BAN 610 Problem set 1 – Normalization, ERD, Consistency, and Recovery

Edit your submission in this word document, attaching the screenshots of the codes used for each question. Include narrative descriptions, outputs screenshot, or short answers when requested.

**Task 1**

Suppose we are processing 100 data records on a CPU cache. Each data record is 32 bytes in size. And each I/O operation will fetch a 64 byte cache line. Suppose the 100 data records on the cache are contiguously stored.

In this case, is sequential reading faster than random reading? How much faster?

What if each data record is 64 bytes in size instead of 32 bytes?

**Answer:** In the random case, we are reading 64 bytes for every 32 bytes we need, so we expect to max out the throughput at least two times sooner. This means that sequential reading is up to two times faster than random reading.

If each data record is 64 bytes in size instead of 32 bytes then sequential and random reading would perform the same since the size of each record is the same size as the cache line no matter how we are accessing the CPU cache.

**Task 2 -** Suppose you are hosting a database server. A table in your database is accessed 20 times / sec on average. The size of the table is 1 GB. The hard drive allocated to store this table cost $10, and the throughput of the hard drive is 100 MB / sec.

The memory of your server cost $20 / MB.

Should you keep the table in the hard disk or in the memory?

Hint: calculate or identify what is the D, I, X, M, and P in the five-minutes rule formula.

Database has a hard drive (disk) and memory (RAM)

Table size: 1000MB (1GB)

Cost of Disk (**D**) = $10

Disk operations throughput (**I**) = 100MB / 1000 MB = .1 MB

Page accessed frequency (**X)** = 1 sec/20 accesses = .05

$10/.1/.05 = $2,000

RAM /Mem Cost (**M**) = $20/MB

Table Size (**P**) = 1000MBs

$20/MB \* 1000MB = $20,000

**Answer:** Since storing this table on disk is much more financially feasible than storing the table in memory.

**Task 3**

We are looking to optimize our database performance by delaying the output of frequently used object.

In the following chart, the object X is frequently accessed by many transactions, and we must repeatedly read X from the disk to memory, update the value of X in memory, and then flush the updated value of X to the disk.

Diagram

Description automatically generated

Please design a better workflow to reduce the number of I/Os for frequently accessed values (considering using pseudocodes). Hint: think about pre-scanning the transactions before executing it, and based on that, optimize the transactions.

**Solution: Caching + Redo logging with checkpoints**

**READING DATA**

Object X is stored in memory in a cache.

A transaction needs the object X, the database checks if it's in the cache.

If the object is in the cache, it's returned directly to the transaction, without needing to perform a disk I/O operation.

Else if the object is not in the cache, the database retrieves it from disk and stores it in the cache for future use.

**UPDATING WITH CHECKPOINTS - Avoid pre-scanning**

If transactions updates the object X

Generate a redo log for current transactions on object X

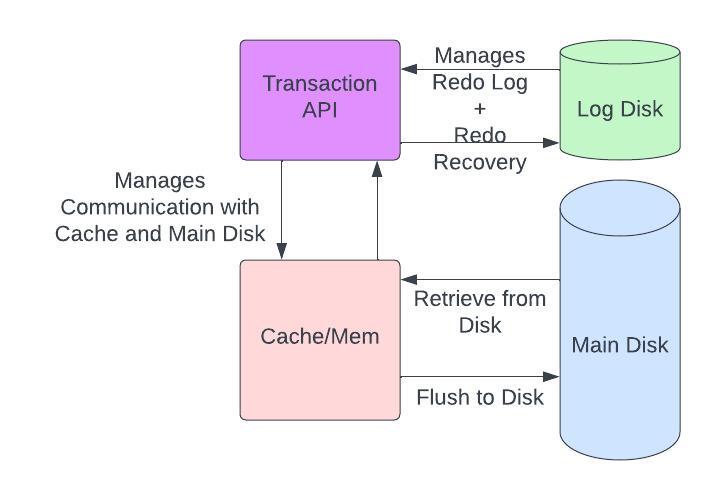
Hold all new transactions for k sec

Actions in redo log are executed and flushed to disk

Redo log transactions completes with <end>

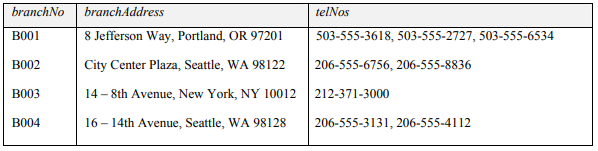
Checkpoint is recorded to disk

Resume taking new transactions and create redo log for next set of transactions



**Task 4**

We have a table below:



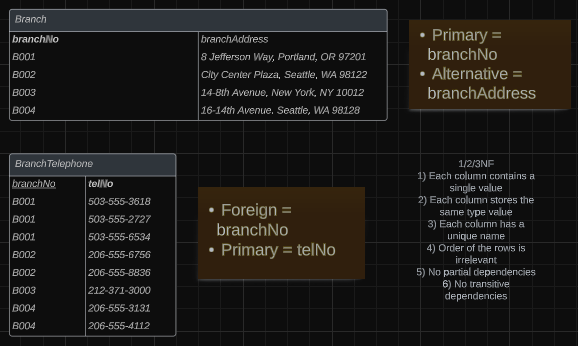
Please answer the following questions:

1. What is the normal form of the table, why?

The normal form of the table above must follow the 1NF rules which state

* Each column contains a single value ⭐ **violated condition**
* Each column stores the same type value
* Each column has a unique name
* Order of the rows is irrelevant

1. Normalize the table to 3NF



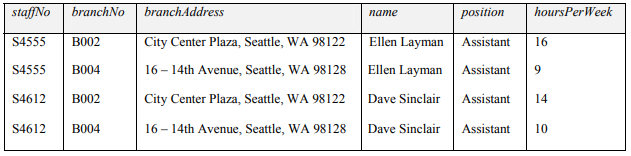
1. Identify the primary keys and foreign keys in the 3NF relations.

Branch table: branchNo is the primary key and branchAddress is the alternative key.

BranchTelephone table: *branchNo* is the foreign key and telNo is the primary key.

**Task 5**

We have a table below:



Also note that each staff work in each branch for a different hoursPerWeek.

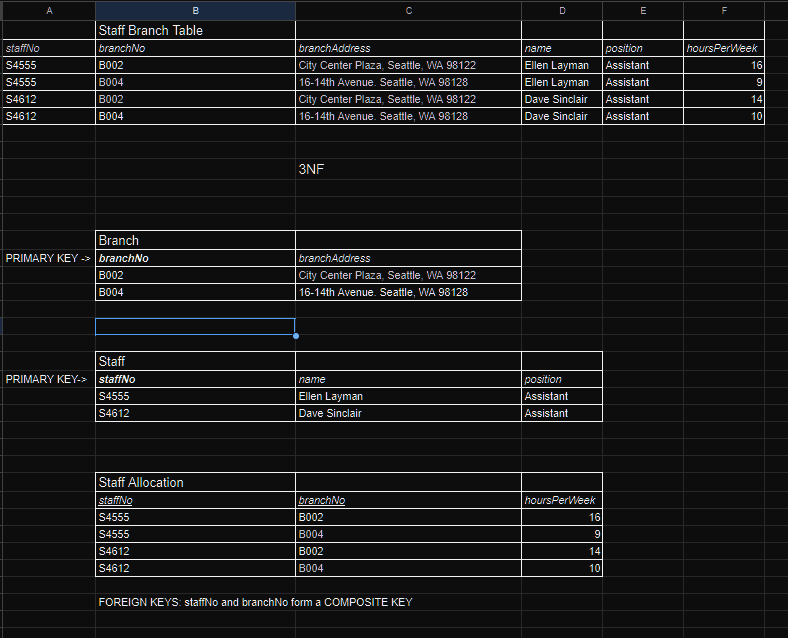
Please answer the following questions:

1. What is the normal form of the table? Why

This table is already in 1st normal form since it follows the 1NF rules which state

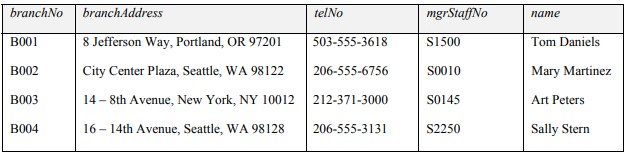
* Each column contains a single value
* Each column stores the same type value
* Each column has a unique name
* Order of the rows is irrelevant

1. Normalize this table to 3NF and mark the primary keys and foreign keys



**Task 6**

We have a table below (telNo is the branch telephone number):

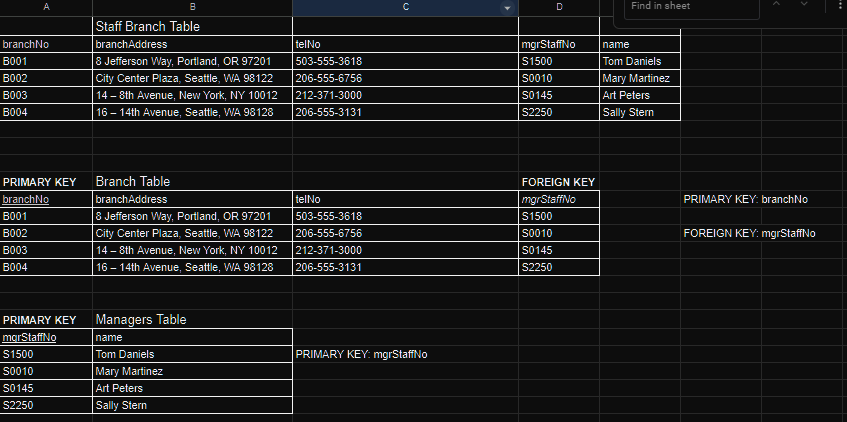


Please answer the following questions:

1. What is the normal form of the table? Why

This table is already in 1NF since it follows the normal form rules. The 3NF of this table is where Branch and Manager Staff are separated into two separate tables to avoid a transitive dependency between the name and mgrStaffNo columns.

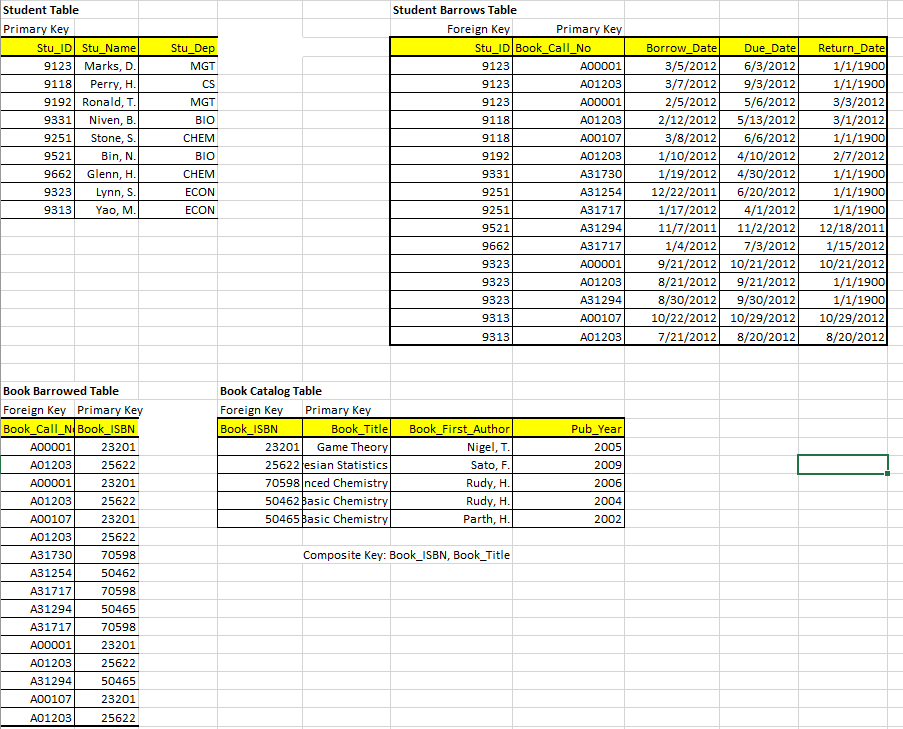
1. Normalize this table to 3NF and mark the primary keys and foreign keys



**Task 7 -**

Please normalize the in the book loan table in Excel file, and then screenshot the output.

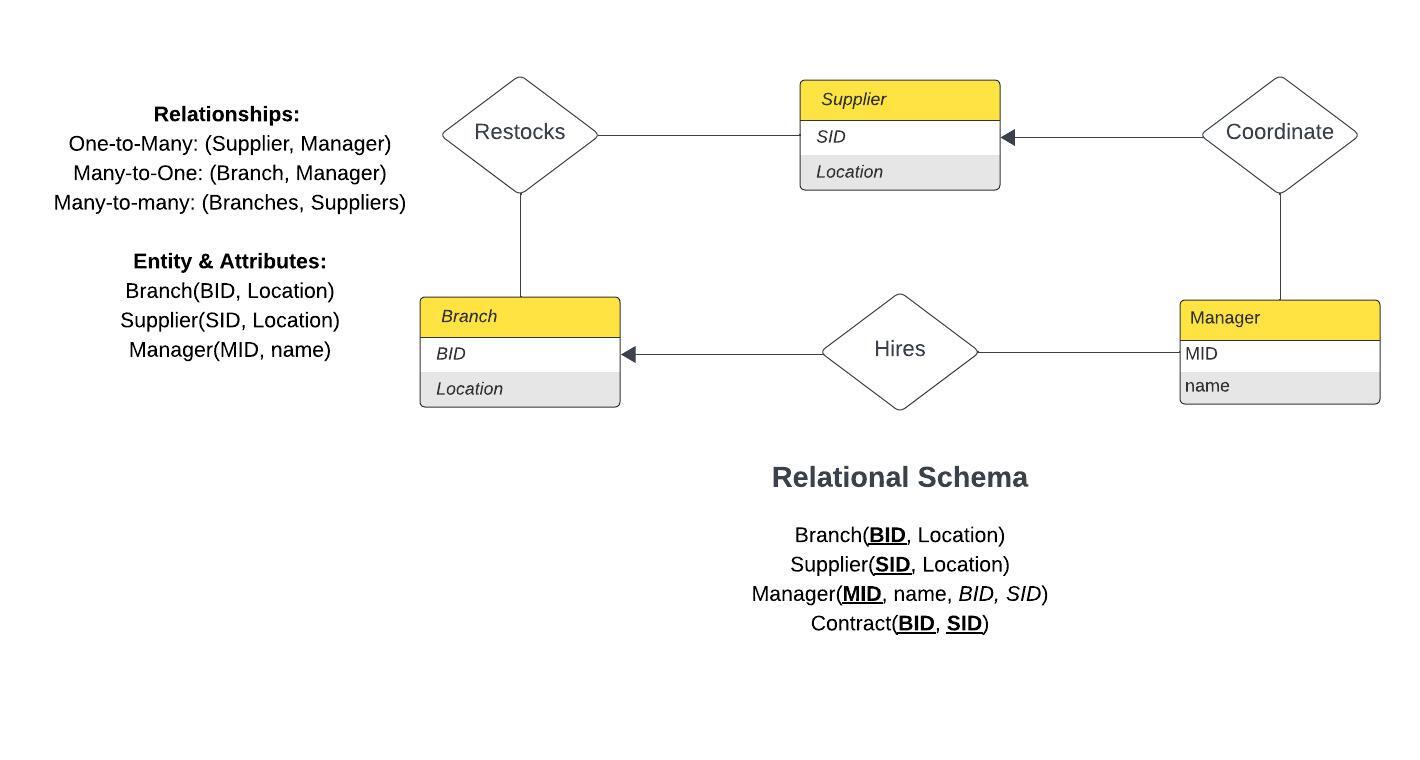
In this book loan system. Each unique Book\_Title has a unique Book\_ISBN, but many different Book\_Call\_No (a title in the library usually has multiple copies). It is possible for a student to borrow multiple book copies (identified by Book\_Call\_No) at one time, or borrow the same book copies across different time.



**Task 8 - IP**

Please draw ER diagram and convert it into relational schema:

1. Each supermarket branch restocks products with multiple suppliers. Each supplier supplies products to multiple supermarket branch. The supermarket branch has attributes BID, and Location; each supplier has attribute SID, and location.
2. Each supermarket branch hires many inventory managers to manage the product restocking. One inventory manager can only be hired by one supermarket branch. Each inventory manager is responsible to coordinate with only one supplier to restock the products. However, each supplier may work with multiple inventory managers at the same time. The supermarket branch has attributes BID, and Location; each inventory manager has attribute MID and name; each supplier has attribute SID, and location.

****

**Task 9**

We store 8 same-sized data blocks (A1, A2, … A8) in three different storage architectures:

1. In a single disk

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 |

**Answer:** For a **single disk** architecture, the time needed to read blocks A1-A4 would take 1ms per block so **the final time needed is 4ms.**

1. In RAID 0 using two disks

|  |  |  |  |
| --- | --- | --- | --- |
| A1 | A3 | A5 | A7 |

|  |  |  |  |
| --- | --- | --- | --- |
| A2 | A4 | A6 | A8 |

**Answer:** For **RAID 0** disk architecture, the time needed to read blocks A1-A4 can be performed in parallel so **the final time needed is 2ms.**

1. In RAID 1 using two disks

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 |

**Answer:** For **RAID 1** disk architecture, the time needed to read blocks A1-A4 can be performed by reading blocks in a simultaneous manner so **the final time needed is 2ms.**

**Task 10:**

We are using undo/redo logging to recover the database after a system crash. The log on the disk looks like the following:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ... | <T1, A, 10, 15> | <T1, end> | <checkpoint> | <T2, start> | <T2, A, 39, 10> | <T3, start> | <T3, B, 20,14> | <T3, commit> | <T2, C, 50, 20> | <T4, start> | <T3, end> | <T4, D, 12, 13> | <T2, commit> | Crash |

How to recover the database (do you redo, undo, or ignore T1, T2, T3, and T4)?

and what are the values of A, B, C, and D before and after the recovery (if the value is unsure, type “unknown”)?

Please fill the two tables below.

|  |  |
| --- | --- |
| Transaction | Recover Plan (redo, undo, or ignore) |
| T1 | Ignore |
| T2 | Redo |
| T3 | Ignore |
| T4 | Undo |

|  |  |  |
| --- | --- | --- |
| Value | Value before Recovery | Value after Recovery |
| A | 10 | 39 |
| B | 20 | 20 |
| C | 20 | 50 |
| D | 13 | Unknown |