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: Is sequential reading faster than random reading in this case? How much faster?**

Yes, sequential reading is faster than random reading in this case.

**Sequential reading:** Each cache line can be fully utilized, loading two consecutive data records with one cache line fetch. This minimizes cache misses and their associated latency.

* 100 data records \* 32 bytes / 64 bytes per cache line = 50 cache line fetches are required

**Random Reading**: Each data record is fetched with a cache line (64 bytes). However, since the data is randomly distributed, the cache line utilization is not effective, leading to more cache misses.

* For random reading, if you assume an average of one cache line fetch per data record due to cache misses, you will need 100 cache line fetches.

This rough estimation suggests that sequential reading might be around 2 times faster than random reading in this specific scenario.

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

**Sequential Reading:** Each data record is 64 bytes, and each cache line is 64 bytes, so when reading sequentially, each cache line fetch perfectly matches a single data record. There's no wasted space in the cache line.

* 100 data records \* 64 bytes / 64 bytes per cache line = 100 cache line fetches are required

**Random Reading:** With 64-byte data records and 64-byte cache lines, random reading is more efficient than in the scenario with 32-byte data records. When accessing data records randomly, each cache line fetch still aligns perfectly with the data record size, ensuring that no space in the cache line is wasted.

* 100 data records \* 64 bytes / 64 bytes per cache line = 100 cache line fetches are required for random reading as well.

If each data record is 64 bytes in size, the time taken for sequential and random reading will be the same. The speedup for sequential reading compared to random reading, in this case, would likely be minimal, as both random and sequential reading efficiently utilize the cache lines.

**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.

**Answer:**

The "Five-Minute Rule" is used in the field of computer systems and data management to determine whether it's more cost-effective to keep data in memory (RAM) or store it on a slower storage medium, such as a hard disk. The rule helps in making decisions about data caching and storage allocation. The rule suggests that if the time it takes to access the data from disk is more than five times the time it takes to access the data from memory, then it's generally more cost-effective to keep the data in memory.

The five-minutes rule formula is: D = I \* X / M \* P

Where: D = Data size (GB)

I = Access rate (queries/sec)

X = Data transfer rate (MB/sec)

M = Memory size (GB)

P = Memory cost per GB ($/GB)

To determine whether the table should be stored on the hard disk or in memory, we need to calculate the cost of storing the table in each location using the formula above.

**Cost of storing the table on the hard disk:**

Given, D = 1 GB

I = 20 times/sec

X = 100MB/sec

M = 1 GB (if the server has 1GB of memory)

P = $10/GB (the cost of hard drive storage)

D = I \* X / M \* P D = 20 \* 100 / 1 \* 10 = $4.

So, the cost of storing the table on the hard disk is $4

**Cost of storing the table in memory:**

Given, D = 1 GB

I = 20 times/sec

X = 100MB/sec

M = 1 GB (if the server has 1GB of memory)

P = $20/MB (the cost of memory)

D = I \* X / M \* P D = 20 \* 100 / 1 \* 20 = $10

So, the cost of storing the table in memory is $10.

Result: Since the cost of storing the table in memory ($10) is more than the cost of storing it on the hard disk ($4), it's **more cost-effective to store the table on the hard disk**.

**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

As you can imagine, this is very inefficient because it results in a lot of I/Os, which takes a long time.

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.

**Answer:**

* One way to optimize database performance for frequently accessed objects like X is to use a caching mechanism.
* pre-scanning transactions to optimize them is a strategy that minimizes I/O operations, making your database more efficient, reducing response times, and improving the performance of frequently accessed objects.
* Instead of reading X from disk every time it is accessed, we can store it in memory in a cache.

**Caching workflow:**

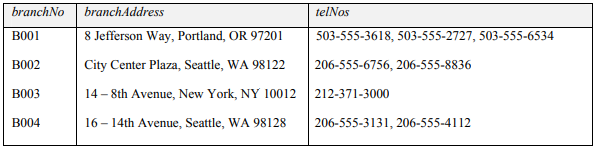
* When a transaction requests the value of X, the database checks if it is in the cache.
* If X is in the cache, it is returned directly to the transaction, eliminating the need for a disk I/O operation.
* If X is not in the cache, the database retrieves it from disk and stores it in the cache before returning it to the transaction.
* When a transaction updates the value of X, the database updates the cache and marks X as "dirty", indicating that it needs to be flushed to disk at some point.
* The database periodically flushes dirty values from the cache to disk, reducing the number of unnecessary write operations and ensuring that the data is durable.

**Pseudocode:**

* Initialization: X is read from the disk to the cache initially.
* Transaction T1: reads X from the cache, updates it, and then writes the updated data back to the cache
* Transaction T2: reads X from the cache, updates it, and then writes the updated data back to the cache
* Transaction T3: reads X from the cache, updates it, and then writes the updated data back to the cache
* Transaction T4: reads X from the cache, updates it, and then writes the updated data back to the cache
* Final Write: The final state of X in the cache is written back to the disk to persist the changes.
* Output: The final state of X is provided as the output

**Task 4**

We have a table below:



Note that branchNo is the primary key of this table.

Please answer the following questions:

1. What is the normal form of the table, why?
2. Normalize the table to 3NF
3. Identify the primary keys and foreign keys in the 3NF relations.

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

* A table in which the intersection of every column and record contains only one value
* The given table is not in 1NF since the telNos column has more than one value
* The modified table below with individual values for every column

Graphical user interface, text, application

Description automatically generated

**2. Normalize the table to 3NF:**

**Step 1:** Making sure the table is in 1NF.

**Step 2:** 2NF

* NF1 is satisfied
* Partial dependency does not exist - all the information in the table must depend on the whole primary key. Since "branchNo" is the primary key, and the "branchAddress" and "telNos" columns depend on it.

**Step 3:** 3NF

* NF2 satisfied
* No transitive dependency exists – meaning that non-key attributes should not depend on other non-key attributes.

Branch Table BranchTelephone Table

A diagram of a number of numbers

Description automatically generated with medium confidenceA close-up of a address

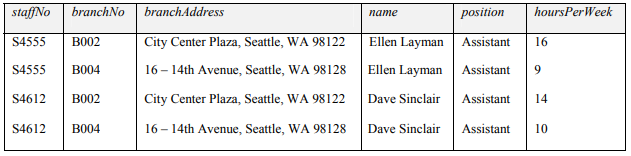
Description automatically generated

**3. Identify the primary keys and foreign keys in the 3NF relations:**

* The Branch table has ‘branch no’ as the primary key.
* The BranchTelephone table has ‘telNo’ as primary key and ‘branch no’ becomes the foreign key.

**Task 5**

We have a table below:



Note that staffNo and branchNo are the composite primary key of this table.

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
2. Normalize this table to 3NF and mark the primary keys and foreign keys
3. **What is the normal form of the table? Why**

* The given table is in 1NF. Each column has individual values.
* The table has composite primary keys: staffNo and branchNo
* Values in branchAddress column can be worked out from only branchNo, so table is not in 2NF.
* Values in name and position columns can be worked out only from staff No, so table is not in 2NF.
* Values in hoursPerWeek column can be worked out only from staffNo and branchNo

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

* The above table can be split into 3 tables to ensure NF2 is satisfied by fixing the partial dependency problem.

**Branch Table:** Primary key: branchNo

A close up of a address

Description automatically generated

**TempStaff Table**: Primary key: staffNo

A close up of a box

Description automatically generated

**TempStaffAllocation Table:** Composite Primary key: staffNo & branchNo

Foreign Keys: staffNo & branchNo

**A screenshot of a computer

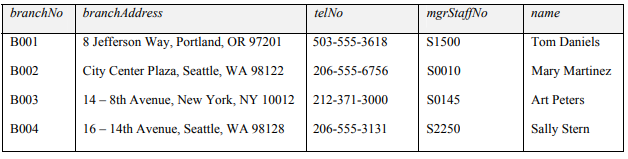
Description automatically generated**

* The above tables satisfy criteria of 2NF.
* All attributes are functionally dependent only on the primary key.

There are no transitive dependencies, meaning that non-key attributes should not depend on other non-key attributes.

**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
2. Normalize this table to 3NF and mark the primary keys and foreign keys
3. **What is the normal form of the table? Why**

Table is in 2NF form. Transitive dependency exists

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

**Branch Table:** Primary key : branchNo

|  |  |  |
| --- | --- | --- |
| branchNo | branchAddress | telNo |
| B001 | 8, Jefferson Way, Portland, OR 97201 | 503-555-3618 |
| B002 | City Centre Plaza, Seattle, WA 98122 | 206-555-6756 |
| B003 | 14 - 8th Avenue, New York, NY 10012 | 212-371-3000 |
| B004 | 16 - 14th Avenue, Seattle, WA 98128 | 206-555-3131 |

**ManagerStaff Table:** Primary Key:mgrStaffNo

Foreign Key: branchNo

|  |  |  |
| --- | --- | --- |
| branchNo | mgrStaffNo | name |
| B001 | S1500 | Tom Daniels |
| B002 | S0010 | Mary Martinez |
| B003 | S0145 | Art Peters |
| B004 | S2250 | Sally stern |

**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.

**Normalized Tables:**

**Student Table:** Primary Key: Stu\_ID

|  |  |  |
| --- | --- | --- |
| Stu\_ID | Stu\_Name | Stu\_Dep |
| 9123 | Marks, D. | MGT |
| 9118 | Perry, H. | CS |
| 9192 | Ronald, T. | MGT |
| 9331 | Niven, B. | BIO |
| 9251 | Stone, S. | CHEM |
| 9521 | Bin, N. | BIO |
| 9662 | Glenn, H. | CHEM |
| 9323 | Lynn, S. | ECON |
| 9313 | Yao, M. | ECON |

**Book Table:** Primary Key: Book\_ISBN

|  |  |  |  |
| --- | --- | --- | --- |
| Book\_ISBN | Book\_Title | Book\_First\_Author | Pub\_Year |
| 23201 | Game Theory | Nigel, T. | 2005 |
| 25622 | Bayesian Statistics | Sato, F. | 2009 |
| 70598 | Advanced Chemistry | Rudy, H. | 2006 |
| 50462 | Basic Chemistry | Rudy, H. | 2004 |
| 50465 | Basic Chemistry | Parth, H. | 2002 |

**Book copies table:** Primary Key: Book\_Call\_No

Foreign Key: Book\_ISBN

|  |  |
| --- | --- |
| Book\_Call\_No | Book\_ISBN |
| A00001 | 23201 |
| A01203 | 25622 |
| A00107 | 23201 |
| A31730 | 70598 |
| A31254 | 50462 |
| A31717 | 70598 |
| A31294 | 50465 |

**Loan Table:** Primary Key: Loan ID

Foreign Key : Stu\_ID, Book\_Call\_No

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Loan ID | Stu\_ID | Book\_Call\_No | Borrow\_Date | Due\_Date | Return\_Date |
| 1 | 9123 | A00001 | 3/5/12 | 6/3/12 | 1/1/00 |
| 2 | 9123 | A01203 | 3/7/12 | 9/3/12 | 1/1/00 |
| 3 | 9123 | A00001 | 2/5/12 | 5/6/12 | 3/3/12 |
| 4 | 9118 | A01203 | 2/12/12 | 5/13/12 | 3/1/12 |
| 5 | 9118 | A00107 | 3/8/12 | 6/6/12 | 1/1/00 |
| 6 | 9192 | A01203 | 1/10/12 | 4/10/12 | 2/7/12 |
| 7 | 9331 | A31730 | 1/19/12 | 4/30/12 | 1/1/00 |
| 8 | 9251 | A31254 | 12/22/11 | 6/20/12 | 1/1/00 |
| 9 | 9251 | A31717 | 1/17/12 | 4/1/12 | 1/1/00 |
| 10 | 9521 | A31294 | 11/7/11 | 11/2/12 | 12/18/11 |
| 11 | 9662 | A31717 | 1/4/12 | 7/3/12 | 1/15/12 |
| 12 | 9323 | A00001 | 9/21/12 | 10/21/12 | 10/21/12 |
| 13 | 9323 | A01203 | 8/21/12 | 9/21/12 | 1/1/00 |
| 14 | 9323 | A31294 | 8/30/12 | 9/30/12 | 1/1/00 |
| 15 | 9313 | A00107 | 10/22/12 | 10/29/12 | 10/29/12 |
| 16 | 9313 | A01203 | 7/21/12 | 8/20/12 | 8/20/12 |

This structure separates students, books, book copies (individual physical copies of a book), and book loans. Each table represents a distinct entity and its attributes. This normalization minimizes redundancy and ensures data integrity.

**Task 8**

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.

**Entities and Their Attributes:**

1. Supermarket Branch (BID, Location)
2. Supplier (SID, Location)
3. Inventory Manager (MID, Name)

**Relationships:**

* A many-to-many relationship between Supermarket Branch and Supplier, indicating that each supermarket branch can restock products from multiple suppliers, and each supplier supplies products to multiple supermarket branches.
* A one-to-many relationship between Supermarket Branch and Inventory Manager, representing that each supermarket branch hires many inventory managers to manage product restocking.
* A one-to-many relationship between Inventory Manager and Supplier, indicating that each inventory manager is responsible for coordinating with one supplier to restock products, and each supplier may work with multiple inventory managers

**ER Diagram:**

Diagram

Description automatically generated

**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 |

1. In RAID 0 using two disks

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

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

1. In RAID 1 using two disks

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

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

Suppose reading one block from the disk to the memory takes 1ms, what is the time needed to read A1, A2, A3, and A4 to the memory in each of the three storage architecture? Ignore seek time rotary latency.

1. **In a single disk:**

* Reading A1, A2, A3, and A4 would take 4ms in total because reading one block from disk to memory takes 1ms and we are reading 4 blocks, so it takes 4 \* 1ms = 4ms.

**2. In RAID 0 using two disks:**

* Reading A1, A2, A3, and A4 would take 2ms in total because data is striped across both disks, meaning that each block is split into two parts and written to different disks. So, reading one block from disk to memory takes 1ms per disk, and we are reading 4 blocks, so it takes 2 \* 1ms = 2ms.

**3. In RAID 1 using two disks:**

* Reading A1, A2, A3, and A4 would take 2ms in total because data is mirrored across both disks, meaning that the same data is written to both disks. So, reading one block from either disk to memory takes 1ms and we are reading 4 blocks, so it takes 2 \* 1ms = 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 | Undo |
| T2 | Redo |
| T3 | Redo |
| T4 | Undo |

|  |  |  |
| --- | --- | --- |
| Value | Value before Recovery | Value after Recovery |
| A | 15 | 15 |
| B | Unknown | 20 |
| C | Unknown | 50 |
| D | Unknown | Unknown |

* To recover the database using the provided log, we need to follow the principles of redo and undo logging.
* Redo ensures that committed changes are reapplied to the database, while undo undoes the changes made by transactions that were not committed.
* We also need to consider the checkpoint record, which indicates which transactions are active at the time of the crash.