**MSCD 600 Database Architecture**

**Scholarly Responses**

For each of your scholarly responses, provide a substantive response (about 2000 words total or 500 words per question) to a selection of at least four rigorous questions listed below.

Each question, (a) calls for demonstration of higher levels of cognition (i.e. analysis, synthesis, and evaluation), and (b) is complex in nature (i.e. calls for a body of knowledge, concise, explicit expository response.

The formulation of your response should follow a logical progression (e.g. introduction, analysis & discussion, and summary/conclusions, be developed with strong referential integrity and citations to academic sources, and strictly follow American Psychological Association (APA) guidelines. Be sure to include a cover page and a references page.

For this assignment, choose a ***minimum of four*** of the following questions:

**1. What is the importance of data normalization in the reduction of deletion and insertion anomalies? Give some examples of deletion and insertion anomalies.**

2. Describe a business scenario and specify the types of constraints that would be appropriate to ensure the integrity of the database.

3. What is a Cartesian product? What are ways to prevent a Cartesian product? When might you want to implement a Cartesian product?

**4. What is metadata and why is it important?**

5. What is the purpose of indexes in a database? Discuss index types introduced in this course and provide examples of each type of index.

**6. What is some functionality of Oracle SQL that can make the life of an SQL application developer easier? Provide specific examples of Oracle functionality, and what this functionality does for the developer.**

7. Describe the importance of database security. What are some methods one can use to increase security in your database?

**8. What does it mean to say that a transaction has “ACID” qualities? Why is it important to properly design transactions and how does it affect the database/information system?**

9. Describe the concept of views in a database, and provide a scenario where views may be useful in a business application.

10. Why is it important to tune SQL statements? What are some steps one might take to tune an SQL statement?

Submit your Scholarly Responses as an attachment which clearly identifies your name in the title to the Dropbox titled for this assignment by the date specified by your facilitator.

For information on how you will be evaluated, see the Scholarly Responses Rubric below.

**1. What is the importance of data normalization in the reduction of deletion and insertion anomalies? Give some examples of deletion and insertion anomalies.**

Data Normalization is a process that assigns attributes to entities so that data redundancies are reduced or eliminated. The main forms of normalization center on the dynamics of determinant keys and dependent keys. Functional dependence is of prime importance here. Normalized forms eliminate partial dependency between attributes and even transitive dependencies as they are increasingly assimilated into greater levels of normalizations.

In summary there are three main types of normalizations. The First Normal Form marks that repeating groups must be eliminated, primary keys should be established and dependencies should be charted. The Second Normal Form marks that the partial dependencies (determinant-->dependent relationship between non key and primary key elements) must be eliminated via the creation of new table. The Third Normal Form marks that the transitive dependencies (determinant-->dependent relationship between non-primes attributes) must be eliminated via the creation of new tables. (Coronel, 222) Finally a special case called the Boyce Codd Normal Form states that every determinant in the table is a candidate key. Meaning, that each determinant references an irreducible primary key element of a composite.

It is important for tables to be in the third normal form to avoid insert and delete anomalies. For example consider a small invoicing system. The system includes CUSTOMER who can buy one or more PRODUCTS thus generating an INVOICE. Because a customer may buy more than one product at a time an invoice may contain several invoice LINEs each providing details about the purchased product. The PRODUCT table should contain the product price to provide a consistent pricing input for each product that appears on the invoice. If the table is not normalized the data entry will not permeate through to all the places where a record needs to be updated. Therefore a product price entered on one table might not change in reference other values linked via foreign keys of other tables. This will cause insert anomaly. Similarly a value deleted in one table may not propagate towards other tables and this will cause a delete anomaly.

In summary if a product was deleted then possible customers and invoices would be affected, a deletion anomaly is the unintended loss of data due to deletion of other data. The resulting data inconsistency is an example of how combining information that does not really belong together into one table can cause problems. An insertion anomaly is the inability to add data to the database due to absence of other data. For example, assume customer address is defined so that only one address is allowed. If a customer has different shipping and billing addresses the database cannot accommodate a new address to ship towards. This results in database inconsistencies due to omission.

**4. What is metadata and why is it important?**

The metadata describe the data characteristics and the set of relationships that links the data found within the database. For example, the metadata component stores information such as the name of each data element, the type of values (numeric, dates or text) stored on each data element, and weather the data element can be left empty. The metadata provide information that complements and expands the value and use of data. In short, metadata present a more complete picture of the data in the database. Given the characteristics of metadata, you might hear a database described as a collection of self-describing data. (Coronel, 7)

The metadata can be used to create a data schema in XML that describes the characteristics and domain of key attributes. The metadata helps us consolidate a data dictionary view of the database in hand and helps us to identify restrictions for values that make up the data set. The data dictionary is a component that stores the definition of data characteristics and relationships. Such data about data is known as metadata. The DBMS data dictionary provides the DBMS with its self-describing characteristic. In effect, the data dictionary resembles an x-ray of the company’s entire data set and is a crucial element in data administration.

Two main types of data dictionaries exist: integrated and standalone. An integrated data dictionary is included with the DBMS. For example, all relational DBMSs include a built in data dictionary or system catalog that is frequently accessed and updated by the RDBMS. Other DBMS’s especially older types do not have a built in data dictionary, instead the DBA may use third party standalone systems.

Data dictionaries can be identified as active or passive. An active data dictionary is automatically updated by the DBMS with every database access to keep its access information up to date. A passive data dictionary is not updated automatically and usually requires a running batch process. Data dictionary access information is usually used by the DBMS for query optimization.

The data dictionary’s main function is to store the description of all objects that interact with the database. Integrated data dictionaries tend to limit their metadata to the data managed by the DBMS. Standalone data dictionary systems are usually more flexible, and allow the DBA to describe and manage all of the organizations data, whether they are computerized or not. Whatever the data dictionary’s format, it provides database designers and end users with a much improved ability to communicate. In addition, the data dictionary is the tool that helps the DBA resolve data conflicts.

The metadata stored in the data dictionary are often the basis for monitoring database use and for assigning access rights to database users. The information stored in the data dictionary is usually based on a relational table format, thus enabling the DBA to query the database with SQL commands. For, example SQL commands can be used to query information about the users of a specific table or the access rights of a particular user. The DBA can also use the data dictionary to support data analysis and design. For example, a DBA can create a report that lists all data elements to be used in a particular application, a list of all the users who access a particular application/a particular program, or a report that checks for data redundancies, duplications and the use of homonyms and synonyms. And a number of other reports that describe data users, data access and data structure. The data dictionary can be also used to see if the naming standards are correct and the validation rules are correct. (Coronel, 696)

**6. What is some functionality of Oracle SQL that can make the life of an SQL application developer easier? Provide specific examples of Oracle functionality, and what this functionality does for the developer.**

SQL commands can be divided into two overall categories: data definition language (DDL) commands and the data manipulation language (DML) commands. The ANSI standard data types are supported by all RDBMS vendors in different ways. The basic data type are NUMBER, NUMERIC, INTEGER, CHAR, VARCHAR and DATE. The basic data definition commands allow you to create tables and indexes. Many SQL constraints can be used with columns. The commands are CREATE TABLE, CREATE INDEX, ALTER TABLE, DROP TABLE and DROP INDEX. DML commands allow you to add modify and delete rows from tables. The basic DML commands are SELECT, INSERT, UPDATE, DELETE, COMMIT and ROLLBACK.

The INSERT command is used to add new rows to tables. The UPDATE command is used to modify data values in the existing rows of a table. The DELETE command is used to delete rows from tables. The COMMIT and ROLLBACK commands are used to permanently save or roll back changes made to the rows. Once you COMMIT the changes, you cannot undo them with a ROLLBACK command.

The SELECT statement is the main data retrieval command in SQL. A SELECT statement has the following syntax: SELECT column list FROM table list WHERE condition list GROUPBY column list HAVING condition list ORDERBY column list- Ascending/Descending.

The column list represents one or more column names separated by commas. The column list may also include computed columns, aliases and aggregate functions. A computed column is represented by an expression or a formula (for example, P\_PRICE \* P\_QOH). The FROM clause contains a list of table names.

The WHERE clause can be used with the SELECT, UPDATE and DELETE statements to restrict the rows affected by the DDL command. The condition list represents one or more conditional expressions separated by logical operators (AND, OR and NOT). The conditional expression can contain any comparison operators (=,>, <,>=, <= and <>) as well as special operators (BETWEEN, IS NULL, LIKE, IN and EXISTS).

Aggregate functions (COUNT, MIN, MAX and AVG) are special functions that perform arithmetic computations over a set of rows. The aggregate functions are usually used in conjunction with the GROUP BY clause to group the output of aggregate computations by one or more attributes. The HAVING clause is used to restrict the output of the GROUP BY clause by selecting only the aggregate rows that match a given condition.

The ORDER BY clause is used to sort the output of a SELECT statement. The ORDER BY clause can sort by one or more columns and can use either ascending or descending order.

You can join the output of multiple tables with the SELECT statement. The join operation is performed every time you specify two or more tables in the FROM clause and use a join condition in the WHERE clause to match the foreign key of one table to the primary key of the related table. If you do not specify a join condition. The DBMS will automatically perform a Cartesian product of the tables you specify in the FROM clause. The natural join uses the join condition to match only rows with equal values in the specified columns. (Coronel, 292)

**8. What does it mean to say that a transaction has “ACID” qualities? Why is it important to properly design transactions and how does it affect the database/information system?**

ACID and BASE are two broad paradigms that describe the nature of database transaction dynamics and consistency.

ACID stands for the following properties:

* Atomicity- Transactions are all or nothing.
* Consistency- Database must be consistent
* Isolation- Transactions are isolated until complete
* Durability- Data persists after transaction completes.

BASE stands for the following properties:

* Basically Available- Response may be failure.
* Soft State- Data state is always changing (soft).
* Eventual Consistency- Data will be consistent at times

Applications that are appropriate for ACID and BASE:

All mission critical applications that require transactions to be absolute and unique must rely on ACID paradigm. Examples of these would include Oracle RDBMS implementations for the banking sector where each transaction has to be maintained by a synchronized and secure thread.

For more analytical ventures that have sparse data characteristics a BASE paradigm might be more appropriate. Examples of these include a business intelligence and analytics mission for a hospital that wants to provide tailored healthcare options to its patients. Data in this scenario would be available from a host of unstructured sources like emails, messages etc. which would be mapped to its appropriate attributes in the NoSQL database. < Primary key, attribute (key), data (value)>

References:

Coronel, Carlos & Morris, Steven. (2015). Database Systems: Design, Implementation, and Management   11thEdition. Cengage Learning.