USN										

Internal Assessment Test 1 – OCT 2024

Sub:	NOSQL DATABASE					Sub Code:	21CS745	Branch:	ISE
Date:	15/10/2024	Duration:	90 min	Max Marks:	50	Sem/Sec:	VII/ A, B & C		OBE

Answer any FIVE FULL Questions

		MARKS	CO	RBT
1.	What is NOSQL? Explain briefly about aggregate data models with, Considering example of Relations and Aggregates.	[10]	1	L2

Scheme : Definition + explanation +example + Diagram – 2+3+2+3 Marks

Solution :

NoSQL databases ("not only SQL") are non-tabular databases and store data differently than relational tables which comes in a variety of types based on their data model. The main types are document, key-value, wide-column, and graph.

Aggregate Data Models

- A data model is the model through which we perceive and manipulate our data.
- For people using a database, the data model describes how we interact with the data in the database
- The relational model takes the information that we want to store and divides it into tuples (rows).
- A tuple is a limited data structure: It captures a set of values, so you cannot nest one tuple within another to get nested records, nor can you put a list of values or tuples within another.

Example of Relations and Aggregates

Let's assume we have to build an e-commerce website; we are going to be selling items directly to customers over the web, and we will have to store information about users, our product catalog, orders, shipping addresses, billing addresses, and payment data. We can use this scenario to model the data using a relation data store as well as NoSQL data stores and talk about their pros and cons. For a relational database, we might start with a data model shown below

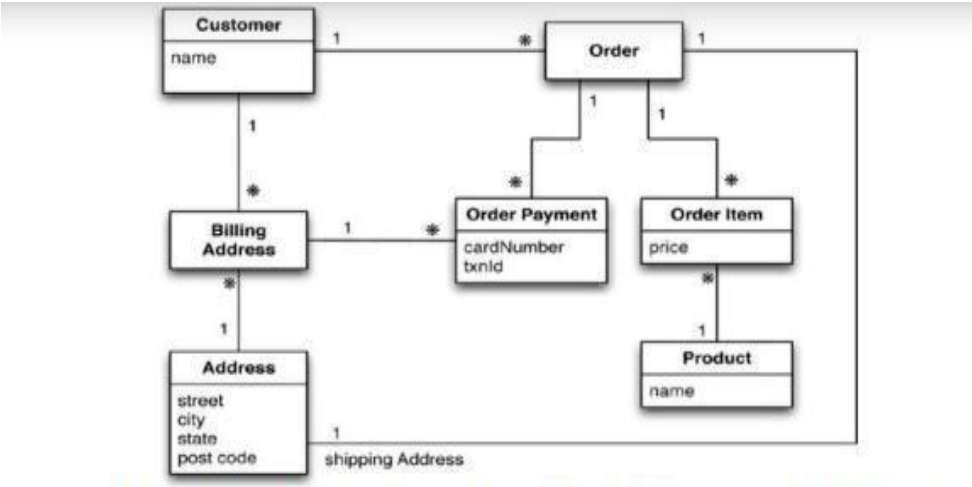


Figure 2.1. Data model oriented around a relational database (using UML notation)

Customer

Id	Name
1	Martin

Orders

Id	CustomerId	ShippingAddressId
99	1	77

Product

Id	Name
27	NoSQL Distilled

BillingAddress

Id	CustomerId	AddressId
55	1	77

OrderItem

Id	OrderId	ProductId	Price
100	99	27	32.45

Address

Id	City
77	Chicago

OrderPayment

Id	OrderId	CardNumber	BillingAddressId	txnId
33	99	1000-1000	55	abelif879rft

Figure 2.2. Typical data using RDBMS data model

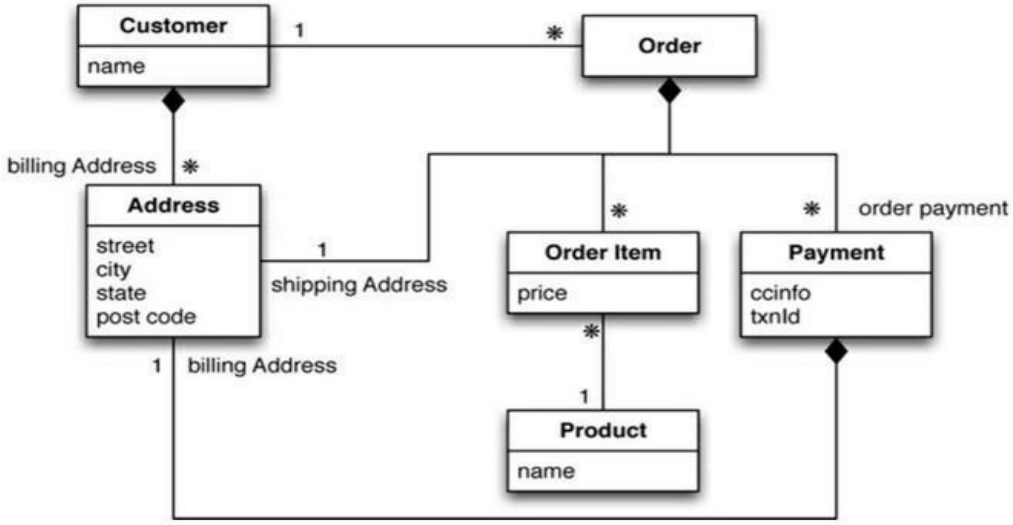


Figure 2.3. An aggregate data model

2. a. Briefly describe the value of Relational databases

Scheme : Definition + explanation – 2+3 Marks

Solution :

Getting at Persistent Data:

The value of a database is keeping large amounts of persistent data. Most computer architectures have the notion of two areas of memory: a fast volatile "main memory" and a larger but slower "backing store." Main memory is both limited in space and loses all data when you lose power or something bad happens to the operating system.

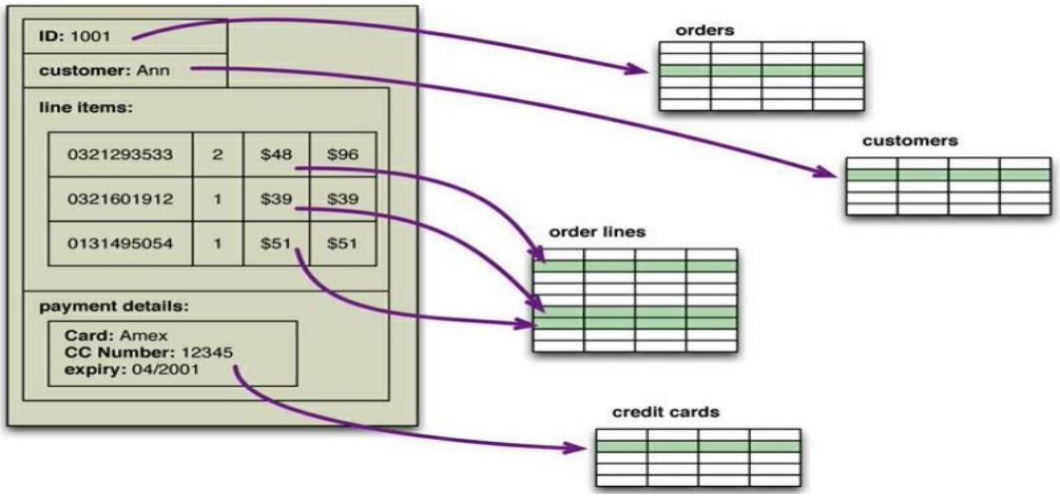
[5] 1 L2

Concurrency:

Enterprise applications tend to have many people looking at the same body of data at once, possibly modifying that data. Most of the time they are working on different areas of that data, but occasionally they operate on the same bit of data.

Integration

Enterprise applications live in a rich ecosystem that requires multiple applications, written by different teams, to collaborate in order to get things done.

	A (Mostly) Standard Model Relational databases have succeeded because they provide the core benefits we outlined earlier in a (mostly) standard way. As a result, developers and database professionals can learn the basic relational model and apply it in many projects.			
	b. Explain briefly about impedance mismatch, with a neat diagram. **Scheme : Definition + explanation + Diagram – 1+2+2 Marks** **Solution :**  • For Application developers using relational databases, the biggest frustration has been what's commonly called the impedance mismatch: the difference between the relational model and the in-memory data structures. • The relational data model organizes data into a structure of tables. Where a tuple is a set of name-value pairs and a relation is a set of tuples. • The values in a relational tuple have to be simple they cannot contain any structure, such as a nested record or a list. This limitation isn't true for in-memory data structures, which can take on much richer structures than relations. • The impedance mismatch lead to relational databases being replaced with databases that replicate the in- memory data structures to disk. • Impedance mismatch has been made much easier to deal with by the wide availability of object relational mapping frameworks, such as Hibernate and iBATIS that implement well-known mapping patterns.	[5]		
3.	Describe the following with reference to aggregate data models: Consequences of Aggregate Orientation, Document data model, Key-value data model and Column family stores. **Scheme : Definition + explanation with Diagram for each – 3+2+3+2 Marks** **Solution :** **Consequences of Aggregate Orientation** • Relational databases have no concept of aggregate within their data model, so we call them aggregate-ignorant. In the NoSQL world, graph databases are also aggregate-ignorant. Being aggregate-ignorant is not a bad thing. It's often difficult to draw aggregate boundaries well, particularly if the same data is used in many different contexts. • An order makes a good aggregate when a customer is making and reviewing orders, and when the retailer is processing orders. • However, if a retailer wants to analyse its product sales over the last few months, then an order aggregate becomes trouble. • An aggregate-ignorant model allows you to easily look at the data in different ways, so it is a better choice when you don't have a primary structure for manipulating your data. **Key-Value Data Model** • Key-value databases are the simplest of the NoSQL databases: The basic data structure is a dictionary or map. You can store a value, such as an integer, string, a JSON structure, or an	[10]	1	L2

array, along with a key used to reference that value.
• For example, a simple key-value database might have a value such as "Douglas Adams". This value is then assigned an ID, such as cust1237.
• Using a JSON structure adds complexity to the database. For example, the database could store a full mailing address in addition to a person's name. In the previous example, key cust1237 could point to the following information: name: "Douglas Adams", street: "782 Southwest St.", city: "Austin", state: "TX"

Document Data Model

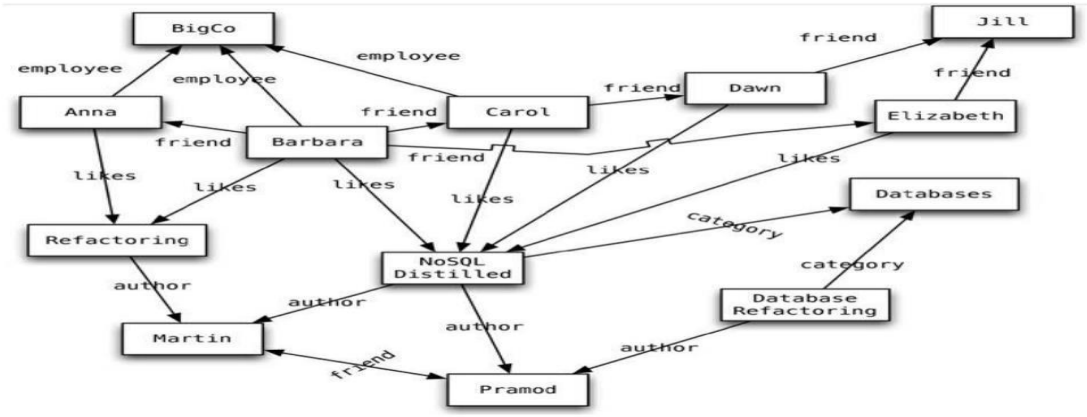
- It is a type of non-relational database that is designed to store and query data as JSON-like documents, which makes it easier for a developer to store and query data in a database.
- It works well with use cases such as catalogues, user profiles etc.
- In document store database, the data which is a collection of key-value pairs is compressed as a document store.
- The flexible, semi-structured and hierarchical nature of documents and document databases allows them to evolve with applications need.
- Example: Book document

```
{
     "id" : "98765432",
     "type" : "book",
     "ISBN": 987-6-543-21012-3,
     "Author":
          {
          "Lname":"Roe", "MI":"T", "Fname":"Richard"
          },
     "Title": "Understanding document databases"
}
```

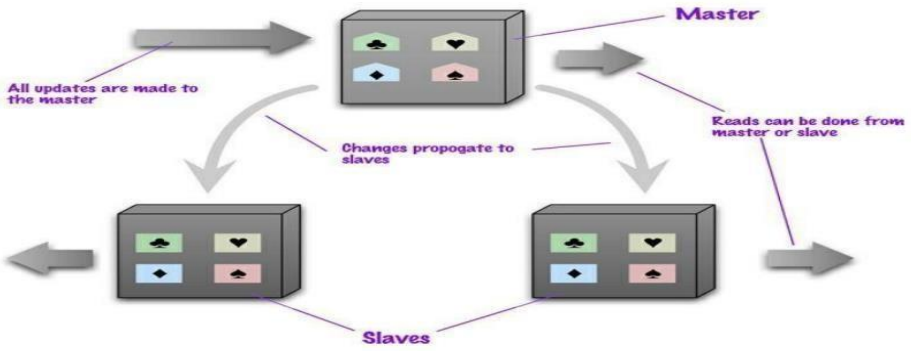
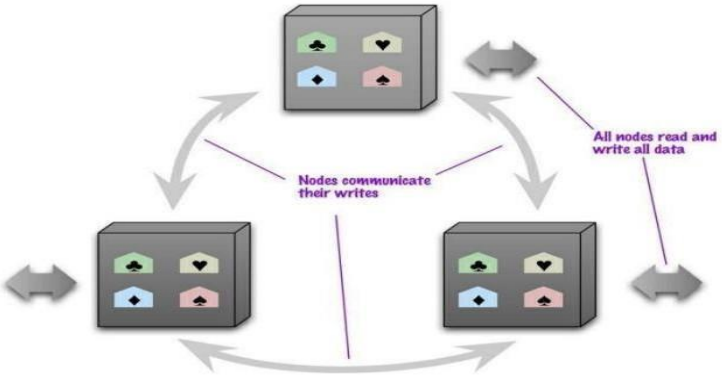
Column-Family Stores

- One of the early and powerful NoSQL databases was Google's BigTable, it is a two-level map. It has been a model that influenced later databases such as HBase and Cassandra.
- These databases with a BigTable-style data model are often referred to as column stores. The thing that made them different was the way in which they physically stored data.
- Most databases have a row as a unit of storage which, in particular, helps write performance. However, there are many scenarios where writes are rare, but you often need to read a few columns of many rows at once.
- In this situation, it's better to store groups of columns for all rows as the basic storage unit—which is why these databases are called column stores.
- BigTable and its next generation follow this notion of storing groups of columns (column families) together, we refer to this as column-family databases.

4.	Explain graph and schemaless databases with a neat diagram **Scheme : Definition + explanation with Diagram for each – 5+5 Marks** **Solution :** **Graph Databases** • Graph databases are an odd fish in the NoSQL pond. • Most NoSQL databases were inspired by the need to run on clusters, which led to aggregate oriented data models of large records with simple connections. • Graph databases are motivated by a different frustration with relational databases and thus have an opposite model—small records with complex interconnections	[10]	1	L2



	Schema less Databases • A common theme across all the forms of NoSQL databases is that they are schemaless. • When you want to store data in a relational database, you first have to define a schema a defined structure for the database which says what tables exist, which columns exist, and what data types each column can hold. • Before you store some data, you have to have the schema defined for it in relational database. **With a schema:** • You have to figure out in advance what you need to store, but that can be hard to do. **Without a schema:** • You can easily store whatever you need. • This allows you to easily change your data storage as you learn more about your project. • You can easily add new things as you discover them. • If you find you don't need some things any more, you can just stop storing them, without worrying about losing old data as you would if you delete columns in a relational schema. • A schema puts all rows of a table into a straightjacket, which becomes awkward if you have different kinds of data in different rows. You either end up with lots of columns that are usually null (a sparse table), or you end up with meaningless columns, like custom column 4.			
5.	Identify the type of conflict in the following scenario: Martin and Pramod share a common Google sheet online. Both read a file. Martin updates the document and forgets to save the file. On the other hand, Pramod updates the sheet and saves the file. The content updated by Martin overwritten by Pramod. The data updated by Martin is lost. Apply suitable technique of consistency in solving the above conflict to handle the NoSQL database. **Scheme : Explanation of Data consistency by applying different techniques with conclusion – 2+2+2+2+2 Marks each** **Solution :** The scenario involves a conflict of data consistency. Specifically, it's a classic case of a last-write-wins issue, where concurrent updates from different users can lead to data loss, as seen when Martin's unsaved changes are overwritten by Pramod's saved updates. To address this conflict in a NoSQL database, need to apply the following techniques: 1.Optimistic Concurrency Control (OCC) - Description: OCC allows multiple users to read the same data simultaneously but requires a check before writing back any changes. Before saving, the application checks whether the original data has been modified by another user since it was read. - Implementation: When Martin or Pramod makes updates, they read the version number or timestamp of the document and before saving, they compare the current version with the one they initially read. If it's the same, they can safely write; if not, they receive an error and must reconcile the changes manually. 2. Versioning -Description: Each update creates a new version of the document, allowing for tracking changes over time. -Implementation: Instead of overwriting data, the system saves each update as a new version. Users can then choose to view, merge, or restore previous versions if necessary. 3. Conflict-Free Replicated Data Types (CRDTs) - Description: CRDTs allow multiple users to make changes concurrently without conflicts. -Implementation: Changes are merged automatically based on predefined rules (e.g., set operations), allowing both users' updates to coexist. This technique is more complex but can be useful in highly collaborative environments. 4. User Notification and Merge Strategy -Description: Notify users of changes and provide a mechanism to resolve conflicts. - Implementation: When Pramod saves his changes, if Martin's updates are still in memory, the system alerts Martin that there have been changes since he last saved. Martin can then review both sets of changes and choose how to merge them. Conclusion: To effectively manage the conflict in your scenario, Optimistic Concurrency Control or Versioning would be the most straightforward techniques to implement, ensuring that users are aware of changes and can handle conflicts gracefully.	[10]	1	L3

6.	Explain single server, master slave and peer to peer distribution models **Scheme : Explanation and Diagram– 3+3+4 Marks** **Solution:** **Single Server: -** ● The first and the simplest distribution option is the one we would most often recommend no distribution at all. ● Run the database on a single machine that handles all the reads and writes to the data store. **Master Slave:-** ● With master-slave distribution, you replicate data across multiple nodes. One node is designated as the master, or primary. ● This master is the authoritative source for the data and is usually responsible for processing any updates to that data. ● The other nodes are slaves, or secondaries. A replication process synchronizes the slaves with the master. **Peer to Peer:-** ● Master-slave replication helps with read scalability but doesn't help with scalability of writes. It provides resilience against failure of a slave, but not of a master. ● Essentially, the master is still a bottleneck and a single point of failure. Peer-to-peer replication attacks these problems by not having a master. ● All the replicas have equal weight, they can all accept writes, andthe loss of any of them doesn't prevent access to the data store.  **Figure 1.2. Data is replicated from master to slaves. The master services all writes; reads maycome from either master or slaves.**  **Figure 1.3. Peer-to-peer replication has all nodes applying reads and writes to all the data.**	[10]	1	L2

Faculty Signature CCI Signature HOD Signature

USN										

Internal Assessment Test 1 – OCT 2024

Sub:	**NOSQL DATABASE**					Sub Code:	**21CS745**	Branch:	**ISE**
Date:	**15/10/2024**	Duration:	**90 min**	Max Marks:	**50**	Sem/Sec:	**VII/ A, B & C**		**OBE**

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1.	What is NOSQL? Explain briefly about aggregate data models with, Considering example of Relations and Aggregates.	[10]	1	L2
2.	a. Briefly describe the value of Relational databases	[5]	1	L2
	b. Explain briefly about impedance mismatch, with a neat diagram.	[5]		
3.	Describe the following with reference to aggregate data models: Consequences of Aggregate Orientation, Document data model, Key-value data model and Column family stores.	[10]	1	L2
4.	Explain graph and schemaless databases with a neat diagram	[10]	1	L2
5.	Identify the type of conflict in the following scenario: Martin and Pramod share a common Google sheet online. Both read a file. Martin updates the document and forgets to save the file. On the other hand, Pramod updates the sheet and saves the file. The content updated by Martin overwritten by Pramod. The data updated by Martin is lost. Apply suitable technique of consistency in solving the above conflict to handle the NoSQL database.	[10]	2	L3
6.	Explain single server, master slave and peer to peer distribution models	[10]	2	L2

Faculty Signature CCI Signature HOD Signature

USN										

Internal Assessment Test 1 – OCT 2024

Sub:	**NOSQL DATABASE**					Sub Code:	**21CS745**	Branch:	**ISE**
Date:	**15/10/2024**	Duration:	**90 min**	Max Marks:	**50**	Sem/Sec:	**VII/ A, B & C**		**OBE**

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1.	What is NOSQL? Explain briefly about aggregate data models with, Considering example of Relations and Aggregates.	[10]	1	L2
2.	a. Briefly describe the value of Relational databases	[5]	1	L2
	b. Explain briefly about impedance mismatch, with a neat diagram.	[5]		
3.	Describe the following with reference to aggregate data models: Consequences of Aggregate Orientation, Document data model, Key-value data model and Column family stores.	[10]	1	L2
4.	Explain graph and schemaless databases with a neat diagram	[10]	1	L2
5.	Identify the type of conflict in the following scenario: Martin and Pramod share a common Google sheet online. Both read a file. Martin updates the document and forgets to save the file. On the other hand, Pramod updates the sheet and saves the file. The content updated by Martin overwritten by Pramod. The data updated by Martin is lost. Apply suitable technique of consistency in solving the above conflict to handle the NoSQL database.	[10]	2	L3
6.	Explain single server, master slave and peer to peer distribution models	[10]	2	L2

Faculty Signature CCI Signature HOD Signature