Physical Design

1 Introduction

When creating physical design, there are many things to consider. These include resolving many to many relationships, determining whether or no keys should be synthetic and what data types to assign to your attributes.

Converting from a login to a physical can be a time-consuming process. However, when done correctly, the design should be self-documenting for the most part.

2 Resolving many to many relationships

There are 2 kinds of resolutions for many to many relationships. These are:

- An associative entity. These are also known as linking table. They are made up of 2 columns.
- An entity with attributes. This is a straight up table that connects 2 or more tables.

2.1 Associative entities

In figure 1, you have an example of an associative entity.

It allows for a relationship between part and company. Unfortunately, that is all it is good for. It is a very limited data structure that assumes to normally have no extra data stored. Due to modern regulatory rules and law, often significantly more data needs to be stored.

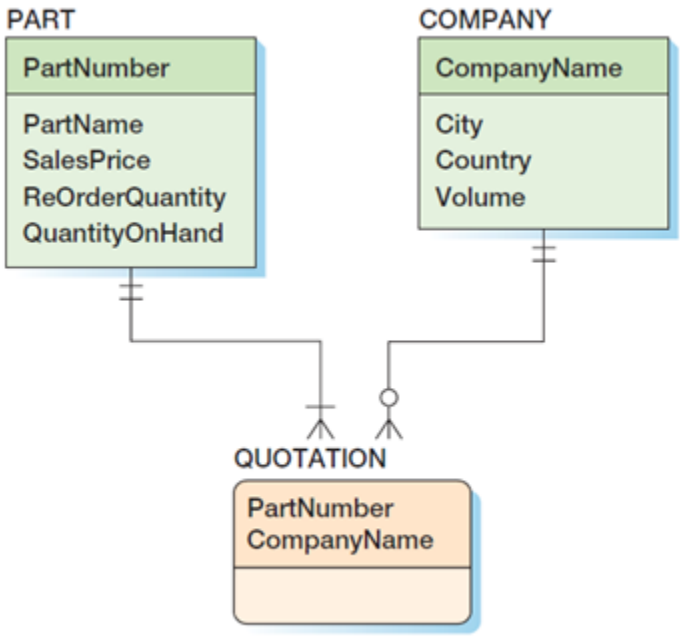


Figure 1

2.2 Entities with attributes

In Figure 2, then, the relationships between PART and QUOTATION and between COMPANY and QUOTATION are both identifying. This fact is shown in Figure 1 by the solid, non-dashed line that represents these relationships.

As with all identifying relationships, the parent entities are required. Thus, the minimum cardinality from QUOTATION to PART is one, and the minimum cardinality from QUOTATION to COMPANY also is one. The minimum cardinality in the opposite direction is determined by business requirements. Here, a PART must have a QUOTATION, but a COMPANY need not have a QUOTATION.

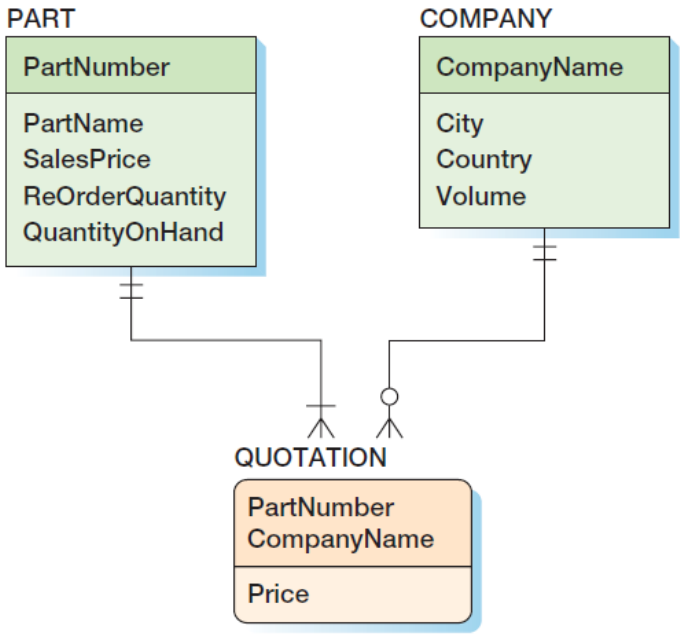


Figure 2

3 NATURAL VS. SYNTHETIC KEYS

Natural versus Synthetic keys is a topic of some debate in the database industry. Although Natural keys have "meaning", they can be difficult to manage due to data collection and potential data entry issues. Natural keys often end up being compound keys. Compound keys make programming both the application and the database calls somewhat more complex.

Synthetic keys suffer from a different problem. They have no meaning outside of the database, thus the values are "alien" to most workflows. Synthetic keys have some very specific advantage.

- They are automatically generated and thus are not prove to data entry errors
- They are often numeric. Numeric keys are easier to index and tend to have much faster retrieval rates.
- They are easier to search against. Synthetic keys are normally single field.
- They allow for a simpler naming convention.

In recent years, it has become preferable to use synthetic keys due to reasons listed above.

4 MAPPING DATA STRUCTURES

Once you start mapping your data structures, you will need to start assigning data types to them. As straight forward as that sounds, there are a few things that go on at this point. The first is proper field design. The second step is assigning data types to a field and finally, you will need to identify relations between entities.

4.1 DATABASE DATA TYPES

4.1.1 Character/text data

There are 3 common data types when dealing with character data. These are:

- CHAR(L) – this is used to store fixed length strings such as postal codes or province/state abbreviations. This field defines the maximum length a string can by. L determines the length of the field up to 255 characters.
- VARCHAR(L) – this is used to store variable length strings. L determines the length of the field up to 255 characters.
- TEXT – this is used to store huge amounts of text. The maximum amount you can store in this field type varies from database system to database system. However, for the most part, the TEXT data type will hold enormous amounts of data, in the upwards of hundreds of megabytes of text.

4.1.2 Numeric data

There are many data type to store numeric data.

- SMALLINT – A small integer. The signed range is -32768 to 32767. The unsigned range is 0 to 65535.
- MEDIUMINT – A medium-size integer. The signed range is -8388608 to 8388607. The unsigned range is 0 to 16777215.
- INT, INTEGER – A normal-size integer. The signed range is: -2147483648 to 2147483647. The unsigned range is 0 to 4294967295.
- BIGINT – A large integer. The signed range is -9223372036854775808 to 9223372036854775807. The unsigned range is 0 to 18446744073709551615.
- FLOAT(precision) – floating-point number. Precision can be <=24 for a single-precision floating-point number and between 25 and 53 for a double-precision floating-point number. These types are like the FLOAT and DOUBLE types described immediately below. FLOAT(X) has

the same range as the corresponding FLOAT and DOUBLE types, but the display size and number of decimals are undefined.

- FLOAT(M,D) – A small (single-precision) floating-point number. Allowable values are -3.402823466E+38 to -1.175494351E-38, 0, and 1.175494351E-38 to 3.402823466E+38. If UNSIGNED is specified, negative values are disallowed. The M is the display width and D is the number of decimals. FLOAT without arguments or FLOAT(X) where X <= 24 stands for a single-precision floating-point number.
- DOUBLE(M,D) – A normal-size (double-precision) floating-point number. Allowable values are -1.7976931348623157E+308 to -2.2250738585072014E-308, 0, and 2.2250738585072014E-308 to 1.7976931348623157E+308. If UNSIGNED is specified, negative values are disallowed. The M is the display width and D is the number of decimals. DOUBLE without arguments or FLOAT(X) where 25 <= X <= 53 stands for a double-precision floating-point number.
- DOUBLE PRECISION(M,D), REAL(M,D), DECIMAL(M[,D]) – An unpacked floating-point number. Behaves like a CHAR column: ``unpacked'' means the number is stored as a string, using one character for each digit of the value. The decimal point and, for negative numbers, the `-' sign, are not counted in M (but space for these is reserved). If D is 0, values will have no decimal point or fractional part. The maximum range of DECIMAL values is the same as for DOUBLE, but the actual range for a given DECIMAL column may be constrained by the choice of M and D. If UNSIGNED is specified, negative values are disallowed. If D is omitted, the default is 0. If M is omitted, the default is 10

4.1.3 Date data

There are several data types to allow a database designer to store dates and times. Some do only dates, others do only time and still others allow you to store the entirety of a date time value.

- DATE – a date. The supported range is '1000-01-01' to '9999-12-31'. MySQL displays DATE values in 'YYYY-MM-DD' format, but allows you to assign values to DATE columns using either strings or numbers
- DATETIME – A date and time combination. The supported range is '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. MySQL displays DATETIME values in 'YYYY-MM-DD HH:MM:SS' format, but allows you to assign values to DATETIME columns using either strings or numbers

- TIMESTAMP – A timestamp. The range is '1970-01-01 00:00:00' to sometime in the year 2037.
- TIME – A time. The range is '-838:59:59' to '838:59:59'. MySQL displays TIME values in 'HH:MM:SS' format, but allows you to assign values to TIME columns using either strings or numbers
- YEAR – A year in 2- or 4-digit format (default is 4-digit). The allowable values are 1901 to 2155, 0000 in the 4-digit year format, and 1970-2069 if you use the 2-digit format (70-69).

4.1.4 Boolean data

Binary data is stored using either the BOOL data type or TINYINT. MySQL uses TINYINT to "emulate" a Boolean data type. Other database systems handle Booleans just fine. The TINYINT is a very small integer. The signed range is -128 to 127. The unsigned range is 0 to 255.

4.2 Field Design

Field design involves naming a field and assigning it a data type. There are several guidelines you can follow when designing your fields. You should always:

- Make your field names meaningful.
- Decide on a naming convention and stick to it throughout the design.
- Try to make an informed guess as the maximum size the data will be.
- Name your relations in an appropriate manner.

Also, you should try to design your database as close to the ANSI standards as possible because all database systems support ANSI SQL naming conventions and data types. You should try and avoid database system specific items as much as possible. Some basic rules to remember are:

- Keep all names lower case
- Do not use spaces in any field, table or database names. Use underscores instead.
- Stick to basic data types wherever possible.

Sometimes you don't have a choice but use database system specific "extensions" to the ANSI standards. Then include storing GIS, special or network data. In this case, make sure you document where and why you used these data types.

4.3 Mapping data types to fields

When mapping data types to a field, you should try and plan for the future. Changes are always easier in the beginning than later in the design process.

Mapping data types to fields is usually a straightforward process. You would look at some sample data, when available, and simply start identifying it properties.

Things you need to ask as you crawl through the data:

- How long does the field need to be?
- Do I need to plan for slightly larger data?
- Is this data text, numeric or a date or time of some sort?
- When looking at numbers:
    - Does the number contain decimal places?
    - How many decimal places of precision do I need to worry about?
    - Can the number be negative?
    - How big can this number get?
- When looking at dates and times, do you need to store both the date and time?

Of course, not all these questions apply to all fields and experience will guide you into making good decisions and speed up the identification process. Once you've identified the properties of your fields, name them and then give them a data type on your database design forms.

As a general guide, you should allow for:

- Addresses: at least 2 address fields with at least 100 characters in length for each.
- Phone numbers:
    - at least 10 digits for North American unformatted phone numbers
    - If you plan to store formatted numbers allow at least 13 characters.
    - You should allow for at least 6 digits for extension numbers
- E-mail address should get at least 100 characters.
- Postal codes should get at least 10 digits if dealing with any outside Canada
- Any body of text longer than 5 words should get a full TEXT(LONGTEXT in MySQL) field.
- Dollar values should have at least 5 digits of precision. You can always round the values in you code later.

4.4 IDENTIFYING RELATIONS

When identifying relations, you should always keep an eye out for the "gotchas". These are fields that you think may not need to be controlled, but end up require extensive control at a later date.

Lists such as order status or publication category are examples of "relation" targets. When identifying your relations, consider that sometimes some data structures can relate to each other more than once.

For example, the invoice form in the Identifying Data Structures chapter can contain multiple relations to an employee table. These relations could include:

- Who created the order?
- Who was the sale person involved?
- Who shipped the order?
- Who billed the order?

At this point, we've identified at least 4 relations just for a single order. Now comes the tricky part. How are you going to create 4 employee ID fields in 1 table? You are not allowed to have more than 1 field with same name in the same table. Normally you would, either prepend or append the field name with the function of the field.

For example:

- Who created the order? entered_by_employee_id or employee_id_creator
- Who was the saleperson involved? salesrep_employee_id or employee_id_salesrep
- Who shipped the order? shipped_by_employee_id or employee_id_shipped_order
- Who billed the order? order_billed_by_employee_id or employee_id_billed_order

Please note that the above examples are just that example and may not reflect the best naming conventions once could ask for.

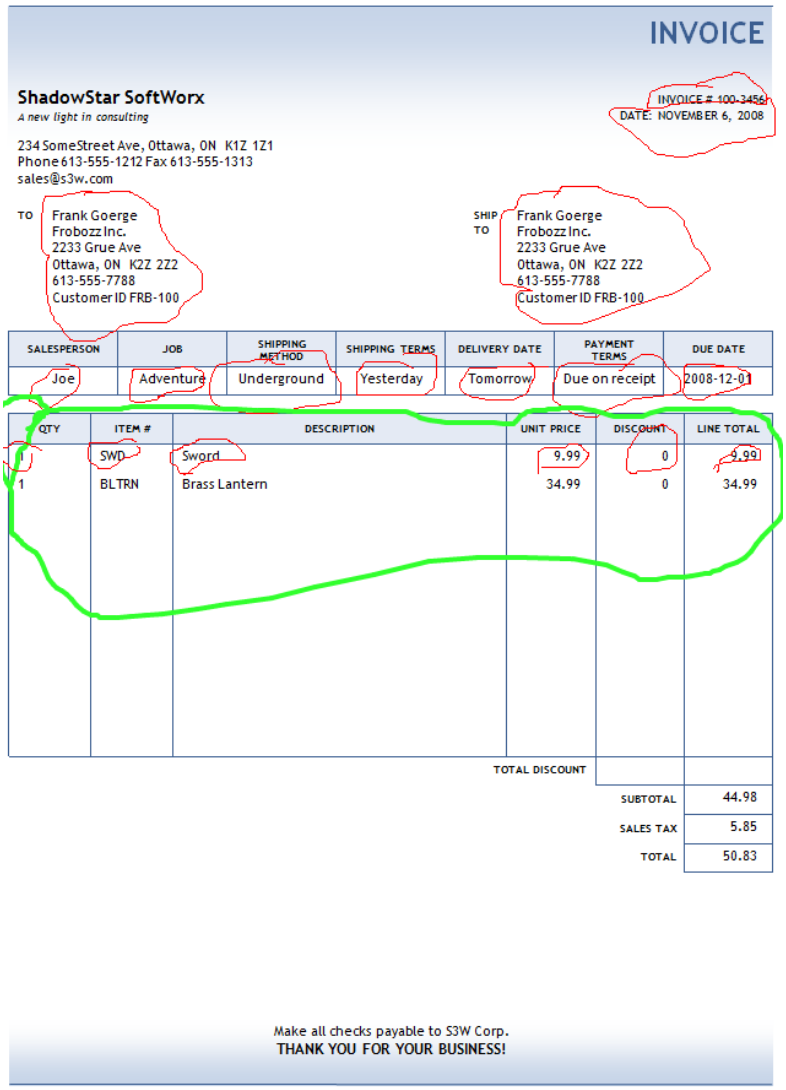
5 DESIGN PROCESS IN THE REAL WORLD

5.1 IDENTIFYING DATA STRUCTURES FROM A REAL WORLD DOCUMENT

Identifying data structures from a real-world document tends to be the easier route to creating a database structure. Normally, you would take an existing form and a marker and start circling all the pieces that may or may not be data. A more refined approach is to use different colored markers to identify different field types. Once you have identified all the data structures, you start filling out a form

for each entity type. Once you have identified all the entities, you then start filling out a “fields form” for each entity.

In the following form, an invoice, all the fields have been identified. The singlet pieces of data are identified in red and since an invoice is a master detail type of form, the details area is circled in green. The master record in this case is the invoice and the details/child record is the line items.



INVOICE

ShadowStar SoftWorx
A new light in consulting

234 SomeStreet Ave, Ottawa, ON K1Z 1Z1
Phone 613-555-1212 Fax 613-555-1313
sales@s3w.com

INVOICE # 100-3456
DATE: NOVEMBER 6, 2008

TO
Frank Goerge
Frobozz Inc.
2233 Grue Ave
Ottawa, ON K2Z 2Z2
613-555-7788
Customer ID FRB-100

SHIP TO
Frank Goerge
Frobozz Inc.
2233 Grue Ave
Ottawa, ON K2Z 2Z2
613-555-7788
Customer ID FRB-100

SALESPERSON	JOB	SHIPPING METHOD	SHIPPING TERMS	DELIVERY DATE	PAYMENT TERMS	DUE DATE
Joe	Adventure	Underground	Yesterday	Tomorrow	Due on receipt	2008-12-01

QTY	ITEM #	DESCRIPTION	UNIT PRICE	DISCOUNT	LINE TOTAL
1	SWD	Sword	9.99	0	9.99
1	BLTRN	Brass Lantern	34.99	0	34.99

TOTAL DISCOUNT	
SUBTOTAL	44.98
SALES TAX	5.85
TOTAL	50.83

Make all checks payable to S3W Corp.
THANK YOU FOR YOUR BUSINESS!

Each of the items circled in red can be broken down into individual items. For example:

- The invoice number
- The date of the invoice
- The sales person
- Etc…

The section circled in green contains line items. The line items are repeated more than once, thus it is a details area of a master detail form. Each detail is made up of several fields. These include:

- Quantity
- Item Number
- Description
- Etc…

In other words, each invoice can contain multiple line items. We have now identified 2 entities with multiple fields in each. So, for this form we will require at least 1 “entities” form and 2 “fields” forms.

However, with further analysis you can find additional tables and fields.

5.2 IDENTIFYING DATA STRUCTURES FROM A VAGUE SPECIFICATION

Often you are asked to create a database from a vague verbal description. This can be a very daunting for beginners and inexperienced designers. In order to help define what needs to be created, there are several "tools" you can employ.

5.2.1 Interviews

Interviewing is your initial source of information. You should ask your customer for as much detail as possible. Unfortunately, the questions themselves vary from customer to customer. However, there are a few that are common. These include:

- Do you plan to support multiple users?
- Do you plan to have different privilege levels for your end users?
- When defining "objects", try to ask if these "objects" can have a finite number of values. For example, order status and article publish state would be finite lists.
- Try to get as many details as possible for each component of the project.

5.2.2 Whiteboard Brainstorming

Whiteboard brainstorming involves taking what your interview produced and exploding them. You can use a whiteboard, pencil and paper or even a mind mapping tool to do this. The usual process includes:

1. List a heading for each component. Try to think about as many different components as possible. Bad ideas can be thrown out later.
2. For each component, list all tables/entities contained therein.
3. For each entity, list all the fields you can think of.
4. If a field is a finite list, mark it as so with a star or in a different color.
5. Once a field is identified as being a finite list, create an entity for it and list it fields.
6. Walk away and have a coffee in another room, or go for a walk in the park. In other words, stop thinking about it.
7. Go back and look at the brainstorming session. Does anything look strange, did you think of anything while you weren't thinking about it? Make your revisions and move on to the forms.

5.2.3 Research

Research usually involves looking at similar products and comparing your brainstorming ideas with what you see in competing products and services.

Once you have defined a general outline of the project and its entities and fields, you would fill out the entities and fields forms.