

Programme Name: \_\_\_\_\_\_\_\_**BCS HONS**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Course Code: \_\_**CSC 1403**\_\_\_\_\_\_\_\_

Course Name: \_\_\_\_\_\_\_\_**Database concept**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Open Book Examination**

Date of Submission: \_\_\_\_\_\_**10/9/2020**\_\_\_\_\_\_\_\_\_\_\_\_\_

**Submitted By: Submitted To:**

Student Name**: CSC 1403\_Dipesh Tha Shrestha** Faculty Name**: Amar Subedi**

IUKL ID: **041902900028** Department**: LMS**

Semester**: Second Semester**

Intake**: September 2019**

SECTION A

1. Business today concurrent execution of transaction as opposed to the serial execution. For example, Nepal Airlines operates a booking system for its tickets reservation. ACID (Atomicity, Consistency, Isolation, and Durability) is a term used to describe the database transaction. In order to maintain database consistency, concurrency control is a requirement in Nepal Airlines’ transaction management.
2. Briefly explain the ACID properties that a transaction should possess and describe how each might relate to the design of transactions within the booking system.

**Answer:**

A **transaction** is a single logical unit of work which accesses and possibly modifies the contents of a database. Transactions access data using read and write operations.  
In order to maintain consistency in a database, before and after the transaction, certain properties are followed. These are called **ACID** properties.

* **Atomicity**: This property ensures that either all the operations of a transaction reflect in database or none. Let’s take an example of banking system to understand this: Suppose Account **A**has a balance of 400$ & **B**has 700$. Account **A**is transferring 100$ to Account **B**. This is a transaction that has two operations a) Debiting 100$ from A’s balance b) Creating 100$ to B’s balance. Let’s say first operation passed successfully while second failed, in this case A’s balance would be 300$ while B would be having 700$ instead of 800$. This is unacceptable in a banking system. Either the transaction should fail without executing any of the operation or it should process both the operations. The Atomicity property ensures that.
* **Consistency**: To preserve the consistency of database, the execution of transaction should take place in isolation (that means no other transaction should run concurrently when there is a transaction already running). For example, account A is having a balance of 400$ and it is transferring 100$ to account B & C both. So we have two transactions here. Let’s say these transactions run concurrently and both the transactions read 400$ balance, in that case the final balance of A would be 300$ instead of 200$. This is wrong. If the transaction were to run in isolation, then the second transaction would have read the correct balance 300$ (before debiting 100$) once the first transaction went successful.
* **Isolation**: For every pair of transactions, one transaction should start execution only when the other finished execution. I have already discussed the example of Isolation in the Consistency property above.
* **Durability**: Once a transaction completes successfully, the changes it has made into the database should be permanent even if there is a system failure. The recovery-management component of database systems ensures the durability of transaction.

1. Describe with examples, TWO (2) types of problems that can occur in a multi user environment when concurrent access to the database is allowed.

Answer:

Concurrent access to a RDBMS is very common and a huge amount of attention must be paid to designing your schema with this in mind. There are different kinds of problems depending on if the database is read only versus read/write. Even a read only database has challenges when concurrent reading happens with very different kinds of queries. There is no way for me to cover all the possibilities here which change depending on your schema.

Let me give you an example though of a poorly designed read write database where concurrency was not designed for that I actually had the experience of working with and trying to fix. This was an online store that was updating inventory from the website as it was sold and updating inventory from the distribution center as it was restocked. The problem here was that programmers didn't use the right kind of transaction monitoring in both cases so the inventory counts were always wrong. This is a characteristic of concurrency problem:

1. Program A gets the count of item N in inventory say it is 20.
2. Program B gets the count 20 of item N in inventory.
3. Program A increments the count of item N in inventory to 21.
4. Program B decrements the count of item N in inventory to 19.
5. Program A updates the database for item N to 21.
6. Program B updates the database for item N to 19.

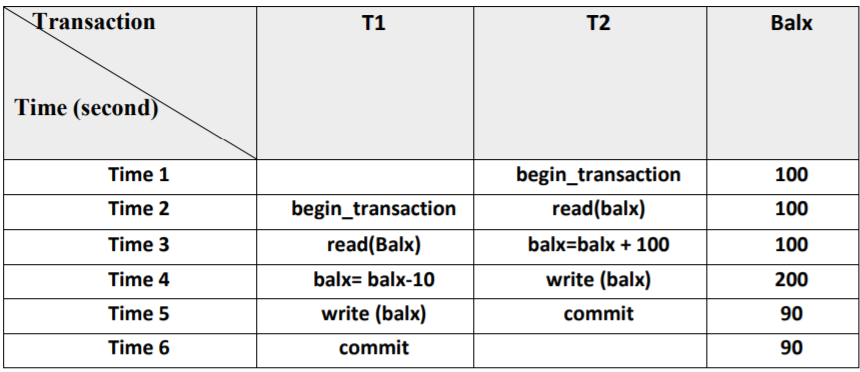
As you can see the database now counts 1 less than the actual physical items in inventory. This is just one kind of problem. Obviously this could be fixed with better programming but it's a typical problem.

Most concurrency problems can kill database performance, but the kind of thing I described above is a real disaster for a company.

When developing multi-user systems, it is important to consider the desired outcome when two or more users or services simultaneously try to access the same data. The most serious problems occur when two users try to modify an item at the same the same time or when one user is editing information whilst others read the original data.

As an example, consider a database that contains all of a company's customer information. If two users simultaneously open the same customer record, make changes to the information and then save their modifications, several data integrity issues are possible.

1. Consider the transactions (Table 1) shown below. Are the transaction below affected by the lost update problem? Explain your answer in details



Answer:

Yes, the transaction below will certainly be affected with lost update problem. At time 3, T2 is adding 100 in the initial balx and before he commits the translation, at time 4 T1 reduce the balx by 10. Now for t2, the balance would be 200 as he has added 100 to the initial transaction, but after reduced by t1, the balx become 90 and at Time4, T2 write the balx and at T5 T2 commit it. The error here occurred due to the interference of next transaction before committing the transaction.

Let transactionsT1 and T2 are concurrently executing, T1 is withdrawing $10 from an account with balance balx, initially $100, and T2 is depositing $100 into the same account. (i.e.Balance=100-10+100=190)

• The loss of T2‘s update is avoided by preventing T1 from reading the value of balx until after T2‘ s update has been completed

• T1 overwriting the previous updates(i.e. 200) to write $90 at t5 therby ‗losing‘ the $100 previously added to the balance

Section B

1. Consider the following information about a university database:

• Professors have an SSN, a name, an age, a rank, and a research specialty.

• Projects have a project number, a sponsor name (e.g., NSF), a starting date, an ending date, and a budget.

• Graduate students have an SSN, a name, an age, and a degree program (e.g., M.S. or Ph.D.).

• Each project is managed by one professor (known as the project's principal investigator).

• Each project is worked on by one or more professors (known as the project's coinvestigators). • Professors can manage and/or work on multiple projects.

• Each project is worked on by one or more graduate students (known as the project's research assistants).

• When graduate students work on a project, a professor must supervise their work on the project. Graduate students can work on multiple projects, in which case they will have a (potentially different) supervisor for each one.

• Departments have a department number, a department name, and a main office.

• Departments have a professor (known as the chairman) who runs the department.

• Professor’s work in one or more departments, and for each department that they work in, a time percentage is associated with their job.

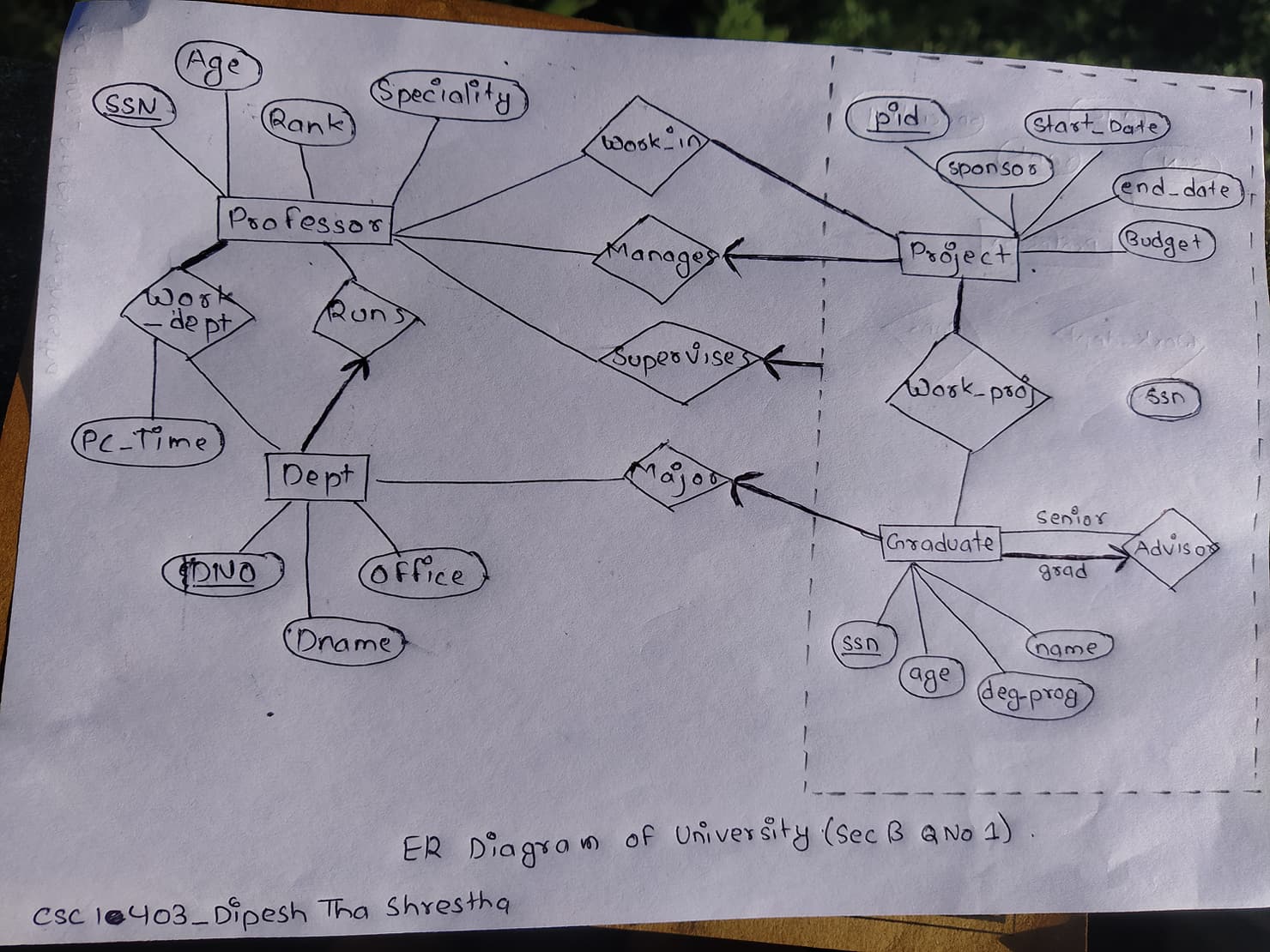
• Graduate students have one major department in which they are working on their degree.

• Each graduate student has another, more senior graduate student (known as a student advisor) who advises him or her on what courses to take.

Design and draw an ER diagram that captures the information about the university. Use only the basic ER model here; that is, entities, relationships, and attributes. Be sure to indicate any key and participation constraints.

Answer :

The ER diagram that captures the information about the university is given below:



1. There are FOUR (4) relations in the following schema. The primary key attributes are underlined. Using the relational scheme, write Data Manipulation (DML) statements for question (a)-(f).

Toy ( ToyNum, Name, Price, AgeGroup, NumInStock, Sold, ManID) Customer ( CustNum, CustName, Address, NumChild, Ages, LastOrderDate) Manufactures ( ManID, ManName, Address, Phone, SalesContact) Orders ( OrderNum, CustNum, ToyNum, Quantity, DateOrdered, DeliveryAddress)

1. Retrieve all the name and price of all toys made by Fisher Price.

**select t.Name, t.Price**

**from Toy as t inner join Manufacturer as m on t.ManID= m.ManID**

**where m.ManName= “Fisher Price”;**

1. Retrieve the toy types manufactured by each manufacturer.

**select m.ManName, count(t.ToyNum) as Total\_toy\_types**

**from Toy as t inner join Manufacturer as m on t.ManID= m.ManID ;**

c. List all customers aged between 20 to 30 and having at least 2 children in descending order of number of children.

**select \* from Customer where**

**Ages > 20 and Ages <30**

**and NumChild >= 2**

**order by Numchild desc;**

1. Change the manufacturer name of M333 to Pensonic.

**update Manufacturer set ManName= “Pensonic” where ManID= “M333”;**

e. Retrieve the customer name who have ordered toys from manufactured id M222.

**select c.CustName from Customer as c**

**inner join Orders as o on c.CustNum = o.CustNum**

**inner join Toy as t on o.ToyNum = t.ToyNum**

**inner join Manufacturer as m on t.ManID = m.ManID**

**where m.ManID=”M222”;**