SQL Programming
Final Project Submission

Patrick Moorehouse

Student Number: Omitted

Email: Omitted

Database Owner: PatDBA

Database Password: patdba

Database Dump File: PatDBA.dmp

Changes

There were no major changes made to the design of the database as the present design was deemed to be a good working model. One column was added to the Products table, p_description, to hold a description for products. This was planned from the beginning, as products need a description so customers know what they are buying, but was mistakenly omitted in the initial specification of the database. A datatype of CLOB had to be used for this new column, as the text descriptions of products are of undetermined length, and may possibly exceed the 255 character limit of VARCHAR2s.

Some changes to data types had to be made. Although no data types were specified in the original specification (Project Specification 1), some data types were intended to be used, but later changed. The Notes column in the order_products table was originally intended to be implemented as a VARCHAR2, however, I realised that the requests submitted by customers, to be stored in the Notes column, can sometimes exceed the character limit of the VARCHAR2 data type, as they can be quite detailed in their description, or may have many requirements. For this reason, the Notes column was implemented as a CLOB type, and not as a VARCHAR2 type as originally intended.

Reflection - Things that could be better

After completing this project, there are a number of things I would do differently if I was to re-design the database. For users, address is stored as a single VARCHAR2 column. I feel a better design would be to use separate tables for country and county, with a foreign key to reference them in the users table, as these may be repeated by different users. I would then store the first line of the address only as a VARCHAR2, containing house number and street name.

I might also drop the passwords table, and include passwords as an attribute of the users table. Passwords were implemented as a separate entity by request of the customer; however, it is not the most common way to implement password security. It does offer a small improvement to security however.

Overall, I am pleased with the outcome of this project. I learned a great deal, especially with regards to PL/SQL and creating procedures and functions within a database. These features will be of great benefit in future database implementation.

Database Layout

The following is a list of all tables, one for each entity, and also extra junction tables where required.

users (u_id, u_fname, u_lname, u_dob, u_phone, u_email, u_address, *pass_id, r_id*)

passwords (pass_id, pass_name)

roles (r_id, r_name)

orders (o_id, *u_id*, *s_id,* date_ordered)

order_status (s_id, s_name)

order_products(o_id, p_id, quantity, notes) products(<u>p_id</u>, p_name, p_price, p_description, *t_id*, *b_id*)

type(<u>t_id</u>, t_name)

brand(<u>b_id</u>, b_name)

category(<u>c_id</u>, c_name)

product_category(<u>p_id</u>, <u>c_id</u>)

Submitted Material

Database Dump

A backup of the database has been included, which can be imported to re-create all tables, data, sequences, triggers, procedures and functions. The file is called PatDBA.dmp and the owner name that should be used to import it is PatDBA.

Screenshots

I have included screenshots of the results for all queries, functions, procedures and trigger activations.

Submitted Code

The code for this project is supplied in text files. The code is too long, and would be too difficult to manage and format in this report, so here I will list the files provided, and explain their contents, so that reviewing them will be easier.

01-CreateTables

This file contains the SQL code for creating all of the tables in the database. All Primary Keys and Foreign Keys are handled at table creation. The text from this file can be pasted directly into SQL Plus to create all the tables.

02-BeforeTriggers

This file contains the code to create the required BEFORE triggers. There are 8 BEFORE INSERT triggers, which are designed to automatically insert Primary Key values into the tables roles, order_status, type, brand, category, products, users and orders. There are 8 sequences created, one for each of the triggers. There is also an addition BEFORE UPDATE trigger, designed to check whether the status of an order has been changed when an order is updated, and notifies the user if the status will remain the same.

02B-AfterTrig

This file just contains code for a single trigger. It is an AFTER UPDATE trigger, and it tests to see if the price has been changed when a product is updated, then prints the new and old values of the price to screen if there has been a change to the price of a product.

03-InsertData

This file holds a list of INSERT statements, which fill all the tables in the database with data.

04-InnerJoins

File containing 4 INNER JOINs.

05-OuterJoins

File containing 6 OUTER JOINs. The first 2 are LEFT OUTER JOINS, the 3rd and 4th are FULL OUTER JOINs, and the 5th and 6th are RIGHT OUTER JOINs.

06-Procedures

File contains 5 procedures. All procedures are named procedures, and are followed by an example of an anonymous block that could be used to call the procedure. All procedures contain EXCEPTION handling, and 2 of the 5 procedures use cursors.

07-Function

A file that contains a single function, CountOrders. The function counts how many orders have been made to date and returns the number.

08-CUBE

This file contains a single query, which is a CUBE query (has CUBE grouping). The query lists information on users, orders, and sale amounts. It displays the total money spent on all orders, the total spent per order, and the total spent by each user, in total, and per order.

09-SubQueries

This file contains 5 queries, all of which contain subqueries.