DATABASE SYSTEMS ASSIGNMENT 2

Name-Vinayak Raghupathy
NYUID-N11568565
NETID-VR840

Q1) What is the difference between controlled and uncontrolled redundancy?
Illustrate with examples.

Ans 1) Redundancy is when the same data is stored multiple times in several places in a database.

For example, say the name of the student with Studentid=10 is Rahul is stored multiple times.

Redundancy is controlled when the DBMS ensures that multiple copies of the same data are consistent.For example, if a new record with Studentid=8 is stored in the database, the DBMS will ensure that this record is for Student Rahul. If the DBMS has no control over this, we have uncontrolled redundancy.

Q2) Cite some examples of integrity constraints that you think can apply to the database shown in Figure 1.2.

Ans 2) Some constraints are-

i)The StudentNumber should be unique for each STUDENT record (key constraint).

ii)The CourseNumber should be unique for each COURSE record (key constraint).

iii)A value of CourseNumber in a SECTION record must also exist in some COURSE record (referential integrity constraint).

iv)A value of StudentNumber in a GRADE_REPORT record must also exist in some STUDENT record (referential integrity constraint).

v) The value of Grade in a GRADE_REPORT record must be one of the values in the set {A, B, C, D, F } (domain constraint).

vi) Every record in COURSE must have a value for CourseNumber (entity integrity constraint).

Q3) If you were designing a Web-based system to make airline reservations and sell airline tickets, which DBMS architecture would you choose from Section 2.5? Why? Why would the other architectures not be a good choice?

Ans3)I would choose Three-Tier Client/Server Architecture for Web Application .Here, the Client consists of Web User Interface. The Web Server contains the application logic which includes all the rules and regulations related to the reservation process and the issue of tickets and the Database Server contains the DBMS.

Other architectures like Centralized DBMS won't work as in centralized DBMS, all the DBMS functionality, application program execution, and user interface processing are carried out on one machine.

Other architectures like Basic Client/Server Architecture and Two-Tier Client/Server Architecture may work but would not be performance efficient as the business logic will pose burden if it is in the client machine or the DBMS server.Although these architectures could work well if the logic is in other servers.

Q4) Consider Figure 2.1. In addition to constraints relating the values of columns in one table to columns in another table, there are also constraints that impose restrictions on values in a column or a combination of columns within a table. One such constraint dictates that a column or a group of columns must be unique across all rows in the table. For example, in the STUDENT table, the Student_number column must be unique (to prevent two different students from having the same Student_number). Identify the column or the group of columns in the other tables that must be unique across all rows in the table.

Ans4)

For Course Table Course_number column must be unique
For Prerequisite Table Prerequisite_number column must be unique
For Section Table Section_Identifier column must be unique
For Grade Table combination of Student_Number and Section_Identifier columns must be unique

Q5) Consider the following relations for a database that keeps track of student enrollment in courses and the books adopted for each course:

STUDENT(Ssn, Name, Major, Bdate)
COURSE(Course#, Cname, Dept)
ENROLL(Ssn, Course#, Quarter, Grade)
BOOK_ADOPTION(Course#, Quarter, Book_isbn)
TEXT(Book_isbn, Book_title, Publisher, Author)

Specify the foreign keys for this schema, stating any assumptions you make.

Ans5) In table BOOK_ADOPTION Course# is Foreign Key which references COURSE
In ENROLL Ssn is Foreign Key which references STUDENT
In ENROLL Course# is Foreign Key which references COURSE
In BOOK_ADOPTION Book_isbn is Foreign Key which references TEXT

Q6) Database design often involves decisions about the storage of attributes. For example, a Social Security number can be stored as one attribute or split into three attributes (one for each of the three hyphen-delineated groups of numbers in a Social Security number—XXX-XX-XXXX). However, Social Security numbers are usually represented as just one attribute. The decision is based on how the database will be used. This exercise asks you to think about specific situations where dividing the SSN is useful.

Ans 6) We require the area code also known as city code in some countries and perhaps the country code for dialing international phone numbers.

Also we can store the numbers in a separate attribute as they have their own independent existence. For example, if an area code region were divided into two regions, it would change the area code associated with certain numbers so if we have it in a separate attribute then we can update the area code easily.

We can also split first name, middle name, and last name into different attributes as it is possible that the names may be sorted and/or retrieved by the last name, etc.

In general, if the each attribute has an independent logical existence based on the application, it would make sense to store it in a separate column otherwise there is no clear benefit. For example, SSN need not be split into its component unless we are using the subsequences to make deductions about validity, geography, etc.

Q7) Recent changes in privacy laws have disallowed organizations from using Social Security numbers to identify individuals unless certain restrictions are satisfied. As a result, most U.S. universities cannot use SSNs as primary keys (except for financial data). In practice, Student_id, a unique identifier assigned to every student, is likely to be used as the primary key rather than SSN since Student_id can be used throughout the system.

a. Some database designers are reluctant to use generated keys (also known as *surrogate keys*) for primary keys (such as Student_id) because they are artificial. Can you propose any natural choices of keys that can be used to identify the student record in a UNIVERSITY database?

b. Suppose that you are able to guarantee uniqueness of a natural key that includes last name. Are you guaranteed that the last name will not change during the lifetime of the database? If last name can change, what solutions can you propose for creating a primary key that still includes last name but remains unique?

c. What are the advantages and disadvantages of using generated (surrogate) keys?

Ans 7a)A combination of First Name and Last Name together can be used as a primary key for identifying a student record.By keeping these values in separate attributes we allow the lookup for these parts of the name.We must leave the MiddleName and Initial separated to avoid the ambiguities from these combination.

Ans7b)No it cannot be guaranteed as new data is being continuously inserted. If suppose the records don't have last name as unique so we can combine other attributes with last name to make the entire combination unique.

For example we can add student attributes like home_number,office_number,cell_number to make the combination unique.However there is a possibility that not all students will have value for these attributes and therefore valueless attribute pairs can exist.

A better solution would be to have an additional relation Phone(SSN, Type, Number) while removing Phone_number from the Student relationship. This new relationship would allow the one to-many relationship from students to phone numbers without creating duplicate data or wasting space on sparse, valueless attributes.

c)Advantage

Immutability

Surrogate keys do not change while the row exists. This has the following advantages:
Applications cannot lose their reference to a row in the database (since the identifier never changes).

The primary or natural key data can always be modified, even with databases that do not support cascading updates across related foreign keys.

Requirement changes

Attributes that uniquely identify an entity might change, which might invalidate the suitability of natural keys. Consider the following example:

An employee's network user name is chosen as a natural key. Upon merging with another company, new employees must be inserted. Some of the new network user names create conflicts because their user names were generated independently (when the companies were separate).
In these cases, generally a new attribute must be added to the natural key (for example, an original_company column). With a surrogate key, only the table that defines the surrogate key must be changed. With natural keys, all tables (and possibly other, related software) that use the natural key will have to change.

Performance
Surrogate keys tend to be a compact data type, such as a four-byte integer. This allows the database to query the single key column faster than it could multiple columns.

Disadvantage
Disassociation
They have no relationship to the real-world meaning of the data held in a row. When inspecting a row holding a foreign key reference to another table using a surrogate key, the meaning of the surrogate key's row cannot be distinguished from the key itself.

Normalization
Surrogate keys can result in duplicate values in any natural keys. It is part of the implementation to ensure that such duplicates are not present.

Inadvertent assumptions
Sequentially generated surrogate keys can imply that events with a higher key value occurred after events with a lower value. This is not necessarily true, because such values do not guarantee time sequence as it is possible for inserts to fail and leave gaps which may be filled at a later time.