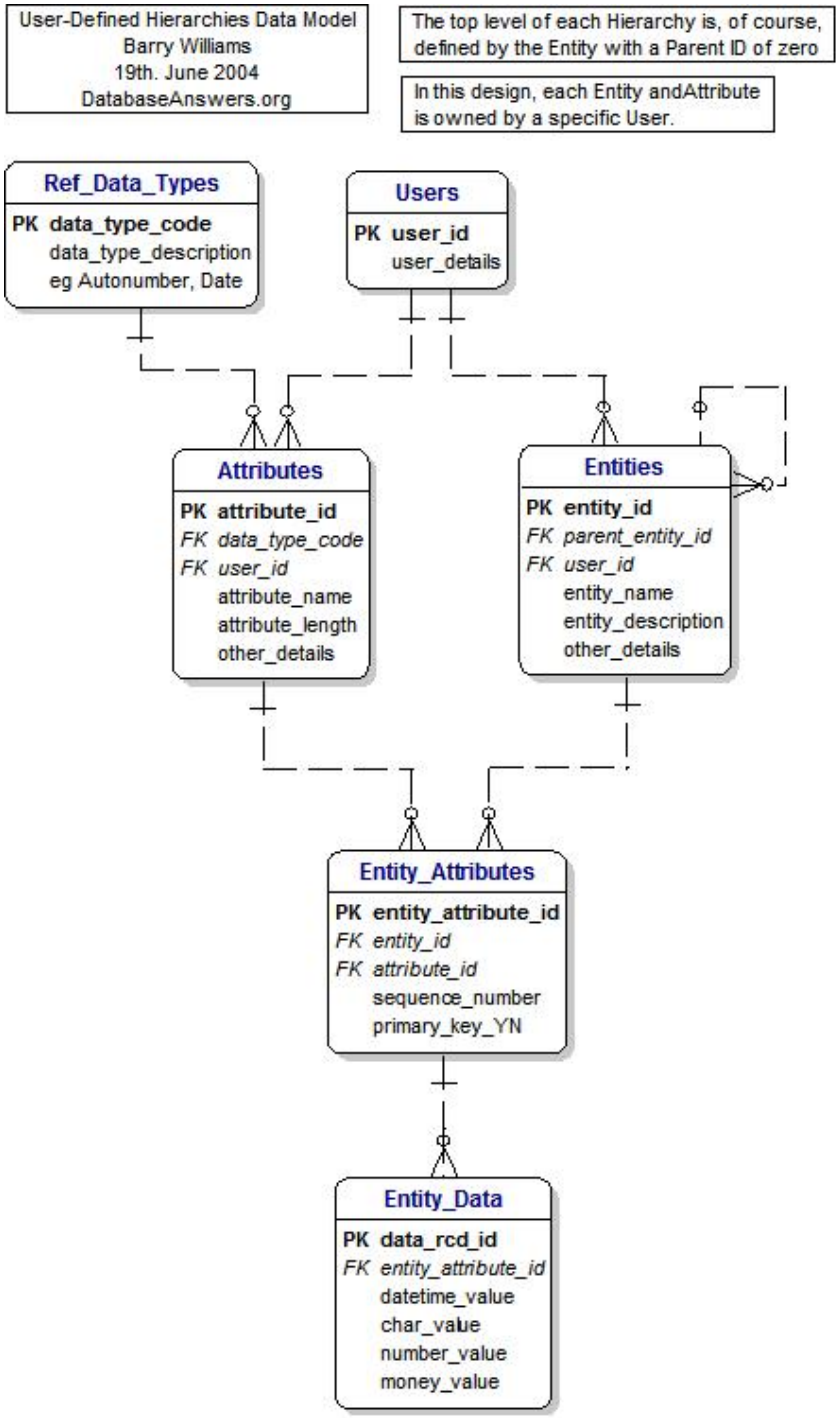
9.7 User-Defined Hierarchies

Here is the page link:

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

The model shows an Entity-Attribute-Value design that is described quite well in Wikipedia:

- http://en.wikipedia.org/wiki/Entity-attribute-value_model



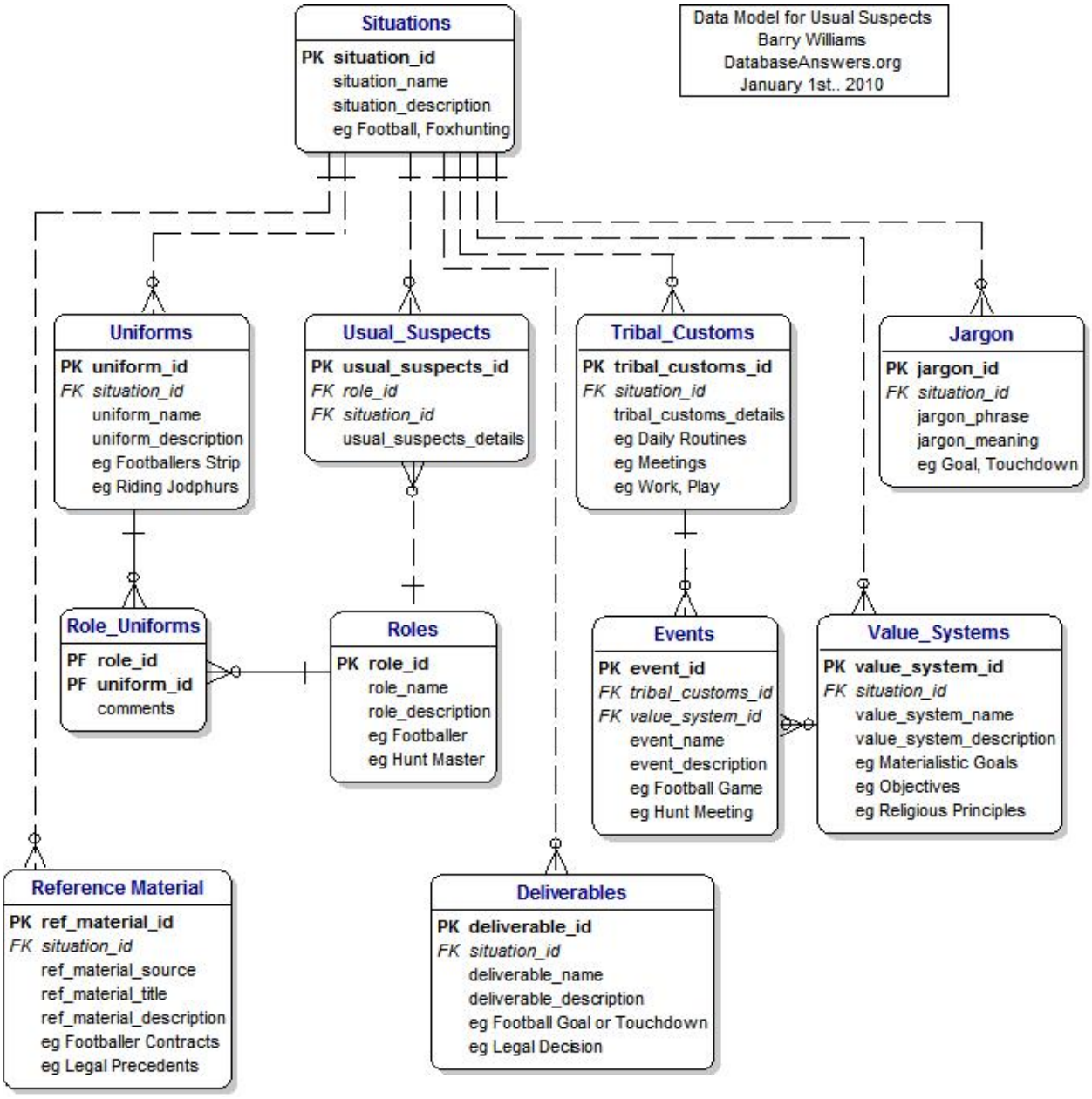
9.8 Usual Suspects

Here is the page link:

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

The striking aspect of this model is that the approach of **usual suspects** applies in many environments in everyday life where this pattern can be identified:

- People wear specific kinds of uniforms
- They use a particular jargon
- They behave in predictable ways, which we can call *tribal customs*



9.9 What have we learned?

In this chapter, we have learned about generic data models. These are models that can apply to many real situations that share a common structure. A good example is usual suspects that can apply to many situations and which we discuss in the section above.

10. Commercial Web Sites

10.1 Introduction

This chapter will discuss data models for a number of different commercial Web sites.

The approach in this chapter is to discuss a number of independent and unrelated data models that have some interesting characteristics.

We selected the data models in this chapter because we found them interesting. Database Answers does not have any association with any of these commercial organizations.

10.1.1 What is this?

A selection of data models for commercial Web sites that have attracted the interest of the author.

10.1.2 Why is it important?

It is important to gain an understanding of how simple or complex the databases can be that support some popular Web sites. For example, Craigslist Web site is very popular but the database behind it could be very simple.

10.1.3 What Will I Learn?

You will learn lessons that will help you if you want to build a Web site similar to an existing one for interest, or to help you create a database to keep track of your Avon purchases or eBay bids.

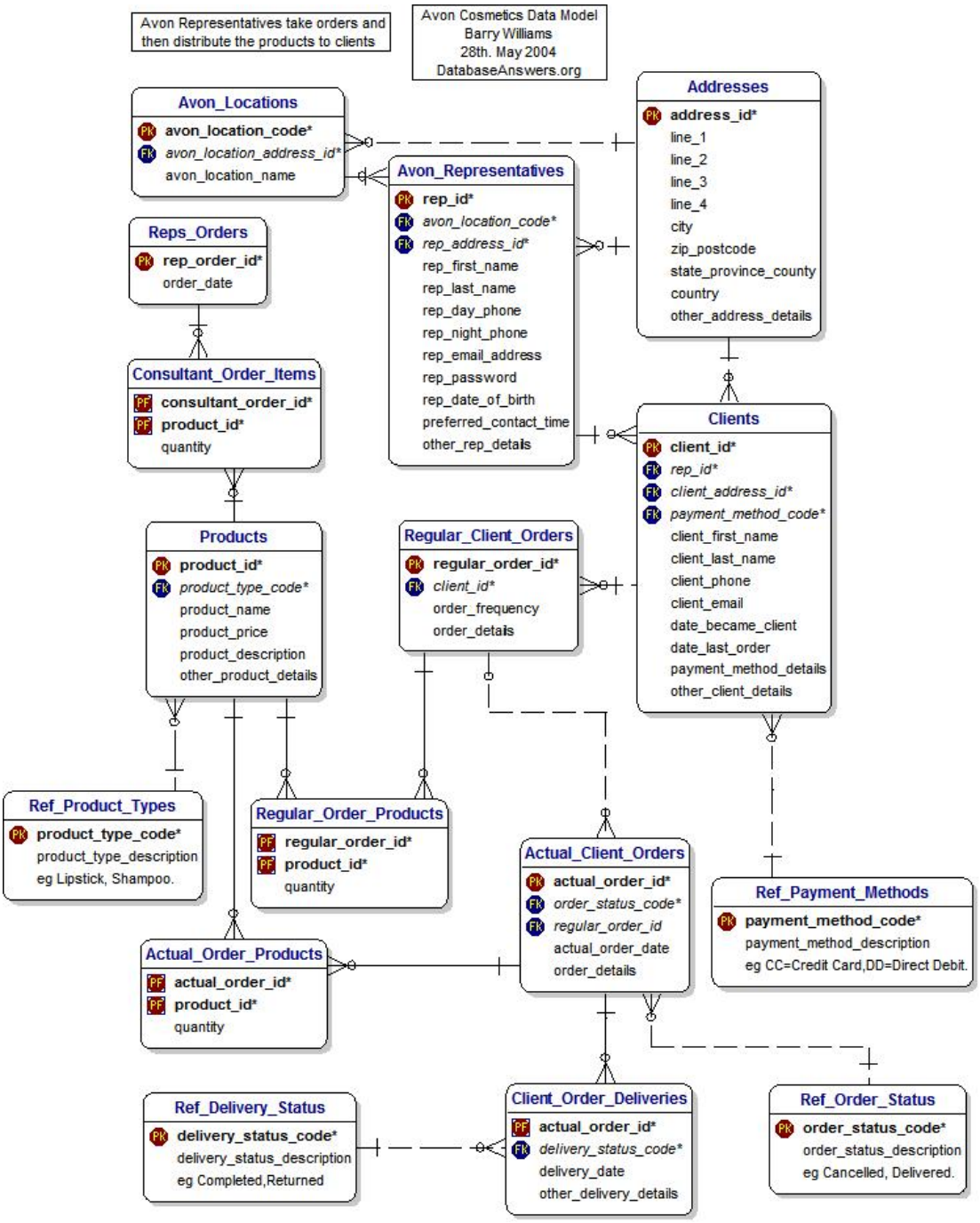
10.2 Avon Cosmetics

Here is the link to the data model on the Database Answers Web site:

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

This model was created by Database Answers in response to a request from a man who had been in a car accident and was not able to work. With the use of our

database, based on this model, he was able to help his wife and feel good about himself and the support he was able to give his wife.



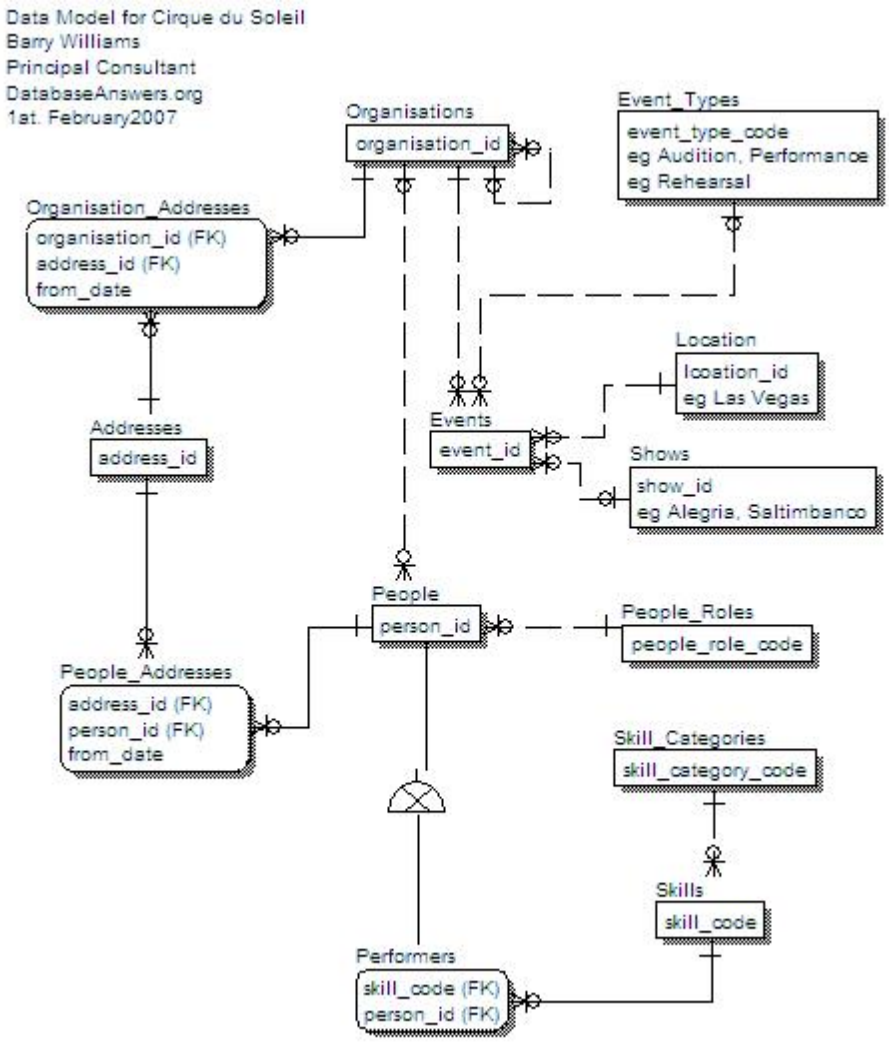
10.3 Cirque du Soleil

Here is the link to the data model on the Database Answers Web site:

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

This model was created by Database Answers because we enjoy Cirque du Soleil and try to arrange company outings to coincide with their visits to London, England.

It shows an example of inheritance, where performers are a sub-type of people.

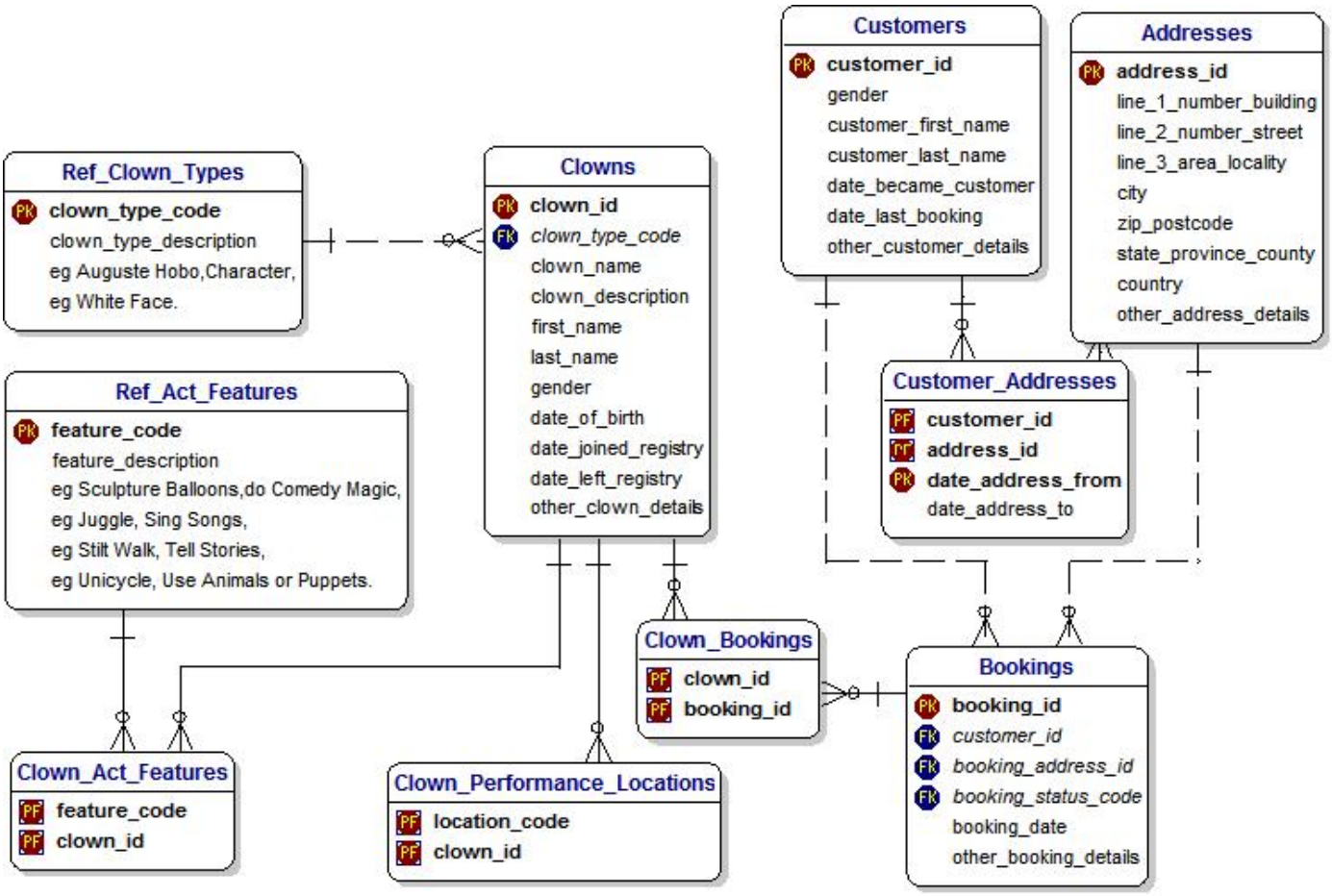


10.4 Clown Registry

Here is the link to the data model on the Database Answers Web site:

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

This model was created after the Clown Registry was featured in an episode of CSI.



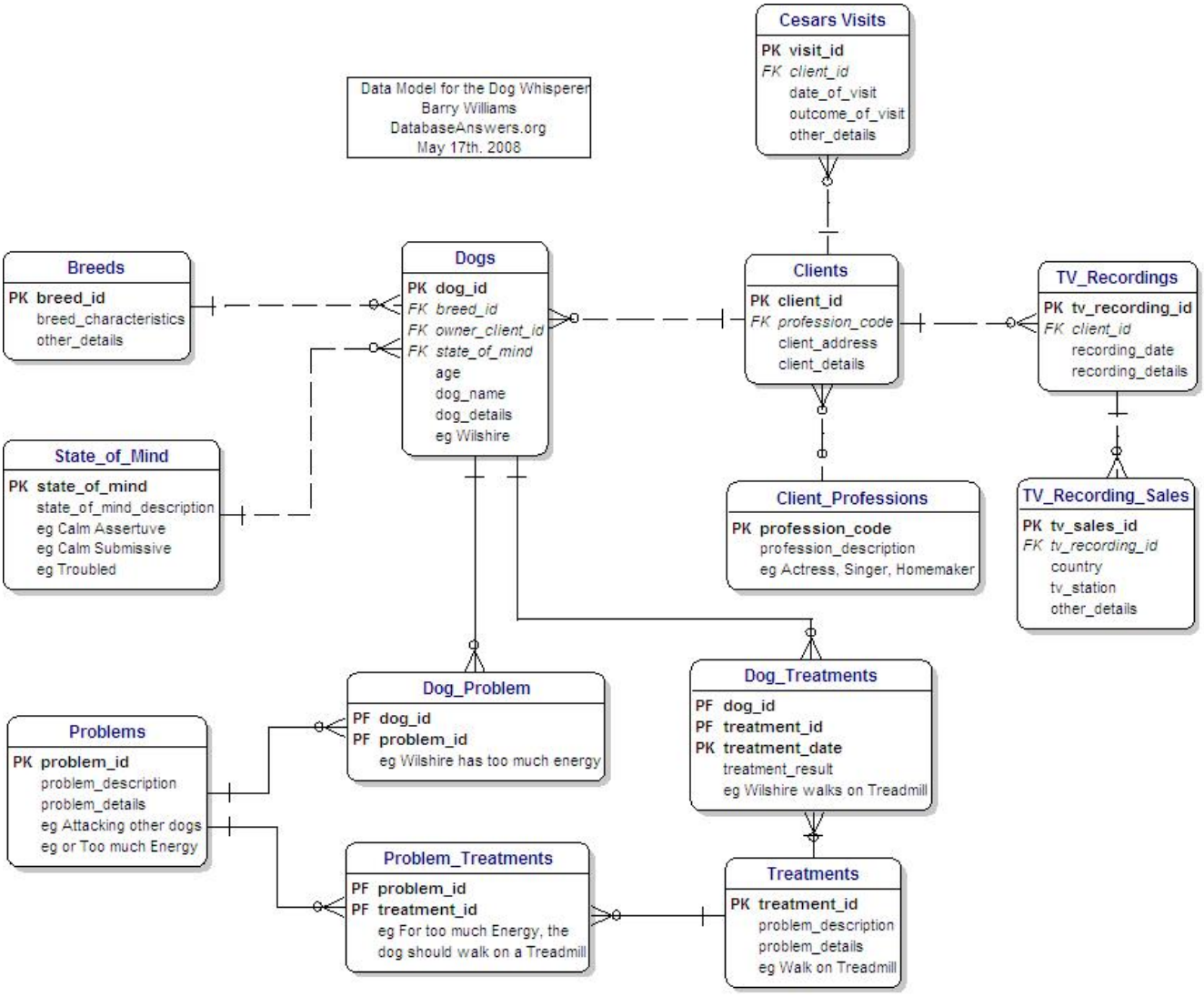
10.5 Dog Whisperer

Here is the link to the data model on the Database Answers Web site:

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

This model is based on the *Dog Whisperer* TV show. It shows an example of inheritance, where performers are a sub-type of people. When we watch the TV show, it soon becomes apparent that Cesar Millan (the so-called 'Dog Whisperer') has a very structured approach to solving dog's behavioral problems by focusing on the dog's state of mind, which always ends as calm and submissive.

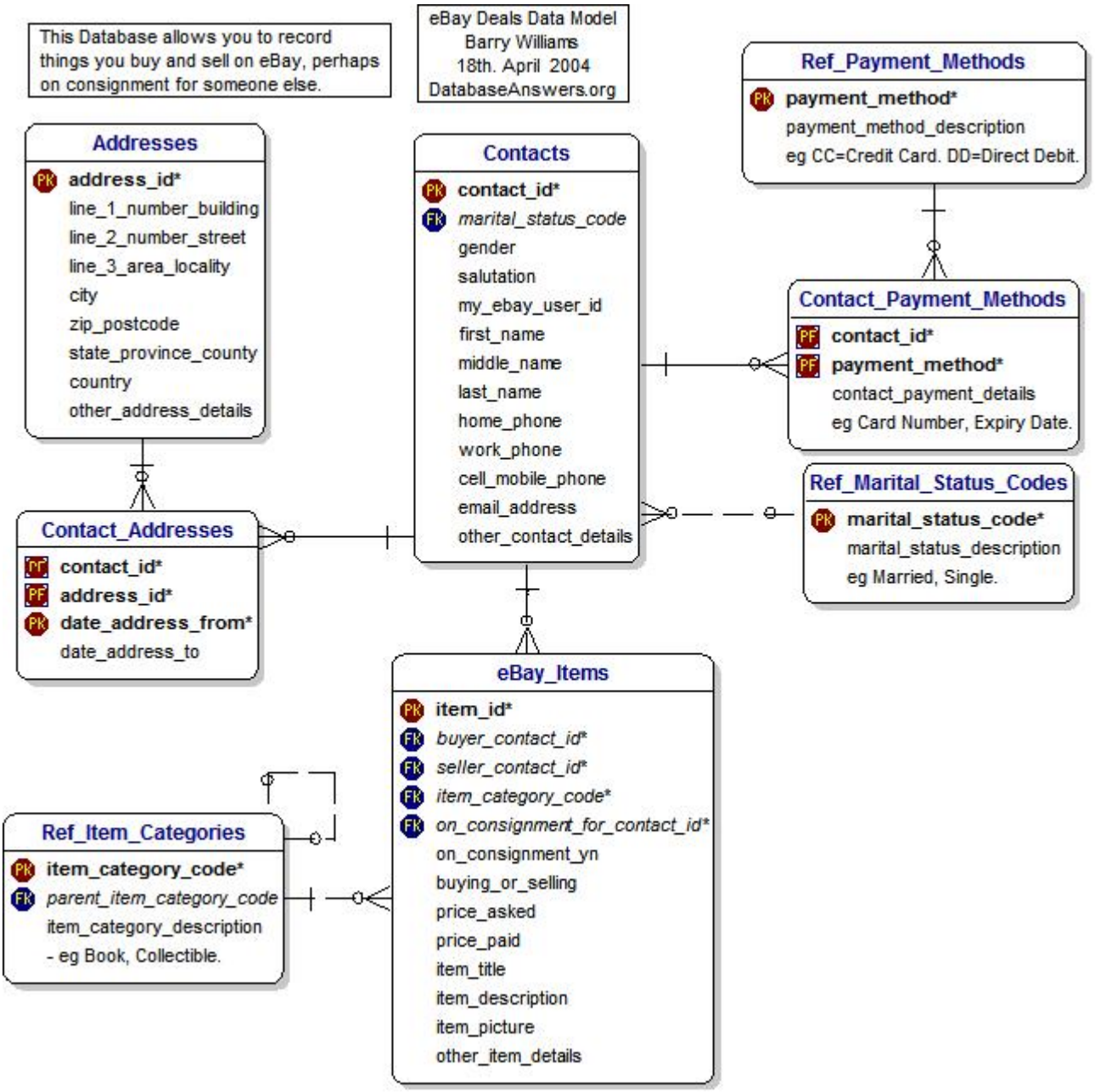
The foundation is that all problems have prescribed treatments that are designed to achieve a desired outcome. This gave the TV program an interest beyond just dogs.



10.6 eBay Deals

This data model was created to help people keep track of their eBay deals:

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

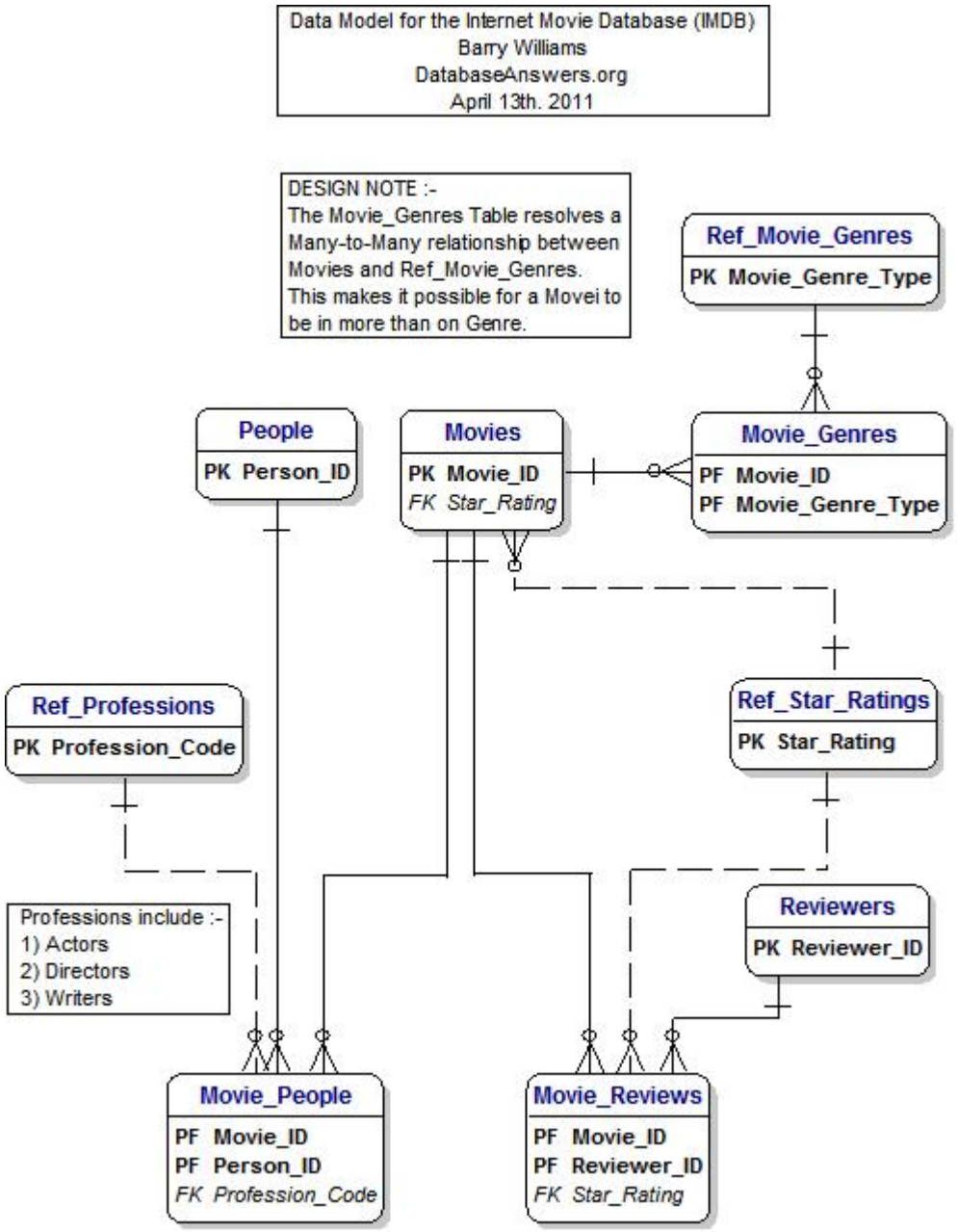


10.7 Internet Movie Database

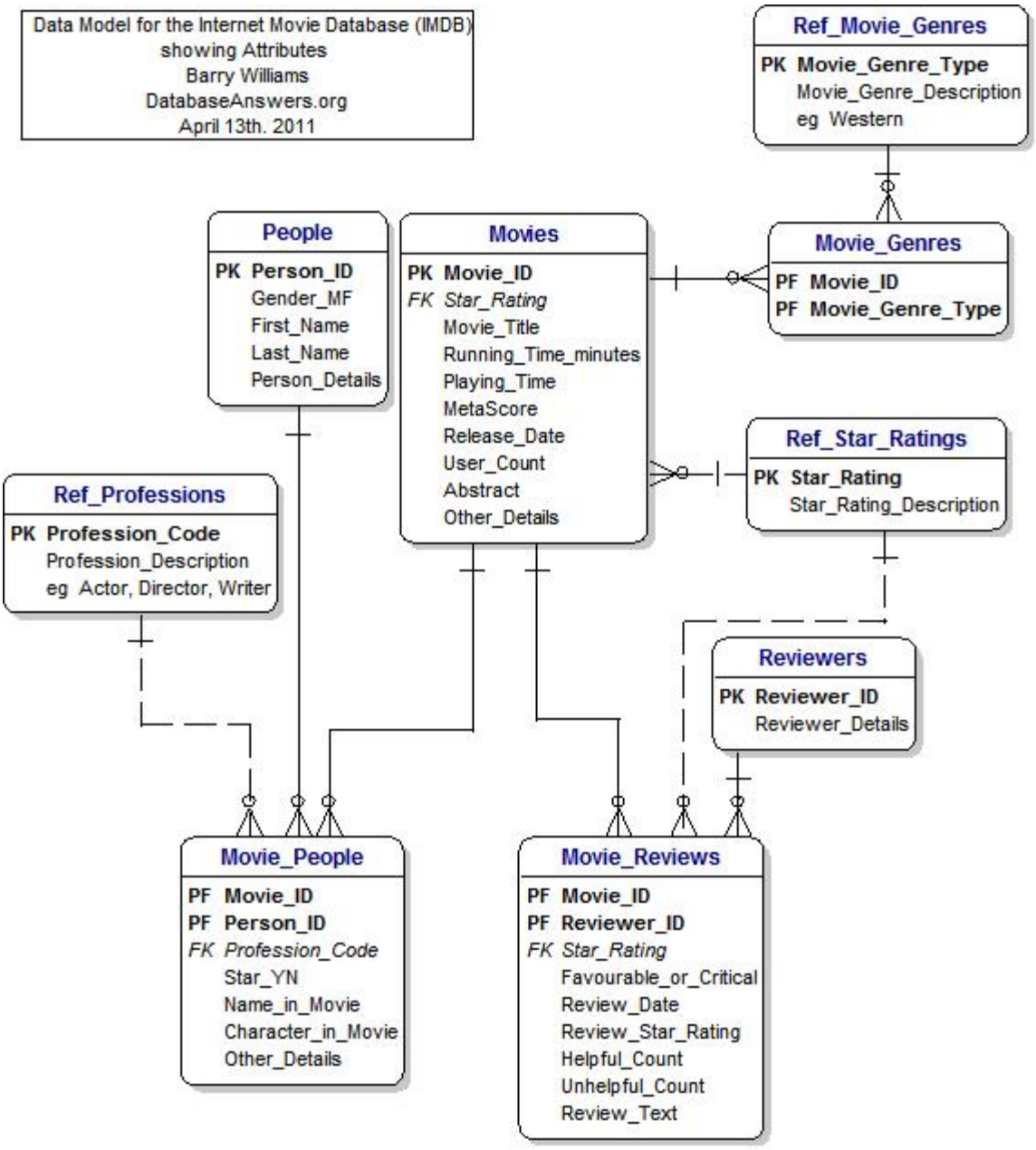
This data model appears on this page:

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

This version shows entities and key fields (but no attributes):



This version also shows the attributes:

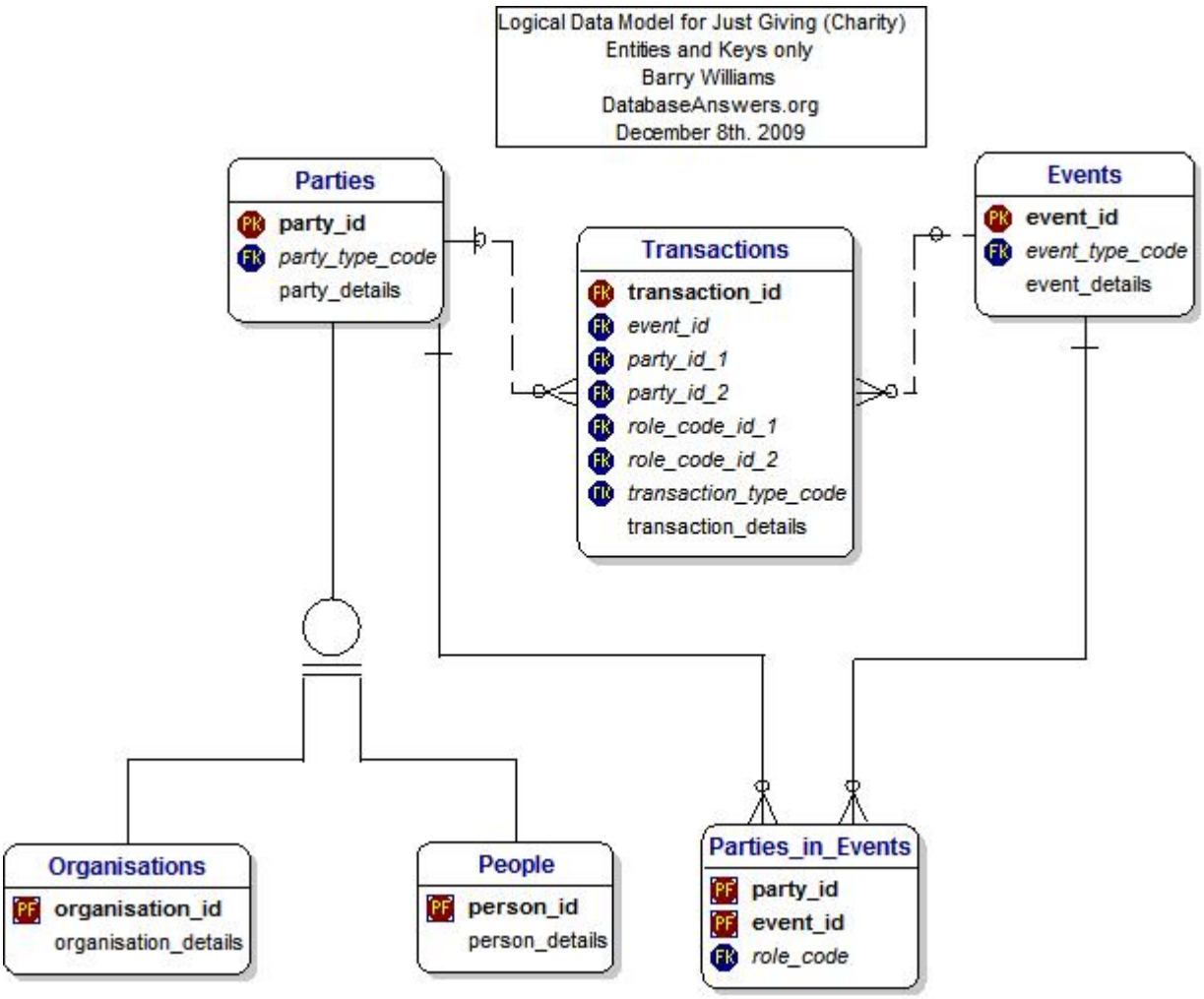


10.8 Just Giving (Charity)

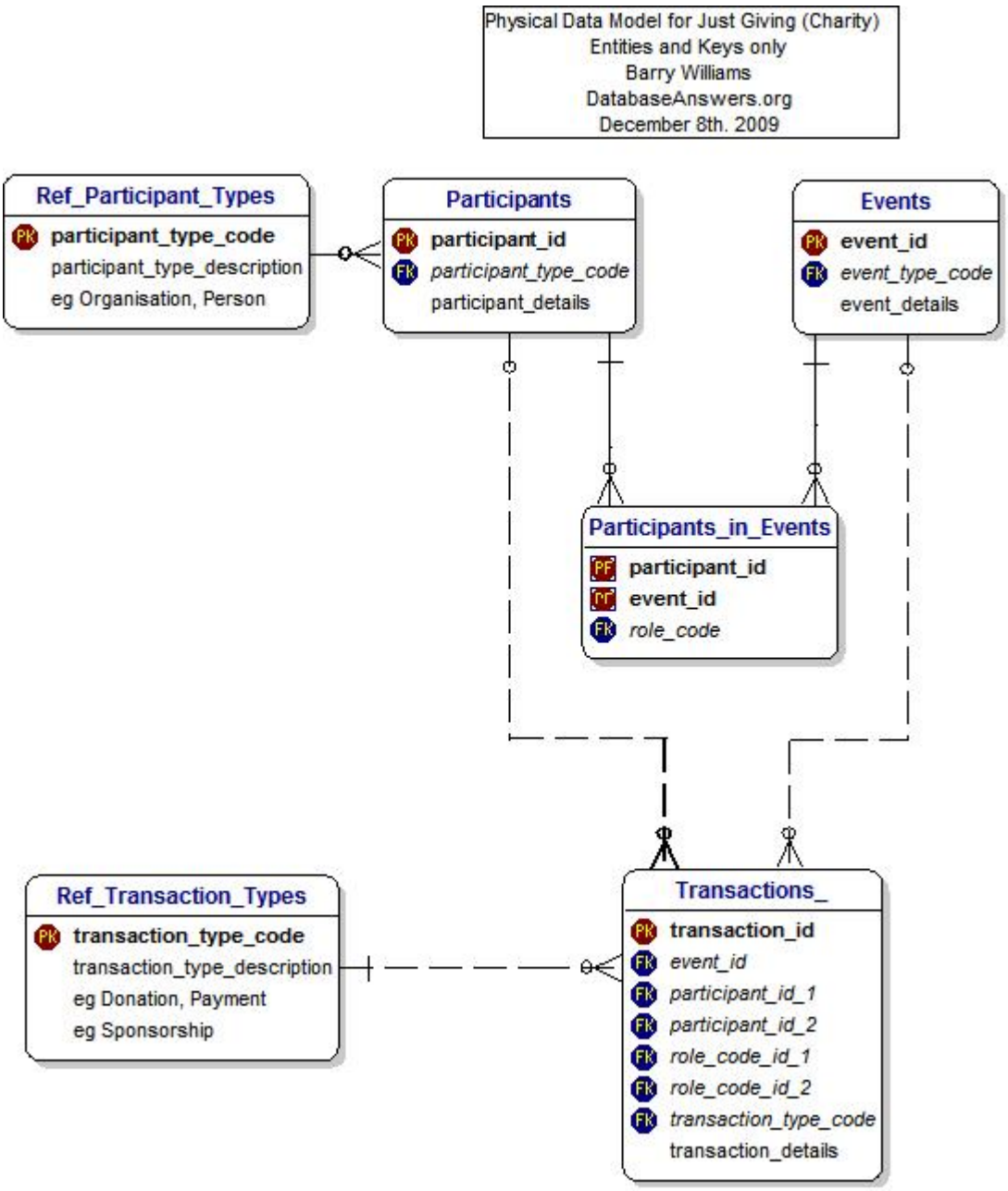
This data model appears on this page:

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

This is a logical model showing inheritance:



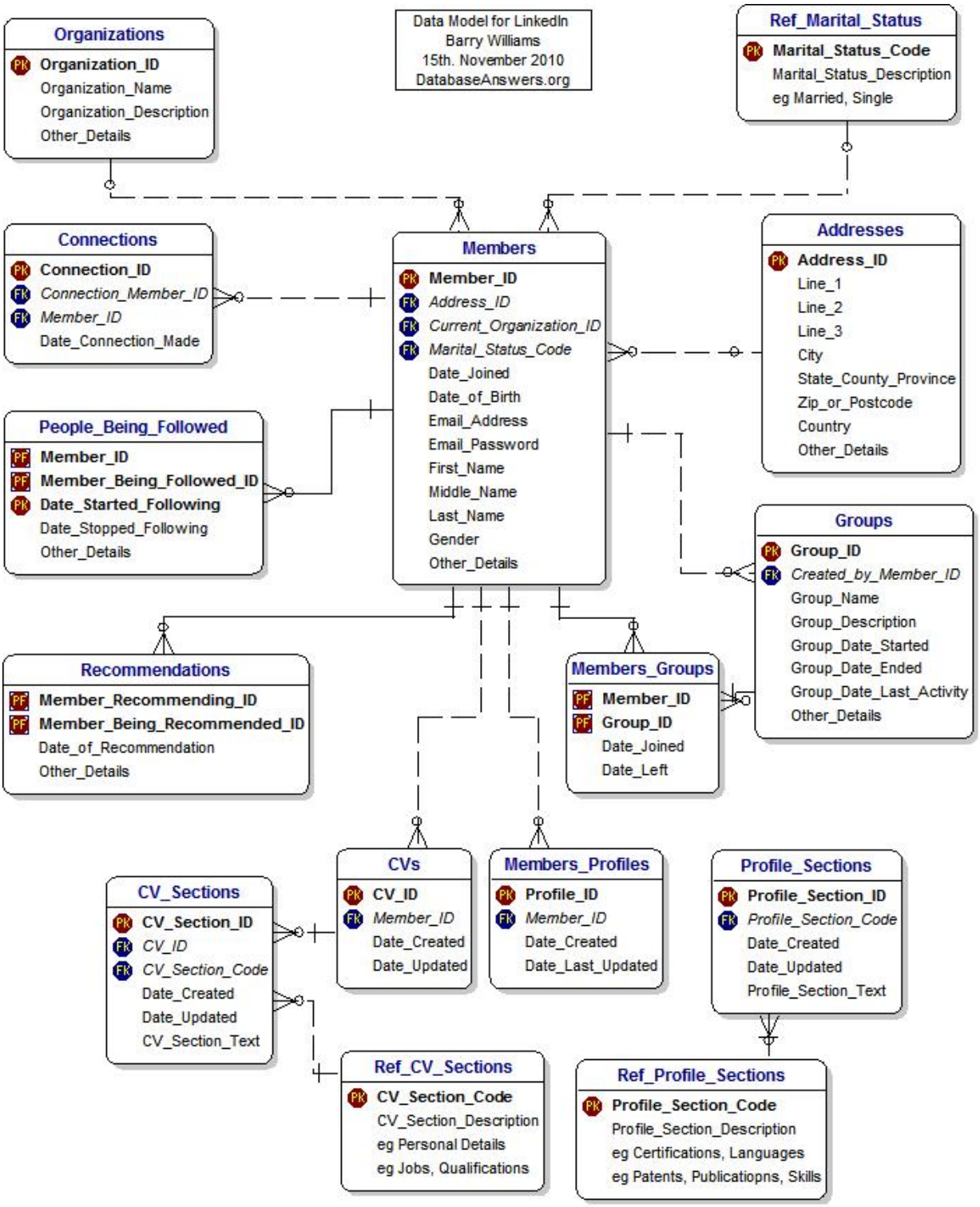
This is a physical model showing how inheritance can be implemented in a relational database:



10.9 LinkedIn

This data model appears on this page:

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



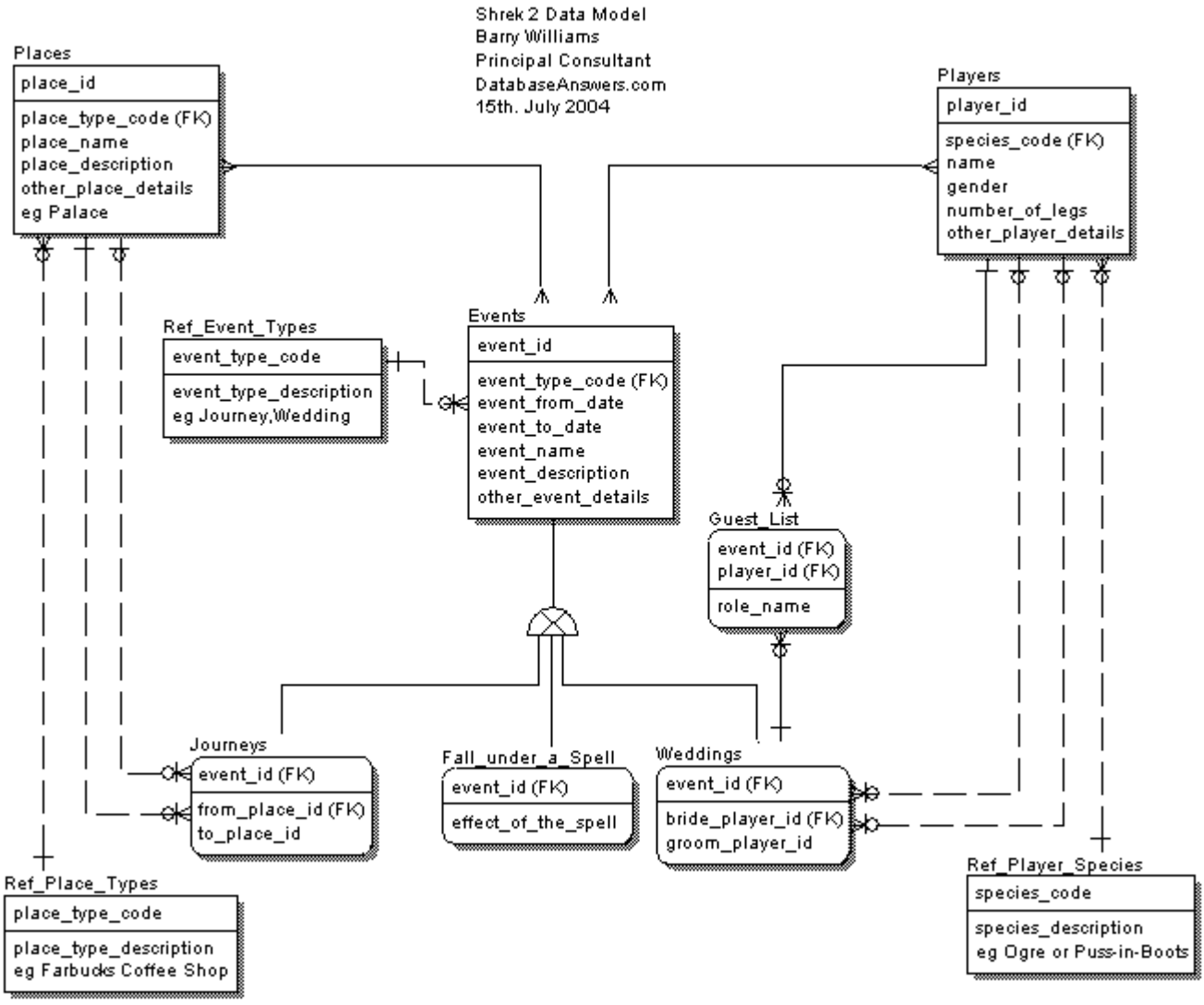
10.10 Shrek Movies

This data model appears on this page:

- http://www.databaseanswers.org/data_models/shrek_2_movie/index.htm

Most of the data models have been created using a very affordable modeling tool called Dezign.

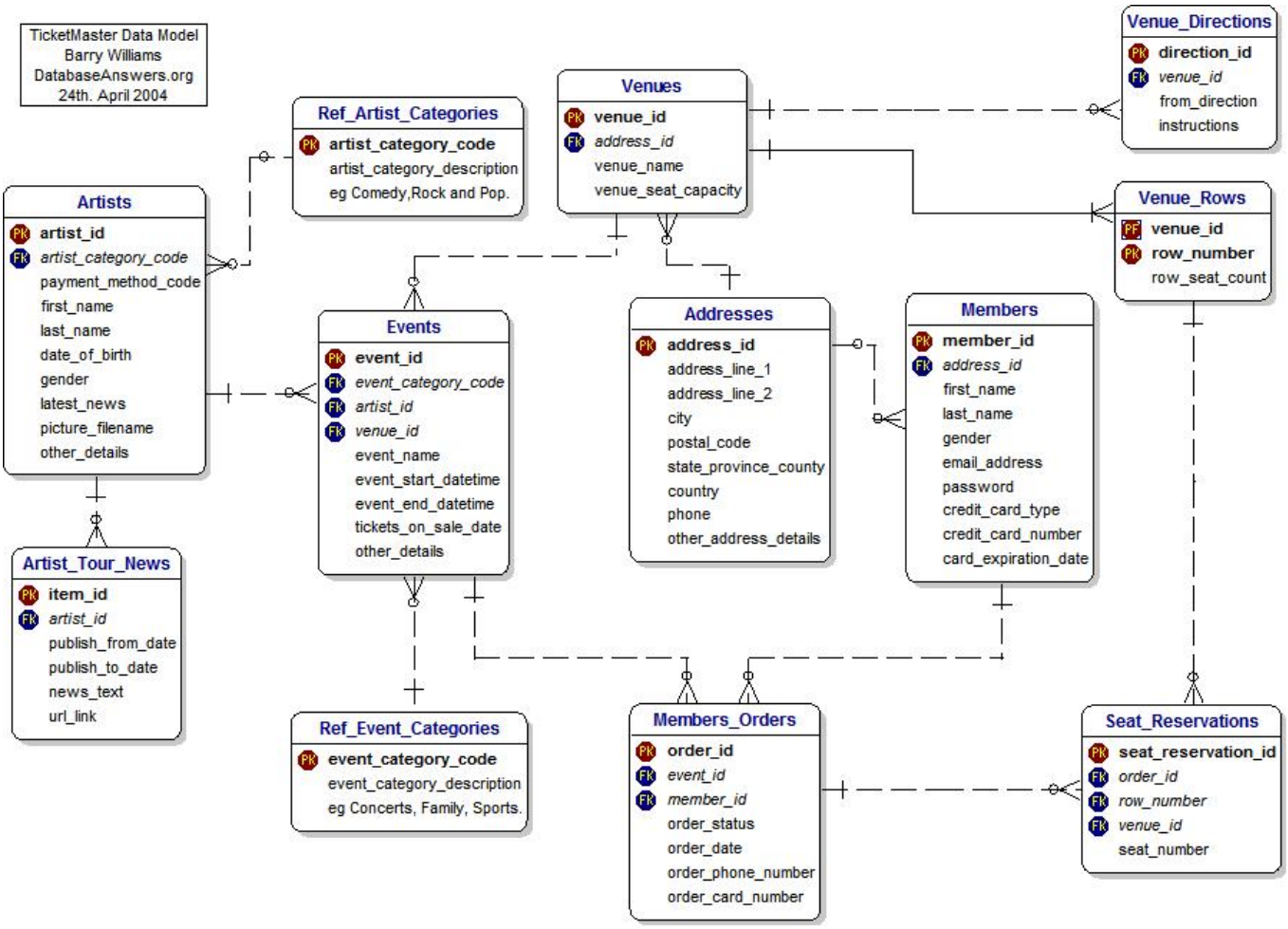
This one was created using ERWin.



10.11 TicketMaster

This data model appears on this page:

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



10.12 What have we learned?

In this chapter we have learned the kind of techniques that are used to create data models for databases that support some of the most popular Web sites on the Internet. These can be very useful if you are thinking about creating your own version of a commercial Web site.

11. From the Cradle to the Grave

11.1 Introduction

This chapter will discuss data models that are appropriate to the stages in our lives from the earliest to the latest.

11.1.1 What is this?

It is structured as a tutorial that takes you step-by-step through each stage and discusses how you create data models at each stage.

11.1.2 Why is it important?

This is important because it helps you understand, starting from first principles, how to deal with increased complexity and conform to the basic principles of a well-designed data model.

11.1.3 What Will I Learn?

The approach in this chapter is to discuss data models covering a typical life cycle from a new-born baby to an old person. This allows us to trace the increasing complexity in life and match it to an increasing complexity in data models.

The 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.

Establishing the Scope of our Data Model

We have decided that the scope is the ‘passages’ in our lives from the cradle to the grave. This will include childhood, teenage years, becoming a student, getting a job, getting married, getting sick, and finally dying

Therefore, any items outside this scope are not '**Things of Interest**'.

Topics covered:

1. Primary Keys and Foreign Keys
2. One-to-Many and Many-to-Many Relationships
3. Hierarchies and Inheritance
4. Reference Data

11.2 I am a new Baby

Topics covered include:

- Entities
- Primary Keys

Baby Elephant in Africa

During the early days, the baby is aware only of its own existence.

In Sweden, new-born babies are issued with unique National Identity numbers.

11.3 Me and Mommy

At this Stage, the baby becomes aware of Mommy's existence so we add her to the model because she is now in scope.

Topics covered include:

1. Foreign Keys

Baby Elephant and Mommy

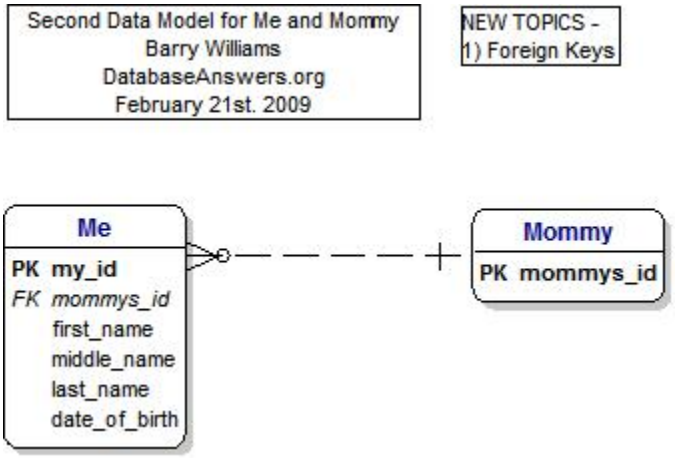
Here we add **relationships** between the entities. When this primary key is used in another table, it is referred to as a *foreign key*. We can see a good example in this diagram, where the 'mommys_id' field appears in the Me Table as a foreign key. This is shown with an 'FK' symbol beside it. The 'mommys_id' field then appears as the primary key in the Mommy Table.

Mandatory Key Fields

A foreign key is usually **mandatory**, in other words, a value for a mommys_id in the Me Table must correspond to the value of the mommys_id for a record in the Mommy Table.

In plain English the business rule would say "Each baby must have a real mommy".

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

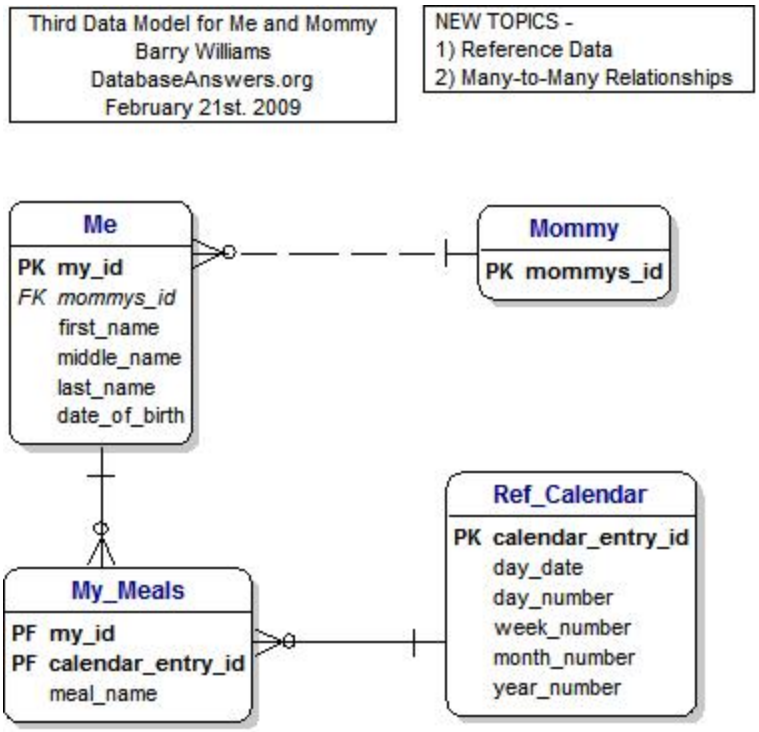


11.4 Me, Mommy and Meals

Now I become aware that I am eating at regular times.

Topics covered include:

1. One-to-Many Relationships



11.5 Children's Playgroups

Topics covered include:

1. Many-to-Many Relationships

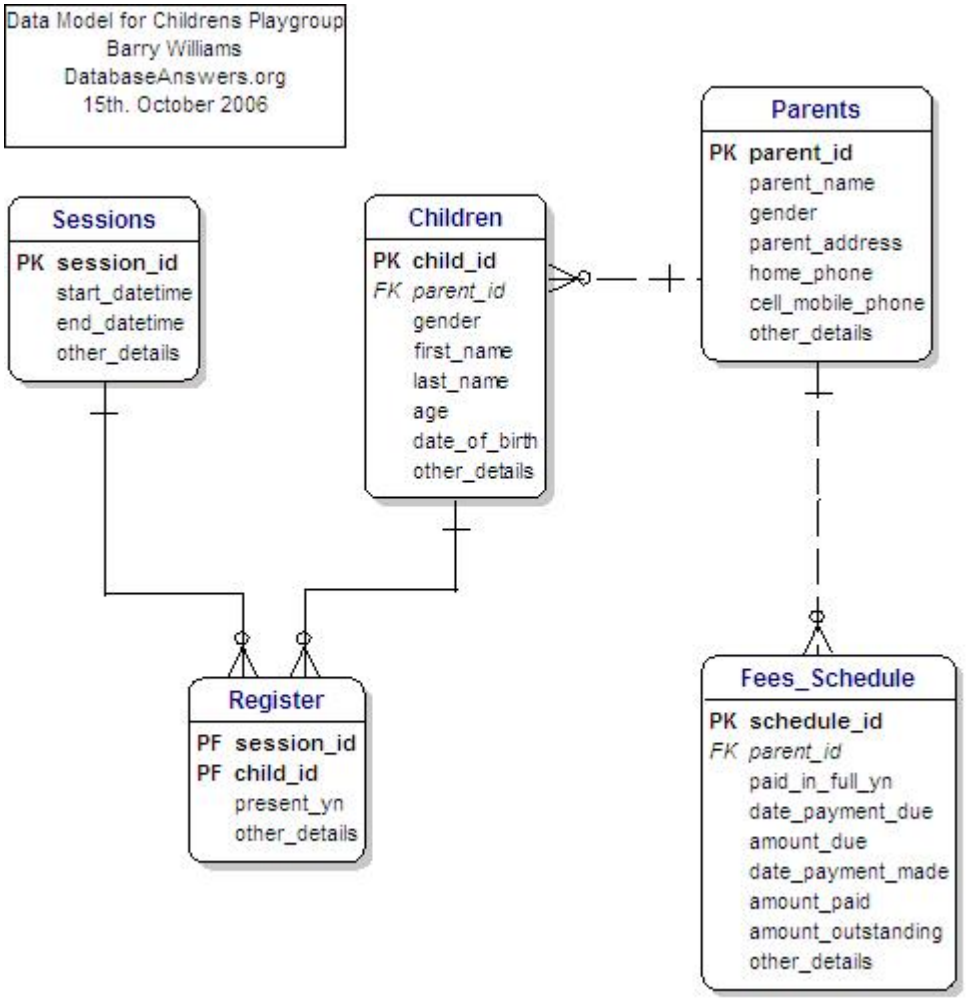
Here we have added the **relationships** between the entities.

1. When this primary key is used in another table, it is referred to as a *foreign ke*y.
2. We can see a good example in this diagram, where the Customer_ID appears in the Customers_Payment_Methods Table as a foreign key.
3. This is shown with an 'FK' symbol beside it

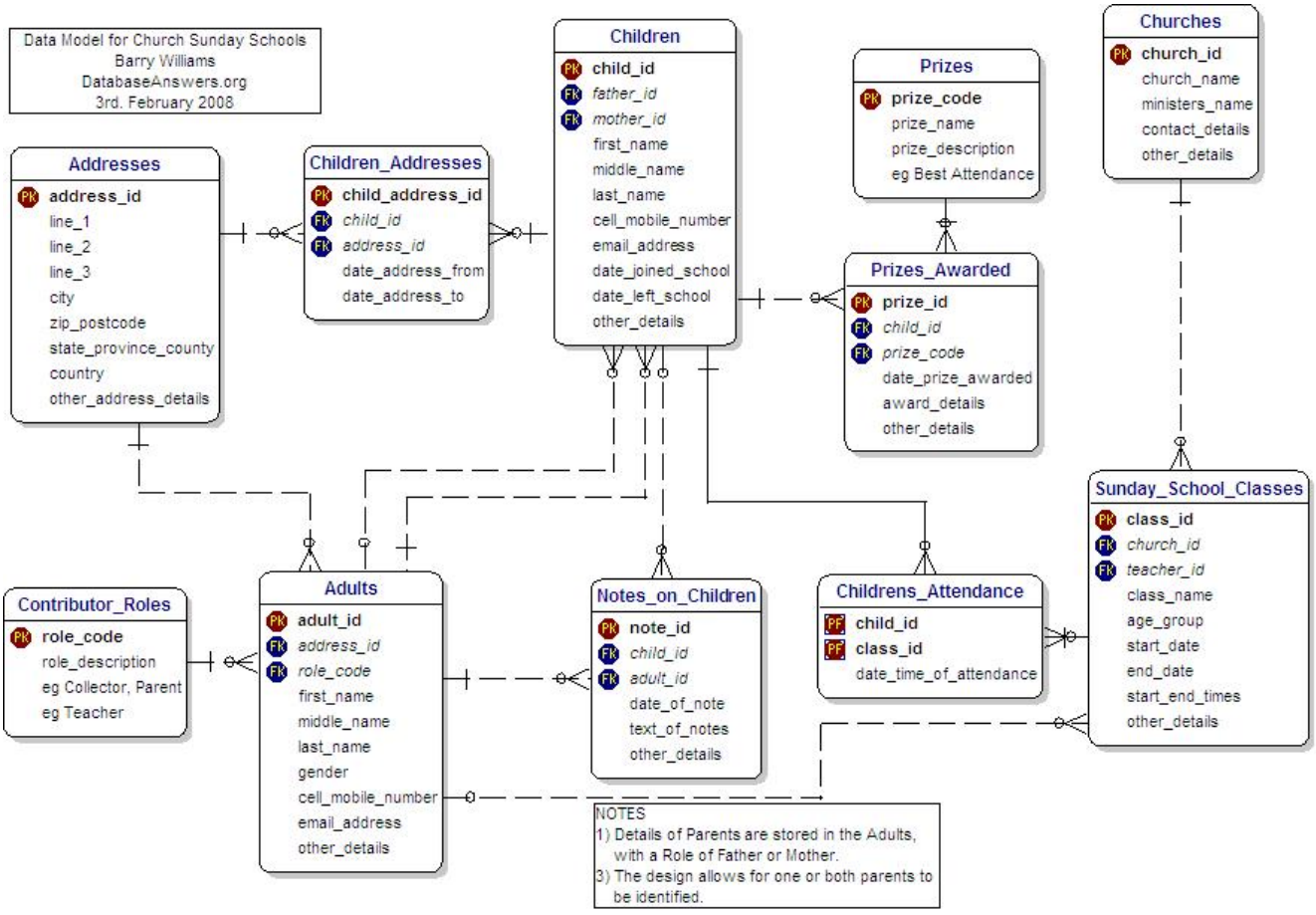
Mandatory Key Fields

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.


Data Model for Childrens Playgroup
Barry Williams
DatabaseAnswers.org
15th. October 2006
Sessions
PK session_id
start_datetime
end_datetime
other_details
Children
PK child_id
FK parent_id
gender
first_name
last_name
age
date_of_birth
other_details
Parents
PK parent_id
parent_name
gender
parent_address
home_phone
cell_mobile_phone
other_details
Register
PF session_id
PF child_id
present_yn
other_details
Fees_Schedule
PK schedule_id
FK parent_id
paid_in_full_yn
date_payment_due
amount_due
date_payment_made
amount_paid
amount_outstanding
other_details

11.6 Church Sunday School



11.7 Student Accommodation

At this Stage, I move into student accommodation.
Topics covered include:

2. Primary and Foreign Keys
3. One-to-Many and Many-to-Many Relationships
4. Reference Data

This diagram shows how the hierarchies of products and product types that we have just discussed are shown in our **Entity-Relationship** diagram.

Rabbit Ears
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.

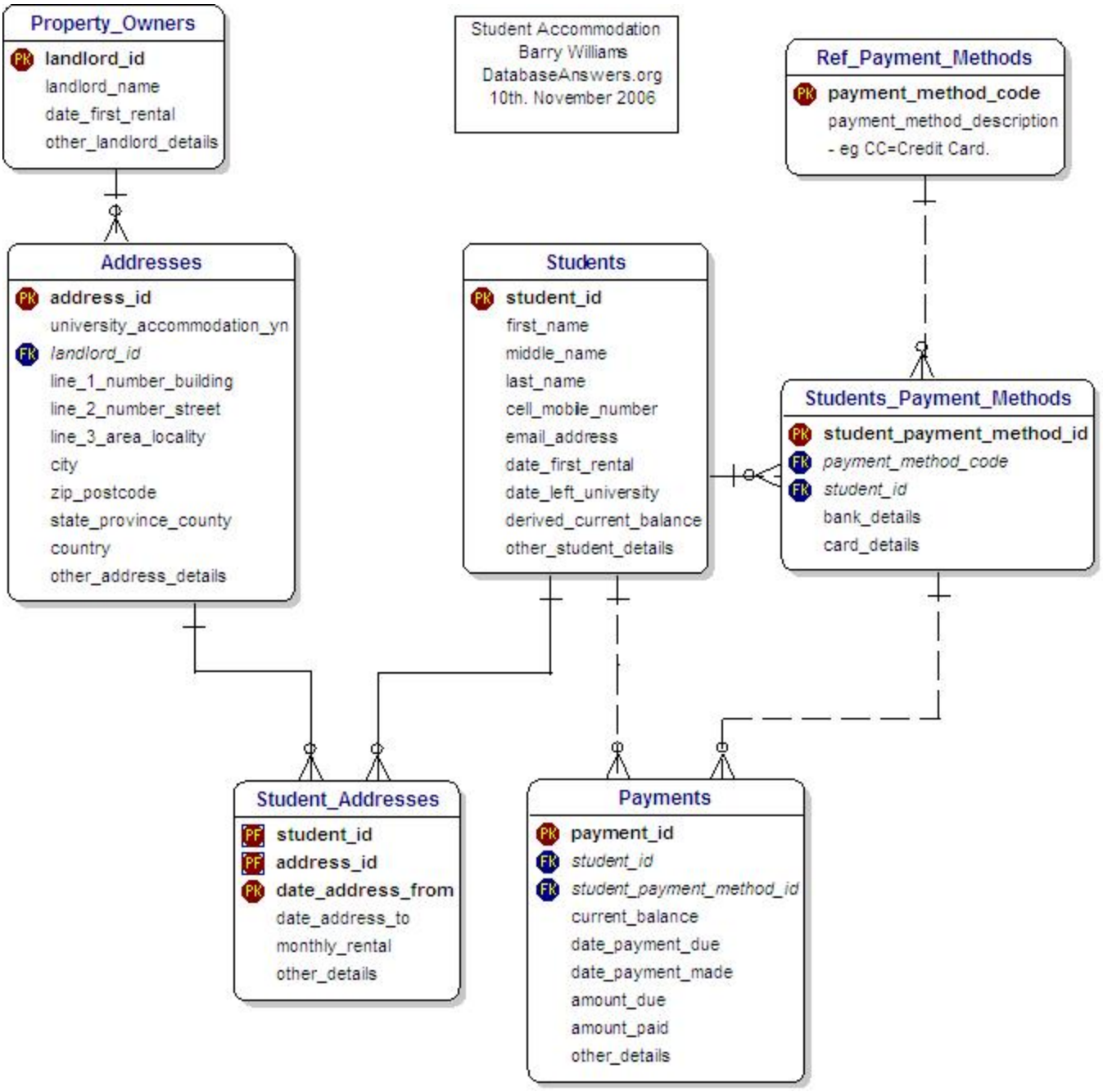
Data analysts call this a *reflexive* relationship, or informally, simply *rabbit ears*.

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 a coffee shop, 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.


Property_Owners
landlord_id
landlord_name
date_first_rental
other_landlord_details
Student Accommodation
Barry Williams
DatabaseAnswers.org
10th. November 2006
Ref_Payment_Methods
payment_method_code
payment_method_description
- eg CC=Credit Card.
Addresses
address_id
university_accommodation_yn
landlord_id
line_1_number_building
line_2_number_street
line_3_area_locality
city
zip_postcode
state_province_county
country
other_address_details
Students
student_id
first_name
middle_name
last_name
cell_mobile_number
email_address
date_first_rental
date_left_university
derived_current_balance
other_student_details
Students_Payment_Methods
student_payment_method_id
payment_method_code
student_id
bank_details
card_details
Student_Addresses
student_id
address_id
date_address_from
date_address_to
monthly_rental
other_details
Payments
payment_id
student_id
student_payment_method_id
current_balance
date_payment_due
date_payment_made
amount_due
amount_paid
other_details

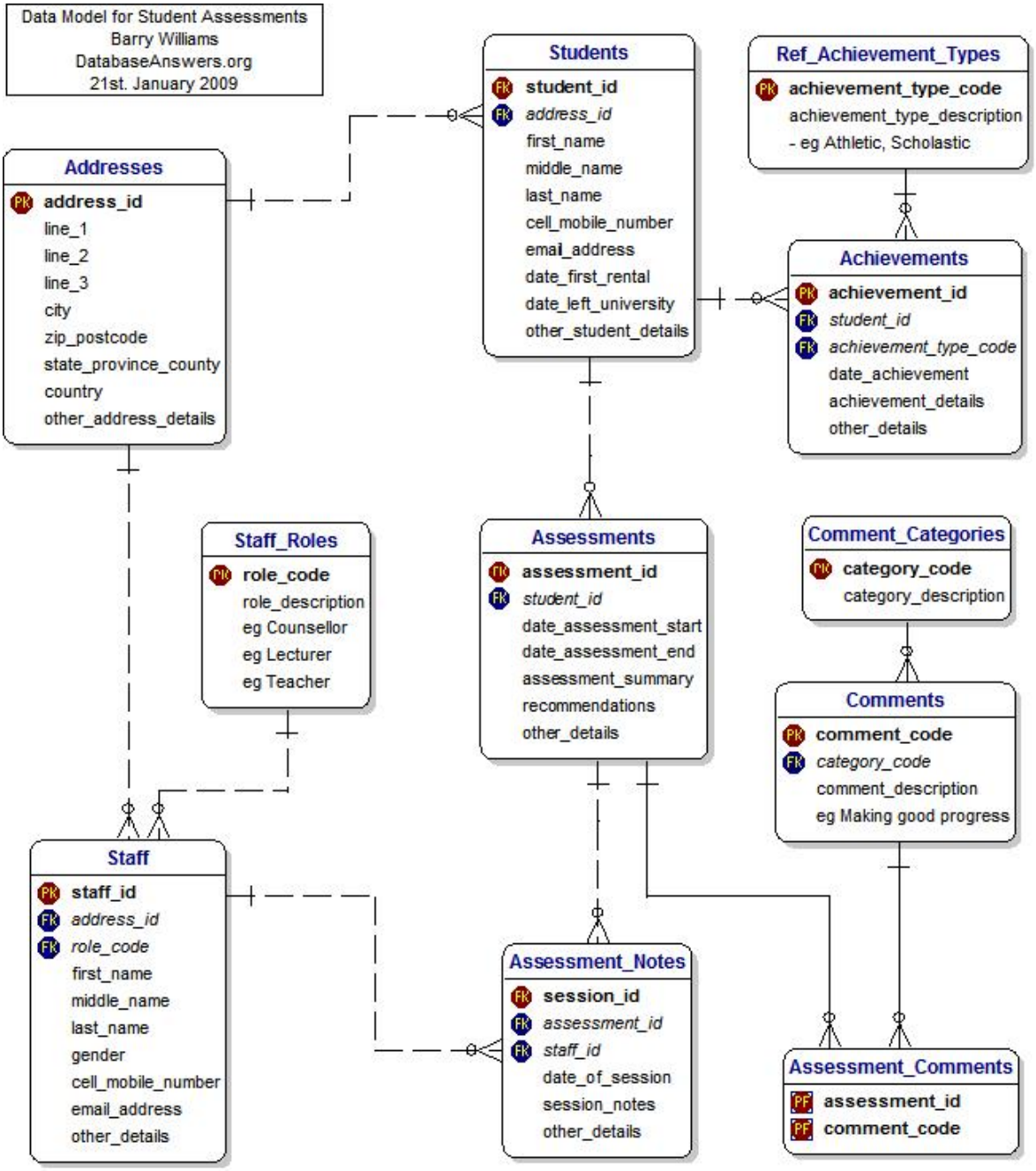
11.8 Student Assessments

Topics covered include:

- Primary and Foreign Keys
- One-to-Many and Many-to-Many Relationships
- Reference Data - Addresses

This model shows that:

- A student can have zero, one or many achievements
- A student can have zero, one or many assessments
- Each assessment can be associated with notes.



11.9 Joining Facebook

Topics covered include:

- Primary and Foreign Keys
- One-to-Many and Many-to-Many Relationships
- Reference Data

This diagram shows address types, which are an example of reference data. This kind of data has the following characteristics:

1. It doesn't change very much.
2. It has a relatively small number of values, usually less than a few dozen and never more than a few hundred.
3. Therefore we can show it with a code as a primary key.
4. Data in Reference Data Tables can be used to populate drop-down lists for users to select from.
5. In this way, it is used to ensure that all new data is valid.

11.113.1 Standards

- In the Address Table, you will see a field called 'iso_country_codes'.
- ISO stands for the 'International Standards Organization'.
- Where possible, it's always good to use national or international standards.

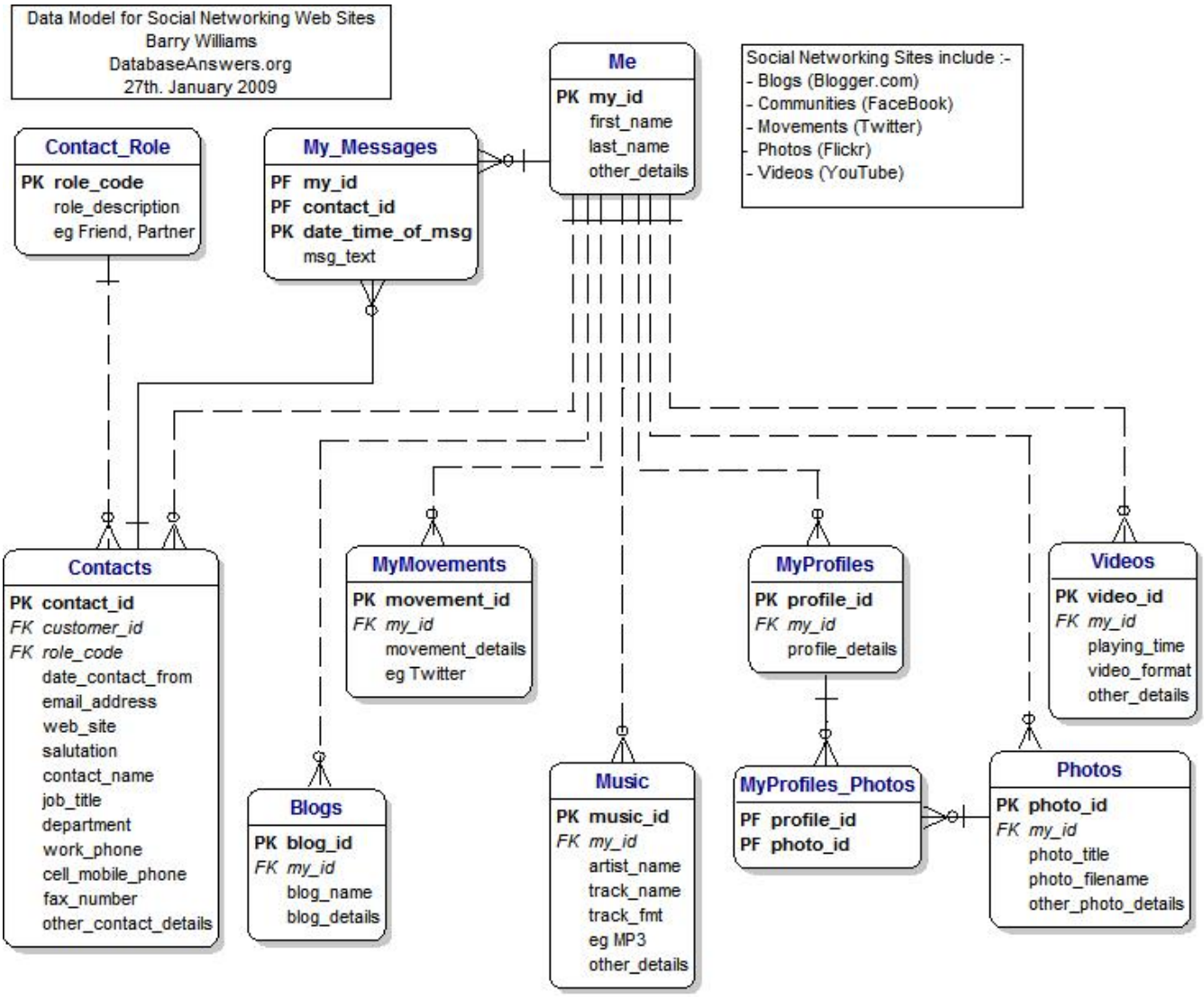
11.13.2 Customer Addresses

This is a general and flexible approach to handling addresses in our data model. We have a separate Address Table, so we can have more than one address for any customer very easily.

This design also has some 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 'Address Validation Software'. The author has used QAS with great success in the past.

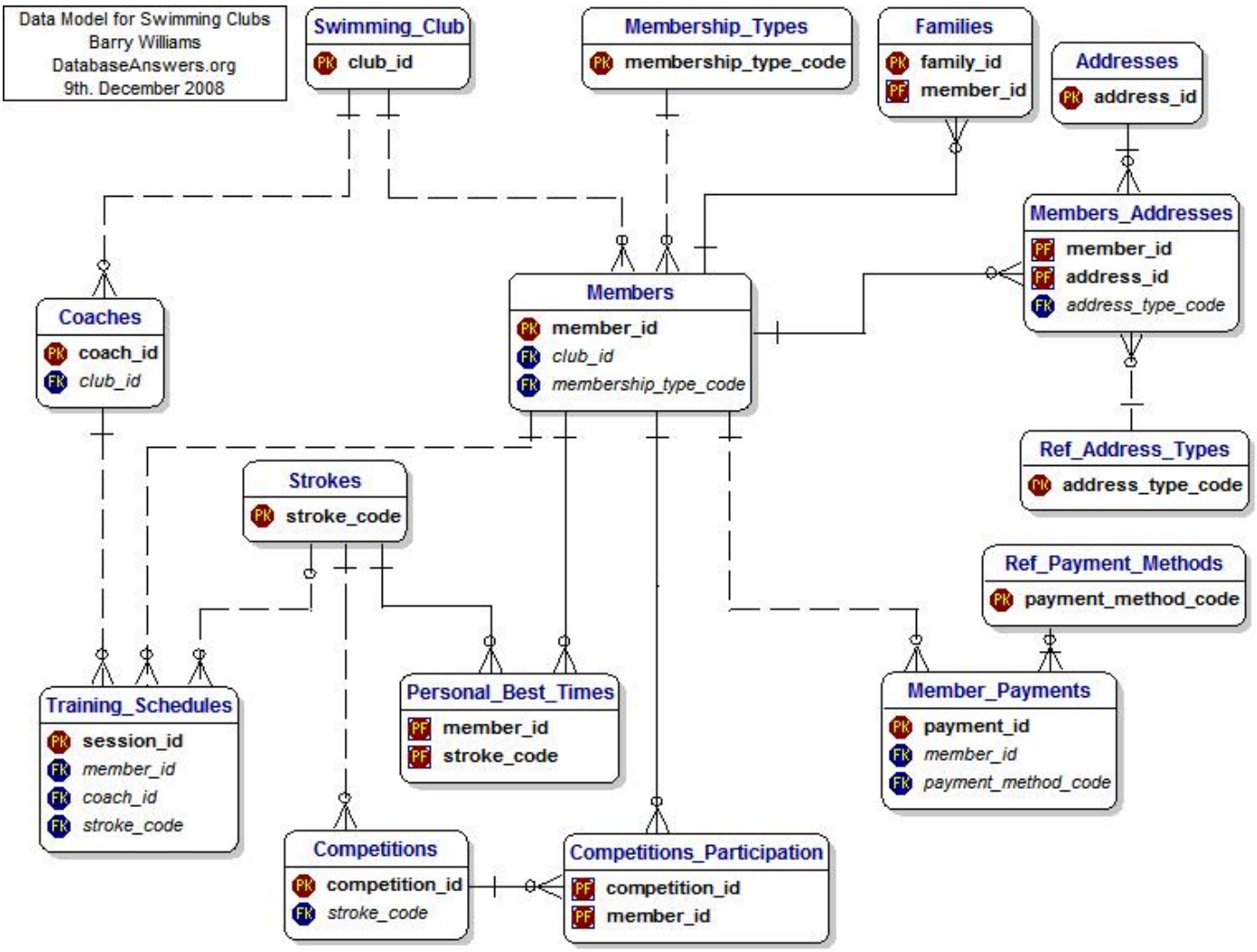
With this approach, we can always be sure that we have 100% good address data in our database.


Data Model for Social Networking Web Sites
Barry Williams
DatabaseAnswers.org
27th. January 2009
Social Networking Sites include :-
- Blogs (Blogger.com)
- Communities (FaceBook)
- Movements (Twitter)
- Photos (Flickr)
- Videos (YouTube)
Me
PK my_id
first_name
last_name
other_details
Contact_Role
PK role_code
role_description
eg Friend, Partner
My_Messages
PF my_id
PF contact_id
PK date_time_of_msg
msg_text
Contacts
PK contact_id
FK customer_id
FK role_code
date_contact_from
email_address
web_site
salutation
contact_name
job_title
department
work_phone
cell_mobile_phone
fax_number
other_contact_details
MyMovements
PK movement_id
FK my_id
movement_details
eg Twitter
MyProfiles
PK profile_id
FK my_id
profile_details
Videos
PK video_id
FK my_id
playing_time
video_format
other_details
Blogs
PK blog_id
FK my_id
blog_name
blog_details
Music
PK music_id
FK my_id
artist_name
track_name
track_fmt
eg MP3
other_details
MyProfiles_Photos
PF profile_id
PF photo_id
Photos
PK photo_id
FK my_id
photo_title
photo_filename
other_photo_details

11.10 Joining a Swimming Club

Mission Viejo Masters Competition
2009 U.S. Masters Swimming Club of the Year

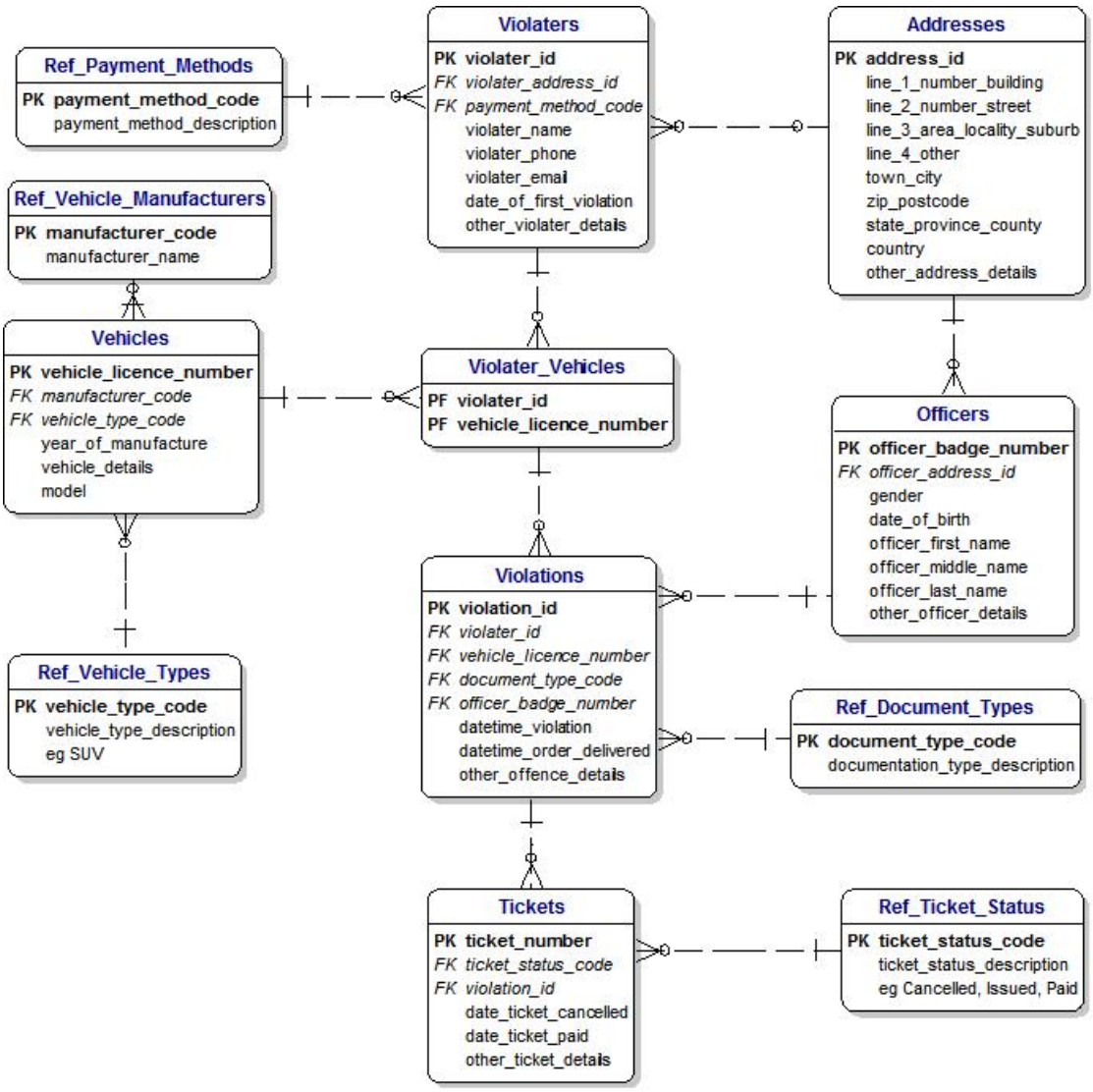
The central entity/table in this data model is members, which emphasizes its importance.



11.11 A Ticket from a Traffic Cop

We start with violators who are always associated with a vehicle. Vehicles in turn are always associated with one or more violations, which result in violations.

Robert Blake as a traffic cop in *Electra Glide in Blue*.



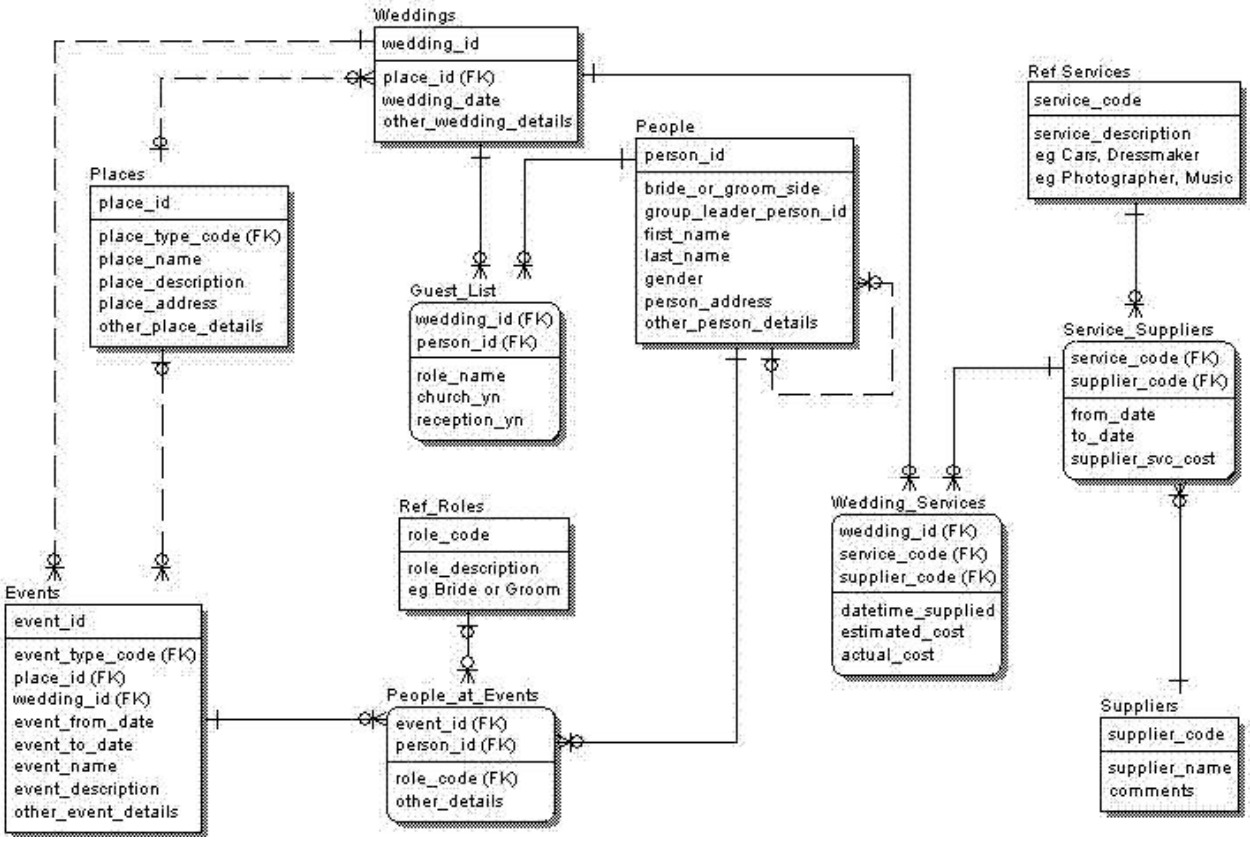
11.12 I Get Married

Topics covered:

- Primary and Foreign Keys
- One-to-Many and Many-to-Many Relationships
- Reference Data

This model was created using a different data modeling tool, called ERWin from Computer Associates. It shows that if you are familiar with the underlying principles that you will be able to understand and ERD.

The **wedding** is the dominant table and is held in one place.

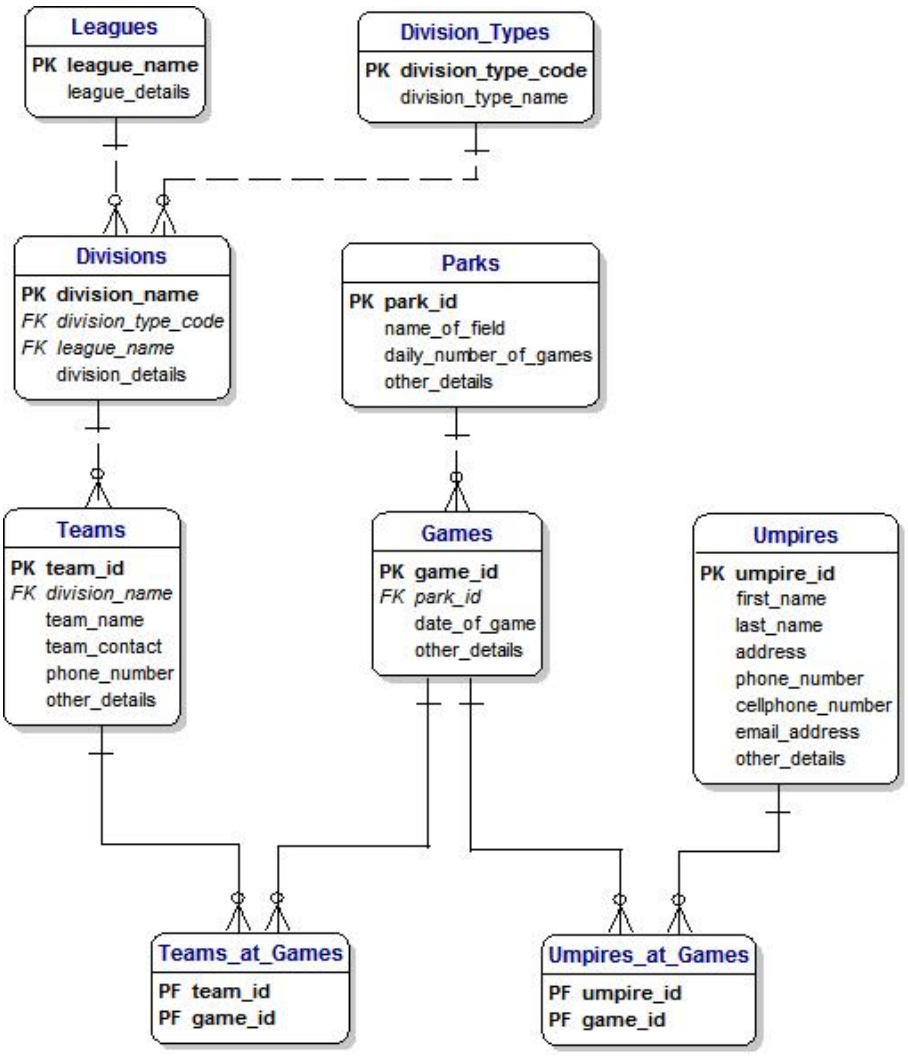


11.13 I Become a Baseball Umpire

Baseball Umpire Calling a Strike

Topics covered:

- Primary and Foreign Keys
- One-to-Many and Many-to-Many Relationships
- Reference Data



11.14 I Go to Hospital

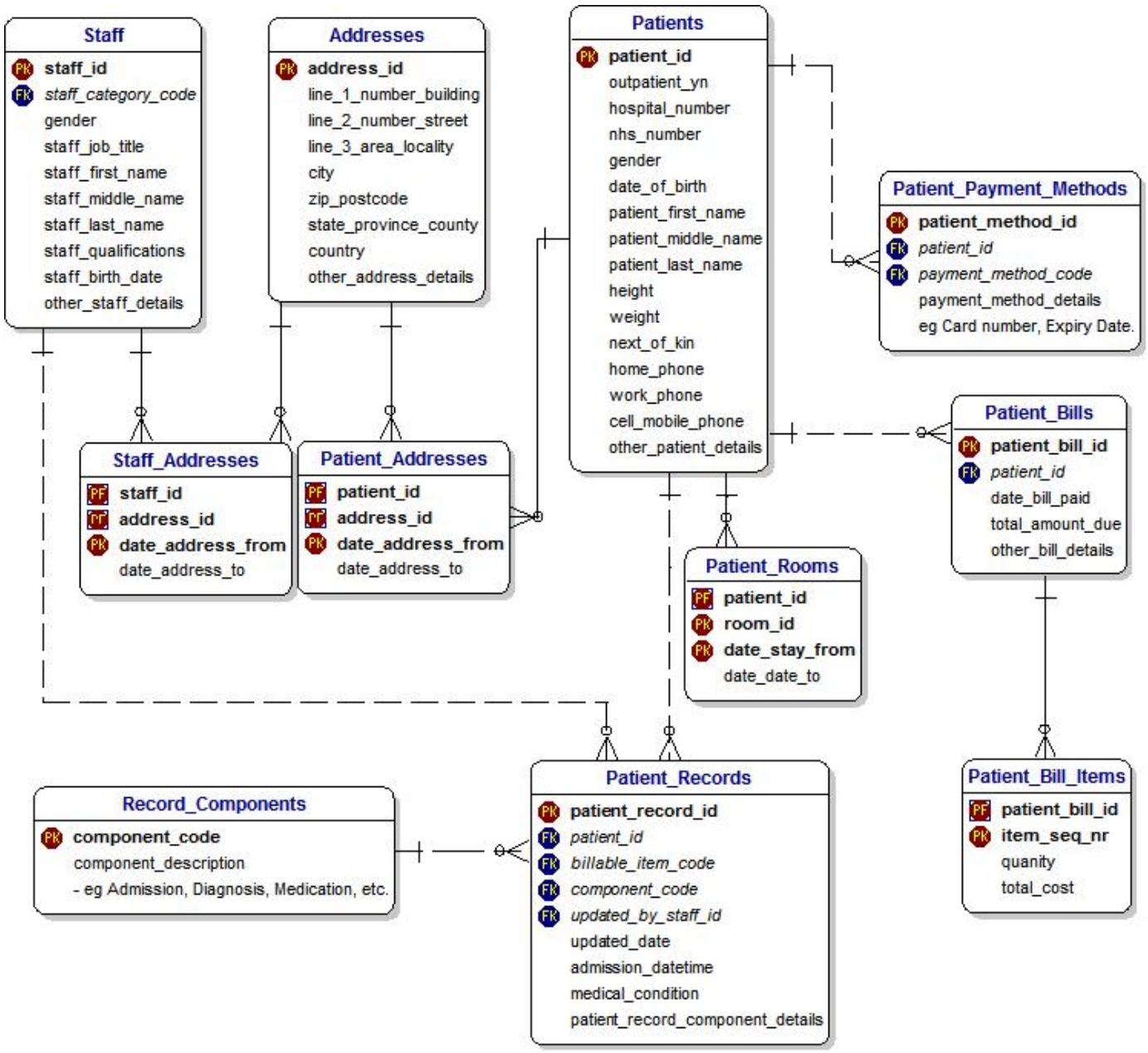
Massachusetts General Hospital, Boston, Mass. USA

Here we see that **patients** is the dominant table.
The rules are:

1. Each patient can have many addresses.
2. Each patient can have zero, one or many payment methods.

3. Each patient can have zero, one or many patient bills.
4. Each patient can be allocated to zero, one or many rooms.

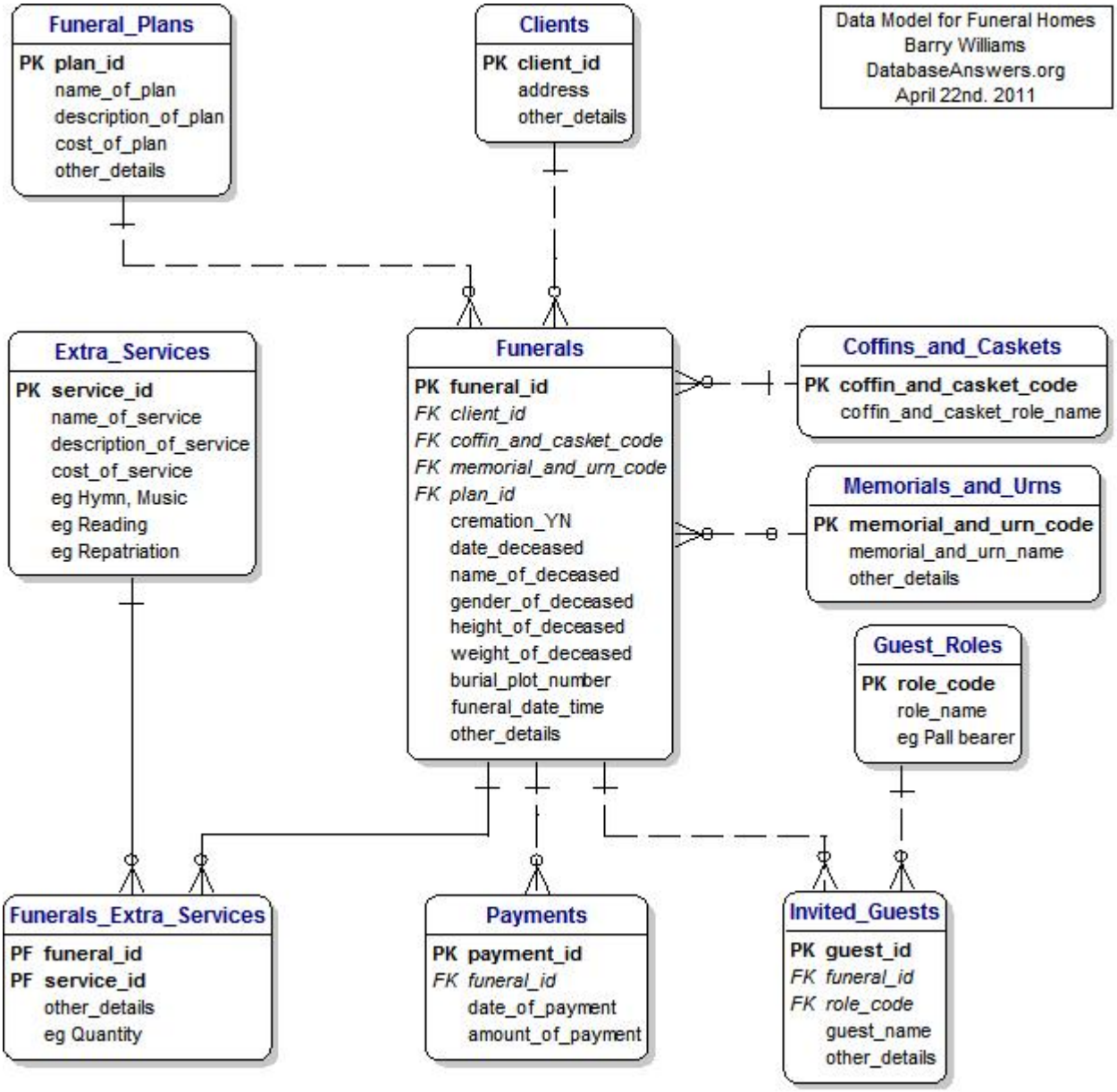
In the US, bills are an important part of a trip to the hospital. This is not the case in Europe and other parts of the world.



11.15 I Visit a Funeral Home

Kelly's Funeral Home, Ottawa, Ontario, Canada
The author lived in Ottawa for six great years.

Here we can see that the dominant thing is **funerals**. Every funeral must be associated with a client and has a funeral plan.



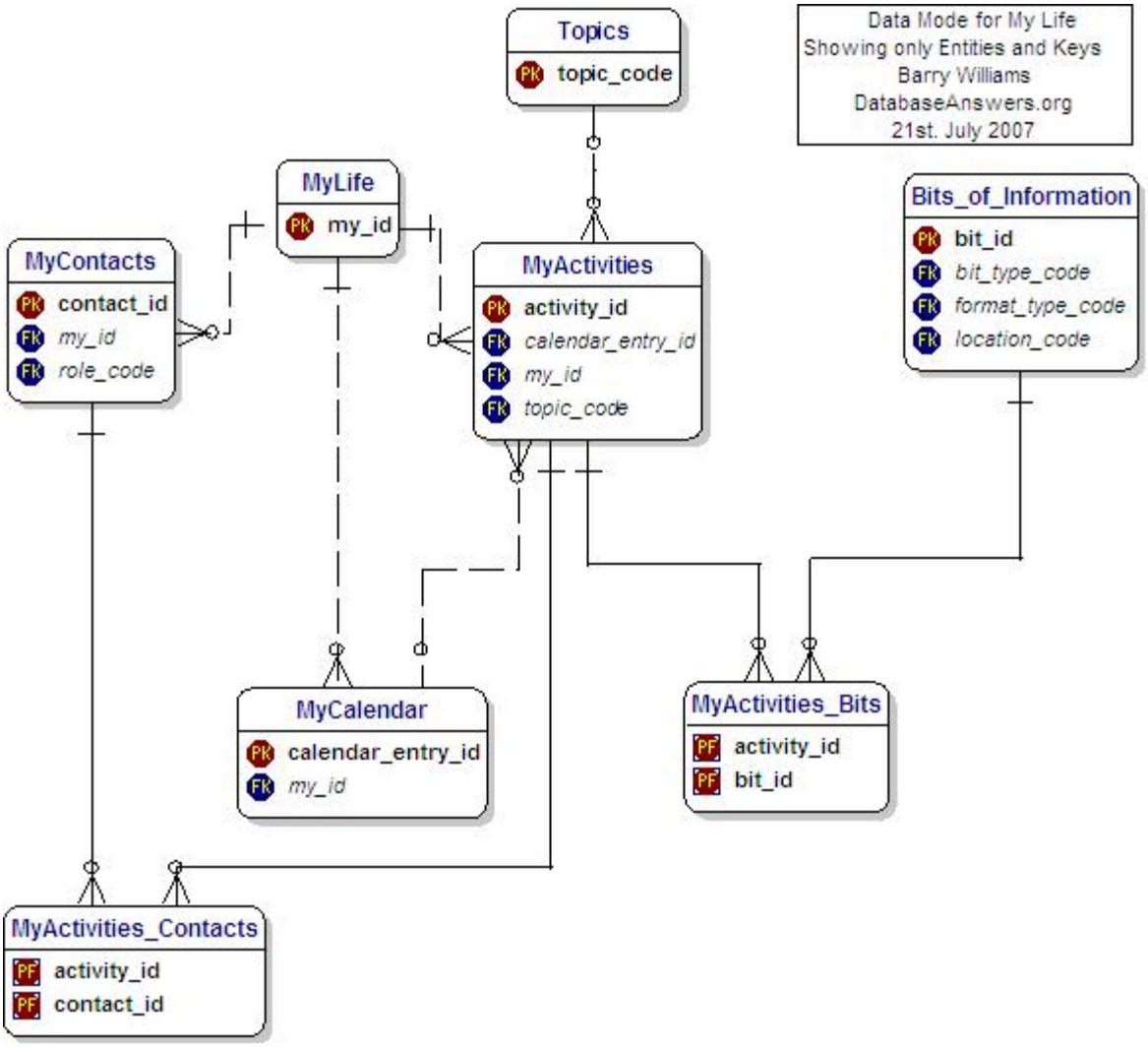
11.16 Events in my Life

Topics covered:

1. Primary Keys
2. Foreign Keys
3. One-to-Many Relationships
4. Many-to-Many Relationships
5. Reference Data

This is based on the 'My Life' data model:

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



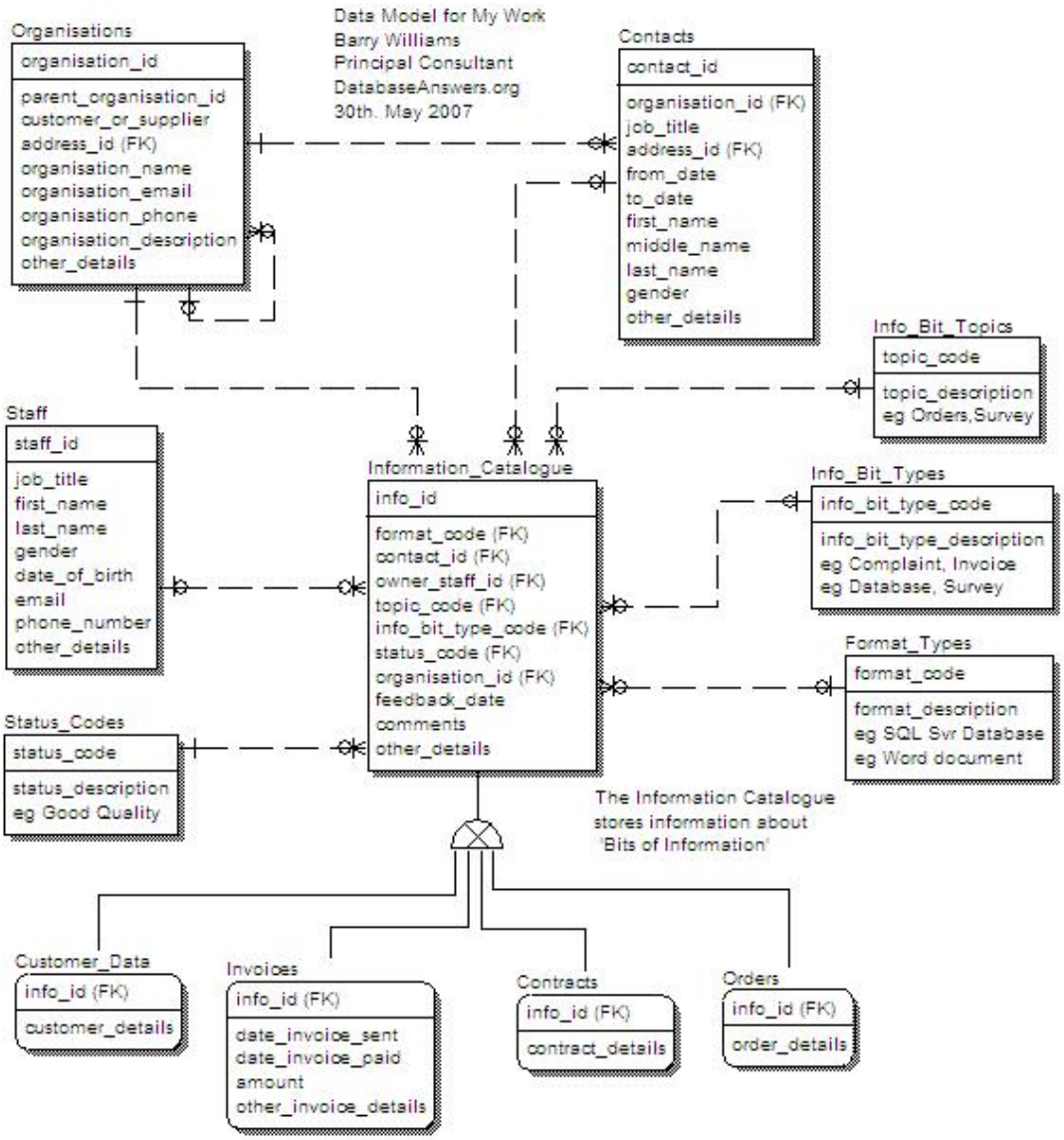
11.17 Events in my Work

Topics covered:

1. Primary Keys and Foreign Keys
2. One-to-Many and Many-to-Many Relationships
3. Hierarchies (e.g. Organizations) and Inheritance
4. Reference Data (e.g. Status Codes)

This is based on the 'My Work' data model:

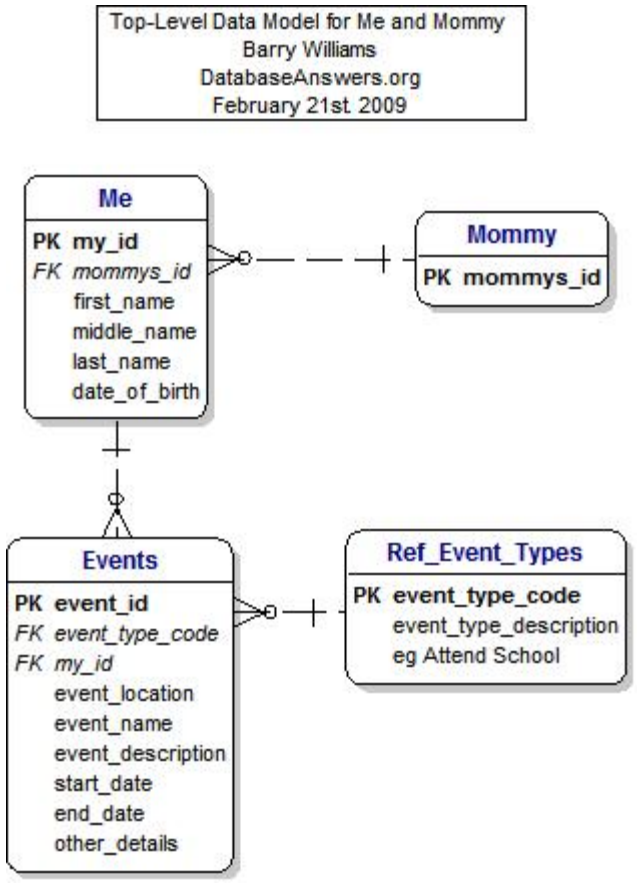
- http://www.databaseanswers.org/data_models/my_work/index.htm



11.18 Canonical Data Model

This is a beautifully simple model which shows that Mommy is the single constant factor and that 'My Life' is a series of events of different types.

A **canonical** data model is one that is stripped of everything superfluous.



11.19 What have we learned?

This chapter aims to bring together a number of data models that cover things that we are all familiar with. Our purpose in bringing them together is to present a range of topics that become increasingly complex in a way that should help us to understand this complexity.

(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/