Barry Williams

1. Introduction

This is Chapter 3 from Volume one of our Amazon Kindle book called 'Data Modelling by Example' –

- http://www.amazon.co.uk/Data-Modeling-Example-Volume-ebook/dp/B007I4Y4PS/ref=sr_1_1?s=digital-text&ie=UTF8&qid=1359889388&sr=1-1

When we design a Data Model we can create a Database from our Model.

Therefore, we consider Data Modelling to be the same as Database design.

2. Purpose

The purpose of this Tutorial is to help you understand the basics of Database design.

When you finish, you will be able to read and understand any Database design or Data Model that you see.

3. Database Design at Windsor Castle in England

3.1 Introduction

This first chapter is a tutorial on database design and data modeling for young people. It provides an introduction to data modeling that we hope you find interesting and easy to read.

It covers the basic concepts and has a very user-friendly approach, featuring a teddy bear and kitten creating a data model on a trip as tourists to Windsor Castle, which is just outside London, England.

You can find this chapter as a tutorial on the Database Answers Web site:

- http://www.databaseanswers.org/tutorial4_data_modeling_dimple_and_toby_visit_windsor_castle/index.htm

In this tutorial, we will follow two young tourists as they visit Windsor Castle and create a data model.

Our tourists are Dimple, a young girl, who likes sightseeing and ice cream and Toby, Dimple's older brother, who likes sightseeing and designing data models.

3.1.1 What is this?

This is a tutorial on data modeling for young people that represents a typical data modeling project and illustrates the basic principles involved.

3.1.2 Why is it important?

Data modeling is important because it is the foundation for so many activities:

It provides a vehicle for communication among a wide variety of interested parties, including management, developers, data analysts, DBAs and more.

A physical database can easily be generated from a data model using a commercial data modeling tool.

3.1.3 What Will I Learn?

You will learn:

How to create a data model, starting from scratch.

What a typical data model looks like.

3.2 Topics

In this chapter, we will cover some basic concepts in data modeling:

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

3.3 Let's Go to Windsor

[Dimple]: Toby, it's great being in London, which is so exciting and buzzing.

[Toby]: I'm glad you like it, Dimple. What would you like to do today?

[Dimple]: Toby, we have seen Buckingham Palace, where the Queen of the United Kingdom lives, and now I'd like to visit Windsor Castle, because it's one of the most popular tourist attractions in the UK, and it's just a short trip from London.

[Toby]: OK. Let's go...

We are starting from Buckingham Palace, where the Queen of the United Kingdom lives …

Toby and Dimple leave London and arrive at Windsor…

3.4 Arriving at Windsor

[Dimple] Wow, Toby, Windsor has a beautiful castle and here is a royal park with lots of deer.

[Toby] Yes, Dimple, and when we look around there are so many banks, cafes, pubs, restaurants, shops, wine bars and hospitals!

The other thing that we see when we look around is people - lots of people.

So we can start thinking about our data model.

3.5 Starting our Data Model

[Dimple]: How do we get started?

[Toby]: Well, we know that we have people and places.

The simplest start is to call all these places **establishments**.

Then we simply have different kinds of establishments.

And we have people - local people, tourists, students, people passing through, people working here, people here on business and so on.

[Dimple]: Hmmm - so how do we translate what we know to help us get started with our data model?

[Toby]: Let's start a diagram with people and establishments.

This simple diagram is going to grow into a data model.

3.6 Identifiers and Primary Keys

[Dimple]: Toby, I am one of these people so how do I create a unique identity for myself to make me different from everybody else?

[Toby]: We will give every person a **unique identifier** and every establishment its own unique identifier.

When we use these we call them **Primary Keys**, and show them in the diagram with a **PK** on the left-hand side.

[Dimple]: That sounds good, Toby, but I don't know what it means.

[Toby]: Well, Dimple, let's look at how we use these identifiers...

Lots of people visit establishments like Starbucks at Windsor ;0)

3.7 Relationships and Foreign Keys

[Toby]: Dimple, now we can add some interesting details because we know that one person can visit many establishments.

We also know that one establishment is visited by many tourists.

Then we call this a **many-to-many relationship** between people and establishments.

To make it easier for you to understand I have expanded the **many-to-many relationship** into two different things, which are called **one-to-many relationships**.

[Dimple]: So Toby, is that like saying that one person can make many visits to many establishments?

[Toby]: Yes, Dimple - that's great - and we can also say that one establishment can have visits from many people.

At this point, we can show how all these boxes are related, and that is a very big step, because it takes us to the idea of 'relationships'.

We can call these boxes **tables** - or *entities* if we want to speak to professional data modelers.

A table simply stores data about one particular kind of ‘Thing of Interest’.

For example, people or establishments.

Each record in a table will be identified by its own unique identifier, which we call the *primary key*.

It is not usually easy to find a specific item of data already in the table that will always be unique.

For example, in the United States, Social Security Numbers (SSNs) are supposed to be unique, but (for various legitimate reasons) that is not always the case.

Also, foreign visitors and tourists will not have SSNs.

Therefore, it is best practice to create a new field just for this purpose.

This will be what is called an **auto-increment** data type, which will be generated automatically by the Database Management System (DBMS) at run-time.

This is called a **surrogate key** and it does not have any other purpose.

It 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.

Now we can see how useful our identifiers can be because we can include the person and establishment identifiers in our Visits Table.

Then the Person_ID field becomes a link to a record for a person in the Person Table.

This link is what is called a **Foreign Key** and we can see it's shown with **'FK'** on the left-hand side.

3.8 Products and Product Types

[Dimple]: Toby, when we go into a shop we want to buy something.

And there are thousands and thousands of possibilities.

How do we deal with all that in our little data model?

[Toby]: Well Dimple, it's really quite easy. It's like all our modeling where we look for simple patterns that cover many situations.

[Dimple]: Hmm - I don't know what that means. Maybe if you showed me I might understand it.

[Toby]: OK.

Everything that we buy is called a **product**, and all we have to do is simply define the type of each product - such as a coffee, muffin or a newspaper.

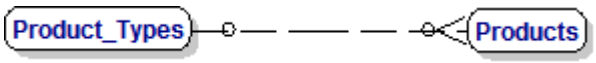
Then we draw a little box called *Products* and say that every product has a type.

In other words, there is a relationship between the *Products* and *Product_Types* boxes.

The lines are called **relationships** and they are very important in data modeling.

We are now creating an Entity-Relationship Diagram or "ERD".

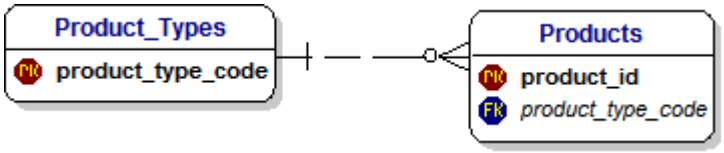
This diagram shows only a line for the relationship:



The symbol at the products end is called *crow's feet* and it shows the *many* end.

The short straight line at the Product_Types end shows the *one* end.

In other words, this line shows a one-to-many relationship.



Dimple, let me explain about the dotted line. It means that the relationship results in a 'Foreign Key' in the products table. This is shown by the 'FK' symbol next to the ***product_type_code*** field and it means that there is a link back to the Product_Types.

However, the primary key is only the Product_ID, and of course, this is shown by the 'PK' symbol next to the **Product_ID** field.

Later, when we talk about inheritance, we will use a straight line, in contrast to this dotted line here. This is to show that the foreign key field is also a primary key.

I have to say something a bit difficult about primary keys right now.

In the Products Table, we have to allow for a very large number of products being stored.

Therefore we use an ID field for the primary key.

We then create this ID field automatically as a number (called an auto-increment integer).

This number has no meaning and is simply used to identify each record uniquely among possibly millions or hundreds of millions.

However, things are different for 'type' fields.

These are what we call enumerated data and are typically **reference data**.

They are always relatively small in number and we choose a code for the primary key because we can create them and review them manually.

It also helps us to create a code that we can use and refer to, in contrast to the ID fields that have no meaning.

Typical examples would be:

Sizes – Small, Medium and Large where we are accustomed to seeing S,M and L.

Gender – Male and Female, where we use M and F.

This menu board at Starbucks shows lots of products.

We know that they are organized in groups, like food and drink, and each of these has more groups and so on, right down to the particular product, like caramel macchiato or a panini.

This top-down organization is called a **hierarchy** and appears all over the place.

Luckily we can show this very easily and neatly in our data model.

3.9 Products, Types and Product Hierarchies

[Dimple]: Toby, when we look closely at the menu board to try to decide what to order we can see lots of possibilities. But after a while we can see a pattern that helps us decide.

How do we deal with all that in our little data model?

[Toby]: Well Dimple, it's really quite easy.

We define something called a *hierarchy*.

Hierarchies are very common and simply mean any situation where there are parents, children, grandchildren and so on.

If we look at the Starbucks menu board on the right-hand side we can see a simple example of 'espresso' and under it a number of different drinks.

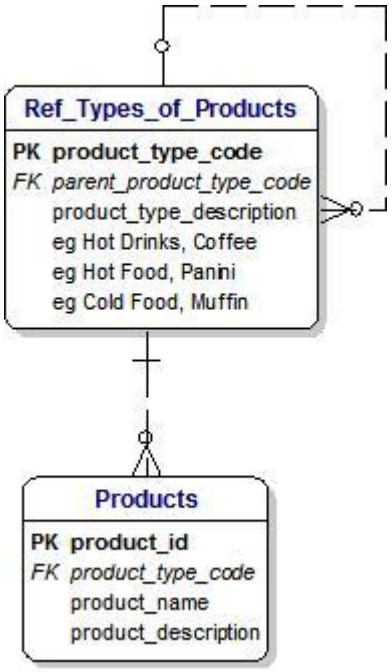
My favorite is caramel macchiato.

So in this case, the top-level of our hierarchy is a product category called espresso, and the next level down is a product called caramel macchiato.

[Dimple]: OK. That sounds OK.

[Toby]: Finally, we show this hierarchy by a dotted line in the top-right hand corner in the entity called 'Ref_Types_of_Products'.

This is formally called a *recursive* or *reflexive* relationship and is informally called **rabbit ears**.



3.10 Types of People

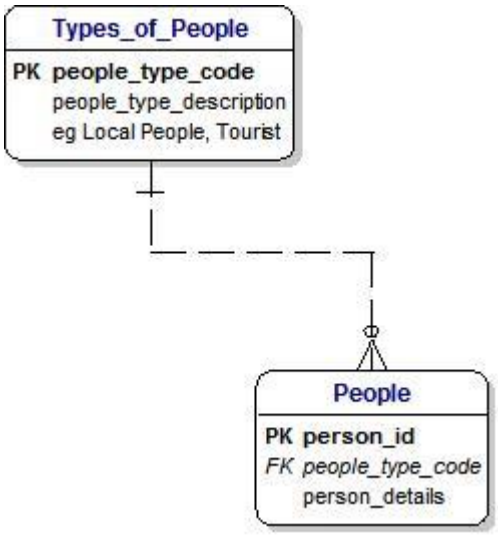
[Dimple]: Toby, that looks OK.

I guess we can deal with types of people the same way, can we?

[Toby]: Yes, Dimple, and types of establishments as well.

[Dimple]: OK, that sounds sensible. And do they use these identifiers in a database?

[Toby]: Yes, and what is even better is that the database will automatically generate a new unique identifier for you and your visits and purchases if you want to get a refund later.



3.11 Types of People and Establishments

[Dimple]: I see, Toby.

I guess we can deal with types of establishments the same way, can we?

[Toby]: Yes, Dimple.

[Dimple]: OK, that sounds sensible. And do they use these identifiers in a database?

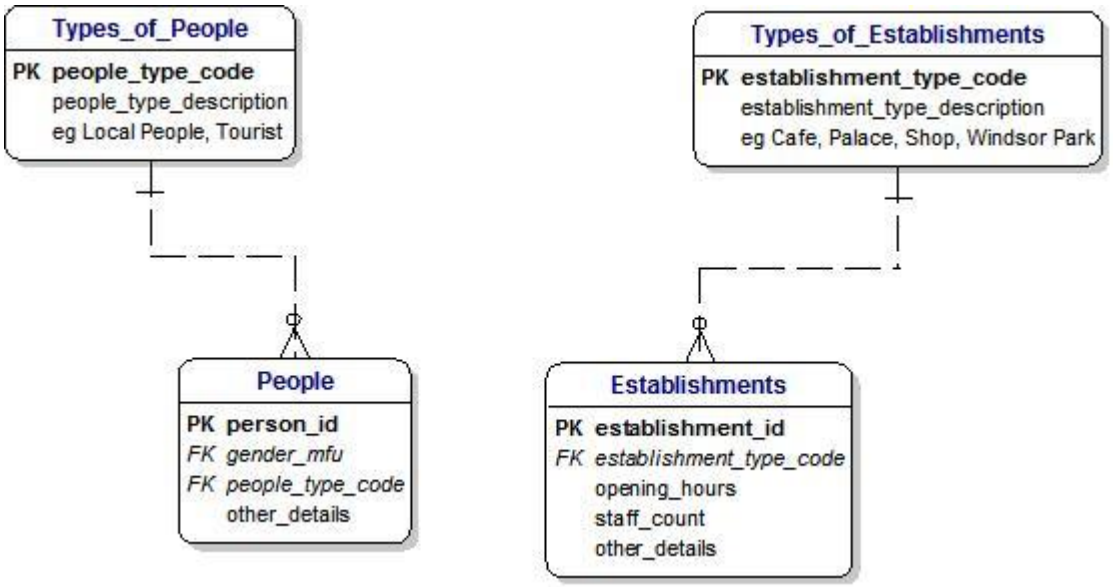
[Toby]: Yes, and we can use our new unique identifier for you and your visits and purchases in case we want to keep track of things.

Like maybe you want to get a refund later so we need to get your details from the database.

[Toby]: Before we move on, let's talk about establishments.

One special thing about Windsor is that it has a castle where the Queen lives and a very large royal park, where she keeps deer.

But when we think about these things, we find that we can simply fit them into our definition of establishments.



3.12 Visits and Purchases:

Here we can see many visitors to Windsor's Royal Shopping Arcade.

[Dimple]: Toby, with so many people, establishments and purchases how do they keep track of everything?

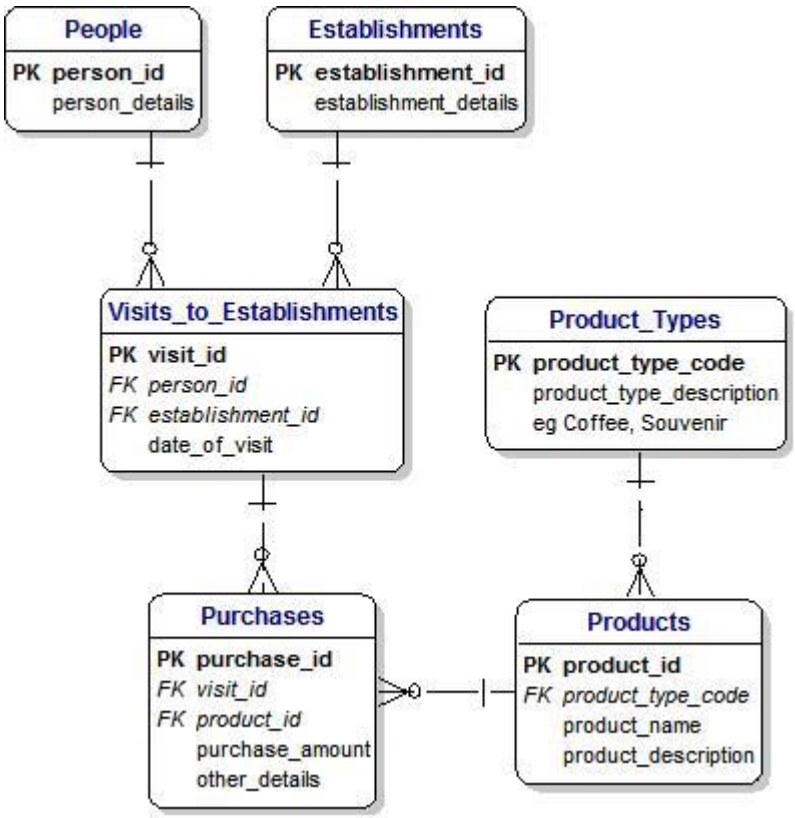
[Toby]: Well, Dimple, by this time, everything has its own identifier that is used wherever they need to keep track.

[Dimple]: OK, that sounds sensible. And do they use these identifiers in a database?

[Toby]: Yes, Dimple, and in this diagram, we can see that we can use the unique identifiers that are shown as 'PK,' for Primary Keys.

We can see that we have a PK for every entity or table so we can be pretty sure we can get from any table to any other table.

This is called *navigating* around the data model and is a good test for a well-designed data model.



3.13 People and Inheritance

[Toby]: Dimple, let's take a closer look at the different types of people we can find in Windsor.

[Dimple]: OK, Toby. I hope I don't have to think too much because I might get a headache?

[Toby]: No, Dimple, I will do the thinking and talking and all you have to do is nod your head when you understand.

[Dimple]: OK, Toby. I promise to do that.

[Toby]: We already said that we have local people and tourists.

There are always lots and lots of people visiting Windsor Castle.

When we look at this picture, we can see ceremonial guards in ceremonial red uniforms, and a big crowd, with mainly tourists but also staff in shops responsible for controlling the crowd, tourists, local people and so on.

Some of these local people are shoppers and some of them will be working in the shops.

We will call the workers **staff** and we know different things about them than the things we know about the tourists.

For example, we will probably know the gender of everybody just by looking at them.

For staff, we will usually also know their date of birth and their home address.

In data modeling we have a very powerful approach that we call **inheritance** that we can use here.

If we want to describe this in English, we would say that staff inherit the People_Type_Code and gender from the parent entity of people, and in addition, they have a date of birth and home address.

For tourists, we don't know much, except for the date of their visit, and maybe, if they buy something in a shop using a credit card, then the shop would know the credit card details.

For the ceremonial guards in red uniforms, we can tell their rank by looking at their uniform and maybe it would also tell us which unit of the army they belong to.

Does that make sense, Dimple?

[Dimple]: I think so, Toby.

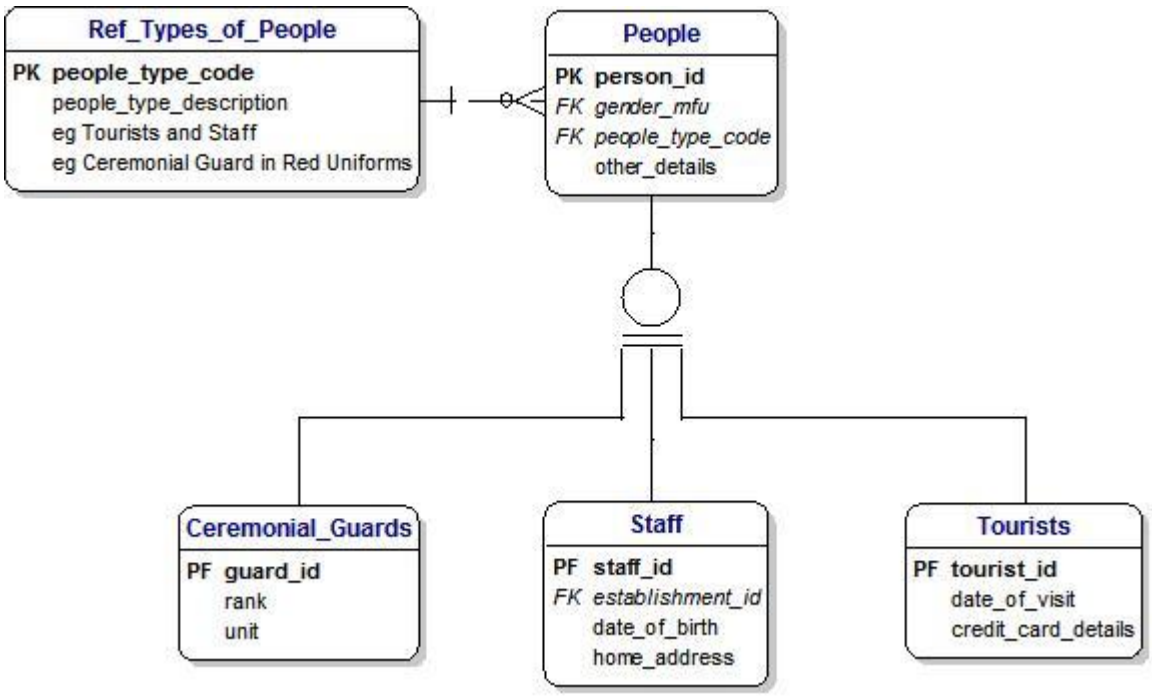
Is it like saying that we inherit having two arms and two legs from our parents because they have two arms and two legs, but that we have also have things that are just us?

[Toby]: Yes, Dimple - that's great - let's take a break and do some shopping!

[Dimple]: I like the sound of that, Toby. Can I have an ice cream?

[Toby]: Yes, of course, Dimple – this diagram shows we are doing well.

It show inheritance between people and the three different types of people:



We can see a field marked as '**PF'** in the three tables for ceremonial guards, staff and tourists.

This is unusual because it means a field that is a **P**rimary Key in the three tables and also a **F**oreign Key to the People Table.

Therefore, if your first record was a ceremonial guard, then we would have a record in the People Table with a Person_ID of 1 and a record in the ceremonial guard with a Guard_ID of 2.

Similarly, if our second record was a member of staff, we would have a record in the People Table with a Person_ID of 2 and a record in the Staff Table with a Staff_ID of 4.

3.14 Staff, Establishments and Derived Fields

[Dimple]: Toby, how do we specify that staff must work in some establishment?

[Toby]: Dimple, that's a very good question.

Fortunately, the answer is very easy.

We add a one-to-many relationship between the staff and establishment entities.

In English, we would say that every member of staff must work in one establishment and every establishment can employ many members of staff.

In the diagram, we show this with a **foreign key** by the Establishment_ID field in the staff entity.

So if we look closely at the staff entity, we will see '**FK**' by the Establishment_ID field.

[Dimple]: OK, that sounds good, and I can see how the identifiers are very important.

[Toby]: I am glad to hear it, Dimple.

There is one more thing I have to say.

We are learning data modeling and one important thing about data modeling is that it has to follow a set of **rules**.

These rules help us to produce good data models and so they are very important.

One of the rules is that we cannot include any bits of data that can be derived from any other bits of data.

For example, we usually want to know how many people work in a shop or cafe.

Therefore we include a **staff count** field with the establishment.

But when it comes to finding the value that goes in here, we will count the records in the Staff Table for each establishment.

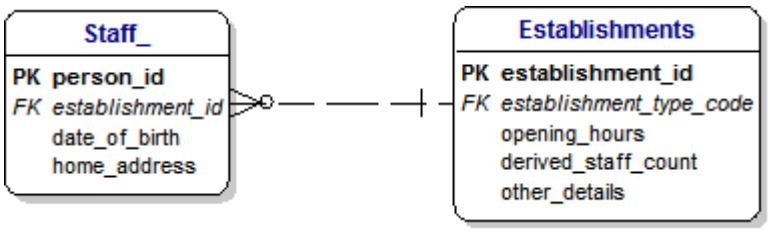
Therefore, it's a **derived field** and we call it a name that starts with 'derived_' to make things clear.

This is because, according to the rules, we should not include derived fields in our data model at this early stage.

I have shown it here simply as an example because it is a situation that occurs quite often so it’s good to recognize it when you see it.

Does that sound sensible, Dimple?

[Dimple]: I suppose so, Toby. But I've got a headache, can we go for an ice cream now?



3.15 Reference Data

[Toby]: Dimple, you can see that I am using a Gender Table and People Types Table.

I have given them both names that begin with 'ref_' to make it clear that they are reference data.

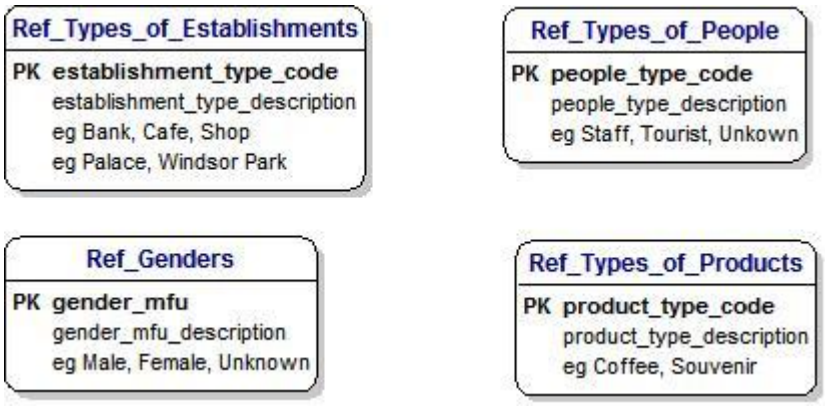
This means that the values don't change much and I can use them to define what the valid values can be.

This is a technique that professional data modelers use but we don't need to worry about it today.

[Dimple]: I'm glad to hear it, Toby!

Although it isn’t difficult to understand and it seems like a good idea.

[Toby]: In our small example, we have only four kinds of reference data altogether - gender, types of establishment, people and products.



3.16 Bringing it all Together

[Toby]: Dimple, if we bring together everything we have talked about, we will see that we have quite a good data model that any professional would be proud of.

[Dimple]: OK, Toby. Do you think I will understand it?

[Toby]: Let me help you by making a list of the **business rules** for our model:

People can be either ceremonial guards, staff or tourists.

There are a number of establishments of different types.

Tourists can make visits to establishments and make purchases.

Staff assist the tourists when they make a purchase.

A purchase involves one product.

[Toby]: OK, Dimple - we have a very nice data model and now we can take the break I promised you.

[Dimple]: That's great, Toby - can I have an ice cream?

[Toby]: Sure, but before we do I should say something about **PF**, which appears in the Staff Table.

It's unusual and it's called **PF** because it means a field which is a **P**rimary Key in the Staff Table and a **F**oreign Key to the People Table.

[Dimple]: Hmmm, I've got a headache, Toby - can we please go and get an ice cream?

[Toby]: OK, Dimple. You've been a very good girl and you deserve a break.

You can admire what we have created, which is this very professional-looking data model.

3.17 Top-Level Model with Names Only

We can show our data model at the top-level, showing only the names of the 'things of interest,' which we call entities or tables if we are thinking about a database.

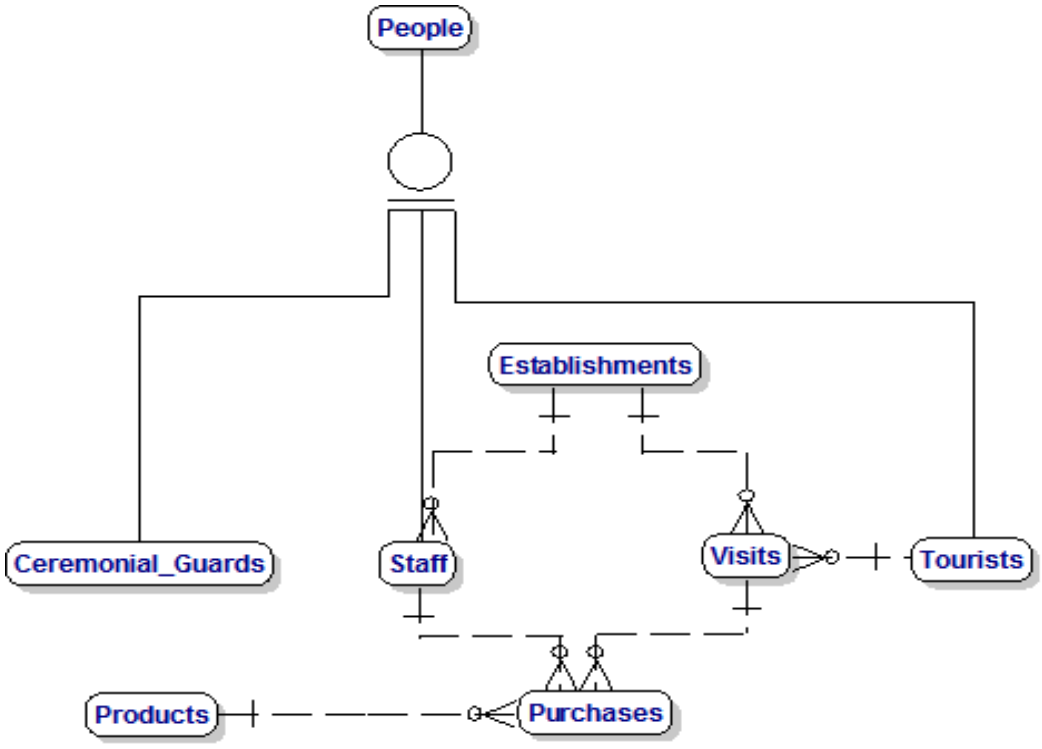
This is suitable for explaining what we saw in Windsor to our family or friends.

If we wanted to describe it, we could simply say:

There are lots of people in Windsor, including ceremonial guards, staff and tourists.

There are also lots of establishments, like shops and the castle.

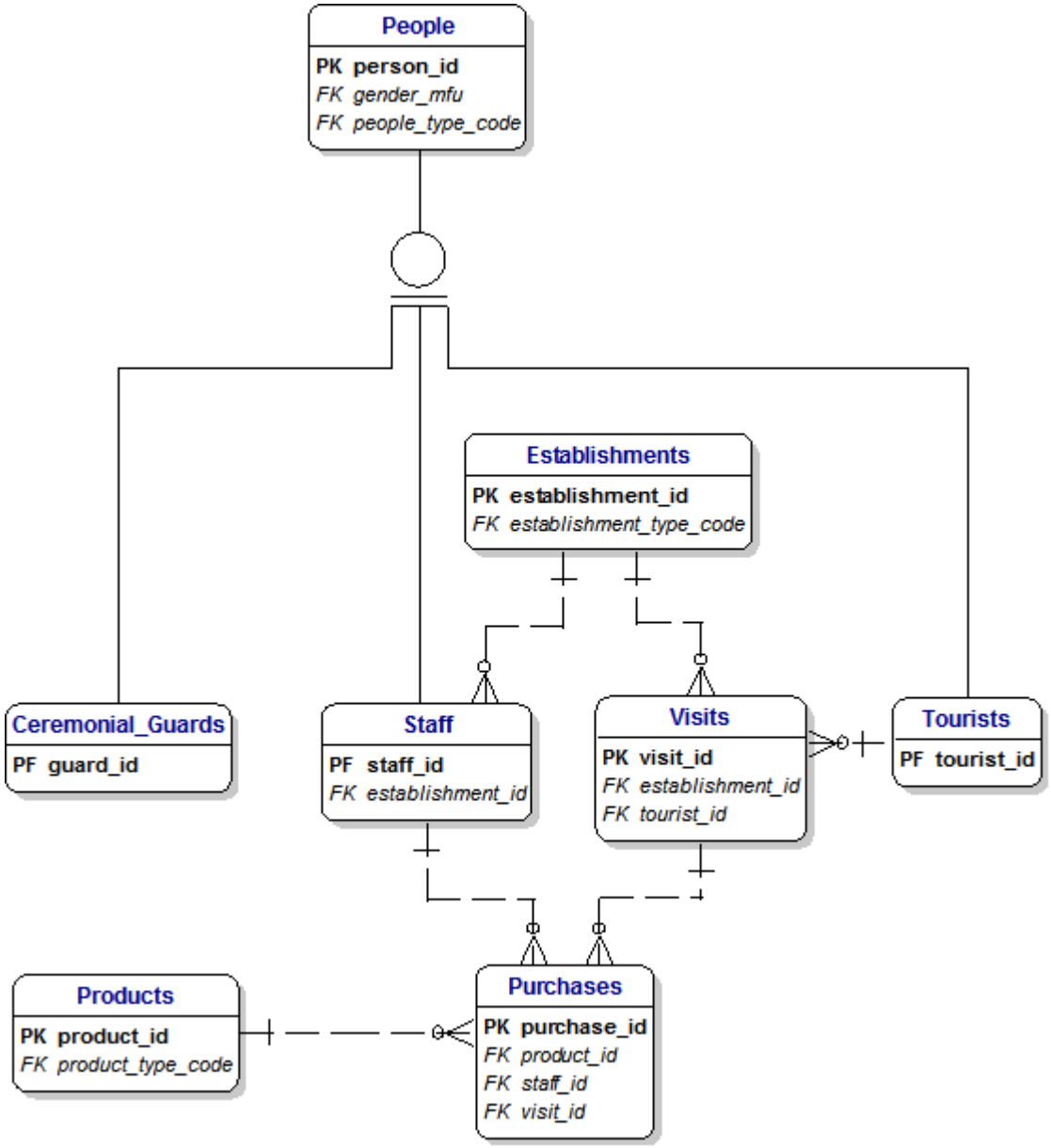
Tourists made visits to establishments where they made purchases of products.



3.18 Top-Level Model with Key Fields

This is what our data model looks like if we show key fields only and leave out the Reference Data Tables.

This level of display is suitable if we want to confirm to each other how the tables (or entities) are related.

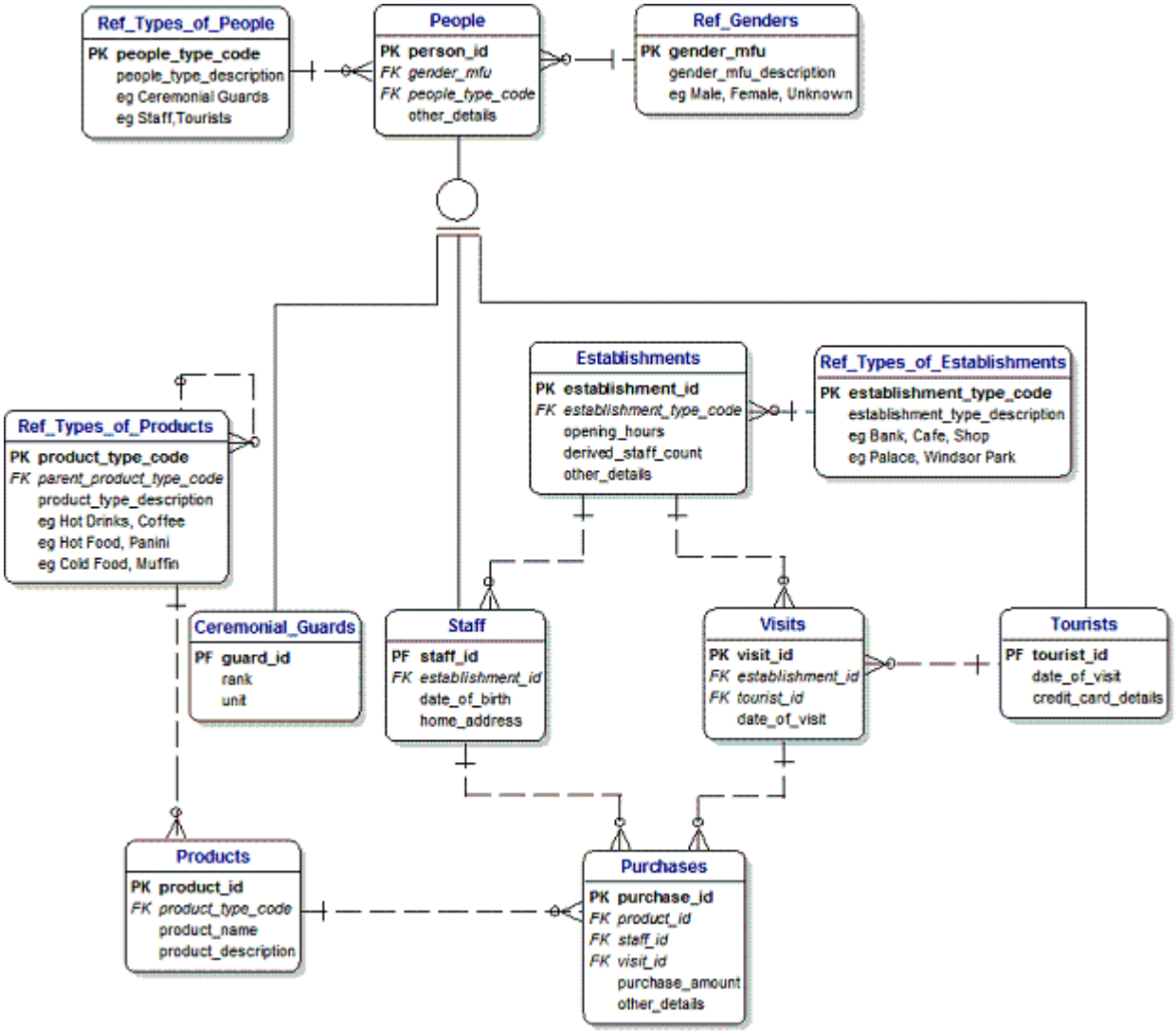


3.19 Top-Level Model with all Details

Finally, this is what our data model looks like if we show the key fields, all the data items only and the Reference Data Tables.

You can see that the amount of detail involved makes it more difficult to understand what's going on and to identify what is important.

This level of display is suitable if we want to talk about details and develop a database from our data model.



3.20 Ice Cream

[Toby]: Dimple, I've got some wonderful news for you.

[Dimple]: I'm glad to hear it, Toby - what is it?

[Toby]: I have found your favorite Baskin-Robbins ice cream here in Windsor ;)

[Dimple]: Toby, are you teasing me?

[Toby]: No, Dimple - look, there it is across the road from Windsor Castle!

[Dimple]: Wow - that's great, so I can have my favorite butter pecan ice cream.

3.21 What have we learned?

In this chapter, we have learned how to think like a data modeler and how to gradually put together a data model in our heads.

We know that if we get in the habit of doing this regularly it gets easier and more natural and soon we will be seeing the world around us as pieces of a data model that we can fit together like a jigsaw puzzle.