Chandan Suri
Introduction to DBs

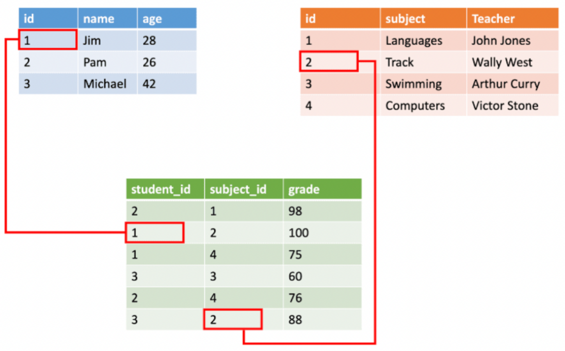
cs4090@columbia.edu
CS4090

# Take Home Question 1 Submission – CS4090

**Question 1:** One way of classifying data is into the categories: structured, semi-structured and unstructured. Give a brief description of each type and an example of each type.

**Answer:**

Structured Data: It is generally tabular data that is represented by columns and rows in a database thus, it has a defined structure. Also, it easier to analyze such form of data in comparison to unstructured and semi-structured data. In particular, databases that hold tables in this form are called relational databases. For instance, SQL (Structured Query Language) is a form of structured data.

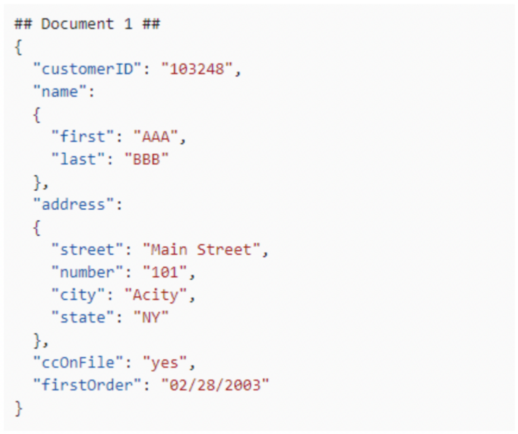


| id | name | age |
|---|---|---|
| 1 | Jim | 28 |
| 2 | Pam | 26 |
| 3 | Michael | 42 |

| id | subject | Teacher |
|---|---|---|
| 1 | Languages | John Jones |
| 2 | Track | Wally West |
| 3 | Swimming | Arthur Curry |
| 4 | Computers | Victor Stone |

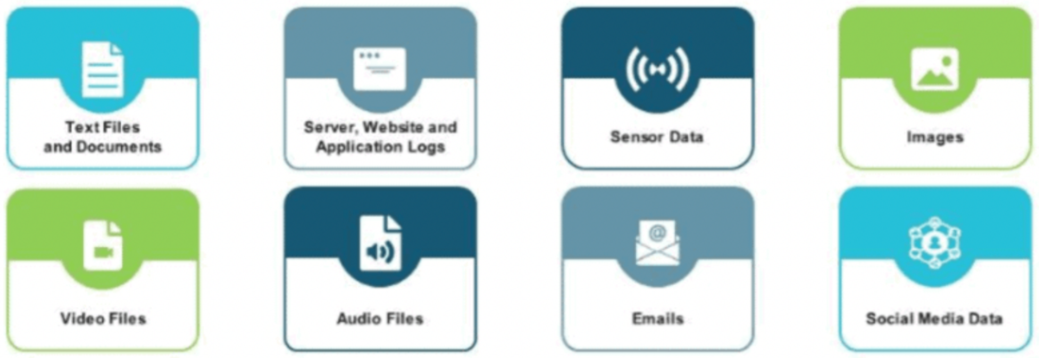
| student_id | subject_id | grade |
|---|---|---|
| 2 | 1 | 98 |
| 1 | 2 | 100 |
| 1 | 4 | 75 |
| 3 | 3 | 60 |
| 2 | 4 | 76 |
| 3 | 2 | 88 |

Semi-Structured Data: This kind of data is information that doesn't consist of structured data (relational DB) but still has some structure to it. Also, this is basically information that does not reside in a relational database but has some organizational properties that makes it easier to analyze. Generally, this consists of data consisting of documents held in JSON format. This also includes key-value stores and graph databases.



```
## Document 1 ##
{
  "customerID": "103248",
  "name":
  {
    "first": "AAA",
    "last": "BBB"
  },
  "address":
  {
    "street": "Main Street",
    "number": "101",
    "city": "Acity",
    "state": "NY"
  },
  "ccOnFile": "yes",
  "firstOrder": "02/28/2003"
}
```

Unstructured Data: This is information that either does not organize in a pre-defined manner or not have a pre-defined model. This consists of information that is a set of text-heavy data, but it might also contain data such as number, dates, and facts as well. For instance, video, audio, and binary data files might not have a definite structure. Such kind of data is often deemed to be unstructured form of data.



References:
1. https://k21academy.com/microsoft-azure/dp-900/structured-data-vs-unstructured-data-vs-semi-structured-data/
2. https://www.geeksforgeeks.org/difference-between-structured-semi-structured-and-unstructured-data/

**Question 2:** What is the difference between a type, e.g., VARCHAR, INT, and a *domain?* Given an example.

**Answer:**

Data Type: The types are basically some predefined data types that determine what type of values can be stored in a column. Those predefined data types allow some domain of values that can be specified. For instance, a VARCHAR type column would only hold alphanumeric data with the number of characters as specified with the type. By default, VARHCAR can hold 65535 characters.

Domain: A domain is based on a particular base type and for many purposes is interchangeable with this base type. However, a domain can have additional constraints on top of the data type assigned that further restricts its valid values to a subset of what underlying base type would allow. In SQL, domains can be created using the command "CREATE DOMAIN". For example,
CREATE DOMAIN dom1 AS INT CHECK
(value >= 0 AND value <= 100)
This would only allow values in the range of 0-100 (both inclusive) for the column on which the DOMAIN is applied. If it were just INT, then the values in a column would just be integer values with some large domain. But, in this case the valid values are restricted to [0, 100].

References:

1. https://www.geeksforgeeks.org/sql-create-domain/
2. https://www.quora.com/What-is-the-difference-between-domain-and-data-type-in-DBMS

**Question 3:** Assume that you know that access to a relational table will be primarily sequential. How would you lay the blocks out on a disk's cylinders, heads, and sectors?

**Answer:**

If the access to the relational table is primarily sequential, then the blocks are laid out on the disk cylinders, heads, and sectors in a way such that reading and writing time is reduced in a sequential manner.

For instance, if a relational table can be stored in a set of sectors and cylinder then the blocks would be written one by one firstly in a sector on all its tracks and then the following records are stored one by one on the sectors in that same cylinder for each of the heads in that cylinder. Let's say after a while, the relational table gets bigger such that a sector is filled completely for all the heads in a cylinder. Then we would move to another sector in the same cylinder and do the same thing.

Let's say after a while, even all the sectors on all the heads on a cylinder are stored with the data and more data needs to be read or written. Then, we move on to the nearest cylinder and do the same process again. This way we would have to seek the least thus, reducing the time for the I/O operations.

**Question 4:** Most databases do not support hash indexes. What are three benefits of B+ Tree indexes over hash indexes? Express your answer by giving examples of SQL queries better supported by B+ Trees.

**Answer:**

3 benefits of B+ tree indexes over hash indexes:

1. If it is a range query retrieval or when the queries include comparison operators other than the equivalency operator, the hash indexing would be less useful in comparison to the B+ tree indexes. Because the original orderly key value may become discontinuous after the hash algorithm, there is no way to use the index to complete the scope query retrieval.
2. While the B+ tree does, hash indexes also do not support the leftmost matching rule for multicolumn join indexes. Due to this, join queries are better supported by B+ tree indexes in comparison to the hash indexes.
3. The keyword retrieval efficiency of the B+ tree index is relatively average, and the fluctuation range is also not very large. In the case of a large number of repeated key values, the efficiency of the hash index is extremely low because there is a so-called hash collision problem. Also, when there are multiple conditions to fulfill to find a record or a set of records, it would be faster for the B+ tree to do so as it has multiple key value pairs at each level of the tree and has clustered indexes which makes the search process quite fast in comparison to using hash indexing to find the same set of rows for the same query.

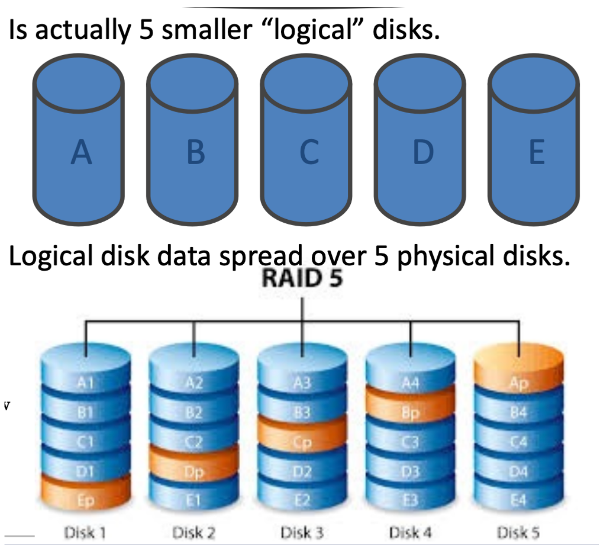
References:

1. https://medium.com/@mena.meseha/what-is-the-difference-between-mysql-innodb-b-tree-index-and-hash-index-ed8f2ce66d69
2. https://dev.mysql.com/doc/refman/8.0/en/index-btree-hash.html
3. Prof. Fergusson's slides

Chandan Suri
Introduction to DBs

cs4090@columbia.edu
CS4090

**Question 5:** What is the primary benefit of RAID-5 compared to RAID-0 and RAID-1?
**Answer:**
RAID-5 is considered to be the most secure and most common RAID implementation. It combines striping and parity to provide a fast and reliable setup. Such a configuration gives the user storage usability as with RAID-1 and the performance efficiency of RAID-0.
This RAID level consists of at least 3 hard drives. Data is divided into data strips and distributed across different disks in the array. This allows for high performance rates due to fast read data transactions which can be done simultaneously by different drives in the array. Also, parity bits are distributed evenly on all the disks after each sequence of data has been saved. This feature ensures that you still have access to the data from parity bits in case of a failed drive. Thus, RAID-5 provides redundancy through parity bits instead of monitoring.



References:
1. https://phoenixnap.com/kb/raid-levels-and-types
2. Prof. Fergusson's slides.

**Question 6:** Briefly explain the difference between fixed size records and variable size records.
**Answer:**
Fixed size records: In such records, all the records in a block/file are of the same size. This could lead to memory wastage. However, access to the records is easier and faster. Also, exact location of the records can easily be determined: location of the ith record would be n*(i-1), where n is the size of every record.

Variable size records: For such records, different records in the file have different sizes. Also, this is more memory efficient in comparison to the fixed size records. However, access of the records is slower. Also, variable length records may arise in a database system in different ways:

- Storage of multiple record types in a file.
- Record types that allow variable lengths for one or more such fields such as strings.
- Record types that allow repeating fields.

References:
1. https://practice.geeksforgeeks.org/problems/differentiate-between-fixed-length-records-and-variable-length-records
2. Prof. Fergusson's slides.

**Question 7:** The database query optimizer may create a hash index to optimize a query. Give an example of a SELECT statement for which the optimizer may create a hash index.

**Answer:**

Generally, for an equivalency query, the hash indexes have an obvious advantage over the B+ tree indexes, because only one algorithm is needed to find the corresponding key value, of course, the premise being that the key value pair is unique. If the key value isn't unique, you need to find the location of the key first, and then scan backward according to the linked list until you find the corresponding data. For instance, hash indexes would be used for the SELECT query below:

```
SELECT employee_id, employee_name, address, phone_number
FROM employees
WHERE employee_id = "E123"
```

Here the employee_id is the primary key for the "employees" table. Thus, there will be a unique key value pair here and the entry would be fetched in O(1) time with hash indexes rather than in O(log n) with B+ tree indexes.

**Question 8:** Least-recently-used is a common buffer pool replacement policy. But sometimes LRU is the worst possible algorithm. Briefly give an example of when LRU is the worst algorithm and why.

**Answer:**

LRU follows the following assumption that the information that will not be used for the longest time is the information that has not been accessed for the longest time. When this assumption would fall short, then the LRU would behave in the worst way possible. For example, in a system when subsequent fetching would result in newer elements all the time, then LRU would fail miserably.

**Question 9:** Briefly explain the concept of conflict serializable and how it differs from serializable.

**Answer:**

Serializability is used to keep the data in the data item in a consistent state. Serializability is a property of a transaction schedule. It relates to the isolation property of a database transaction. Conflict serializability is a subset of serializability. If a schedule is conflict serializable, it is implied that this schedule is serializable. It is computationally easier to determine if a schedule is conflict serializable as opposed to just serializable, we can simply construct a precedence graph.

In simple Words, suppose there is a schedule(S) with two transactions T1,T2. Let Result1 and Result2 be two variables, where Result1 is produced after running T1 and then T2, i.e., T1->T2 (serially) and Result2 is produced after running T2 and then T1, i.e., T2->T1(serially).

Now suppose we interleave the actions of the two transaction, let us call it schedule S, now if the net result produced after running S is equivalent to Result1 or Result2 then we call it serializable. But, if we can swap the non-conflicting actions and produce a serial schedule that is equal to a schedule in which T1 is run first and then T2(T1->T2), or T2 is run first and then T1(T2->T1) then we call it conflict-serializable.

Now if a schedule is conflict-serializable then it is bound to be serializable as it can be changed to some serial order by swapping the non-conflicting actions. Hence, we can conclude, every conflict-serializable schedule is serializable but not all serializable schedules are conflict-serializable.

References:

1. https://stackoverflow.com/questions/20529800/whats-the-difference-between-conflict-serializable-and-serializable
2. https://www.geeksforgeeks.org/conflict-serializability-in-dbms/
3. Prof. Fergusson’s slides

**Question 10:** What is a deadlock? How do DBMS transaction managers break deadlock?
**Answer:**
In a database, a deadlock is an unwanted situation in which two or more transactions are waiting indefinitely for one another to give up locks. Deadlock is said to be one of the most feared complications in DBMS as it brings the whole system to a halt.

Deadlocks can occur if the four conditions hold simultaneously (necessary conditions):

1. Mutual exclusion: One or more than one resource is non-shareable. (Only one process can use a resource at a point in time)
2. Hold and Wait: A process is holding at least one resource and waiting for resources.
3. No Preemption: A resource cannot be taken from a process unless the process releases the resource.
4. Circular wait: A set of processes are waiting for each other in a circular form.

Deadlocks can be prevented! A deadlock can be prevented if the resources are allocated in such a way that deadlock never occurs. The DBMS analyzes the operations whether they can create a deadlock or not, if they do, the transaction is never allowed to be executed. Deadlock prevention could work in 2 ways:

1. Wait-Die Scheme: If a transaction requests a resource that is locked by another transaction, then the DBMS simply checks the timestamp of both transactions and allows the older transaction to wait until the resource is available for execution.
2. Wound-Wait: If an older transaction requests for a resource held by a younger transaction, then an older transaction forces a younger transaction to kill the transaction and release the resource. The younger transaction is started with a delay but with the same timestamp. If the younger transaction is requesting a resource that is held by an older one, then the younger transaction is asked to wait till the older one releases it.

References:

1. https://www.geeksforgeeks.org/introduction-of-deadlock-in-operating-system/
2. https://www.geeksforgeeks.org/deadlock-in-dbms/