# Database Design 4: Lists

Learning outcome 1: to introduce a special type of design: a multi item list.

List ID X DateCompiled: 1/1/07

ListItem1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ListItem2. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ListItem3. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Etc.

In any application that has a list, you must be able to find ( or add in) **some** identifier (key) for the list e.g. listid field above. Typical applications include a list of things to do (a Do-To list); a list of parts to be shipped in a consignment (Shipping list) etc. Within a given list form, we have a number of items in the list, one item per line (drawn on paper). Reference the CIT Claims and Refund forms in Design 1 handout.

Learning outcome 2: To establish a link between design and programming. The intention is to show that a table design derived from a high level modelling may be incorrect and problems only appear when the developer begins to use that design. In many cases developer will ‘stick with’ the poor design and try to work ‘around the problem’ E.g. by writing complicated queries/reports. DB queries usually involve multiple tables.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The following is a possible design for this type of list design problem:

ProductInfo table describing all the products the company can sell, PK = **ProductId**

|  |  |  |
| --- | --- | --- |
| ProductId | Description | Cost |
|  |  |  |

OrderHistory table. The Total attribute value is calculated when the order is made and its value is the sum of (product cost \* quantity ordered). **OrderID** is the primary key:

|  |  |  |  |
| --- | --- | --- | --- |
| OrderId | CustomerId | Total | Date |
|  |  |  |  |

LineItem Table consists of 3 fields: an order can be for multiple items and this table stores the quantity of each item ordered. Primary Key (**LineitemId, OrderId**)

|  |  |  |
| --- | --- | --- |
| LineitemId | OrderId | Quantity |
|  |  |  |

Customer table with PKey = CustomerID.

|  |  |  |
| --- | --- | --- |
| CustomerId | Custname | Address |
|  |  |  |

This design problem was given to me by a student with the following text:

“I am having major problems with my database queries/reports when I try to print a report with a full listing of all the Products ordered by a certain customer e.g. CustomerID = 'C134'”

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Your assignment is to

1. Analyze the tables to deduce what application the tables are designed for i.e. what are the tables trying to store? Describe in clear terms e.g. is it for a Video rental? This can help you clarify your thinking from a logical perspective.
2. Populate the tables with dummy data. Does the design correctly store the data according to the meaning you have given the application? Alternatively, think about this problem from the processing perspective: why does the programmer run into problems of duplicate or incorrect data. What does (s)he mean by this?

1. Identify the problem with the design i.e. what is the design flaw that has come to light when the query processing of the tables won’t work correctly. Note we don’t know the SQL for this query exactly, but we should be able to work out what data is needed for the query; where that data is; and then envisage how the tables would have to be processed to obtain that data.
2. Redesign the tables to solve the problem(s).

Note: the solution to this problem will be dealt with in class. But you should be able to identify the problem, and offer some solution(s).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_