ST. XAVIER’S COLLEGE

**(Affiliated to Tribhuvan University)**

**Maitighar, Kathmandu**

****

**Database Management System**

**Assignment # 13**

**SUBMITTED BY:**

Anlisha Maharjan

013BSCCSIT006

**SUBMITTED TO:**

|  |  |
| --- | --- |
| **Er. Sanjay Kr. Yadav**  **Lecturer** |  |

Submission Date: 05th November 2015

**GRANT AND REVOKE AUTHORIZATIONS**

The GRANT statement is used to grant specific permissions to users. The REVOKE statement is used to revoke permissions. The grant and revoke privileges are: 

* DELETE
* EXECUTE
* INSERT
* SELECT
* REFERENCES
* TRIGGER
* UPDATE

**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The Data Encryption Standard (DES) is an outdated symmetric-key method of data encryption.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 Once the go-to, symmetric-key algorithm for the encryption of electronic data, DES has been superseded by the more secure Advanced Encryption Standard (AES) algorithm.

**PROPERTIES OF FD’s**

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](https://en.wikipedia.org/wiki/Armstrong%27s_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](https://en.wikipedia.org/wiki/Axiom), where the other two are proper [inference rules](https://en.wikipedia.org/wiki/Inference_rules), more precisely giving rise to the following rules of syntactic consequence.

\vdash X \rightarrow \varnothing  
X \rightarrow Y \vdash XZ \rightarrow YZ  
X \rightarrow Y, Y \rightarrow Z \vdash X \rightarrow Z.

These three rules are a [sound](https://en.wikipedia.org/wiki/Soundness) and [complete](https://en.wikipedia.org/wiki/Completeness_(logic)) 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](https://en.wikipedia.org/wiki/Schema_(logic)), meaning that the X, Y and Z range over all ground terms (attribute sets).

**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 super key

Remember: nontrivial means Y is not contained in X

Remember, a super key is any superset of a key (not necessarily a proper superset).

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

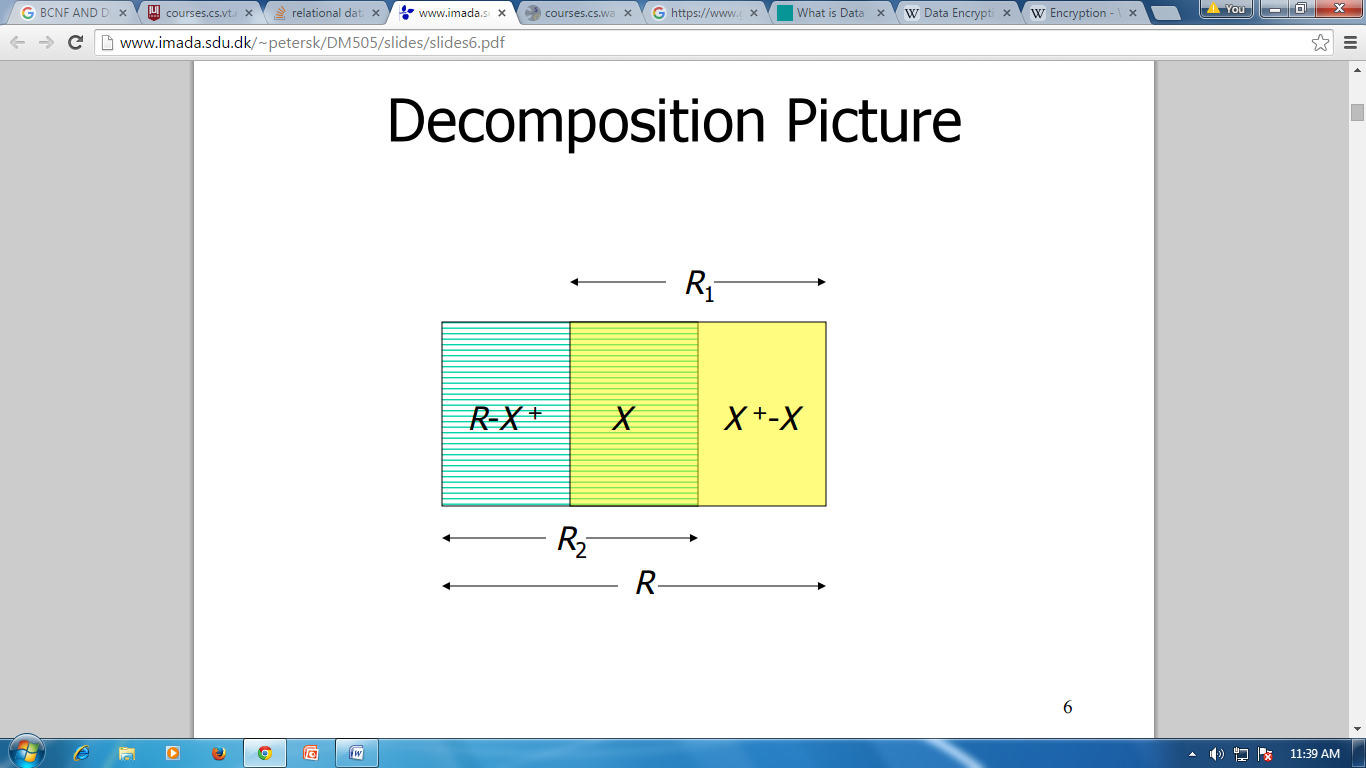
Look among the given FD’s for a BCNF violation X → Y

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

**DECOMPOSITION PICTURE**



**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.

**CHARACTERIZING SCHEDULES BASED ON 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., S has the same result database state as S0.

**TRANSACTIONS 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 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. No repeatable read**: A transaction may read a given value from a table.

**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.