ST. XAVIER’S COLLEGE

**(Affiliated to Tribhuvan University)**

**Maitighar, Kathmandu**

****

**Database Management System**

**TheoryAssignment #13**

**SUBMITTED BY:**

**Biraj Poudyal**

**013BSCCSIT015**

**SUBMITTED TO:**

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

Submission Date: 5th November 2015

**GRANT and REVOKE authorizations**

The SQL GRANT statement lets you grant explicit privileges to authorization IDs. The REVOKE statement lets you take them away. Only a privilege that has been explicitly granted can be revoked.

Granting privileges is very flexible. For example, consider table privileges. You can grant all the privileges on a table to an ID. Alternatively, you can grant separate, specific privileges that allow that ID to retrieve data from the table, insert rows, delete rows, or update specific columns. By granting or not granting those privileges on views of the table, you can effectively determine exactly what action an ID can or cannot take on the table.

You can use the GRANT statement to assign privileges as follows:

* Grant privileges to a single ID or to several IDs in one statement.
* Grant a specific privilege on one object in a single statement, grant a list of privileges, or grant privileges over a list of objects.
* Grant ALL, for all the privileges of accessing a single table or for all privileges that are associated with a specific package.

 The following examples show specific table privileges that you can grant to users.

* GRANT SELECT ON DEPT TO PUBLIC;

This statement grants SELECT privileges on the DEPT table. Granting the select privilege to PUBLIC gives the privilege to all users at the current server.

* GRANT UPDATE (EMPNO,DEPT) ON TABLE EMP TO NATZ;

This statement grants UPDATE privileges on columns EMPNO and DEPT in the EMP table to user NATZ.

* GRANT ALL ON TABLE EMP TO KWAN,ALONZO WITH GRANT OPTION;

This statement grants all privileges on the EMP table to users KWAN and ALONZO. The WITH GRANT OPTION clause allows these two users to grant the table privileges to others.

The same ID that grants a privilege can revoke it by issuing the REVOKE statement. If two or more grantors grant the same privilege to an ID, executing a single REVOKE statement does not remove the privilege for that ID. To remove the privilege, each ID that explicitly granted the privilege must explicitly revoke it.

Here are some examples of revoking privileges that were previously granted.

**Revoke example 1:**

* REVOKE SYSOPR FROM NICHOLLS;

This statement revokes SYSOPR authority from user NICHOLLS.

* REVOKE UPDATE ON EMP FROM NATZ;

This statement revokes the UPDATE privilege on the EMP table from NATZ.

* REVOKE ALL ON TABLE EMP FROM KWAN,ALONZO;

This statement revokes all privileges on the EMP table from users KWAN and ALONZO.

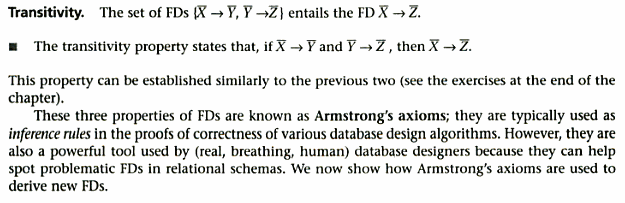
An ID with SYSADM or SYSCTRL authority can revoke privileges that are granted by other IDs.

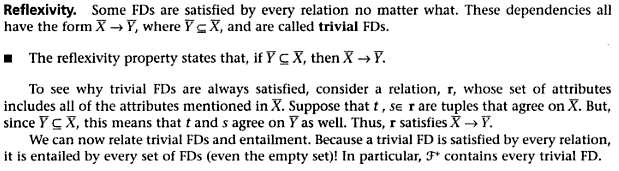
**Data encryption**

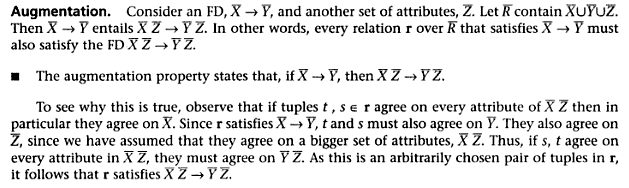
The Data Encryption Standard (DES) is an outdated symmetric-key method of data encryption. DES works by using the same key to encrypt and decrypt a message, so both the sender and the receiver must know and use the same private key. Once the go-to, symmetric-key algorithm for the encryption of electronic data, DES has been superseded by the more secure Advanced Encryption Standard (AES) algorithm.

Despite having reached the end of its useful life, the arrival of the Data Encryption Standard served to promote the study of cryptography and the development of new encryption algorithms. Until DES, cryptography was a dark art confined to the realms of military and government intelligence organizations. The open nature of DES meant academics, mathematicians and anyone interested in security could study how the algorithm worked and try to crack it. As with any popular and challenging puzzle, a craze - or in this case, a whole industry was born.

**Transivity, reflexivity & augmentation properties of FDs**

****

****

****

**BCNF & 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

§ Remember: nontrivial means Y is not contained in X

§ Remember, 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

**Decompose R Using X → Y**

§ Replace R by relations with schemas: 1. R1 = X + 2. R2 = R – (X + – X )

**Characterizing schedules based on Recoverability**

Transaction schedule or history:

 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.

Schedules classified on recoverability:

 Recoverable schedule:

 One where no transaction needs to be rolled back.

 A schedule S is recoverable if no transaction T in S commits until all transactions T’ that have written an item that T reads have committed.

 Cascadeless schedule:

 One where every transaction reads only the items that are written by committed transactions.

 Schedules requiring cascaded rollback:

 A schedule in which uncommitted transactions that read an item from a failed transaction must be rolled back.

 Strict Schedules:

 A schedule in which a transaction can neither read or write an item X until the last transaction that wrote X has committed.

**Characterizing schedules based on serializability**

Serial schedule:

 A schedule S is serial if, for every transaction T participating in the schedule, all the operations of T are executed consecutively in the schedule.

 Otherwise, the schedule is called nonserial schedule.

 Serializable schedule:

 A schedule S is serializable if it is equivalent to some serial schedule of the same n transactions.

 Result equivalent:

 Two schedules are called result equivalent if they produce the same final state of the database.

 Conflict equivalent:

 Two schedules are said to be conflict equivalent if the order of any two conflicting operations is the same in both schedules.

 Conflict serializable:

 A schedule S is said to be conflict serializable if it is conflict equivalent to some serial schedule S’.

 View equivalence:

 A less restrictive definition of equivalence of schedules

 View serializability:

 Definition of serializability based on view equivalence.

 A schedule is view serializable if it is view equivalent to a serial schedule.

**Transactions supports in SQL**

A single SQL statement is always considered to be atomic.

 Either the statement 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.

 Every transaction must have an explicit end statement, which is either a COMMIT or ROLLBACK.