**ST. XAVIER’S COLLEGE**

**(Affiliated to Tribhuvan University)**

Maitighar, Kathmandu



**Database Management System**

**theory Assignment # 11**

**Submitted by:**

PRANESH DHUNJU SHRESTHA  
013BSCCSIT026

**Submitted to:**

|  |  |
| --- | --- |
| **Er. Sanjay Kumar Yadav**  Lecturer  St. Xavier’s College |  |

**Date of Submission:** November 5, 2015

* **GRANT and REVOKE authorization**

**Authorization ID’s**

A user is referred to by authorization ID, typically their login name. There is an authorization ID PUBLIC. Granting a privilege to PUBLIC makes it available to any authorization ID.

**Granting Privileges**

You have all possible privileges on the objects, such as relations, that you create. You may grant privileges to other users (authorization ID’s), including PUBLIC. You may also grant privileges WITH GRANT OPTION, which lets the grantee also grant this privilege.

**The GRANT Statement**

To grant privileges, say: GRANT ON TO ;

If you want the recipient(s) to be able to pass the privilege(s)

to others add: WITH GRANT OPTION.

**Example: GRANT**

Suppose you are the owner of Sells.

You may say: GRANT SELECT, UPDATE(price) ON Sells TO sally;

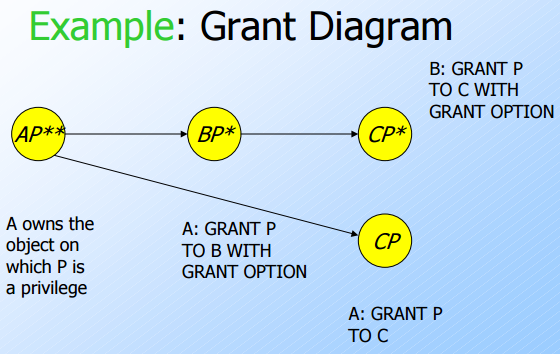
Now Sally has the right to issue any query on Sells and can update the price component only.

**Example: Grant Option**

Suppose we also grant: GRANT UPDATE ON Sells TO sally WITH GRANT OPTION;

Now, Sally not only can update any attribute of Sells, but can grant to others the privilege UPDATE ON Sells.

Also, she can grant more specific privileges like UPDATE(price)ON Sells.



**Revoking Privileges**

REVOKE ON FROM ; Your grant of these privileges can no longer be used by these users to justify their use of the privilege. But they may still have the privilege because they obtained it independently from elsewhere.

**REVOKE Options**

We must append to the REVOKE statement either:

1. CASCADE. Now, any grants made by a revokee are also not in force, no matter how far the privilege was passed.

2. RESTRICT. If the privilege has been passed to others, the REVOKE fails as a warning that something else must be done to “chase the privilege down.”

**Revoke example :**

* REVOKE SYSOPR FROM NICHOLLS;

This statement revokes SYSOPR authority from user NICHOLLS.

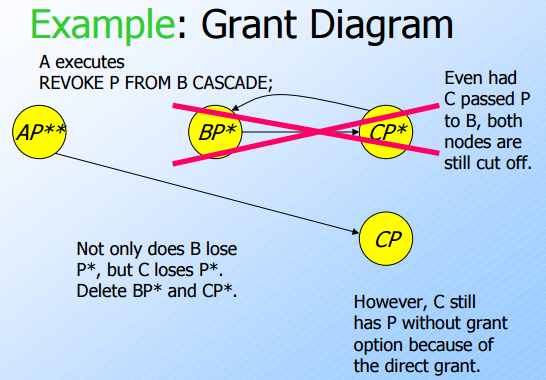
* REVOKE UPDATE ON EMP FROM NATZ;

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

* REVOKE ALL ON TABLE EMP FROM KWAN,ALONZO;

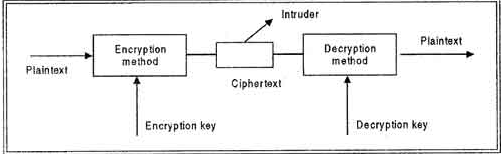
This statement revokes all privileges on the EMP table from users KWAN and ALONZO.

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



* **DATA ENCRYPTION**

A [DBMS](http://ecomputernotes.com/fundamental/what-is-a-database/advantages-and-disadvantages-of-dbms) can use encryption to protect [information](http://ecomputernotes.com/fundamental/information-technology/what-do-you-mean-by-data-and-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.



In encryption, the message to be encrypted is known as plaintext. The plaintext is transformed by a function that is parameterized by a key. The output of the encryption process is known as the cipher text. Ciphertext is then transmitted over the network. The process of converting the plaintext to ciphertext is called as Encryption and process of converting the ciphertext to plaintext is called as Decryption. Encryption is performed at the transmitting end and decryption is performed at the receiving end. For encryption process we need the encryption key and for decryption process we need decryption key as shown in figure. Without the knowledge of decryption key intruder cannot break the ciphertext to plaintext. This process is also called as Cryptography.

The basic idea behind encryption is to apply an encryption algorithm, which may' be accessible to the intruder, to the original data and a user-specified or DBA-specified encryption key, 'which is kept secret. The output of the algorithm is the encrypted version of the data. There is also a decryption algorithm, which takes the encrypted data and the decryption key as input and then returns the original data. Without the correct decryption key, the decryption algorithm produces gibberish. Encryption and decryption keys may be same or· different but there must be relation between the both which must me secret.

**Techniques used for Encryption**

There are following techniques used for encryption process:

• Substitution Ciphers

• Transposition Ciphers

**Substitution Ciphers:** In a substitution cipher each letter or group of letters is replaced by another letter or group of letters to mask them For example: a is replaced with D, b with E, c with F and z with C. In this way *attack*becomes *DWWDFN.*The substitution ciphers are not much secure because intruder can easily guess the substitution characters.

**Transposition Ciphers:** Substitution ciphers preserve the order of the plaintext symbols but mask them-;-The transposition cipher in contrast reorders the letters but do not mask them. For this process a key is used. For example:*iliveinqadian*may be coded as *divienaniqnli.*The transposition ciphers are more secure as compared to substitution ciphers.

* **FUNCTIONAL DEPENDENCY INFERENCE RULES**

**Transitivity**: If A functionally determines B and B functionally determine C then A functionally determines C. For example:

{name, location} -> {initials} (as {name, location} functionally determines {name} and {name} functionally determines {initials})

**Reflexivity**: If B is a subset of A then A functionally determines B.

For example:

{name, location} -> {name}

**Augmentation**: If B is a subset of A and C functionally determines D then A and C functionally determine B and D. For example:

{name, location} and {birthdate, time} ->{name} and {age}

(as {name} is a subset of {name, location} and {birthdate, time} functionally determines {age})

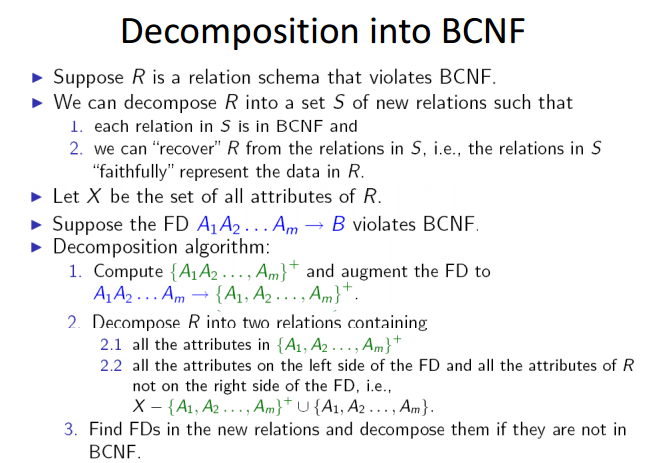
* **BCNF Decomposition into BCNF**

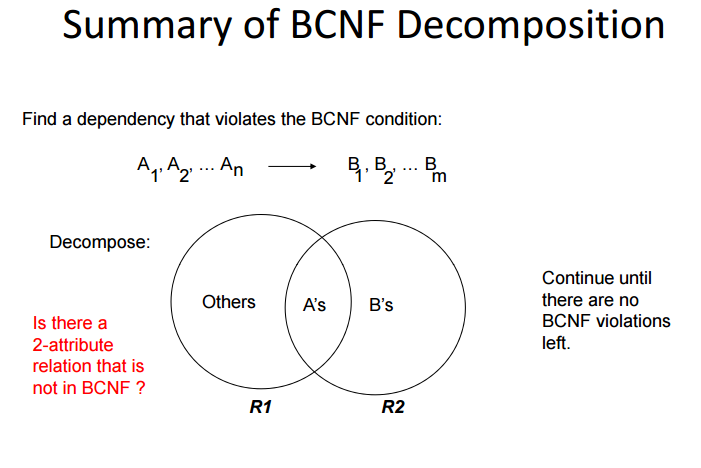
### The Boyce-Codd normal form

A relational schema R is considered to be in **Boyce–Codd normal form (BCNF)** if, for every one of its dependencies X → Y, one of the following conditions holds true:

* X → Y is a [trivial functional dependency](http://www.vertabelo.com/blog/functional-dependencies) (i.e., Y is a subset of X)
* X is a [superkey](http://www.vertabelo.com/blog/on-keys) for schema R

Informally the Boyce-Codd normal form is expressed as “Each attribute must represent a fact about the key, the whole key, and nothing but the key.”





* **TRANSACTION SUPPORT 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.

* **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**
* 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’.
* Being serializable is not the same as being serial.
* Being serializable implies that the schedule is a correct schedule.
  + It will leave the database in a consistent state.
  + The interleaving is appropriate and will result in a state as if the transactions were serially executed, yet will achieve efficiency due to concurrent execution.
* Serializability is hard to check.
  + Interleaving of operations occurs in an operating system through some scheduler
  + Difficult to determine beforehand how the operations in a schedule will be interleaved.

**Reference :**

[1] ”Transitive, reflexivity properties of FDs”. Internet: http://class- editor.sourceforge.net/docs/Databases.pdf

[2] <http://courses.cs.vt.edu/~cs4604/Fall08/lectures/lecture14.pdf>

[3] https://www.classle.net/book/transaction-support-sql