ST. XAVIER’S COLLEGE

**(Affiliated to Tribhuvan University)**

**Maitighar, Kathmandu**

****

**Database Management System Assignment**

**SUBMITTED BY:**

**Utsav Luitel**

**013BSCCSIT046**

**SUBMITTED TO:**

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

**THEORY ASSIGNMENT**

Report on :

->GRANT and REVOKE authorizations

->Data encryption

->Transivity, Reflexivity and Augmentation properties of FDs

->BCNF and decomposition into BCNF

->Characterizing Schedule based on Recoverability

->Transaction supports in SQL.

1. 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.

Examples of grant privileges

The following examples show how to grant some system privileges, use privileges, and table privileges.

Grant example 1: To grant the privileges of system operator authority to user NICHOLLS, the system administrator uses the following statement:

GRANT SYSOPR TO NICHOLLS;

Assume that your business decides to associate job tasks with authorization IDs.

Grant example 2: In the following examples, PKA01 is the ID of a package administrator, and DBA01 is the ID of a database administrator. Suppose that the system administrator uses the ADMIN authorization ID, which has SYSADM authority, to issue the following GRANT statements:

* GRANT PACKADM ON COLLECTION GOLFS TO PKA01 WITH GRANT OPTION;

This statement grants PACKADM authority to PKA01. PKA01 acquires package privileges on all packages in the collection named GOLFS and the CREATE IN privilege on that collection. In addition, specifying WITH GRANT OPTION gives PKA01 the ability to grant those privileges to others.

* GRANT CREATEDBA TO DBA01;

CREATEDBA grants DBA01 the privilege to create databases, and DBA01 acquires DBADM authority over those databases.

* GRANT USE OF STOGROUP SG1 TO DBA01 WITH GRANT OPTION;

This statement allows DBA01 to use storage group SG1 and to grant that privilege to others.

* GRANT USE OF BUFFERPOOL BP0, BP1 TO DBA01 WITH GRANT OPTION;

This statement allows DBA01 to use buffer pools BP0 and BP1 and to grant that privilege to others.

Grant example 3: 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.

Examples of revoke privileges

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.

Revoke example 2: A user with SYSADM or SYSCTRL authority can issue the following statements:

* REVOKE CREATETAB ON DATABASE DB1 FROM PGMR01 BY ALL;

In this statement, the CREATETAB privilege that user PGMR01 holds is revoked regardless of who or how many people explicitly granted this privilege to this user.

* REVOKE CREATETAB, CREATETS ON DATABASE DB1 FROM PGMR01 BY DBUTIL1;

This statement revokes privileges that are granted by DBUTIL1 and leaves intact the same privileges if they were granted by any other ID.

1. Data encryption

**Data encryption** is the act of changing electronic information into an unreadable state by using algorithms or ciphers. Originally, **data encryption** was used for passing government and military information electronically.

The primary purpose of encryption is to protect the confidentiality of digital data stored on computer systems or transmitted via the Internet or other computer networks. Modern encryption algorithms play a vital role in the security assurance of IT systems and communications as they can provide not only confidentiality, but also the following key elements of security:

* Authentication: the origin of a message can be verified.
* Integrity: proof that the contents of a message have not been changed since it was sent.
* Non-repudiation: the sender of a message cannot deny sending the message.

1. Transitivity, Reflexivity and Augmentation properties of FDs

**Transitivity rule** − Same as transitive rule in algebra, if a → b holds and b → c holds, then a → c also holds. a → b is called as a functionally that determines b.

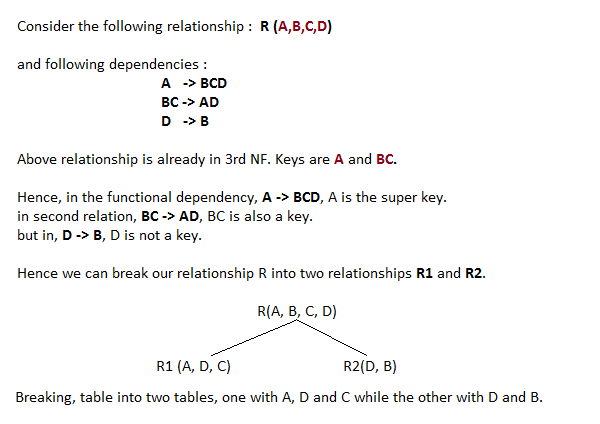
**Reflexive rule** − If alpha is a set of attributes and beta is\_subset\_of alpha, then alpha holds beta

**Augmentation rule** − If a → b holds and y is attribute set, then ay → by also holds. That is adding attributes in dependencies, does not change the basic dependencies.

1. BCNF and decomposition into BCNF

**Boyce and Codd Normal Form** is a higher version of the Third Normal form. This form deals with certain type of anamoly that is not handled by 3NF. A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF. For a table to be in BCNF, following conditions must be satisfied:

* R must be in 3rd Normal Form
* and, for each functional dependency ( X -> Y ), X should be a super Key.



1. Characterizing Schedule based on Serialibility.

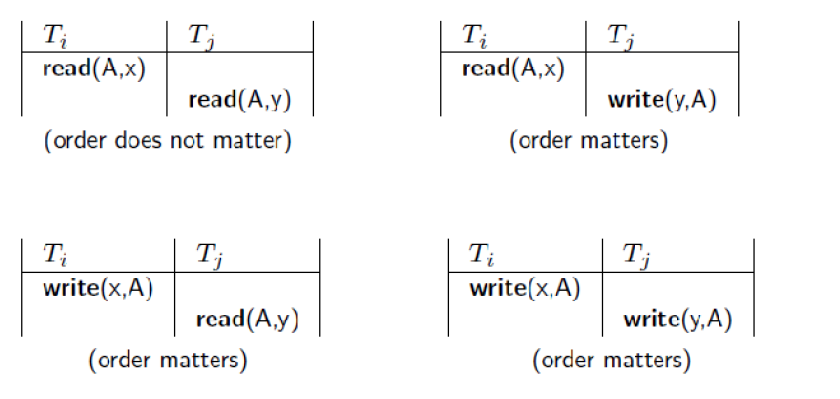
When transactions are executing concurrently in an interleaved fashion, the order of execution of operations from the various transactions forms what is known as a transaction schedule (orhistory). A schedule (or history) S of n transactions T1, T2, …, Tn:

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.

**Serializability:**

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., has the same result database state as S0.

**How to ensure serializability of concurrent transactions?**



1. Transaction 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. If a transaction executes at a lower isolation level than SERIALIZABLE, then one or more of the following three violations may occur:

1. **Dirty read:** A transaction may read the update of a transaction , which has not yet committed. If fails and is aborted, then would have read a value that does not exist and is incorrect.

2. **Nonrepeatable read:** A transaction may read a given value from a table. If another transaction later updates that value and reads that value again, will see a different value.

3. **Phantoms:** A transaction may read a set of rows from a table, perhaps based on some condition specified in the SQL WHERE-clause. Now suppose that a transaction inserts a new row that also satisfies the WHERE-clause condition used in, into the table used by. If is repeated, then will see a phantom, a row that previously did not exist.

Table summarizes the possible violations for the different isolation levels. An entry of "yes" indicates that a violation is possible and an entry of "no" indicates that it is not possible.