ST. XAVIER’S COLLEGE

**(Affiliated to Tribhuvan University)**

**Maitighar, Kathmandu**

****

**Database Management System**

**TheoryAssignment #13**

**SUBMITTED BY:**

**Apil Neupane**

**013BSCCSIT008**

**SUBMITTED TO:**

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

Submission Date: 5th November 2015

**GRANT AND REVOKE AUTHORIZATION:**

The Grant and Revoke commands allow system system administrators to create users. Grant is implemented in MySQL Version 3.22.11 or later, for earlier versions, the Grant statment does nothing.

**GRANT:**

The GRANT statement is used to give permissions to a user or role. By using the GRANT statement, it is possible to assign permissions to both statements as well as objects. You can use the GRANT statement with the WITH GRANT OPTION clause to permit the user or role receiving the permission to further grant/revoke access to other accounts

**REVOKE:**

The REVOKE statement is used to remove a previously granted or denied permission from a user in the current database. You can use the REVOKE statement to remove both statements and objects permissions. You can specify the GRANT OPTION FOR clause with the REVOKE statement to remove the WITH GRANT OPTION permissions. Therefore, the user will have the objects permissions, but cannot grant the permissions to other users. Specify the CASCADE clause along with the WITH GRANT OPTION clause, if the permissions being revoked were originally granted using the WITH GRANT OPTION setting.

**DATA ENSCRYPTION:**

Data encryption refers to mathematical calculations and algorithmic schemes that transform plaintext into cyphertext, a form that is non-readable to unauthorized parties. The recipient of an encrypted message uses a key which triggers the algorithm mechanism to decrypt the data, transforming it to the original plaintext version.

**Types of Data Encryption**

There are many different types of data encryption, but not all are reliable. In the beginning, 64-bit encryption was thought to be strong, but was proven wrong with the introduction of 128-bit solutions. AES (Advanced Encryption Standard) is the new standard and permits a maximum of 256-bits. In general, the stronger the computer, the better chance it has at breaking a data encryption scheme.

Data encryption schemes generally fall in two categories: symmetric and asymmetric. AES, DES and Blowfish use symmetric key algorithms. Each system uses a key which is shared among the sender and the recipient. This key has the ability to encrypt and decrypt the data. With asymmetric encryption such as Diffie-Hellman and RSA, a pair of keys is created and assigned: a private key and a public key. The public key can be known by anyone and used to encrypt data that will be sent to the owner. Once the message is encrypted, it can only be decrypted by the owner of the private key. Asymmetric encryption is said to be somewhat more secure than symmetric encryption as the private key is not to be shared.

Strong encryption like SSL (Secure Sockets Layer) and TLS (Transport Layer Security) will keep data private, but cannot always ensure security. Websites using this type of data encryption can be verified by checking the digital signature on their certificate, which should be validated by an approved CA (Certificate Authority).

**Transivity, Reflexivity and Augumenteation 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, usually called Armstrong's axioms:

• Reflexivity: If Y is a subset of X, then X → Y

• Augmentation: If X → Y, then XZ → YZ

• Transitivity: If X → Y and Y → Z, then X → Z

"Reflexivity" can be weakened to just X \rightarrow \varnothing, i.e. it is an actual axiom, where the other two are proper inference rules, more precisely giving rise to the following rules of syntactic consequence:

|- X ->ф

X -> Y |- XZ -> YZ

X ->Y, Y ->Z |-X ->Z.

These three rules are a sound and complete axiomatization of functional dependencies. This axiomatization is sometimes described as finite because the number of inference rules is finite, with the caveat that the axiom and rules of inference are all schemata, meaning that the X, Y and Z range over all ground terms (attribute sets).

**BCNF AND DECOMPOSITIONN INTO BCNF:**

It is defined as a relation schema R, with FD set, F is in BCNF if:

For all nontrivial X🡪Y in F+:

X🡪R (i.e. X a superkey)

Example:

R= R1 U R2

R1 = (A, B) , R2 = (B,C)

F = (A🡪B, B🡪C)

Are R1, R2 in BCNF?

Ans: Yes, both non-trivial FDs define a key in R1, R2

**Decomposition Algorithm**

Algorithm BCNF(R: relation, F: FD set)

Begin

1. Compute F+

2. Result 🡪 {R}

3. While some Ri in Result not in BCNF Do

a. Chose (X🡪Y) in F+ s.t.

(X🡪Y) covered by Ri

X -/-> Ri  ( X not a superkey for Ri )

b. Decompose Ri on (X🡪Y)

Ri1  🡨 X U Y

Ri2 🡨 Ri - Y

c. Result 🡨 Result - {Ri} U { Ri1, Ri2}

4. return Result

End

**BCNF Decomposition**

Example:

R= (A, B, C, D)

F = (A🡪B, AB🡪D, B🡪C)

Decompose R into BCNF

Ans: Fc = {A 🡪BD, B🡪C}

R=(A, B, C, D)

B🡪 C is covered by R and B not a superkey

**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.

**Transaction Support 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.

**Sample SQL transaction:**

EXEC SQL whenever sqlerror go to UNDO;

 EXEC SQL SET TRANSACTION

READ WRITE

DIAGNOSTICS SIZE 5

ISOLATION LEVEL SERIALIZABLE;

 EXEC SQL INSERT

INTO EMPLOYEE (FNAME, LNAME, SSN, DNO, SALARY)

VALUES ('Robert','Smith','991004321',2,35000);

EXEC SQL UPDATE EMPLOYEE

SET SALARY = SALARY \* 1.1

WHERE DNO = 2;

EXEC SQL COMMIT;

GOTO THE\_END;

UNDO: EXEC SQL ROLLBACK;

THE\_END: ...