Ans 1 XML database product-Db2 PureXML

DB2 PureXML offers sophisticated capabilities to store, process and manage XML data in its native hierarchical format. By integrating XML data intact into a relational database structure, users can take full advantage of DB2's relational data management features.

Lower IT costs

It provides intelligent XML data management services without forcing you to transform or "shred" your XML data into tabular structures behind the scenes. This minimizes administrative overhead, simplifies your database design, and reduces the complexity of your XML applications.

Strong performance and scalability

Key features that enable DB2 to achieve strong XML runtime performance include:

1)Flexible XML indexing

2)Sophisticated cost-based query optimization

3)Support for parallel processing environments

4)Storage management and compression options

Increase business agility

Data and schema updates are easy and fast

Less Integration work needed

Example-

In DB2, you can query XML data in four different ways, using plain SQL, SQL/XML, XQuery, and XQuery with embedded SQL.

sample.xml

```
<!-- sample xml file -->
<Customer>
<Name>
<FirstName>Vinayak</FirstName>
<LastName>Raghupathy</LastName>
 </Name>
 <Address country="US">
 <City>Jersey City</City>
```

```
<Zip>7306</Zip>
</Address>
<Phone type="work">123-456-7890</Phone>
<Email>vr840@nyu.edu</Email>
</Customer>
```

The DB2 function db2-fn:sqlquery allows us to embed SQL statements in XQuery. It accepts as an argument an SQL full select statement that must return XML data.

```
for $cust in db2-fn:sqlquery('SELECT DOC FROM XPS')/Customer
let $name := concat($cust/Name/FirstName, " ", $cust/Name/LastName)
let $info := if (exists($cust/Email))
then($cust/Email[last()]/text())
else($cust/Phone[last()]/text())
return (concat($name, " : ", $info));
```

Result

Vinayak Raghupathy : vr840@nyu.edu

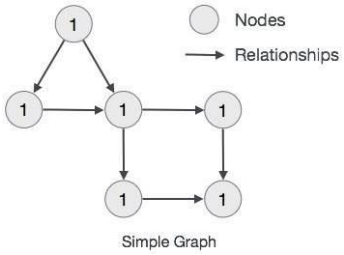
Graph Database product-Neo4j

Neo4j is an open source graph database implemented in Java and accessible from software written in other languages using the Cypher Query Language through a transactional HTTP endpoint.

Features-

1)SQL like easy language Neo4j CQL

2)Follows property graph data model.



Here we have represented Nodes by using Circles. Relationships by using arrows. Relationships are directional. We can represent Node's data in terms of Properties(key-value pairs).

3)It supports full ACID(Atomicity, Consistency, Isolation and Durability) rules.

4)It uses Native graph storage with Native GPE(Graph Processing Engine)

5)It is very easy to represent connected data.

6)It is very easy and faster to retrieve/traversal/navigation of more Connected data.

7)It does NOT require complex Joins to retrieve connected/related data as it is very easy to retrieve it's adjacent node or relationship details without Joins or Indexes.

Example- Cypher is a declarative, SQL-inspired language for describing patterns in graphs. It allows us to describe what we want to select, insert, update or delete from a graph database without requiring us to describe exactly how to do it.

Here is a simple example of a cypher query (cast of movies starting with “T”)

```
MATCH (actor:Person)-[:ACTED_IN]->(movie:Movie)
WHERE movie.title =~ "T.*"
RETURN movie.title as title, collect(actor.name) as cast
ORDER BY title ASC LIMIT 10
```

NOSQL Database product-MongoDB

MongoDB is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document. Database is a physical container for collections.

Collection is a group of MongoDB documents. Collections do not enforce a schema. A document is a set of key-value pairs. Documents have dynamic schema.

Features-

1)**Ad hoc queries**

MongoDB supports field, range queries, regular expression searches. Queries can return specific fields of documents and also include user-defined JavaScript functions.

2)**Indexing**

Any field in a MongoDB document can be indexed.

3)**Replication**

MongoDB provides high availability with replica sets.A replica set consists of two or more copies of the data. Each replica set member may act in the role of primary or secondary replica at any time. The

primary replica performs all writes and reads by default. Secondary replicas maintain a copy of the data of the primary using built-in replication.

4)**Load balancing**

MongoDB scales horizontally using sharding.The user chooses a shard key, which determines how the data in a collection will be distributed. MongoDB can run over multiple servers, balancing the load and/or duplicating data to keep the system up and running in case of hardware failure .

5)**Aggregation**

MapReduce can be used for batch processing of data and aggregation operations.

Example-

You can use the insert() method to add documents to a collection in MongoDB. If you attempt to add documents to a collection that does not exist, MongoDB will create the collection for you. Insert a document into a collection named restaurants. The operation will create the collection if the collection does not currently exist.

```
db.restaurants.insert(
{
  "address" : {
  "street" : "2 Avenue",
   "zipcode" : "10075",
   "building" : "1480",
   "coord" : [ -73.9557413, 40.7720266 ],
 },
"borough" : "Manhattan",
"cuisine" : "Mexican",
"grades" : [
{
"date" : ISODate("2014-10-01T00:00:00Z"),
 "grade" : "A-",
  "score" : 10
},
{
"date" : ISODate("2014-01-16T00:00:00Z"),
```

```
   "grade" : "B+",
   "score" : 17
  }
 ],
"name" : "Ben’s Pizza",
"restaurant_id" : "41704620"
}
)
```