CS107: Analysis of a Sample Database

William J Townsend

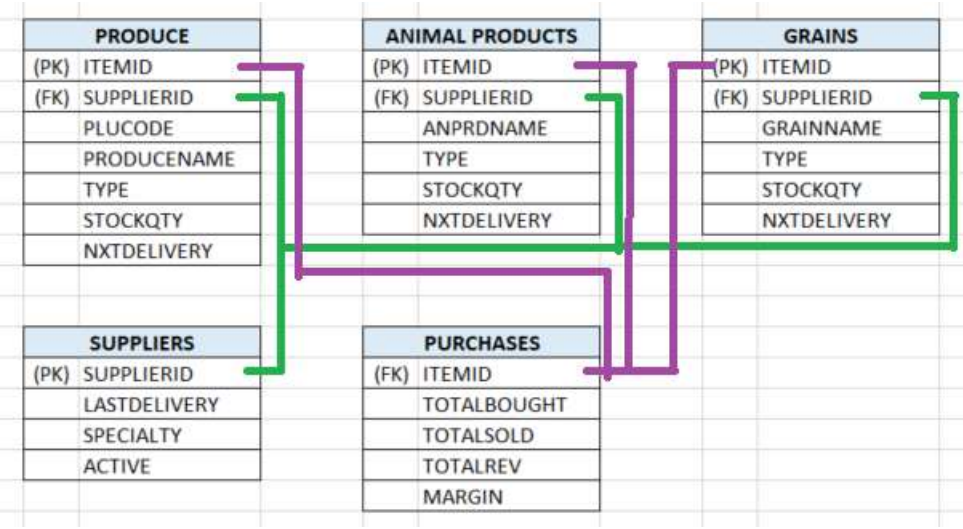
ABSTRACT

This essay analyzes a sample database provided in Computer Science 107: Database Fundamentals, to generate various conclusions about the database. This analysis is based solely on provided table names, column names, and column data types, without any sample records to demonstrate the practical usage of the database. As a result, assumptions regarding the usage of the database are made and provided within the essay. An entity relationship Diagram is generated to help depict the Primary and Foreign Keys within the database and the relationships between tables, which are also explained narratively. The normalization of each table is assessed against database standards of first, second, and third normalization forms, with four of the five tables being in first normal form and the fifth being in second normal form. Based upon provided information, the database software for this sample database is also inferred to be Oracle Database.

CS107: Analysis of a Sample Database

Provided with the tables and column names of a small sample database, we can draw several conclusions about the nature and characteristics of that database. This may include the brand or version of the database used, the normalization of the database, and the chosen keys and dependencies within the sample database. It is important to note, however, that the conclusions which can be drawn solely from column headings are limited and require some assumptions on the reader's part, as there are no sample records provided of this database, which would demonstrate how the database is actually being used. The assumptions underpinning this analysis will be explained prior to explaining the conclusions derived from the analysis of this sample database.

Underlying Assumptions



This is a very simple Entity Relationship Diagram (ERD) built with Microsoft Excel & Paint, with each table, the associated columns, and the inferred Primary Keys (PK) and Foreign Keys (FK) from the following assumptions:

1) It is assumed that items exist in each of the Product, Animal Products, and Grains tables and as new orders or an item come in, the corresponding row for that item is updated. For example, if Granny Smith apples (Produce table) are ITEMID "PR001" and are ordered a second time, the fields (NXTDELIVERY, STOCKQTY) are updated. In this way, the ITEMID functions as a Primary Key in each of these three tables. While this isn't how I would set up such a database, it is the most sensible way that I can make sense of the provided database and columns.
2) It is also assumed that in the Purchases table, that purchases of an item (such as Granny Smith apples) recorded in this table are updated in a fashion similar to the assumed usage of the Produce, Animal Products, and Grains tables. Again, this is not how I would set up such a database, but this is the assumption that allows me to most clearly understand this database as it is currently built, rather than declaring it to be incomplete and badly broken.
3) It is further assumed with regard to the ITEMID in each of the Produce, Animal Products, and Grains tables that the Primary Key (ITEMID field) is not only unique within its own table, but also unique across all three tables. The ITEMID field is defined as being of datatype "CHAR (5)", which allows for the use of letters, such as "PR001", "AP001", "GR001". This is a specific choice to not use a process of automatically generating a numeric ITEMID instead, such as "00001", "00002", and so on. If ITEMID's were numeric in the form of "00001", then there would be multiple items (in different tables)

with an identical ITEMID, which would almost certainly result in identical records occurring in the Purchases table. Based on the ability to use ITEMIDs in the form of "PR001", the ITEMID field in the Purchases table should remain unique itself.

4) It is assumed that the SUPPLIERID in the Suppliers table is a Primary Key, which has a 1:Many relationship with items that may populate the Produce, Animal Products, and Grains tables. Thus, the SUPPLIERID in those three tables functions as a Foreign Key.

As mentioned, these are some lengthy assumptions to make about this database, given absolutely no populated records or data. In making these assumptions, I can infer how the database works, even if I don't think it would work very efficiently. Absent these assumptions, I would be at a loss to figure out how the database works at all, given the absence of records to observe.

Keys & Dependencies

As shown in the above ERD, there are several keys present in the database. The ITEMID functions as a Primary Key in the Produce, Animal Products, and Grains tables (Kaitlin Oglesby for Study.com, n.d.). SUPPLIERID functions as a Primary Key in the Suppliers tables. This is because each of these IDs is unique and each specifies a specific record in the database. SUPPLIERID functions as a Foreign Key in the Produce, Animal Products, and Grains tables, as it stems from somewhere else, may not be unique, and doesn't specify a particular record in these tables, though it could be used to join tables together (Kaitlin Oglesby for Study.com, n.d.). Similarly, ITEMID is a Foreign Key in the Purchases table. The Purchases Table does not at this

point have a Primary Key, but it could be given one by giving each purchase a specific PURCHASEID, as is often done in relational databases for retail environments.

Normalization & Dependencies

First normal form (1NF) requires that each column have a single value (not multiple values put together), each column have a unique name, all values in a field have the same data type, and no two records are identical (Temitayo Odugbesan for Study.com, n.d.). What is normal form in DBMS?, study.com). Each of the Product, Animal Products, Grains, and Suppliers tables meet these requirements, the last (unique records) being assured by the presence of a Primary Key. The Purchases table is being assumed to meet the last requirement (unique records) given the stated assumption that ITEMID across the Product, Animal Products, and Grain tables is unique. As all tables are 1NF, the database as a whole could be said to be normalized to 1NF.

None of the Product, Animal Products, Grains, or Suppliers tables are in second normal form (2NF). 2NF requires that there be no partial dependencies of any column on the primary key for that table, along with being 1NF (Temitayo Odugbesan for Study.com, n.d.). Each of these tables has partial dependencies on the primary key. The PLUCODE, PRODUCENAME, ANPRDNAME, and GRAINNAME are each dependencies of the ITEMID for their associated tables, as each would be the same for a given ITEMID. The TYPE column in each table is unclear but would also be the same for each ITEMID as well. In order to be 2NF, these attributes would have to be removed to a new table, using ITEMID as a Foreign Key. The SPECIALTY field in the supplier table is also a dependency on the SUPPLIERID, so this table is

also not 2NF (but it could be if detailed supplier information, including specialty, were placed in another table). None of these four tables can even be considered for being in third normal form, because 2NF is a prerequisite for such (Temitayo Odugbesan for Study.com, n.d.).

The Purchases table is 2NF, because there are no dependencies on the ITEMID in this table. However, it is the only table which is 2NF, the rest of the database remains at 1NF. The Purchases table is not in the third normal form (3NF) because it does have transitive dependencies (Temitayo Odugbesan for Study.com, n.d.). Specifically, the MARGIN field is a transitive dependency of the TOTALREV field, if I'm understanding the columns and the intended math correctly (the margin would come from TOTALREV minus a total cost, which is not provided). While this field prevents the table from being in 3NF, it is a useful metric to have available at a glance, so it may be preferable to keep the table in second normal form.

Database/SQL Version

I attempted to identify the database software for this sample database, by looking at the specified DATA_TYPE limitations provided for each table in the sample database. The database version appears to be Oracle Database. In Oracle Database, the CHAR datatype requires a size argument (each CHAR field is followed by at least one integer), and it can take an optional length argument as well (some of the CHAR fields are followed by a second integer) (Oracle Data Types: CHAR, n.d.). In Oracle Database, NUMBER requires two arguments, one each for precision and scale, and in each of the NUMBER fields, 2 arguments were provided by the designer of the sample database (Oracle Data Types: NUMBER, n.d.).

REFERENCES

Odugbesan (n.d.). What is Normal Form in DBMS? - Types & Examples. (2017, December 30). Retrieved from https://study.com/academy/lesson/what-is-normal-form-in-dbms-types-examples.html.

Ogilsby (n.d.). What is a Primary Key in SQL? (2016, June 28). Retrieved from https://study.com/academy/lesson/what-is-a-primary-key-in-sql.html.

Ogilsby (n.d.). What is a Foreign Key in SQL? (2016, June 24). Retrieved from https://study.com/academy/lesson/what-is-a-foreign-key-in-sql.html.

Oracle (n.d.). https://docs.oracle.com/en/database/oracle/oracle-database/21/sqlrf/Data-Types.html#GUID-85E0A0DD-9E90-4AE1-9AD5-93C89FDCFC49

Oracle (n.d.). https://docs.oracle.com/en/database/oracle/oracle-database/21/sqlrf/Data-Types.html#GUID-75209AF6-476D-4C44-A5DC-5FA70D701B78