SFWRENG 3DB3 Assignment 1

Wednesday October 6th 2021

Mohinder Kallay – 400034724 – kallaym

1. State the two properties of data independence provided in a DBMS. Why are they important? Give an example of when data independence is useful.

   The two properties of data independence provide in a DBMS are logical data independence and physical data independence. Logical data independence shields users from changes in the choice of relation to be stored or changes in the logical structure of the data of the data of the DBMS. Meanwhile the conceptual schema of the DBMS shields the user from changes in the physical storage details. The combination of logical data independence and physical data independence combine together and are able to shield users and programs from changes in how the data is stored and structured which is a very important abstraction for DBMS. Also, data independence itself is particularly important because the rate of change of a platform is much faster than the rate of change of a database application which is slow. An example of when data independence is useful is when a relation within a database is split into two or more relations. For instance, suppose you have a database representing a university and one of the relations within the database is Faculty with the relational schema Faculty (fid: string, fname: string, sal: real), this relation is then further broken down into Faculty_public (fid: string, fname: string, office: integer) and Faculty_private(fid: string, sal: real). Although the initial relation of Faculty is changed in the conceptual schema of the database, users who are viewing the database do not experience a change in their respective views and are still able make the same queries as a result of data independence. In this case data independence is able to insulate users from changes in the way the data is stored for the database as well as how it is structured (structure of the Faculty relation changed in this case).

2.
   a. The primary key for the relation BRANCH would be (branchName, address). This key would be valid under the assumption that the name of the branch of the bank and address of each branch is unique. For example, if you can have 2 TD bank and RBC branches, we can identify each one uniquely by using the name of the branch as well as it's address. If the branchName alone was the unique identifier it would not be possible to accurately differentiate 2 different branches of the same bank. The reverse of the situation would be having 2 branches with the same address where each one could be uniquely identified with the name of the branch.

      The primary key for the relation CUSTOMER would be (cID). This key would be valid under the assumption that each customer receives their own unique customer id.

      The primary key for the relation LOAN would be (loanNum). This key would be valid under the assumption that each loan is associated with it's own unique loan number.

The primary key for Borrower would be (cID, loanNum). This key would be valid under the assumption that each borrower is assigned their own unique customer id and a unique loan number for their loan.

The primary key for the relation ACCOUNT would be (acctNum). This key would be valid under the assumption that each account belonging to someone has a unique account number that is different from the account number of all other individuals. An individual can have multiple accounts, each of which can be uniquely identified with a unique account number.

The primary key for the relation DEPOSITER would be (cID, acctNum). This key would be valid under the assumption that each customer is assigned a unique customer id and each customer can have multiple accounts which each have their own unique account number.

b. The first relation we will look at where this applies is the BORROWER relation. The primary key for this relation is (cId, loanNum), both of these attributes are foreign keys which means they reference other relations in the database (BORROWER is the referencing relation in this case). More specifically, the cID key is a foreign key in the BORROWER relation that references the CUSTOMER relation (referred relation) and it's primary key of cID. Likewise, the loanNum key is also a foreign key in the BORROWER relation that references the LOAN relation (referred relation) and it's primary key of loanNum. Since cID is a foreign key of BORROWER using referential integrity constraints this means that the cID of each tuple in BORROWER can only contains cIDs that exist in the primary key column of cID in the CUSTOMER relation (customer id must be real). Also, the referential integrity constraint in this case would mean that insertions of CUSTOMER tuples would not violate referential integrity but if BORROWER tuples were inserted where a cID value is not present in the cID field of the CUSTOMER relation a violation would occur. Similarly, the deletion of tuples in the CUSTOMER relation could potentially violate referential integrity if they are referenced in any tuples contained in the BORROWER relation while the reverse of the situation where BORROWER tuples are deleted in the BORROWER relation would not violate referential integrity. Like cID, loanNum is also a foreign key of the BORROWER relation which means that using the referential integrity constraint every tuple in the BORROWER relation must only contain loanNums that exist in the primary key column of the LOAN relation (loanNum must be a real loan). In addition, the referential integrity constraint in this case would indicate that insertions of tuples into the LOAN relation would not violate referential integrity meanwhile insertions of tuples into the BORROWER relation could potentially violate referential integrity if the inserted tuple contains a loanNum that doesn't occur in the primary key of the LOAN relation. Likewise, the deletion of tuples in the LOAN relation could violate referential integrity if the deleted tuples contain loanNums that are referenced in the BORROWER relation while deletion of tuples from BORROWER that reference loanNums in the LOAN relation do not violate referential integrity.

For the second relation DEPOSITER, whose primary key is cID and acctNum are also the foreign keys of this relation. The cID key is a foreign key that links the DEPOSITER relation (referencing relation) to the CUSTOMER relation (referred relation) and it's primary key of cID. Similarly, the acctNum foreign key links the DEPOSITER relation (referencing relation) to the ACCOUNT relation (referred relation) and it's primary key of acctNum. Since cID is a foreign key of DEPOSITER using referential integrity constraints this means that the cID of each tuple in DEPOSITER can only contains cID's that exist in the primary key column of cID in the CUSTOMER relation (customer id must be real). Also, the referential integrity constraint in this case would mean that insertions of CUSTOMER tuples would not violate referential integrity but if DEPOSITER tuples were inserted where a cID value is not present in the cID field of the CUSTOMER relation a violation would occur. Similarly, the deletion of tuples in the CUSTOMER relation could potentially violate referential integrity if they are referenced in any tuples contained in the DEPOSITER relation while the reverse of the situation where DEPOSITER tuples are deleted in the DEPOSITER relation would not violate referential integrity. Like cID, acctNum is also a foreign key of the DEPOSITER relation which means that using the referential integrity constraint every tuple in the DEPOSITER relation must only contain acctNums that exist in the primary key column of the ACCOUNT relation (acctNum must be a real account). In addition, the referential integrity constraint in this case would indicate that insertions of tuples into the ACCOUNT relation would not violate referential integrity meanwhile insertions of tuples into the DEPOSITER relation could potentially violate referential integrity if the inserted tuple contains an acctNum that doesn't occur in the primary key of the ACCOUNT relation. Likewise, the deletion of tuples in the ACCOUNT relation could violate referential integrity if the deleted tuples contain acctNums that are referenced in the DEPOSITER relation while deletion of tuples from DEPOSITER that reference acctNums in the ACCOUNT relation do not violate referential integrity.