

# **Stock and Manpower Management for Pick and Pack Operations**

## **Proposal for Reporting Database**

Greg O'Keefe

August 14, 2008

This proposal describes a database system providing reports to help manage stock and casual labour in the pick and pack operations of Completely Different Mail-Order Pty. Ltd. It is a response to the request for proposals dated 24 August 2007.

The first section describes the general rationale for the proposed solution, and our assumptions about the operational environment in which it would work. The following two sections present UML diagrams, a class diagram and an object diagram. The class diagram describes the kinds of relevant objects and their relationships, whilst the object diagram shows a specific example end-of-day scenario as a test case. The database design schema is derived from the class diagram, and SQL code to create the database schema is shown in Section 4. Similarly, the test data given in the following section as SQL INSERT commands, is derived from the object diagram. SQL queries to satisfy each of the 4 specific system requirements are presented in Section 6.

## **1 Background and Assumptions**

Each business day, the company receives orders through its online ordering system. At the close of business, picking slips are printed for these orders. The following working day, casual staff pick and pack the products described on the picking slips.

Currently, inadequate stock of a given product is detected when the last one is picked. This results in unacceptable delays in delivering orders. Delays are also caused by lack of casual staff, but calling in too many staff results in unnecessary cost. The proposed system will inform the operations manager about products that will run out on the following day, and also provide an estimate of the person hours required to pick and pack the orders.

We assume that the ordering system will provide the required information to the proposed reporting database at the close of business. If the proposal is accepted, code will be developed to load the database from the data provided.

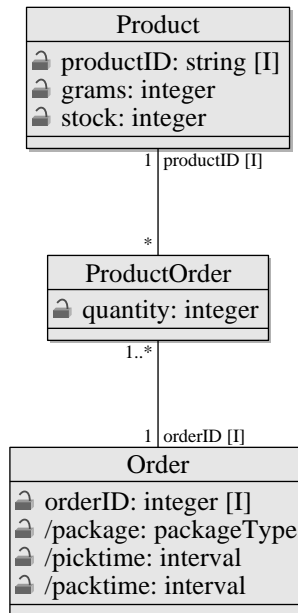


Figure 1: Class Diagram

## 2 Class Diagram

A UML class diagram is shown in Figure 1. This shows the classes of object that we need to consider to provide the required information.

The class **Product** represents the products that customers can order through the online ordering system and they are identified by the attribute **productID**, which is the same as the product code used by the ordering system. The attribute **grams** is the weight of a single item of the given product in grams, and **stock** is the number of items of this product available in the warehouse for picking.

An **Order** represents an order taken by the online ordering system and is identified by the **orderID** assigned to it by that system. Attributes **/package**, **/picktime** and **/packtime** are *derived*, as indicated by the '/' in front of their names. These values depend on other values, and in an implementation are calculated as needed rather than stored explicitly.

The type **packageType** is an enumeration, given below.

$$packageType = \{smallEnvelope, largeEnvelope, boxes\}$$

The actual contents of the order, the products ordered and their quantities are represented by a separate class **ProductOrder**. Thus, a **ProductOrder** represents a single line on an order picking-slip, specifying a product and a quantity.

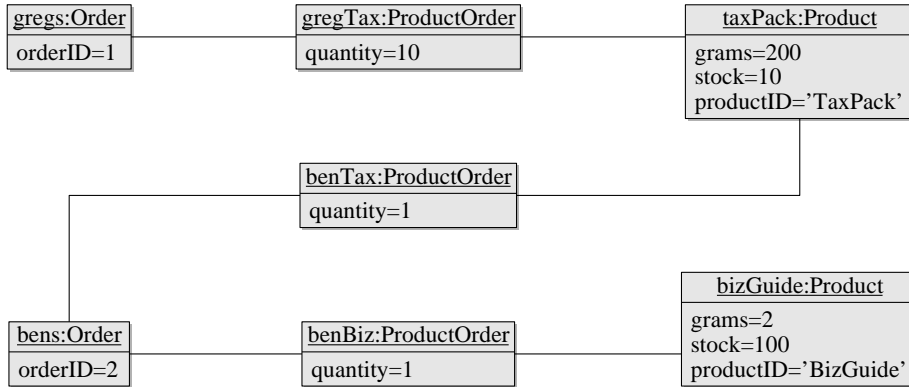


Figure 2: Object Diagram

### 3 Object Diagram

A specific situation instantiating the class diagram of the previous section is shown in the object diagram of Figure 2. It shows two orders, one for 10 tax packs, and one for one tax pack and a business guide. This scenario was chosen to exercise most possibilities relevant to the queries with a minimal amount of data.

### 4 Schema Creation

The file `pickpackEgCreateSchema.sql` contains SQL `CREATE TABLE` statements translated from the UML class diagram above. The translation is straightforward, except for the derived attributes.

The derived attributes in the class model are implemented as *views*. The derivation of the attributes proceeds in stages. View `orderWeight` derives the weight of an order. This is then used in view `orderPackage` to derive the packaging type for that order. The picking time for each product order (line item in an order form) is derived in view `productOrderPickTime`. This data is then assembled to yield the view `orderDerived`, which corresponds to the `Order` class of the UML class model.

See attached file `pickpackEgCreateSchema.sql`.

### 5 Test Data

The file `pickpackEgPopulate.sql` contains SQL code to populate the database with the example data shown in the object diagram above.

## 6 Queries

The four SQL queries in `pickpackEgQueries.sql` each correspond to the four items in the requirements statement. Because very similar data are used to satisfy the first two requirements, the view `productStockOrdered` is defined in `pickpackEgCreateSchema.sql` and used in both of these queries. Utilising the views has enabled very simple queries. We anticipate that these views would also greatly simplify the satisfaction of similar requirements.

## 7 Conclusion

The proposed solution is a simple one, but satisfies the requirements and is easily extended to related problems.