Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

AY: 2024 - 25

Class:	BE	Semester:	VII
Course Code:		Course Name:	BDA

Name of Student:	BARI ANKIT VINOD
Roll No. :	61
Assignment No.:	3
Title of Assignment:	
Date of Submission:	
Date of Correction:	

Evaluation

Performance Indicator	Max. Marks	Marks Obtained
Completeness	5	4
Demonstrated Knowledge Legibility	3	3
Legibility	2	2
Total	10	9

Performance Indicator	Exceed Expectations (EE)	Meet Expectations (ME)	Below Expectations (BE)
Completeness	5	3-4	1-2
Demonstrated Knowledge Legibility	3	2	1
Legibility	2	1	0

Checked by

Name of Faculty : Ms. Sweety Patil

Signature :

Date : 4/9/25

Assignment No. - 3

1) Apply the concept of column family store and graphs store of NoSQL architecture patterns on student management systems.

(i) Column-family store - (modeling principles) -

- design for queries - pick partition keys + clustering keys so the most frequent queries are served by single partition scans.
- denormalize - duplicate data as needed to avoid joins.
- wide rows - store sequential / time-series events as clustering.
- Eg.

```
CREATE TABLE student_profiles (
    student_id  uuid,
    name        text,
    dob         date,
    email       text,
    phone       text,
    address     text,
    major       text,
    PRIMARY KEY (student_id) );
```

(ii) Typical Graph store - (Graph model elements) -

- Nodes : Student, Course, Instructor, Department, Assignment, Group.
- Relationships : :ENROLLED_IN, :ADVISED_BY, :TA_FOR, :PREREQ_OF, :WORKED_WITH, :SUBMITTED.
- Properties : node / edge properties like grade, semester, role, timestamp
- Eg. Cypher Neo4j constructions -

```
CREATE (s1 : Student {id: '123', 'name' : 'Alice' }),
       (c1 : Course {id : 'CS101', 'title' : 'Intro to CS'}),
       (c2 : Course {id : 'CS201', 'title' : 'Data Structures'}),
       (i1 : Instructor {id : 'I10', 'name' : 'Dr. Rao'});
CREATE (s1) - [: ENROLLED_IN {semester : '2025-Fall', grade : 'A'}]
CREATE (c1) - [: PREREQ_OF] -> (c2);
CREATE (s1) - [: ADVISED_BY] -> (i1);
```

Q. 2) Consider the case of Netflix adapting Apache cassandra (NoSQL). show how bussiness problems like high availability, real-time, personalise, and global scalability were solved faster, cheper, and more effiectively. state the bussiness drivers and the findings in the implementations.

(i) Business drivers for netflix -

- High availability - (Always on service) - Streaming must never go down - down time means instant user churn. User expect 24×7 uptime across all regions, even during failures.

(ii) Real-time personalization -

- Netflix's value is not just movies but personalized recomendations.
- Need sub-second updates of user activity to adapt recomendations.

(iii) Global scalability -

- Tens of thousands of microservices handle petabytes of data.
- User world wide needs low-latency access data must be across regions.

(iv) Cost-effectiveness -

- Traditional RDBMS shading and replication across constraints would be too costly and complex.
- Need elastic, commodity-hardware - friddly system.

• Example use cases of casandra in netflix -

- viewing history - stores every video watched by users.
- Users preferences - stores like / dislike, device activity, profiles.
- Recommendations & search - cassandra backs graph & ML models for personalization results.
- billing & accounting events - distributed, durable writes across regions.
- faster development cycle - cassandra's schema flexibility reduced engineering overhead.
- cheper operations - commonly cloud infrastructure + linear scale.
- resilience - survide large-scale AWS outage without downtime.
- Community contribution - Netflix created tools like Priam.