**ST.XAVIER’S COLLEGE**

**MAITIGHAR, KATHMANDU**

****

**DATABASE MANAGEMENT SYSTEM**

**THEORY ASSIGNMENT# 11**

**SUBMITTED BY:**

Shaurabh Chapagain

013BSCCSIT037

**SUBMITTED TO:**

|  |  |  |
| --- | --- | --- |
|  | **Signature** | **Remarks** |
| Er. Sanjay Kumar Yadav  Lecturer  Department of Computer Science |  |  |

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

**GRANT and REVOKE authorizations**

**GRANT Statement**

The GRANT Statement grants access privileges for database objects to other users. It has the following general format:

GRANT privilege-list ON [TABLE] object-list TO user-list

Privilege-list is either ALL PRIVILEGES or a comma-separated list of properties: SELECT, INSERT, UPDATE, DELETE. object-list is a comma-separated list of table and view names. user-list is either PUBLIC or a comma-separated list of user names.

The GRANT statement grants each privilege in privilege-list for each object (table) in object-list to each user in user-list. In general, the access privileges apply to all columns in the table or view, but it is possible to specify a column list with the UPDATE privilege specifier:

UPDATE [ ( column-1 [, column-2] ... ) ]

If the optional column list is specified, UPDATE privileges are granted for those columns only. The user-list may specify PUBLIC. This is a general grant, applying to all users (and future users) in the catalog. Privileges granted are revoked with the REVOKE Statement. The optional specificier WITH GRANT OPTION may follow user-list in the GRANT statement. WITH GRANT OPTION specifies that, in addition to access privileges, the privilege to grant those privileges to other users is granted.

GRANT Statement Examples:

GRANT SELECT ON s,sp TO PUBLIC

GRANT SELECT,INSERT,UPDATE(color) ON p TO art,nan

GRANT SELECT ON supplied\_parts TO sam WITH GRANT OPTION

**REVOKE Statement**

The REVOKE Statement revokes access privileges for database objects previously granted to other users. It has the following general format:

span>

REVOKE privilege-list ON [TABLE] object-list FROM user-list

The REVOKE Statement revokes each privilege in privilege-list for each object (table) in object-list from each user in user-list. All privileges must have been previously granted. The user-list may specify PUBLIC. This must apply to a previous GRANT TO PUBLIC.

REVOKE Statement Examples:

REVOKE SELECT ON s,sp FROM PUBLIC

REVOKE SELECT,INSERT,UPDATE(color) ON p FROM art,nan

REVOKE SELECT ON supplied\_parts FROM sam

**Data encryption**

Encryption is the conversion of electronic [data](http://searchdatamanagement.techtarget.com/definition/data) into another form, called [ciphertext](http://searchcio-midmarket.techtarget.com/definition/ciphertext), which cannot be easily understood by anyone except authorized parties. The primary purpose of encryption is to protect the confidentiality of digital data stored on computer systems or transmitted via the [Internet](http://searchwindevelopment.techtarget.com/definition/Internet) or other computer [networks](http://searchnetworking.techtarget.com/definition/network). Modern encryption [algorithms](http://whatis.techtarget.com/definition/algorithm) 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](http://searchsecurity.techtarget.com/definition/authentication): the origin of a message can be verified.
* [Integrity](http://searchdatacenter.techtarget.com/definition/integrity): proof that the contents of a message have not been changed since it was sent.
* [Non-repudiation](http://searchsecurity.techtarget.com/definition/nonrepudiation): the sender of a message cannot deny sending the message.

**Transitivity, Reflexivity and Augmentation properties of FDs**

**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 and decomposition into BCNF**

**BCNF**

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 [super-key](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.”

**Decomposition into BCNF**

To go from non-BCNF normal form to BCNF, you must decompose your table using these two steps.

1. Find a nontrivial functional dependency X → Y which violates the BCNF condition (where the X is not a super-key)
2. Split your table in two tables:
   * one with attributes XY (all attributes from the dependency),
   * one with X attributes together with the remaining attributes from the original relation

Then you keep repeating the decomposition process until all of your tables are in BCNF. After sufficient iterations you have a set of tables, each in BCNF, such that the original relation can be reconstructed.

**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.

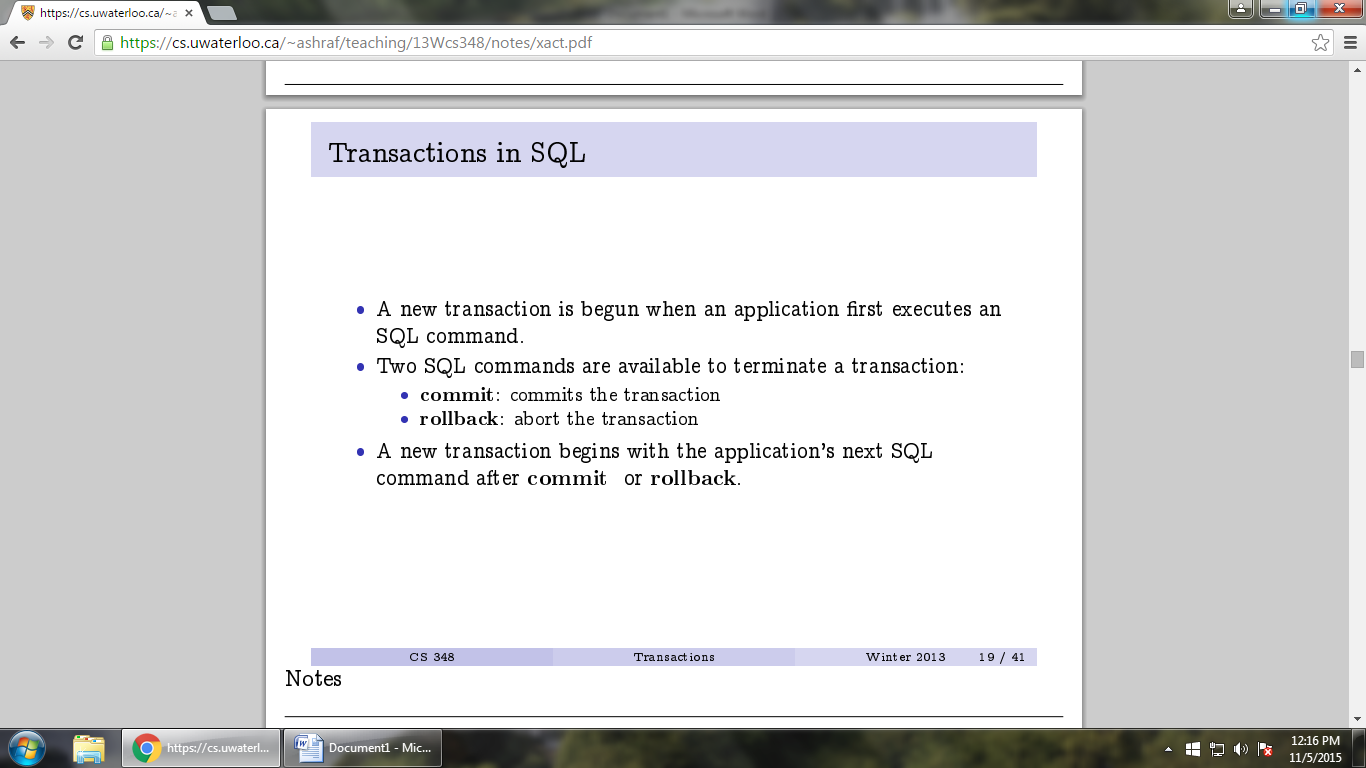
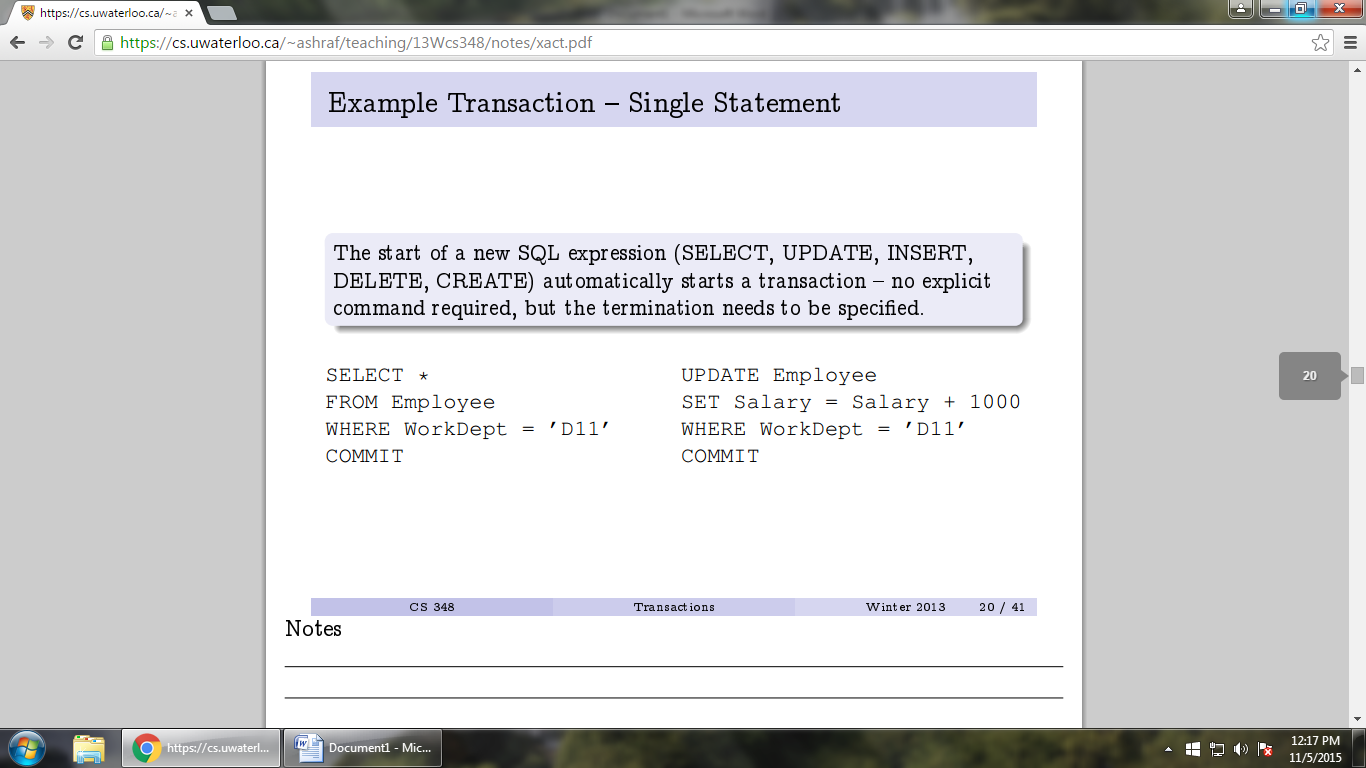
* **Strict Schedules**:

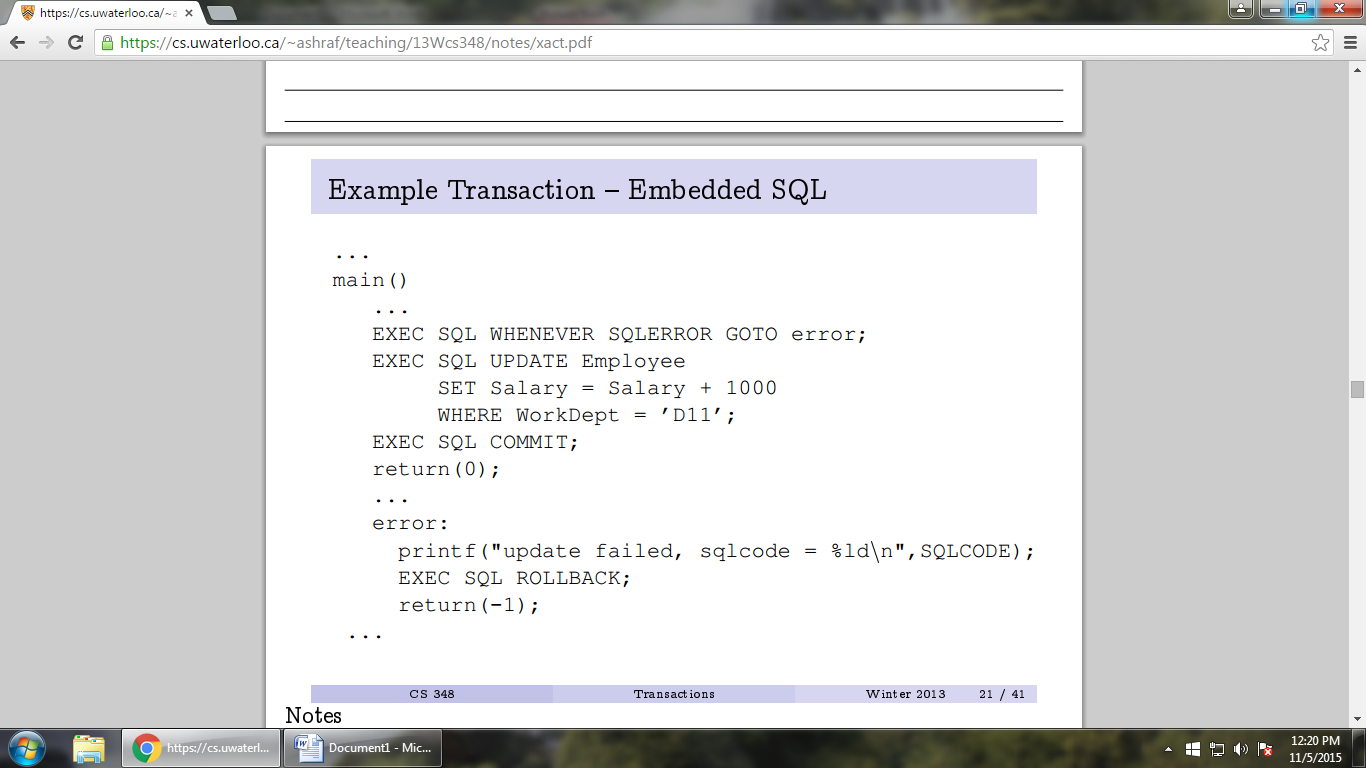
A schedule in which a transaction can neither read or write an item X until the last transaction that wrote X has committed.

**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 non-serial 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’.
* 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.

**Transactions supports in SQL**

** **

****