**ST.XAVIER’S COLLEGE**

MAITIGHAR, KATHMANDU

****

**Database Management System**

Theory Assignment

Submitted By:

Shreesha Pokharel

013BSCCSIT038

2nd year/ 4th semester

Submitted to:

|  |  |
| --- | --- |
| **Er. Sanjay Kumar Yadav**  Lecturer  Department of Computer Science |  |

1. **GRANT and REVOKE authorizations**

You can grant or revoke system privileges to users and roles. If you grant system privileges to roles, then you can use the roles to manage system privileges. For example, roles permit privileges to be made selectively available.

In general, you grant system privileges only to administrative personnel and application developers. End users normally do not require and should not have the associated capabilities.

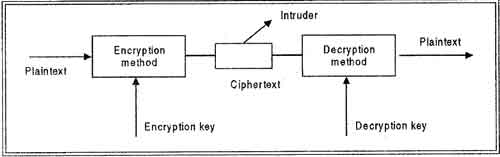
**Who Can Grant or Revoke System Privileges?**

Only two types of users can grant system privileges to other users or revoke such privileges from them:

* Users who have been granted a specific system privilege with the **ADMIN** **OPTION**
* Users with the system privilege **GRANT** **ANY** **PRIVILEGE**

1. **Data encryption**

A DBMS can use encryption to protect information in certain situations where the normal security mechanisms of the DBMS are not adequate. For example, an intruder may steal tapes containing some data or tap a communication line. By storing and transmitting data in an encrypted form, the DBMS ensures that such stolen data is not intelligible to the intruder. Thus, encryption is a technique to provide privacy of data.

****

1. **Transitivity, reflexivity and augmentation properties of FDs**

Given that *X*, *Y*, and *Z* are sets of attributes in a relation *R*, one can derive several properties of functional dependencies. Among the most important are the following:

* **Reflexivity**: A1 A2 ... An → a subset of A1 A2 ... An
* **Augmentation**: A1 A2 ... An → B1 B2 ... Bm then

A1 A2 ... An C1 C2 ... Ck→B1 B2 ... Bm C1 C2 ... Ck

* **Transitivity**: – A1 A2 ... An → B1 B2 ... Bm and B1 B2 ... Bm → C1 C2 ... Ck

then 7 A1 A2 ... An → C1 C2 ... Ck

1. **BCNF and decomposition into BCNF**

We say a relation R is in BCNF if whenever X → Y is a nontrivial FD that holds in R, X is a superkey

* nontrivial means Y is not contained in X
* a superkey is any superset of a key (not necessarily a proper superset)

**Example:**

Drinkers (name, addr, beersLiked, manf, favBeer)

FD’s: name → addr favBeer, beersLiked → manf

* Only key is {name, beersLiked}
* In each FD, the left side is not a superkey
* Any one of these FD’s shows Drinkers is not in BCNF

**Decomposition into BCNF**

Given: relation R with FD’s F

* Look among the given FD’s for a BCNF violation X → Y
* If any FD following from F violates BCNF, then there will surely be an FD in F itself that violates BCNF
* Compute X +
* Not all attributes, or else X is a superkey

1. **Characterizing schedules based on recoverability**

When transactions are executing concurrently in an interleaved fashion, the order of execution of operations from the various transactions forms what is known as a transaction schedule (or history).  
A schedule (or history) S of n transactions T1, T2, …, Tn:  
It is an ordering of the operations of the transactions subject to the constraint that, for each transaction Ti that participates in S, the operations of T1 in S must appear in the same order in which they occur in T1.  
Note, however, that operations from other transactions Tj can be interleaved with the operations of Ti in S.

1. **Characterizing schedules based on serializibility**

DBMS must control concurrent execution of transactions to ensure read consistency, i.e., to  
avoid dirty reads etc.  
A (possibly concurrent) schedule S is serializable if it is equivalent to a serial schedule S0, i.e., S  
has the same result database state as S0.

1. **Transactions supports in SQL**

The definition of an SQL-transaction is that it is a logical unit of work and is guaranteed to be atomic. A single SQL statement is always considered to be atomic—either it completes execution without error or it fails and leaves the database unchanged.

With SQL, there is no explicit Begin\_Transaction statement. Transaction initiation is done implicitly when particular SQL statements are encountered. However, every transaction must have an explicit end statement, which is either a COMMIT or a ROLLBACK. Every transaction has certain characteristics attributed to it. These characteristics are specified by a SET TRANSACTION statement in SQL2. The characteristics are the *access mode,*the *diagnostic area size,*and the *isolation level.*

The **access mode**can be specified as READ ONLY or READ WRITE. The default is READ WRITE, unless the isolation level of READ UNCOMMITTED is specified, in which case READ ONLY is assumed. A mode of READ WRITE allows update, insert, delete and create commands to be executed. A mode of READ ONLY, as the name implies, is simply for data retrieval.

The **diagnostic area size**option, DIAGNOSTIC SIZE *n,*specifies an integer value *n,*indicating the number of conditions that can be held simultaneously in the diagnostic area. These conditions supply feedback information (errors or exceptions) to the user on the most recently executed SQL statement.

The **isolation level**option is specified using the statement ISOLATION LEVEL <isolation>, where the value for <isolation> can be READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, or SERIALIZABLE. The default isolation level is SERIALIZABLE, although some systems use as READ COMMITTED their default. The use of the term SERIALIZABLE here is based on not allowing violations that cause dirty read, unrepeatable read, and phantoms, and it is thus not identical to the way serializability.