SUMMARY OF TRANSACTION

Transaction in a database system is similar to database storage and querying, transaction is something that is finished by the database system, thus we do not have to care much about it when applying the knowledge of database and designing applications. However, the concept of transaction contains a lot of important thoughts, and such thoughts can benefit us in other aspects well.

1. Basic Concepts

Transaction is a set of operations which forms a single logical working unit. In another word, a transaction contains a serial of operations on the database, and these operations must form an intact logical unit, which indicates that the unit ought to be conceptually reasonable to human being. Basically, the transaction has following properties:

Atomicity: it is an indication of “all or nothing” property of transaction. All operations in a transaction must be displayed correctly in the database; otherwise nothing will be done to change the data.

Consistency: this requires the database should be in a stable state. No matter what kind of transaction has been implemented, the database must remain consistent.

Isolation: this property ensures that all transactions will not feel like there are other parallel transactions in the system. It is similar to the transparency in DDBS or encapsulation to some extent.

Durability: as long as one transaction is completed, it makes a permanent alternation to the database even if the system has the probability to break down.

To be precise, a transaction includes reading and writing data, which can be summarized as the interaction with the database; and operations on the data when it is in the main memory. Now that the operations on the data is done in the main memory, it does not involves in the interaction with database, thus the study on transaction can be simplified into two operations: read data from database, and write data to database.

2. Concurrency

(1) Necessity of Concurrency

When a large number of transactions have to be done, it is still possible to force the implementing these transaction in a serialized way. But from the definition of transaction, it is easy to figure out that some operations in a transaction involve in I/O, and others involves in the computation in CPU. Since I/O and CPU can work in a parallel way, several transactions can also be implemented this way. Concurrency can increase the throughput, the transactions been implemented in a

unit of time. Also, the utilization of CPU and hard disk is also improved, while the average response time for transactions can be reduced.

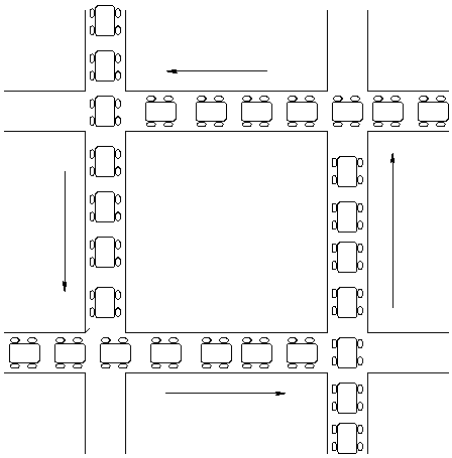
(2) Concurrency Control

One approach to concurrency control is to ensure the isolation of transactions. In this case, the approach of data should be exclusive. One method is the lock-based protocol. There are many kinds of locks, and the widely used ones include:

Shared-mode lock (S): a transaction with this kind of lock can only read certain data, but cannot write to the data. Thus, this lock ensures each transaction can get the unchanged, correct data.

Exclusive-mode lock (X): a transaction with this kind of lock can both read and write data. This ensures that when one transaction is finished or the data is unlocked, other transactions cannot read or write the certain data, thus isolates the data.

Properly lock and unlock the data been read and written, concurrency control can be accomplished. However, there are still other problems left, for example, if two transactions are waiting for each other and applying for locks, neither of them can continue implementing. This is called **deadlock**. The following picture gives an illustration about deadlock. There are discussions about such problems, but I will not cover this part here.



(3) Further Discussion about Concurrency

As Mr. Koo mentioned in the lecture, concurrency is an important concept but uncovered by our current curriculum. In computer science, concurrency is a property of systems in which several computations are executing simultaneously, and potentially interacting with each other. A number of mathematical models have been developed for general concurrent computation including, among which is the Petri net, a topic discussed in the lecture last week.

Another concurrency model is **SCOOP** (Simple Concurrent Object Oriented Programming) for the Eiffel programming language, created by Bertrand Meyer. SCOOP works by allowing references to certain objects to be declared as separate. Besides the concept of separateness, SCOOP applies the principle of **design by contract (DbC)** to synchronize access to separated resources. It covers the idea of preconditions and postconditions, all can be found in the lecture by Idea Factory.

Source: http://toyhouse.cc/profiles/blogs/summary-of-transaction