INVENTORY MANAGEMENT SYSTEM

Submitted to:

Dr. Shelly Sachdeva

Head of Department

Computer Science

NIT Delhi

Submitted by :

Hitendra (171210028)

Ankit (171210009)

Irshad (171210029)

Akhil (171210005)

CONTENTS

INTRODUCTION

In this world, at this time, we have a ubiquitous and essential entity called database management system. It is a software used to manage databases. Database management has evolved from a specialized computer application to a central component of a modern computing environment, and, as a result, knowledge about database systems has become an essential part of an education in computer science.

The project INVENTORY MANAGEMENT SYSTEM is an example of working database system where we manage inventory and analyze the day-to-day records of a dairy.

LIST OF ABBREVIATIONS

- **IMS :** Inventory Management System
- **DBMS :** Database Management System
- **DDL :** Data Definition Language
- **DML :** Data Manipulation Language
- **SQL :** Structured Query Language
- **SSMS :** SQL Server Management Studio Express
- **ER :** Entity Relationship

DECLARATION

We hereby declare that the work reported in this project work on “Inventory Management System” submitted to National Institute of Technology, Delhi is our original work done under the supervision of Dr. Shelly Sachdeva, Head of Department (Computer Science), National Institute of Technology, Delhi. The material contained in the report has not been submitted to any university or institution for any award of any degree.

ACKNOWLEDGEMENT

This project is prepared in the partial fulfillment of the requirement for the degree of Bachelor's in Technology in Computer Science. The satisfaction and completion of this task would be incomplete without the heartfelt thanks to people whose constant guidance, support and encouragement made this work successful. On doing this undergraduate project we have been fortunate to have help, support and encouragement from many people we would like to acknowledge them for their cooperation.

Our essence spirit of gratitude to National Institute of Technology, Delhi for designing such a worthy syllabus and making us do this project. Our next essence of gratitude goes to Dr. Shelly Sachdeva, Head of Department (Computer Science) without whose help our project would have never been possible to bring into existence. We would also like to thank all the batch mates who, without any hesitation, came to our aid whenever needed. Last but not the least we want to thank every direct and indirect hands that were involved in completion of this project.

This project has been a learning curve for all of us which strengthened the concepts of database functionality, and it has been a wonderful experience where we have learnt and experienced many beneficial things.

With Regards

Hitendra (171210028)

Ankit (171210009)

Irshad (171210029)

Akhil (171210005)

THEORY

Database Theory :

A database is a collection of information that is organizes so that it can easily be accessed, managed and updated. In one view, database can be classified according to types of content: bibliography, full-text, numeric, and image. In computing, database is sometime classified according to their organizational approach. A distributed database is one that can be dispersed or replicated among different points in a network.

Relational Database :

IMS has the relational database model. A relational database is a digital database whose organization is based on the relational model of data. This model organizes data into one or more tables of rows and columns. These tables here have the relation.

The relation is maintained by the unique key defined in each row. The key can be primary and foreign depending on their nature of connection. The standard user and application program interface to a relational database is the structured query language (SQL). SQL statement are used both for interactive queries for information from relational database and for gathering data for reports.

Primary Key :

The primary key of a relational table uniquely identifies each record in the table. It can either be a normal attribute that is guaranteed to be unique or it can be generated by the DBMS. A primary key's main features are:

- It must contain a unique value for each row of data.
- It cannot contain null value.

Foreign Key:

A foreign key is a column or group of column in a relational database table that provides a link between data in two tables. In foreign key reference, a link is created between two tables when the column or columns that hold the primary key value for one table are referenced by the column or column in another table thereby establishing a link between them. Creating a foreign key manually includes the following advantages:

- Changes to primary key constraints are checked with foreign key constraints in relation table.
- An index enables the Database Engine to quickly find related data in the foreign key tables.

Structured Query Language (SQL) :

The structured Query language (SQL) is the set of instructions used to interact with a relational database. In fact, SQL is the only language the most database actually understands. Whenever you interact with such a database, the software translates your commands into SQL statement that the database knows how to interpret. SQL has three major Components:

- Data Manipulation Language (DML)
- Data Definition Language (DDL)
- Data Control Language (DCL)

ACID Property :

Every database transaction obeys the following rules:

Atomicity – Either the effects of all or none of its operation remain ("all or nothing" semantics) when a transaction is completed (committed or aborted respectively). In other words, to the outside world a committed transaction appears (by its effects on the database) to be indivisible, atomic, and an aborted transaction does not leave effects on the database at all, as if never existed.

Consistency – every transaction must leave the database in a consistent (correct) state, i.e., maintain the predetermined integrity rules of the database (constraints upon and among the database's objects). A transaction must transform a database from one consistent state to another consistent state (however, it is the responsibility of the transaction's programmer to make sure that the transaction itself is correct, i.e., performs correctly what it intends to perform (from the application's point of view) while the predefined integrity rules are enforced by the DBMS). Thus since a database can be normally changed only by transactions, all the database's states are consistent. An aborted transaction does not change the database state it has started from, as if it never existed (atomicity above).

Isolation – Transactions cannot interfere with each other (as an end result of their executions). Moreover, usually (depending on concurrency control method) the effects of an incomplete transaction are not even visible to another transaction. Providing isolation is the main goal of concurrency control.

Durability – Effects of successful (committed) transactions must persist through crashes (typically) by recording the transaction's effects and its commit event in a non-volatile memory.

FUNCTIONALITIES & REQUIREMENTS

- **Total Sales** – Date wise, Product wise, Week wise, Month wise, Previous Months Data, Variance – Day wise, Week wise, Month wise --- **All in graphs**
- **Products**: Milk, Ghee – 2 types, Curd, Nasika, Ark – 3 types, Phenyl, Mustard Oil
- Customer Orders: In order to see the **seasonal effect in sale** we need to trace out orders from customers for our products, though it can be tracked with sales also.
- Sales - **Target vs Actual**, **Profit vs Loss** - Monthly
- Auto **message if loss is more than 5%** in each product and cumulative(annual) sales.
- **Capital Items Tracing** --- Purchase date and Total amount, Depreciation, etc.
- **Connectivity with customers** – as soon as we are launching new products or stores.
- Tracking of **customer feedback and suggestions**.
- **Which Customers are using our products frequently** – Product wise/ Month Wise (SQL Query)

Entity-Relationship Diagram

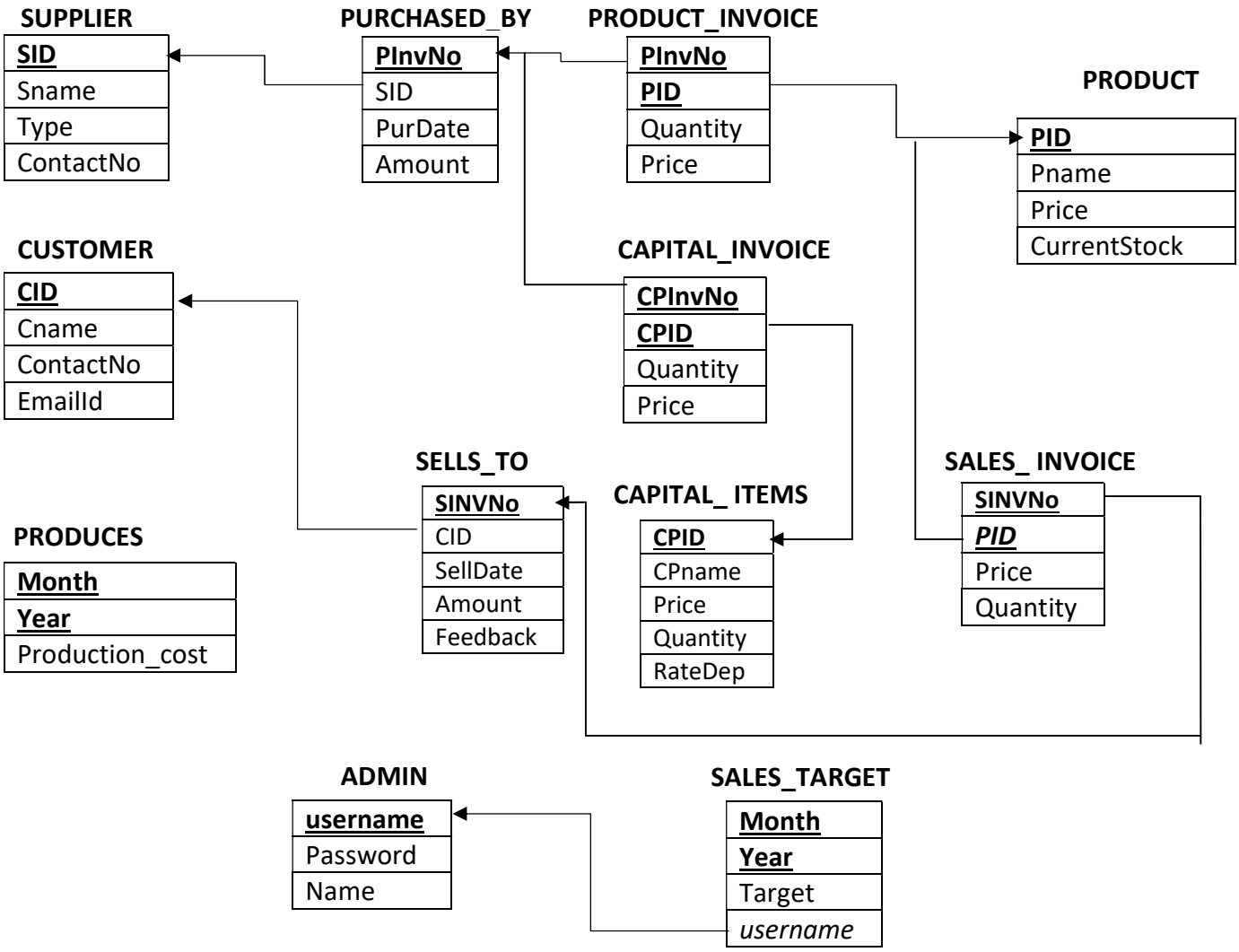
On the next page, is attached the approved ER diagram of Inventory Management System.

(Please turn over)

NORMALIZED DATABASE SCHEMA

- Following is the database schema after passing through the normalization stages:
- 1NF – First Normalized Form, 2NF – Second Normalized Form, 3NF – Third Normalized Form, BCNF – Boyce-Codd Normalized Form

The final schema is :



RELATIONAL ALGEBRA OPERATIONS USED

- **SELECT(σ)**

  -Notation: σ p(r)

  -p is called the selection predicate

  -Defined as: σp(r) = {t | t $\in$ r and p(t)} Where p is a formula in propositional calculus consisting of terms connected by : $\wedge$ (and), $\vee$ (or), $\neg$ (not)

  Each term is one of:

  <attribute> op <attribute> or <constant>

  where *op* is one of: =, $\neq$, >, $\geq$. <. $\leq$

- **PROJECT (π)**

  -Notation: $\prod_{A_1, A_2, \mathrm{K}, A_k}(r)$

  where A1, A2 are attribute names and r is a relation name.

  The result is defined as the relation of k columns obtained by erasing the columns that are not listed and duplicate rows removed from result, since relations are sets

➢ CARTESIAN PRODUCT (A X B)

Notation r x s

Defined as: r x s = {t q | t $\in$ r and q $\in$ s}

Assume that attributes of r(R) and s(S) are disjoint. (That is, R $\cap$ S = $\varnothing$).

If attributes of r(R) and s(S) are not disjoint, then renaming must be used.

➢ JOIN($\bowtie$)

Let r and s be relations on schemas R and S respectively.

Then, r s is a relation on schema R $\cup$ S obtained as follows:

- Consider each pair of tuples tr from r and ts from s.
- If tr and ts have the same value on each of the attributes in R $\cap$ S, add a tuple t to the result, where
  - t has the same value as tr on r
  - t has the same value as ts on s

- **UNION ($\cup$)**

  Notation: $r \cup s$

  Defined as: $r \cup s = \{t \mid t \in r \text{ or } t \in s\}$

  For $r \cup s$ to be valid.

  1. r, s must have the same arity (same number of attributes)
  2. The attribute domains must be compatible (example: 2nd column of r deals with the same type of values as does the 2nd column of s)

- **INTERSECTION ($\cap$)**

  Notation: $r \cap s$

  Defined as: $r \cap s = \{ t \mid t \in r \text{ and } t \in s \}$

  Assume: r, s have the same attributes of r and s are compatible

DDL QUERIES INCORPORATED

- ***CREATE DATABASE***

  Creates a database repository for the project.

```
CREATE DATABASE INVENTORY;
```

- ***CREATE TABLE***

  To create a new table specifying its:
  - Attributes
  - Data type
  - Constraints such as NOT NULL, KEY, INTEGRITY CONSTRAINTS etc

```
CREATE TABLE customer
 (
      Cid INT NOT NULL,
      Cname VARCHAR(20) NOT NULL,
      ContactNo NUMBER(10),
      EmailId VARCHAR(20),

      PRIMARY KEY(Cid)

  )
```

- ***ALTER TABLE***

  To change the definition of a base table. ALTER TABLE is used to:
  1. Add a column (attribute)
  2. Drop a column (attribute)
  3. Change column definition
  4. Add or Dropping table constraints.

```
ALTER TABLE productinvoice
ADD CONSTRAINT productinvoice_PInvNo_FK
FOREIGN KEY(PInvNo) REFERENCES purchasedby(PinvNo);
```

- ***DROP TABLE***

  To drop an entire relation.

```
DROP TABLE IF EXISTS `capitalinvoice`;
```

DML QUERIES & CLAUSES USED

- ***INSERT INTO:***

  Used to populate the relations with data.

```
INSERT INTO produces
VALUES( ${data.month}, ${data.year}, ${data.cost});
```

- ***SELECT… FROM… WHERE…-***

  It gives a list of attributes from a set of the relation satisfying a given conditional (Boolean) expression.

```
SELECT Cname, SInvNo, Amount
FROM customer, sellsto
WHERE customer.Cid = sellsto.cid;
```

- ***ORDER BY -***

  It allows the user to order the tuples in the result of a query by the values of one or more attributes.

```
SELECT WEEK(SellDate) as week_wise,SUM(Amount) AS total_sales
FROM sellsto
where YEAR(SellDate) = ${data.year}
group by week_wise
order by week_wise;
```

- ***GROUP BY -***

  Specifies the grouping attributes, so that the aggregate function could be applied to each group.

```
SELECT WEEK(SellDate) as week_wise,SUM(Amount) AS total_sales
FROM sellsto
where YEAR(SellDate) = ${data.year}
group by week_wise
order by week_wise;
```

- ***UPDATE-***

  The UPDATE statement is used to modify the data in a table. It is used to modify attribute values of one or more selected tuples.

```
UPDATE products
SET CurrentStock =  CurrentStock + ${data.newStock}
WHERE Pid = ${data.pid};
```

- ***FUNCTIONS USED-***

  Following functions were used –

  - SUM()
  - YEAR()
  - WEEK()
  - MONTH()
  - Monthname()
  - Concat()

ADVANCED SQL FEATURES USED

- ***TRIGGER-***

A trigger is a statement that is executed automatically by the system as a side effect of a modification to the database. In our program a trigger is to be executed when a new product is added and an email is to be sent to the customers to notify them. Also if loss is more than five percent a trigger is executed to inform the admin as a *snackbar*.

BIBLIOGRAPHY

- https://www.google.com/
- https://www.scholar.google.com/
- https://www.wikipedia.org/
- https://creately.com/
- Oracle MySQL documentation
- Database System Concepts - Abraham Silberschatz, Henry F. Korth, S. Sudarshan, [6th edition]
- Fundamentals of Database Systems - Ramez Elmasri, Shamkant B. Navathe [6th edition]