Database Answers

Welcome to my Tutorial in Ge

INTRODUCTION :

This Tutorial is based on designing an **Order Processing Database** to deal with Customers placing Orders, firstly at Starbucks and then with Amazon.
I have done it this way because Starbucks is quite simple so it provides us with a good starting-point.

Then I look at the questions of extending this simple Database Design to deal with the more complex situation with Amazon.
I hope you find this Tutorial interesting and helpful.
Please email me and let me know.

GETTING STARTED :

- We stand outside Starbucks, planning to go in.
- We can see Customers, and inside we can anticipate seeing a wide range of things to eat and drink.
- In this Tutorial, I will start by designing a Database for Starbucks and then extend it to include Amazon.
- **My Approach** has three Steps :-
  1. Establish the Scope of the Database
  2. Identify the 'Things of Interest' that are within the Scope, (called **Entities** or Tables in a Database).
  3. Determine the **Relationships** between them.
- At the end of this Tutorial, We have will produced a diagram of the Database Design, which is commonly referred as an 'Entity-Relationship Diagram', or **ERD**
- During the Tutorial, I will refer to Tables as the 'Things of Interest', whereas Data Modellers would refer to Entities, but I think that 'Tables' is more User-Friendly.

DECIDING THE SCOPE OF OUR DATABASE

- When we step inside, we see that Starbucks sells a wide range
to decide which of them should be included in our Database.
- Right now, we are interested only in something to eat and som
- Therefore, all the mugs and other items shown in this picture
Scope of our Database, and are not **'Things of Interest'.**

LOOKING AT THE PRODUCTS :

- Turning away from the mugs, we can see a display of Food and Drink, and that is what we are looking for.
- Right now, we are thinking about getting something to eat and something to drink.
- When we go in as an ordinary Customer, we are thinking about what we would like to eat and drink.
- However, when we go in as a Database Designer, we look at the display from a different point of view.
- We want to discover the structures in the data and how they are related.

Starbucks Cash Customers (version 1)
Barry Williams
1st. August 2008
DatabaseAnswers.org

Customers_version1

COMMENTS :

- Starbucks (and Retailers in general) know nothing about their

COMMENTS :

- Retailers can find out the Name,Address and other details about Customers who pay with a Card.
- Reference data is an important topic and I will come back to it later.
- **Primary Keys**

1. You will notice that the first field in the Customers_version2 Table is the Customer_id.
2. It has a 'PK' symbol beside it, which indicates that it is the Primary Key for the Table.
3. The Primary Key is very important and is the way that we can recognise each individual record in the Table.
4. In the case of Starbucks, they will have hundreds of thousands of Customers over a period of time. Therefore we need a way of automatically generating this ID field.
5. The way we do this is to use what is called an **'Auto-Increment' field**. This means that every time you add a new Record, a new Customer ID is automatically generated.
6. Every DBMS has an Auto-Increment field. It is called a 'UniqueIdentifier' data type in Access, an 'Identity' in SQL Server and a 'Sequence' in Oracle.

- **Foreign Keys**

1. When this Primary Key is used in another Table, it is referred to as a'Foreign Key'.
2. We can see a good exapmple 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**

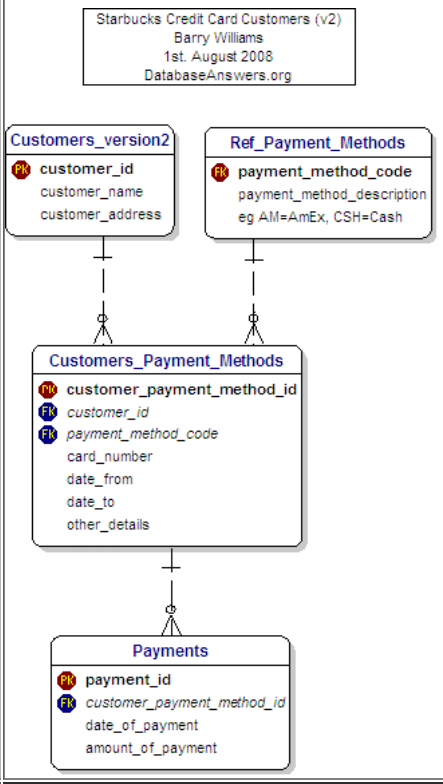
1. A Foreign Key is usually **mandatory**, in other words, a value for a customer_id in the Customers_Payment_Methods Table must correspond to an actual value of the customer_id in the Customers_Version_1 Table.
2. This is shown in the diagram by the short straight line at the end of the dotted loine close to the Customers Table.

- **Optional Key Fields**

1. Not every Customer will have a Payment_Method. In general, they would but we need to allow for situations where Customers change their minds and don't buy anything.
2. In other words, we would say that the Relationship is **optional** at the Customers_Payment_Methods Table end.
3. This is shown by the little 'O' at that end of the Relationship dotted line.

- **One-to-Many Relationships**

1. A Customer can have more than one Payment_Method, for example, American Express or Cash.
2. In other words, we would say that the Relationship is **optional** at the Customers_Payment_Methods Table end.
3. This is shown by the symbol that has three small lines at that end of the Relationship dotted line, which is referred to as **Crow's Feet**.



COMMENTS :

- We can make a start on the Product hierarchy by looking at th

- At the top level, we have :-

1. Coffeehouse Favourites
2. Expresso
3. Frappuccino

I always like to put things in alphabetical order as soon as possil

Then looking more closely, we can see the Products listed under hand corner, are as follows :-

Blended Drinks

- Caramel Cream
- Chocolate CremeExpresso
- Srawberries and Cream
- Vanilla Cream

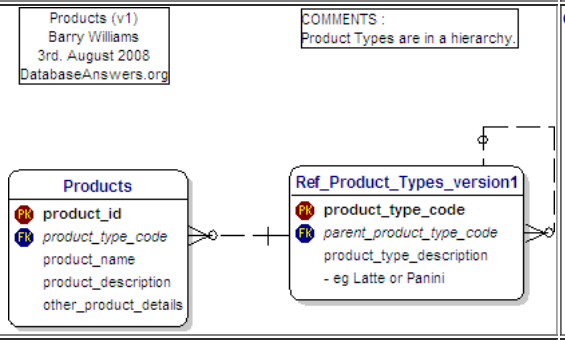
At this point, we have defined three levels within our Product Hi

- The top level 1

1. Coffeehouse Favourites
2. Expresso
3. Frappuccino
   - Blended Drinks
     1. Caramel Cream
     2. Chocolate CremeExpresso
     3. Srawberries and Cream

4. Vanilla Cream

Then we can see that Caramel Cream, Chocolate CremeExpresso Vanilla Cream all have a common Parent Product, which is 'Blen



COMMENTS :

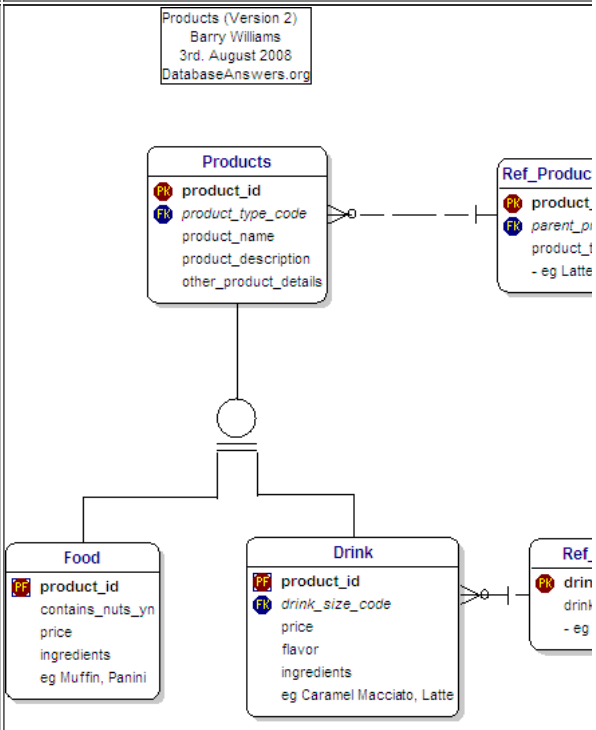
- This diagram shows how the hierarchies of Products and Produ discussed are shown in our **Entity-Relationship Diagram**.

COMMENTS :

- Food and Drink are specific examples of the more general Thing called a Product.
- They inherit some common attributes from the Product, and also have some of their own.
- For example, Food can contain Nuts but Drink do not contain nuts, but both have a Product Name.

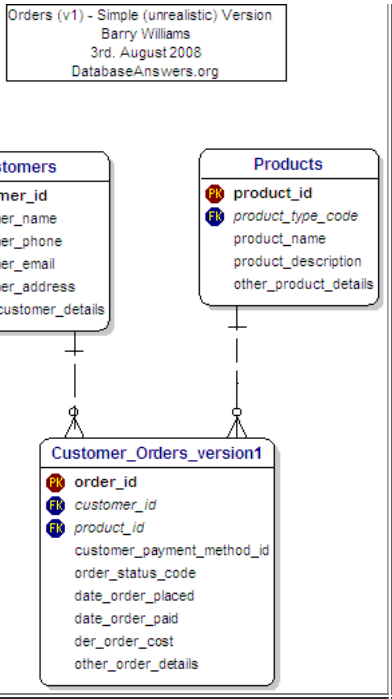
- **Rabbits Ears**
- You will notice that the table called 'Product_Types_v1' has a dotted line coming out on the right-hand side and going back in again on the top-right corner.
- Data Analysts call this a Reflexive Relationship, or informally, simply 'Rabbits Ears'.
- In plain English, we would say 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 'Miscelleneous 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 Starbucks, take a look at the borad behind the counter and try to decide how you design the Products area of the Database.
- 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.
- It's important to think through this level of detail otherwise you will get caught out somewhere down the line when real data can't be stored in your Database.
- In practice, it's smart to have a checklist of things like this to run through before you expose the first draft of your new Database design to a critical audience.
- It's good practice even if you don't have a critical audience because otherwise your design will need to be corrected.
- This topic is discussed again in the final Slide that looks at Deliveries.
- If you find this kind of thing interesting then maybe you would be happy earning a living as a Database specialist.
- If not, then keep well away from ti, and make sure your livelihood sdoesn't depened on your abaility to produce a well-design ed Database.
- You can trust me when I say that if you don't like it, or have an aptitude for it, it quickly becomes a chore that you would hate, because it requires concentrated thought and attention for ;prolonged periods of time.
- You could think of it **like solving crosswords** eight hours a day, five days a week.

- **Hierarchies in Products and Product Types**
- It's important to understand the difference between these two things, which is that the Product Type Hierarchy defines the Category names and levels, while the Product Hierarchy defines where specific Products fit into the Product Type Hierarchy.



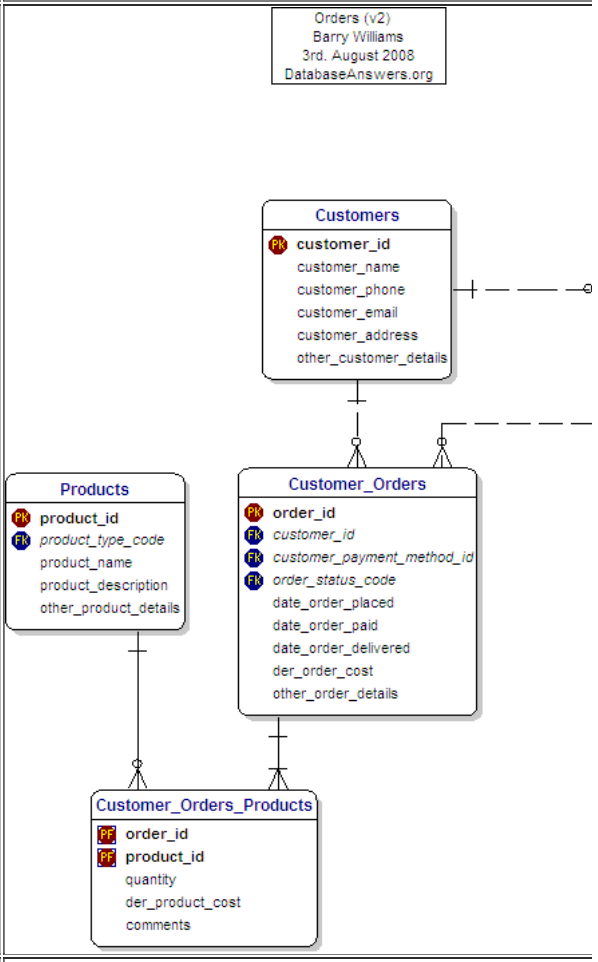
COMMENTS :

- This is a very simple (and unrealistic) Diagram.
- It shows that Customers order Products,such as Latte and a Mu



COMMENTS :

- This is a realistic Diagram showing the detail necessary to correctly model Customers, Products and Order details.

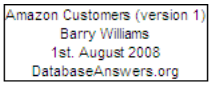


1. **What have we found in Starbucks ?**
   - Customers
   - Orders
   - Products
     - Food
     - Drink

Now we move on to Amazon ...

COMMENTS :

- Amazon Customers need to give their Delivery Address, their I Home Address. These can all be different.
- We could simply include them all in the Customer record, as w Version 1. However, this looks ugly. It also breaks one of the ve Database Thoery, which is that 'Repeating Groups are not allowe clearly Repeating Groups, and so we must find another way to re

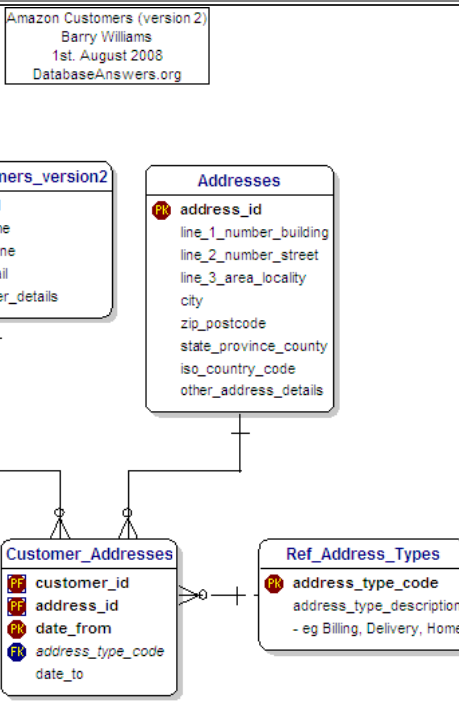


COMMENTS :

- And this is how we do it.
- We have a separate Address Table, which allows us to have more than one Address for any Customer very easily.
- This design also has some other benefits :-

1. We can accomodate 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.
2. With a separate table of Addresses, we can easily use commercial software to validate our Addresses. To find this kind of software, simply Google for "Address Validation Software". I have 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.

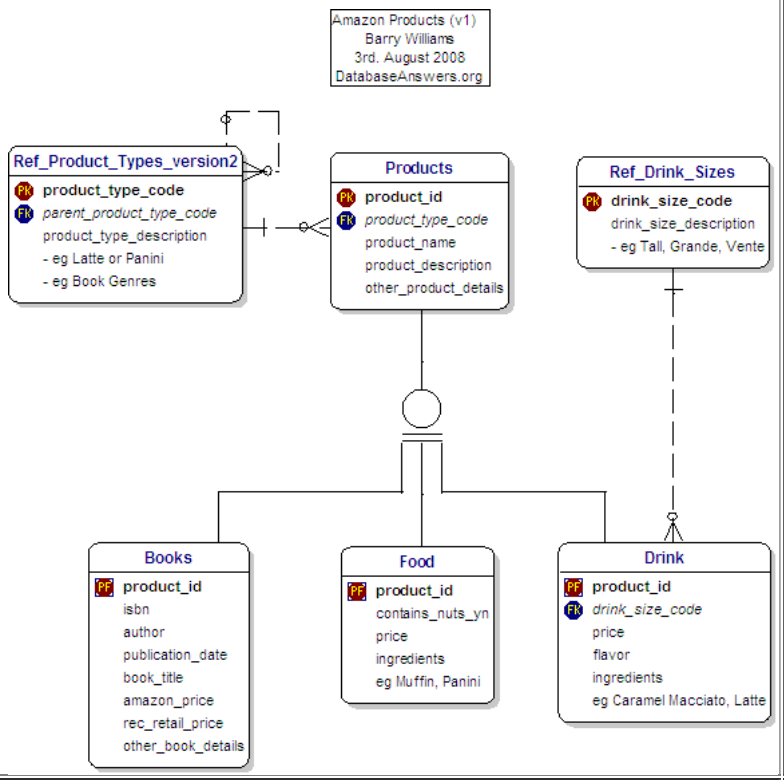
- **Reference Data**
  - This diagram shows Address Types, which are an example of Reference Data.
  - This kind of data has the following characteristics :-
    1. it doesn't change very much
    2. it has a relatively small number of values, usually less than a few dozen and never more than a few hundred.
    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.
  - **Standards**
    - In the Address Table, you will see a field called 'iso_country_codes'.
    - iso stands for the 'International Standards Organisation'.
    - Where possible, it's always good to use national or international standaRds.



COMMENTS :

- Different kinds of Products, like Books, are added quite easily.
- We simply add different kinds of Products, and I have shown E
- You will see that the attibutes for Books are specific to Books
- For example, an Author and an ISBN.
- Because all Products are handled in a Database in a similar wa to extend its range of Products.

- **Inheritance**
- The unusual symbol in the middle of the diagram, composed o underneath it is how Inheritance is shown using the particular Da using,which is called Dezign.
- Inheritance is a very important topic when you are designing a
- In plain English, we would say that Inheritance occurs where a between Things of Interest (or Entities).
- You can ask the simple **'Is-a' question** - in this case, if we as clearly the answer is 'Yes' so we think there is an Inheritance rel
- In the example of Inheritance shown in this diagram, we can s and Descriptions. Therefore, Books, Food and Drink will inherit t parent Product.
- However, each type of Product will have specific characteristic



other types of Products. For example, Books have ISBNs and Aut not.

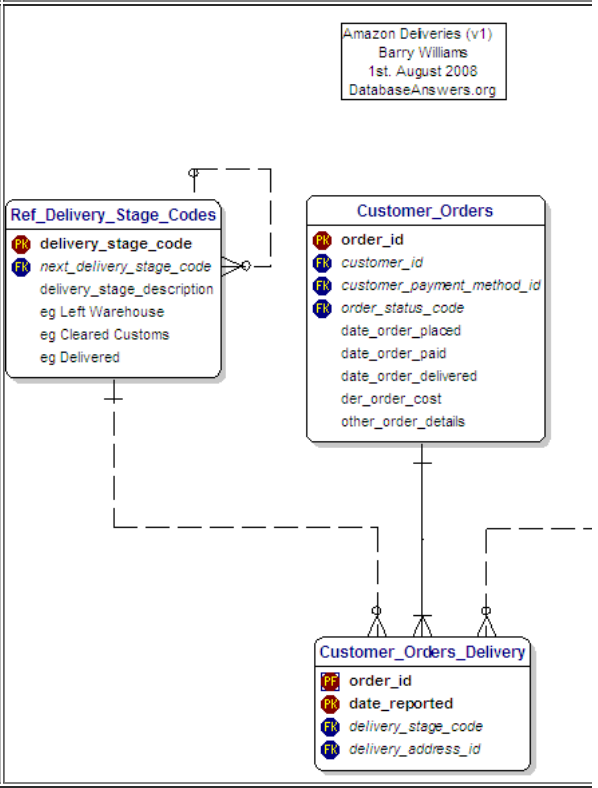
- One of the important things in your Database design is to be s Inheritance relationships.
- However, from many years of experince as a DBA, I should pc blurred in a real physical Database because it can be clumsy to i
- I sometimes find myself showing Inheritance in a Logical Data when I design the Physical Database, which is what ultimately be

- **There are three different approaches to implementing Inh**

1. Implement nthe design as you see it , in other words, have a Product Entity in this diagram.
2. Add all attributes in the Product Super-Type to the Books, Fo end up with three Tables. This is appealing because it seems more difficulty when it comes to development.
3. Add all attributes from the Books, Food and Drink Sub-Types This results in one large Table, where two-thirds of all Attribu record.
   However, software development can be easier, depending on followed. Currently, wasted space is not such an issue, so this

COMMENTS :

- Although Deliveries are complex in the real world, it is easy to add them to our Database design.
- We simply add a Reference Data Table of Delivery Status Codes and one Table for tracking Order Deliveries.
- You will notice that the name of this Table is 'Customer_Orders_Delivery', which follows our convention of combining the names of Tables that are combined in the data that is stored in the Table.
- **Rabbits Ears**
- You will notice that the table called 'Ref_Delivery_Stage_Codes' 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 'Rabbits 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 Delivery Stage is related to another one.
- When you buy a product from Amazon, you can track its progress as it leaves the Warehouse, clears Customs and is finally delivered to you.
- 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 'Next Delivery Stage Code' is optional, because the last Stage in Delivery will not have a Next one.
- It's important to think through this level of detail otherwise you will get caught out somewhere down the line when real data can't be stored in your Database.
- In practice, it's smart to have a checklist of things like this to run through before you expose the first draft of your new Database design to a critical audience.
- It's good practice even if you don't have a critical audience because otherwise your design will need to be corrected.



Finally, here's a Summary of the Rules for designing a Database

1. Define the **Scope** of your Database.
2. Define the **"Things of Interest"**,(e.g. **Customers** and **Order**), that are within Scope.
3. Establish how these Things are related and write down the 'Business Rules'.
   For example, "A **Customer** can have zero, one or many **Addresses**.
4. Determine what else you know about these Things.
   For example, "Books have an ISBN and one or many **Authors**
5. Identify the Reference Data, such as **Address Types** and **Product Types**.

- You need to define a Primary Key for all Tables.
- For Reference Tables, use the'Code' as the Key, often with only one other field, which is the Description field.
- For all other Data, you can use a generated number as the Primary Key.

This has some major benefits, for example, it provides flexibility, and it's really the only choice for a Database supporting a Web Site.

- **How to Validate your Database Design**

t's always useful to validate your design.
One good way is to talk through it with someone who knopw a littlw about what you are doing, or about Amazon or Stabuckls !!
Another good way is to check that you will be able to store actual data in your new Database.

- Firstly, obtain a small set of Sample Data by thinking about your last trip to Starbucks, or the last book you bought from Amazon.
- Confirm the first draft of the Database design against the Sample Data.
- Obtain from the users some representative enquiries for the Database,
e.g. "How many Hot Drinks are on offer in Starbucks ?"
- Review Code or Type Data which is (more or less) constant, which can be classified as **Reference Data**.
Look for external standards which can be national or international.
For example, Currency or Country Codes might have ISO Codes.
- Finally, define User Scenarios and step through them with some sample data to check that that Database supports the required functionality.
- **Please email me with your comments**
- I hope you have found this Tutorial interesting and useful.
- Please email me with your questions or suggestions so I can improve this first draft Tutorial.

Good luck with designing your first Database !

Barry Williams
Principal Consultant
Database Answers Ltd.
London, England