ORDER MANAGEMENT SYSTEM

DATABASE DESIGN DOCUMENT

ABSTRACT

Order management system deals with creating a platform for customers, administrators and technicians to view or add various aspects in accordance to their roles within the organization.

1. INTRODUCTION

Objective   
The Order management system project aims to implement functionalities for administrator, buyer to view, place and edit orders within system. We are going to create a fully functional database with table schemas and relations with independent roles.

Analysis

Analysis can be defined as breaking up of any whole so as to find out their nature, function etc. It defines design as to make preliminary sketches of; to sketch a pattern or outline for plan. To plan and carry out especially by artistic arrangement or in a skillful wall. System analysis and design can be characterized as a set of techniques and processes, a community of interests, a culture and an intellectual orientation.

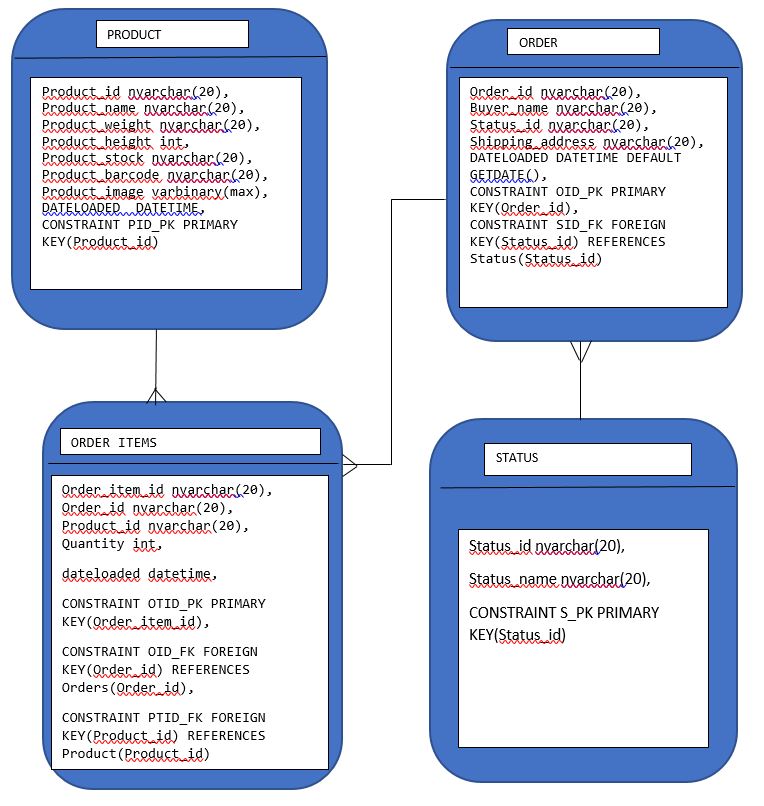
The various tasks in the system analysis include the following.

* Understanding application.
* Planning.
* Scheduling.
* Developing Database models.
* Performing relationship studies.
* Performing table schema analysis.
* Mapping of tables.
* Providing key relationships.
* Supervising, installing and maintaining the system.

1. PURPOSE  
     
    The purpose of this database is to provide a fully functional order management system with details of every order and its related products with customer details, Buyers should be able to view only the order details that are placed under their name unlike administrator who should be able to view the complete data.
2. SCOPE, APPROACH and METHODS

The Database Design is composed of definitions for database objects derived by mapping entities to tables, attributes to columns, unique identifiers to unique keys and relationships to foreign keys. The design is done with respect to requirements of the database and specific roles.

1. ER diagram( Entity Relationship) :



The audit tables doesn’t relate with any main tables they are used for tracking history, So they need not be included in ER diagram.

1. DATABASE DESIGN :

STATUS TABLE:  
  
To Create a table for status to map with orders.

|  |  |  |  |
| --- | --- | --- | --- |
| FIELDS | DATATYPE | CONSTRAINT | DESCREPTION |
| Status\_id | nvarchar(20) | PRIMARY KEY(S\_PK) | Status ID |
| Status\_name | nvarchar(20) |  | Status name |

ORDER TABLE:  
  
To create a table for orders, to store details of every order.

|  |  |  |  |
| --- | --- | --- | --- |
| FIELDS | DATATYPE | CONSTRAINT | DESCREPTION |
| Order\_id | nvarchar(20) | PRIMARY KEY(OID\_PK) | Order ID |
| Buyer\_name | nvarchar(20) |  | Name of customer |
| Status\_id | nvarchar(20) | FOREIGN KEY(SID\_FK) | Status ID |
| Shipping\_address | nvarchar(20) |  | Shipping address of customer |
| DATELOADED | DATETIME |  | Time and Date of data load |

Product Table:

To create a table to store product details.

|  |  |  |  |
| --- | --- | --- | --- |
| FIELDS | DATATYPE | CONSTRAINT | DESCREPTION |
| Product\_id | nvarchar(20) | PRIMARY KEY(PID\_PK) | Product ID |
| Product\_name | nvarchar(20) |  | Name of Product |
| Product\_weight | nvarchar(20) |  | Weight of product |
| Product\_height | int |  | Height of product |
| Product\_stock | nvarchar(20) |  | Stock of product |
| Product\_barcode | nvarchar(20) |  | Barcode |
| Product\_image | varbinary(max) |  | Image |
| DATELOADED | DATETIME |  | Time and Date of data load |

Order Items :  
  
 To view bills and to access any kind of information regarding customers and their tests.

|  |  |  |  |
| --- | --- | --- | --- |
| FIELDS | DATATYPE | CONSTRAINT | DESCREPTION |
| Order\_item\_id | nvarchar(20) | PRIMARY KEY(OTID\_PK) | Order item ID |
| Order\_id | nvarchar(20) | FOREIGN KEY(OID\_FK) | Order ID |
| Product\_id | nvarchar(20) | FOREIGN KEY(PTID\_FK) | Product ID |
| Quantity | Int |  | Quantity of product |
| dateloaded | DATETIME |  | Time and Date of data load |

ORDER AUDIT HISTORY Table:  
  
Contains all the information about changes made in Order table, acts as history table for Orders.

|  |  |
| --- | --- |
| FIELDS | DATATYPE |
| Order\_id | nvarchar(20) |
| Buyer\_name\_OLD | nvarchar(20) |
| Buyer\_name\_NEW | nvarchar(20) |
| Status\_id\_OLD | nvarchar(20) |
| Status\_id\_NEW | nvarchar(20) |
| Shipping\_address\_OLD | nvarchar(20) |
| Shipping\_address\_NEW | nvarchar(20) |
| AuditOperation | nvarchar(10) |
| Dateloaded\_OLD | DATETIME |
| Dateloaded\_NEW | DATETIME |

ORDER ITEMS AUDIT HISTORY Table:  
  
Contains all the information about changes made in Order items table, acts as history table for Order items table.

|  |  |
| --- | --- |
| FIELDS | DATATYPE |
| Order\_item\_id | nvarchar(20) |
| Order\_id | nvarchar(20) |
| Product\_id | nvarchar(20) |
| Quantity\_Old | int |
| Quantity\_New | int |
| AuditOperation | varchar(50) |
| Dateloaded\_Old | datetime |
| Dateloaded\_New | datetime |

PRODUCT AUDIT HISTORY Table:

Contains all the information about changes made in Product table, acts as history table for product table.

|  |  |
| --- | --- |
| FIELDS | DATATYPE |
| Product\_id | nvarchar(20) |
| Product\_name\_NEW | nvarchar(20) |
| Product\_name\_OLD | nvarchar(20) |
| Product\_weight\_NEW | nvarchar(20) |
| Product\_weight\_OLD | nvarchar(20) |
| Product\_height\_NEW | Int |
| Product\_height\_OLD | Int |
| Product\_stock\_NEW | nvarchar(20) |
| Product\_stock\_OLD | nvarchar(20) |
| AuditOperation | varchar(50) |
| DATELOADED\_NEW | DATETIME |
| DATELOADED\_OLD | DATETIME |

1. Normalization

Normalization is a process of organizing the data in database. It is a systematic approach of decomposing tables to eliminate data redundancy there by increasing data consistency.

1NF:

1. Data in each column should be atomic, no multiple values separated by commas.

2. Table should not contain any repeating column groups.

3. Identify each record uniquely using primary key.  
  
Our Order management system project satisfies the above rules and hence it is in First normal form.  
  
  
2NF:

1. The Table meets all the conditions of 1nf.

2. Move redundant data to separate table.

3. Create relationship between these tables using foreign keys.

Our Order management project satisfies the above rules and hence it is in Second normal form.  
3NF:

1. Meets all the conditions of 1NF and 2NF.

2. Does not contain columns that are not functionally dependent on the primary key.

Our Order management project satisfies the above rules and hence it is in Second normal form.

1. CODING and IMPLEMENTATION:

The script file has been made with tables such as Order, Order items, Product and status with relationships across them, An Audit history table has been made separately for Order, Order items and Product tables to record changes. For inserting Product image and bar code we take help of temp tables first and then we push the values to the main Product table. For Barcode generation we have used a Scalar valued function within SSMS. Once the script file is done and tables are exported to database, Users has to be added with validations for each one of them.

Stored procedures has been created for Audit trail of changes made to tables and to store it into history tables.  
Another Stored procedure has been created for Bulk inserting and updating of Orders and Order item information.

1. Stored procedure for Audit trail:

We have created a Stored procedure named DBO.PR\_ADD\_UPDATE\_AUDIT\_HISTORY\_TABLES in the database to track the changes made in Product, Order and Order items table, On executing the procedure for first time the values are getting inserted into the audit tables as following image and for every further update the audit table will be updated accordingly. We have used temp tables inside procedure since the data in audit table are subjected to change every now and then, we need to make sure that only the final correct data is pushed to the main table.

1. Stored procedure for Bulk insert and update of data in Order and Order item table:  
     
   We have created a Stored procedure named DBO.PR\_ADD\_UPDATE\_ORDER\_TABLES in the database to bulk insert and update data in Orders table, We have used temp tables inside procedure since the data in Order tables are subjected to change every now and then, we need to make sure that only the final correct data is pushed to the main table. We have used bulk insert function to push data to table using a txt or csv file.
2. User roles has been created as per the script file with predicate function for Row level security, Administrator will have admin privileges for all the tables whereas each particular user will have privileges to view only the order details related to that particular buyer name. In similar fashion for requested Dev ops and application roles can be created with specific privileges.
3. Conclusion:

The Application analysis is done and the database design with table schema and key relationships are given with entity relationship to pave way for further Implementation of this Order management system.