**Design 6b Modelling Lists**

We have seen clear guidelines on how to model and transform models to tables. Model to table transformation rules

* 1:\* no new table, put key from 1 side into the other table.
* \*.\* always generates a new table, with composite key made from the keys from both participating objects.

We also have introduced guidelines for viable objects. However, in this section we will discuss different ways we might model a list design. We should know that from a normalisation point of view, a list requires at least two tables. One for the list and another for the items on the list.

List (Design 4).

The issue with List modelling is that text books tend to model the link as a 1.\* association.

Order 1.\* Lineitem

OrderID LineItemid

So, one Order has many lineitems, but a lineitem CAN ONLY appear on one Order. Using our transformation rules, we take the key from the Order table and put it into the Lineitem table. And in this case it becomes the primary key of that table (OrderID, LineitemId).

In a way, this seems to work. However, we know that LineItemId will repeat, i.e. 1 ,2 ,3 repeats on each Order list. But, if it’s 1.\*, then the lineitemId is not allowed repeat. Therefore, in modelling terms we are modelling the physical line on which the order might be written if we had a paper invoice form. These are unique. Compare this to if the order was for a list of products (as in Design4).

When we use a notion like Product, and we have an order for Product e.g. P1; we do not model the order on the physical product that was taken by the Customer. We are using the general notion of Product P1. If we wanted the exact physical Product, we’d need some identifier like serial number. These are unique for each physical product P1.

If we replace the notion of Lineitem with an actual List of products we can see an alternative model.

UML diagram for an Order list of products. OrderId (key) with attributes date, Cust etc. Order can have many products. Products are PID, name, colour.

Transform diagram into a set of tables

Order \*.\* Products

OrderId Pid

1 Order has many products

1 product can appear on many Orders

I.e. it is a many to many association. Therefore according to our rules we generate a 3rd table, for the list details. Key is (OrderId, Pid). This is in line with the tables we get if we normalise.

But if we model the link as 1.\* as in most text books, what happens when we add in product? Does Lineitem also now link to Product? What is the multiplicity of that link?

Order 1.\* \*.1

Products

Li

IineItems

Li

So in a database, the Lineitems table key is (OrderId, LineItemId) and the ProductId is also in that table (because of the 1.\* association).

But, if you scan across this latest diagram you see that in fact Order is linked (indirectly) to Product via a \*.\* association. The Lineitem Object (table) is really just complicating the model ( unnecessarily in my opinion).

Note: an alternative model of a list might put the lineitem# as a field in the association just like qty is in design 1 and mark in design 2.

List orders(\*.\*) Product

Lineitem

Design List: a list of nothing in particular!!

e.g. a daily to-do(study) list. The items on the list are just a text description and a time. We will use the LineItemNo field to Identify the items on the list.

List 9: 2/12/12

1. Study DB , 3-4
2. GO to Gym 4-5

List 10: 4/4/2012

1. Go to the pub 3-4
2. Go to town 4-6

List 11: 14/4/2012

1. Go to town 9-10
2. Study DB 3-10

|  |
| --- |
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |

ListId, date

Li

List \*.\* LineItemNo

Need Table for List Table for \*.\*

????????

Key (ListId, LineitemNo)-> Desc, time

Do we really need to implement a table of one column just to store the integer lineItemNos? Not according to our rules of viable objects (see Design 6 modelling notes). No descriptor attributes. Therefore in this list design model we would generate a table for the List object, and a table for the \*.\* association using the individual PKs from the participants to form a composite index (ListId, LineItemNo). However, we would not generate a table just for LineItemNo. 1 List can have many itemnos and an itemno can appear on many Lists

So this would be also be correct for our design transformation rules.

Reference: <http://www.tdan.com/view-special-features/8457>