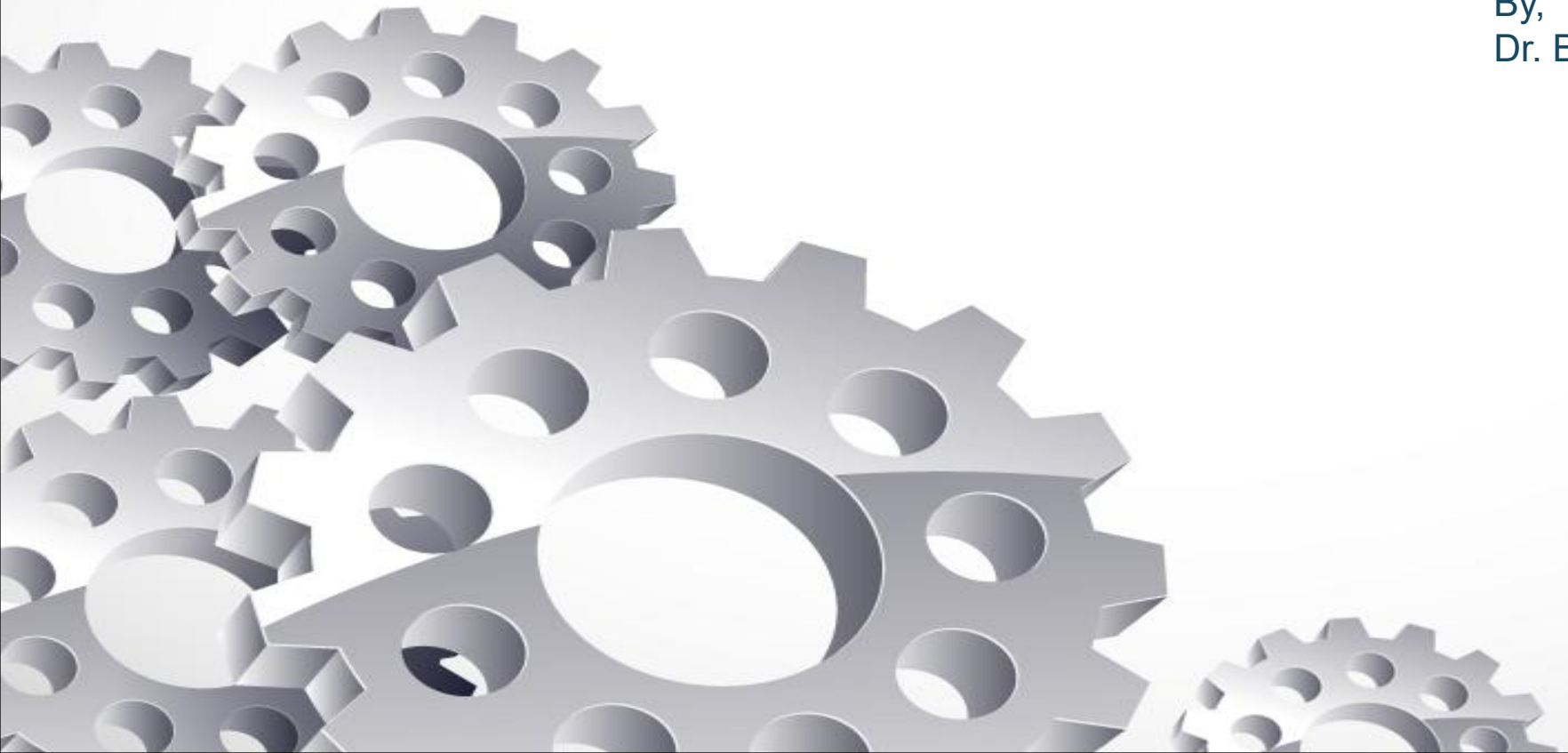
NO SQL and It's Understanding

By,
Dr. Bhargava R



SQL vs No SQL

SQL	NO SQL
Relational DB	Distributed DB
Defined Schema	Dynamic Schema
Vertical Scalable	Horizontal Scalable
Low Availability	Highly Available
Support Complex Queries	Not Supported for Complex Queries

CAP Theorem

The three letters in CAP refer to three desirable properties of distributed systems with replicated data:

- Consistency (among replicated copies)
- Availability (of the system for read and write operations)
- Partition tolerance (in the face of the nodes in the system being partitioned by a network fault).

Consistency

- Consistency means that the nodes will have the same copies of a replicated data item visible for various transactions.
- A guarantee that every node in a distributed cluster returns the same, most recent and a successful write.
- Consistency refers to every client having the same view of the data.

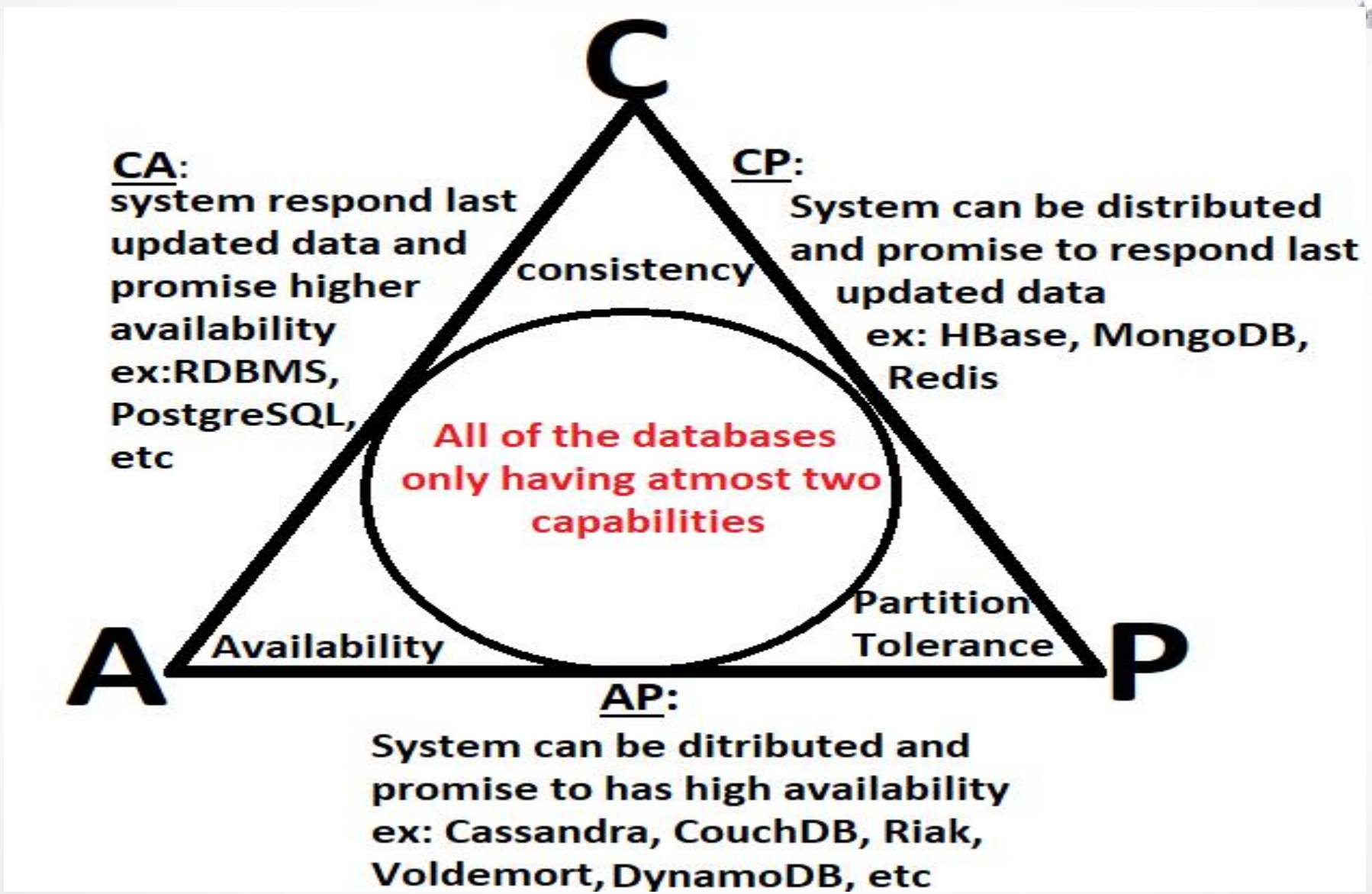
Availability

- Availability means that each read or write request for a data item will either be processed successfully or will receive a message that the operation cannot be completed.
- Every non-failing node returns a response for all the read and write requests in a reasonable amount of time. The key word here is “every”.
- In simple terms, every node (on either side of a network partition) must be able to respond in a reasonable amount of time.

Partition Tolerance

- Partition tolerance means that the system can continue operating even if the network connecting the nodes has a fault that results in two or more partitions, where the nodes in each partition can only communicate among each other.
- That means, the system continues to function and upholds its consistency guarantees in spite of network partitions. Network partitions are a fact of life.
- Distributed systems guaranteeing partition tolerance can gracefully recover from partitions once the partition heals.
-

CAP


C
CA:
system respond last updated data and promise higher availability
ex:RDBMS, PostgreSQL, etc
CP:
System can be distributed and promise to respond last updated data
ex: HBase, MongoDB, Redis
consistency
All of the databases only having atmost two capabilities
A
Availability
Partition Tolerance
P
AP:
System can be ditributed and promise to has high availability
ex: Cassandra, CouchDB, Riak, Voldemort, DynamoDB, etc

Types of NoSQL

Key Value

Example:
Riak, Tokyo Cabinet, Redis server, Memcached, Scalaris

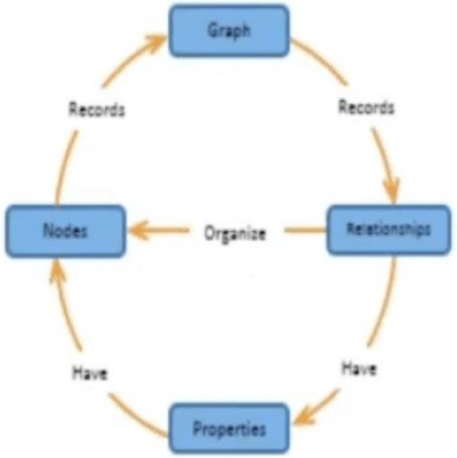
Document-Based

Example:
MongoDB, CouchDB, OrientDB, RavenDB

Column-Based

Example:
BigTable, Cassandra, Hbase, Hypertable

Graph-Based

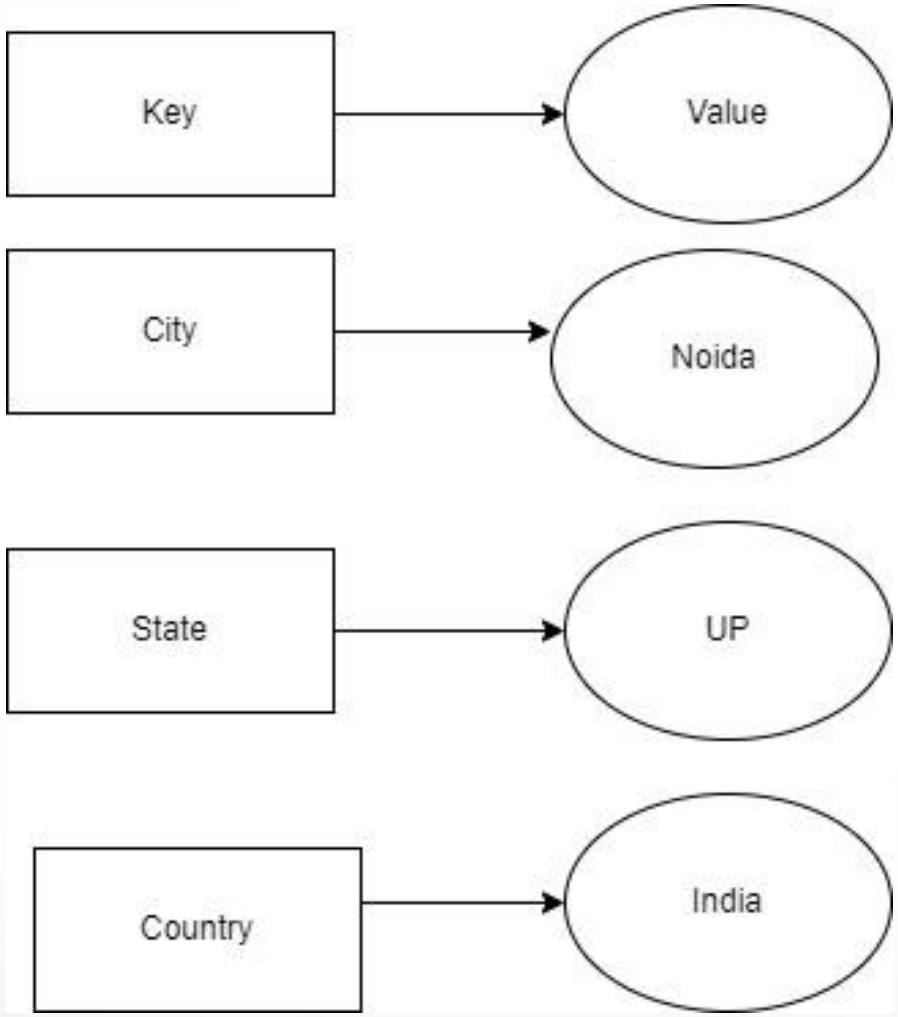


Example:
Neo4J, InfoGrid, Infinite Graph, Flock DB

Key-Value Store

- A key-value data model or database is also referred to as a key-value store.
- Array is used as a basic database in which an individual key is linked with just one value in a collection.
- For the values, keys are special identifiers.
- The collection of key-value pairs stored on separate records is called key-value databases and they do not have an already defined structure.

Key-Value Store



Key-Value Store

When to use a key-value database:

Here are a few situations in which you can use a key-value database:-

- User session attributes in an online app like finance or gaming, which is referred to as real-time random data access.
- Caching mechanism for repeatedly accessing data or key-based design.
- The application is developed on queries that are based on keys.

Features:

- One of the most un-complex kinds of NoSQL data models.
- For storing, getting, and removing data, key-value databases utilize simple functions.
- Querying language is not present in key-value databases.
- Built-in redundancy makes this database more reliable.

Key-Value Store

Advantages:

- It is very easy to use.
- Its response time is fast.
- Key-value store databases are scalable vertically as well as horizontally.
- Built-in redundancy makes this database more reliable.

Disadvantages:

- As querying language is not present in key-value databases, transportation of queries from one database to a different database cannot be done.
- The key-value store database is not refined. You cannot query the database without a key.

Document-Based

- A Document Data Model is a lot different than other data models because it stores data in JSON, BSON, or XML documents.
- It works as a **semi-structured** data model in which the records and data associated with them are stored in a single document which means this data model is not completely unstructured.
- The main thing is that data here is stored in a document.

```
{
"Name" : "Yashodhra",
"Address" : "Near Patel Nagar",
"Email" : "yahoo123@yahoo.com",
"Contact" : "12345"
}
```

Document-Based

Features:

- **Document Type Model:** As we all know data is stored in documents rather than tables or graphs, so it becomes easy to map things in many programming languages.
- **Flexible Schema:** Overall schema is very much flexible to support this statement one must know that not all documents in a collection need to have the same fields.
- **Distributed and Resilient:** Document data models are very much dispersed which is the reason behind horizontal scaling and distribution of data.
- **Manageable Query Language:** These data models are the ones in which query language allows the developers to perform CRUD (Create Read Update Destroy) operations on the data model.

Applications of Document Data Model :

- **Content Management:** These data models are very much used in creating various video streaming platforms, blogs, and similar services Because each is stored as a single document and the database here is much easier to maintain as the service evolves over time.
- **Book Database:** These are very much useful in making book databases because as we know this data model lets us nest.
- **Catalog:** When it comes to storing and reading catalog files these data models are very much used because it has a fast reading ability if incase Catalogs have thousands of attributes stored.
- **Analytics Platform:** These data models are very much used in the Analytics Platform.

Document-Based

Advantages:

- Schema-less:
- Faster creation of document and maintenance:
- Open formats:
- Built-in versioning:

Disadvantages:

- **Weak Atomicity:** It lacks in supporting multi-document ACID transactions. A change in the document data model involving two collections will require us to run two separate queries i.e. one for each collection. This is where it breaks atomicity requirements.
- **Consistency Check Limitations:** One can search the collections and documents that are not connected to an author collection but doing this might create a problem in the performance of database performance.
- **Security:** Nowadays many web applications lack security which in turn results in the leakage of sensitive data. So it becomes a point of concern, one must pay attention to web app vulnerabilities.

Column-Based

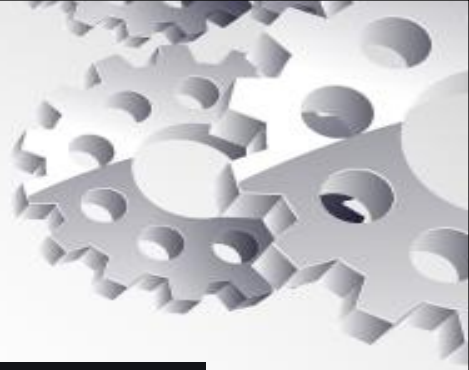
- Basically, the relational database stores data in rows and also reads the data row by row, column store is organized as a **set of columns**.
- So if someone wants to run analytics on a small number of columns, one can read those **columns directly** without consuming memory with the unwanted data.
- Columns are somehow are of the same type and gain from more efficient compression, which makes reads faster than before.
- Examples of Columnar Data Model: **Cassandra and Apache Hadoop Hbase.**
- In Columnar Data Model instead of organizing information into rows, it does in columns. This makes them function the same way that tables work in relational databases.

Column-Based

Row-Oriented Table:

S.No.	Name	Course	Branch	ID
01.	Tanmay	B-Tech	Computer	2
02.	Abhishek	B-Tech	Electronics	5
03.	Samriddha	B-Tech	IT	7
04.	Aditi	B-Tech	E & TC	8

Column-Based



Column – Oriented Table:

S.No.	Name	ID
01.	Tanmay	2
02.	Abhishek	5
03.	Samriddha	7
04.	Aditi	8

S.No.	Course	ID
01.	B-Tech	2
02.	B-Tech	5
03.	B-Tech	7
04.	B-Tech	8

S.No.	Branch	ID
01.	Computer	2
02.	Electronics	5
03.	IT	7
04.	E & TC	8

Column-Based

Advantages of Columnar Data Model :

- Well structured:
- Flexibility:
- Aggregation queries are fast:
- Scalability:
- Load Times:

Disadvantages of Columnar Data Model:

- Designing indexing Schema: To design an effective and working schema is too difficult and very time-consuming.
- Suboptimal data loading: incremental data loading is suboptimal and must be avoided, but this might not be an issue for some users.
- Security vulnerabilities: If security is one of the priorities then it must be known that the Columnar data model lacks inbuilt security features in this case, one must look into relational databases.
- Online Transaction Processing (OLTP): Online Transaction Processing (OLTP) applications are also not compatible with columnar data models because of the way data is stored.

Column-Based

Applications of Columnar Data Model:

- Columnar Data Model is very much used in various Blogging Platforms.
- It is used in Content management systems like WordPress, Joomla, etc.
- It is used in Systems that maintain counters.
- It is used in Systems that require heavy write requests.
- It is used in Services that have expiring usage.

Graph-Based

Graph Based Data Model in NoSQL is a type of Data Model which tries to focus on building the relationship between data elements.

As the name suggests Graph-Based Data Model, each element here is stored as a node, and the association between these elements is often known as Links.

Association is stored directly as these are the first-class elements of the data model. These data models give us a conceptual view of the data.

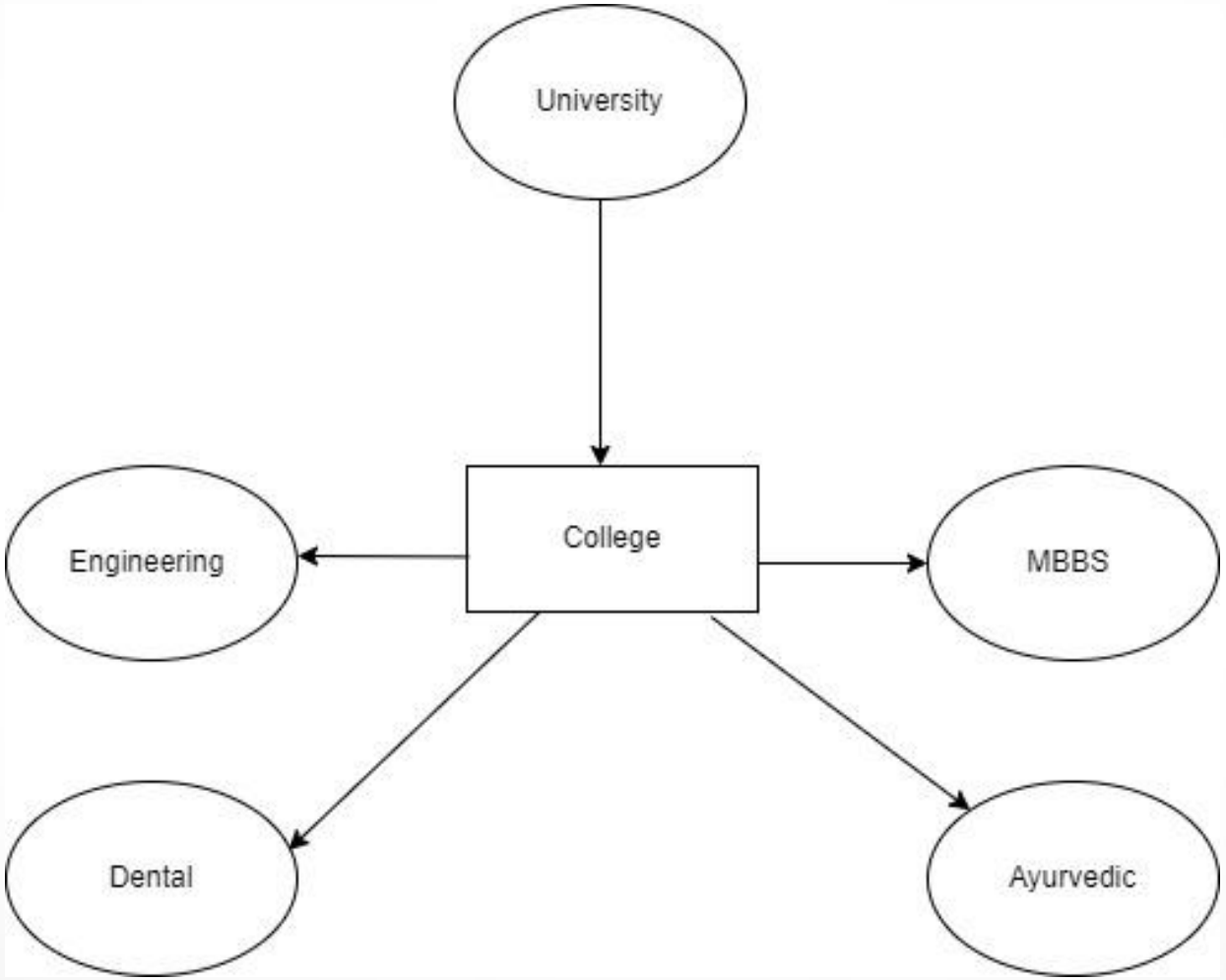
These are the data models which are based on topographical network structure.

Nodes: These are the instances of data that represent objects which is to be tracked.

Edges: As we already know edges represent relationships between nodes.

Properties: It represents information associated with nodes.

Graph-Based



Graph-Based

- In these data models, the nodes which are connected together are connected physically and the physical connection among them is also taken as a piece of data.
- Connecting data in this way becomes easy to query a relationship.
- This data model reads the relationship from storage directly instead of calculating and querying the connection steps.
- Like many different NoSQL databases these data models don’t have any schema as it is important because schema makes the model well and good and easy to edit.

Graph-Based

Advantages of Graph Data Model :

- **Structure:** The structures are very agile and workable too.
- **Explicit Representation:** The portrayal of relationships between entities is explicit.
- **Real-time O/P Results:** Query gives us real-time output results.

Disadvantages of Graph Data Model :

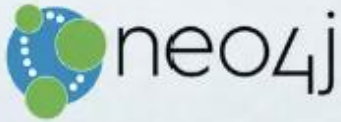
- **No standard query language:** Since the language depends on the platform that is used so there is no certain standard query language.
- **Unprofessional Graphs:** Graphs are very unprofessional for transactional-based systems.
- **Small User Base:** The user base is small which makes it very difficult to get support when running into a system.

Graph-Based

Applications of Graph Data Model:

- Graph data models are very much used in fraud detection which itself is very much useful and important.
- It is used in Digital asset management which provides a scalable database model to keep track of digital assets.
- It is used in Network management which alerts a network administrator about problems in a network.
- It is used in Context-aware services by giving traffic updates and many more.
- It is used in Real-Time Recommendation Engines which provide a better user experience.

WHY GRAPHS?

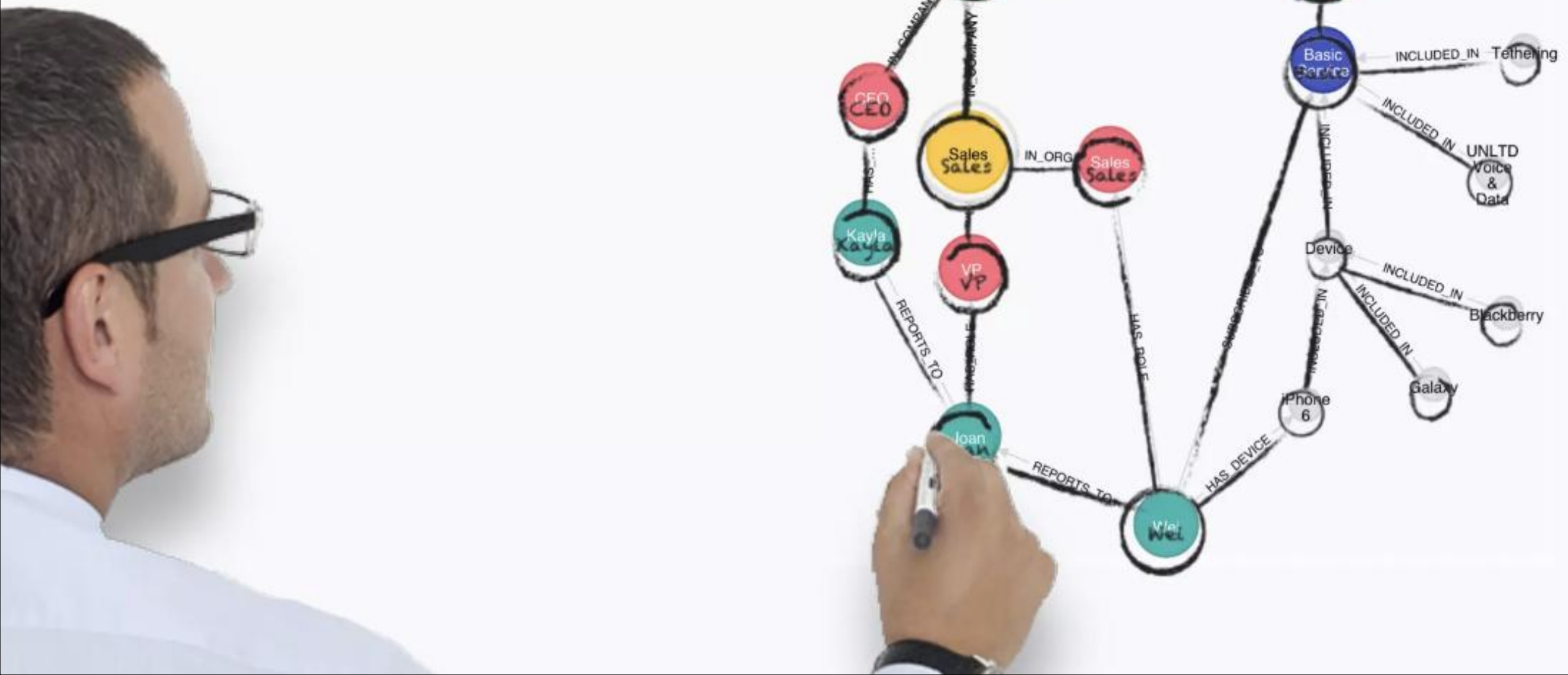


Intuitivness

Speed

Agility

Intuitiveness



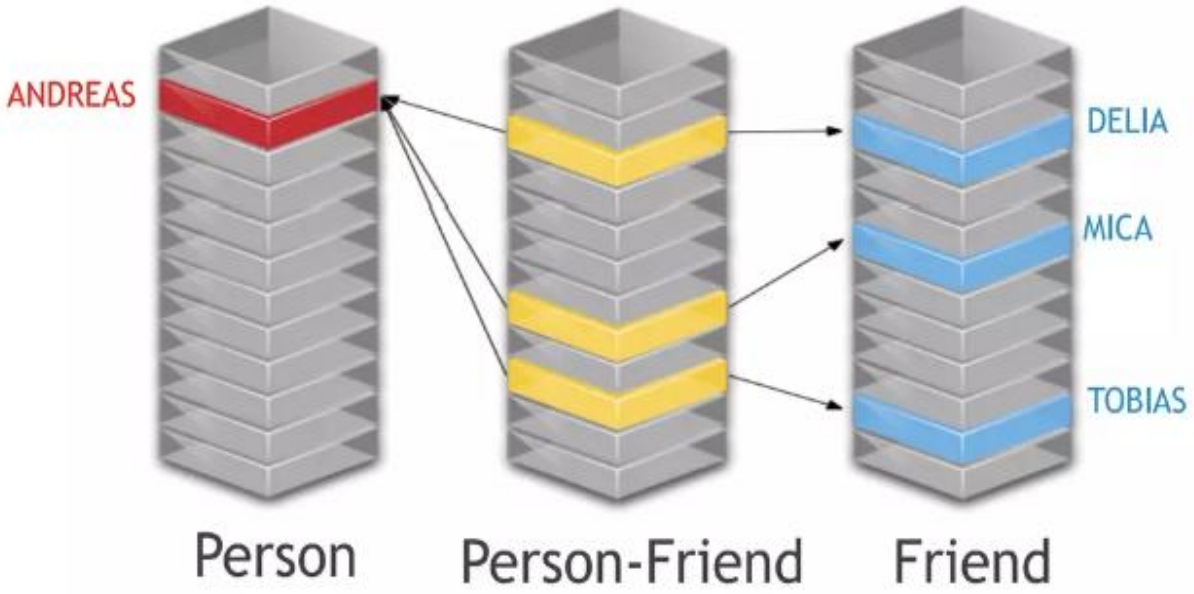
Intuitivness

Speed

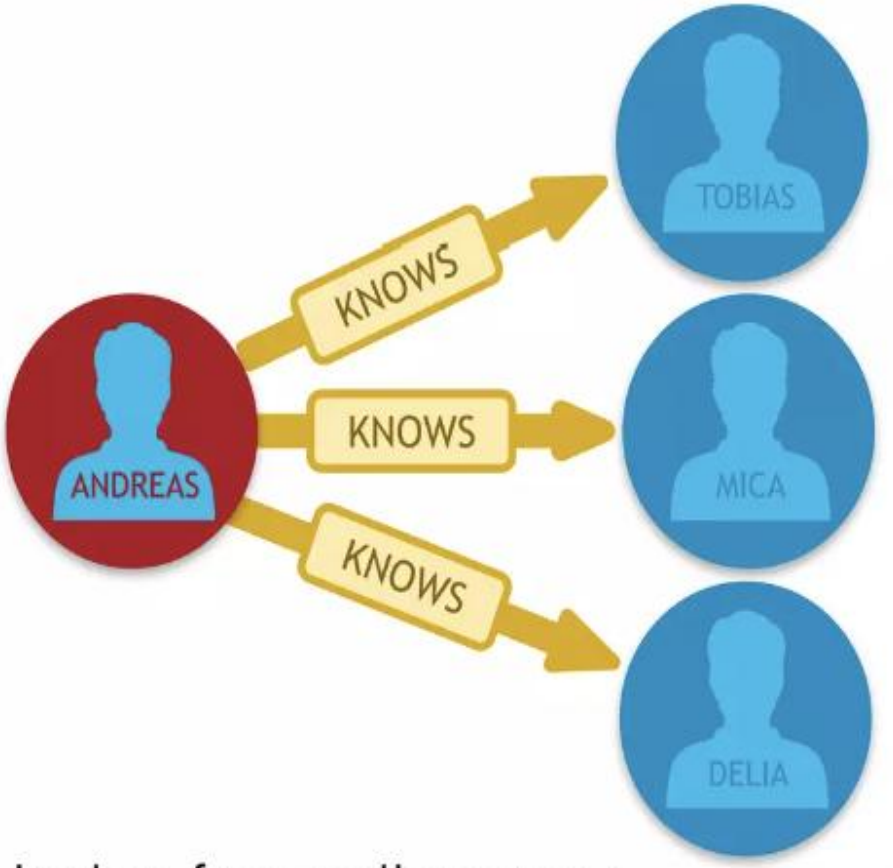
Agility

Relational Versus Graph Models

Relational Model



Graph Model



Index free adjacency

Intuitivness

Speed

Agility

Agility =

A Naturally Adaptive Model

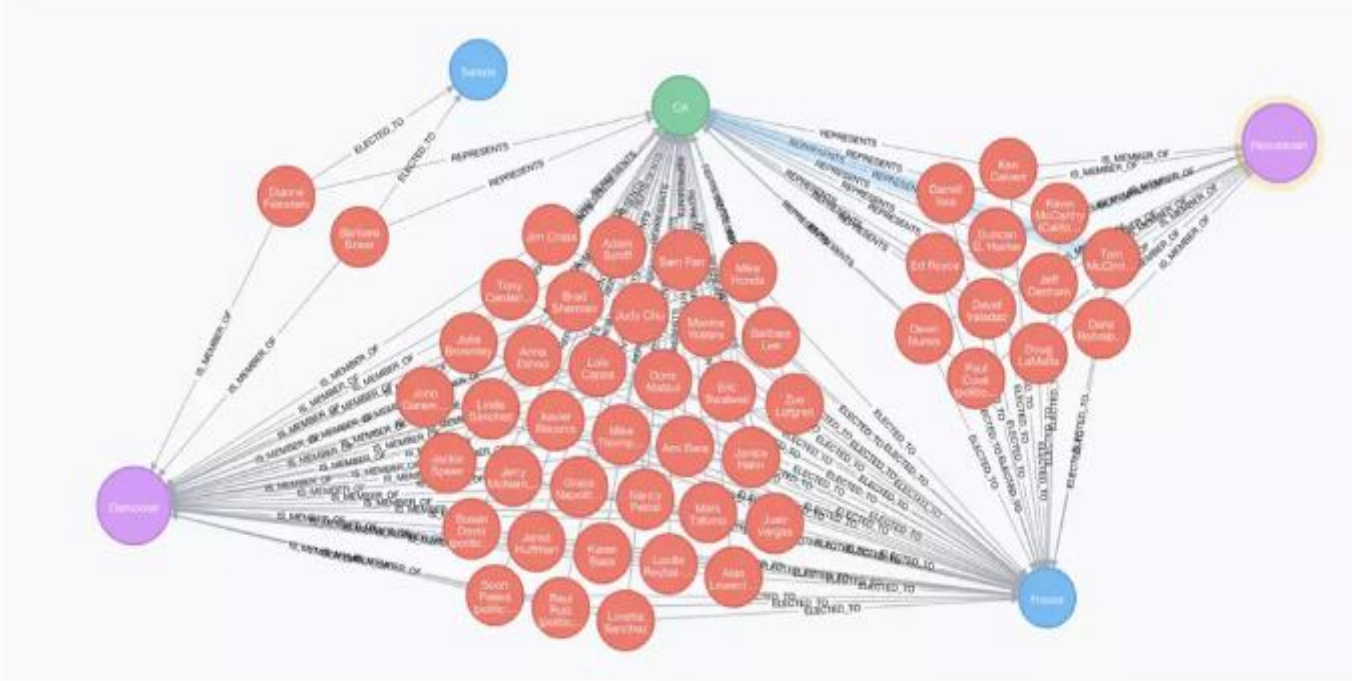
+

A Query Language Designed for Connectedness

Neo4j

Graph Database

- Property graph data model
  - Nodes and relationships
- Native graph processing
- (open)Cypher query language



neo4j.com

Neo4j - Key Product Features

Native Graph Storage
Ensures data consistency and performance

Native Graph Processing
Millions of hops per second, in real time

"Whiteboard Friendly" Data Modeling
Model data as it naturally occurs

High Data Integrity
Fully ACID transactions

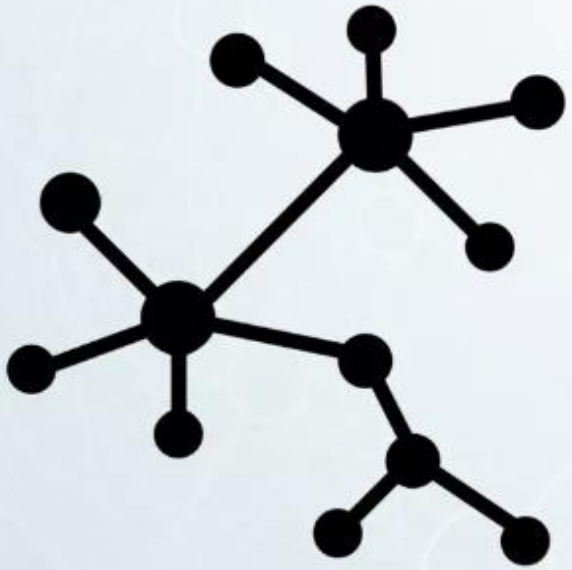
Powerful, Expressive Query Language
Requires 10x to 100x less code than SQL

Scalability and High Availability
Vertical and horizontal scaling optimized for graphs

Built-in ETL
Seamless import from other databases

Integration
Drivers and APIs for popular languages

How do you use Neo4j?



CREATE MODEL

LOAD DATA

QUERY DATA

How do you use Neo4j?

localhost:7474/browser/

Neo4j 2.3.0

Node labels

* Category Customer Employee Order Product Supplier

Relationship types

* PART_OF PRODUCT PURCHASED REPORTS_TO SOLD SUPPLIES

Property keys

_GA_TIMER_MODULE_NR address categoryID categoryName city code companyName contactName contactTitle country created_at customerID customerName description discontinued distance employeeID favorites fax firstName followers following homePage id id_str lastName location method name nodeRank orderID phone picture postalCode productID productName profile_image_url

$

$ MATCH (e:Employee)<-[:REPORTS_TO]-(sub:Employee) OPTIONAL MATCH (e)-[]-(o:...

Graph Rows Code

*(32) Employee(2) Order(30)

*(31) REPORTS_TO(1) SOLD(30)

Displaying 32 nodes, 31 relationships (completed with 31 additional relationships). AUTO-COMPLETE ON

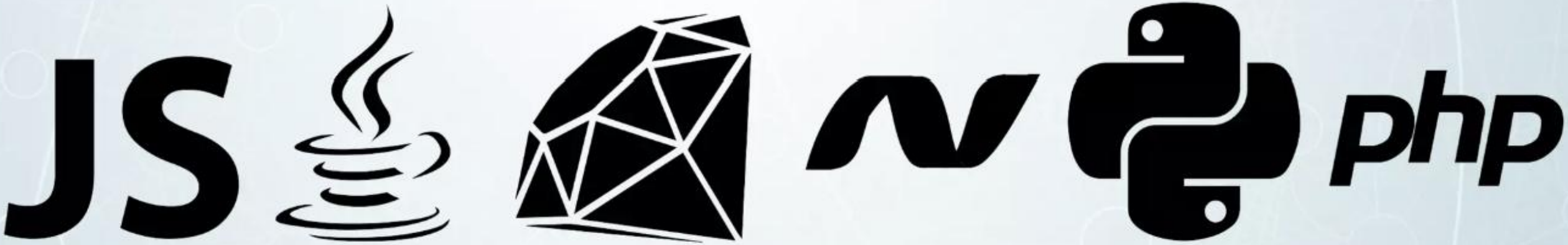
$ MATCH (e:Employee)<-[:REPORTS_TO]-(sub:Employee) OPTIONAL MATCH (e)-[]-(o:...

Graph

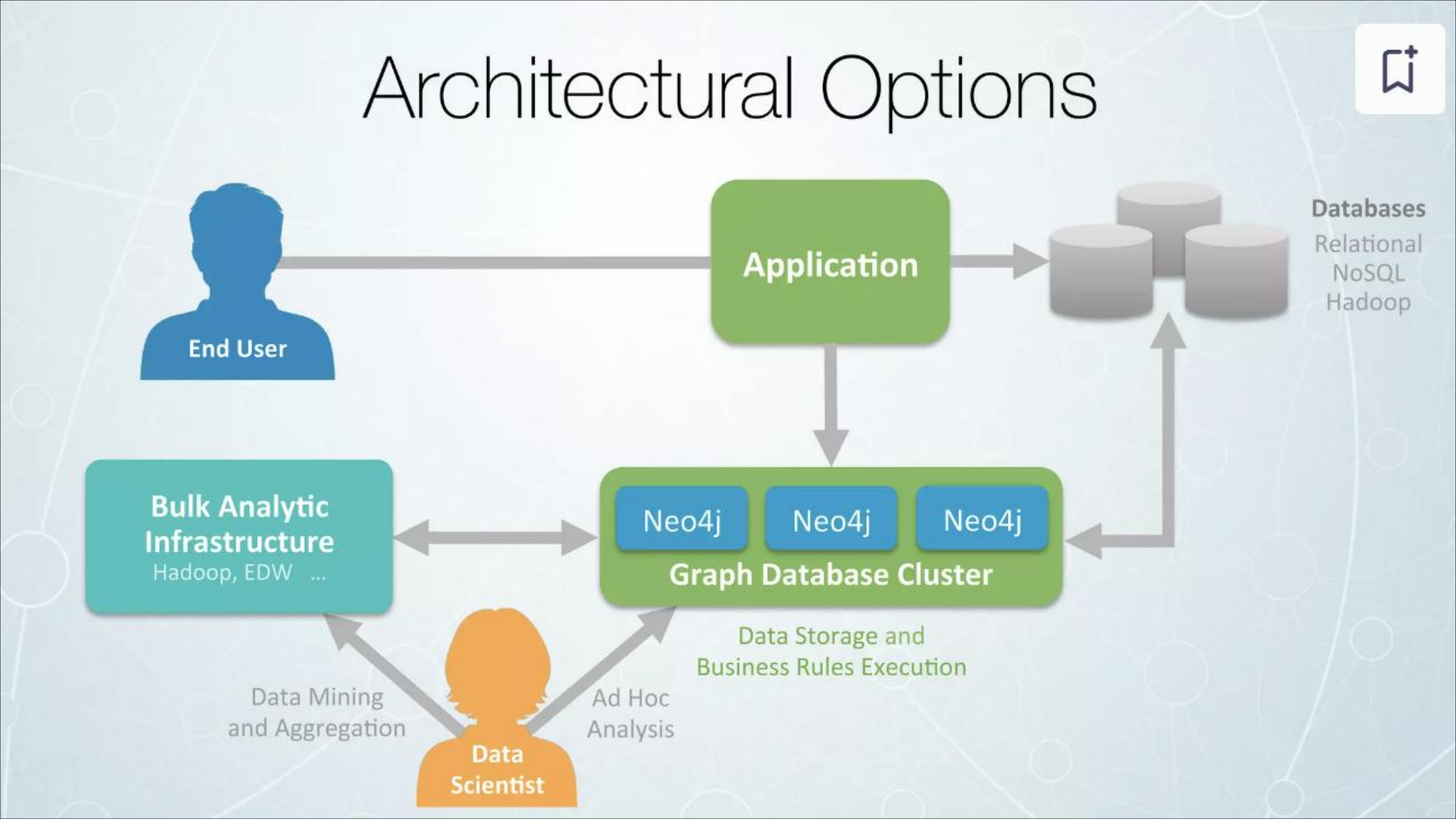
*(22) Employee(2) Order(20)

*(21) REPORTS_TO(1) SOLD(20)

Language Drivers



Architectural Options



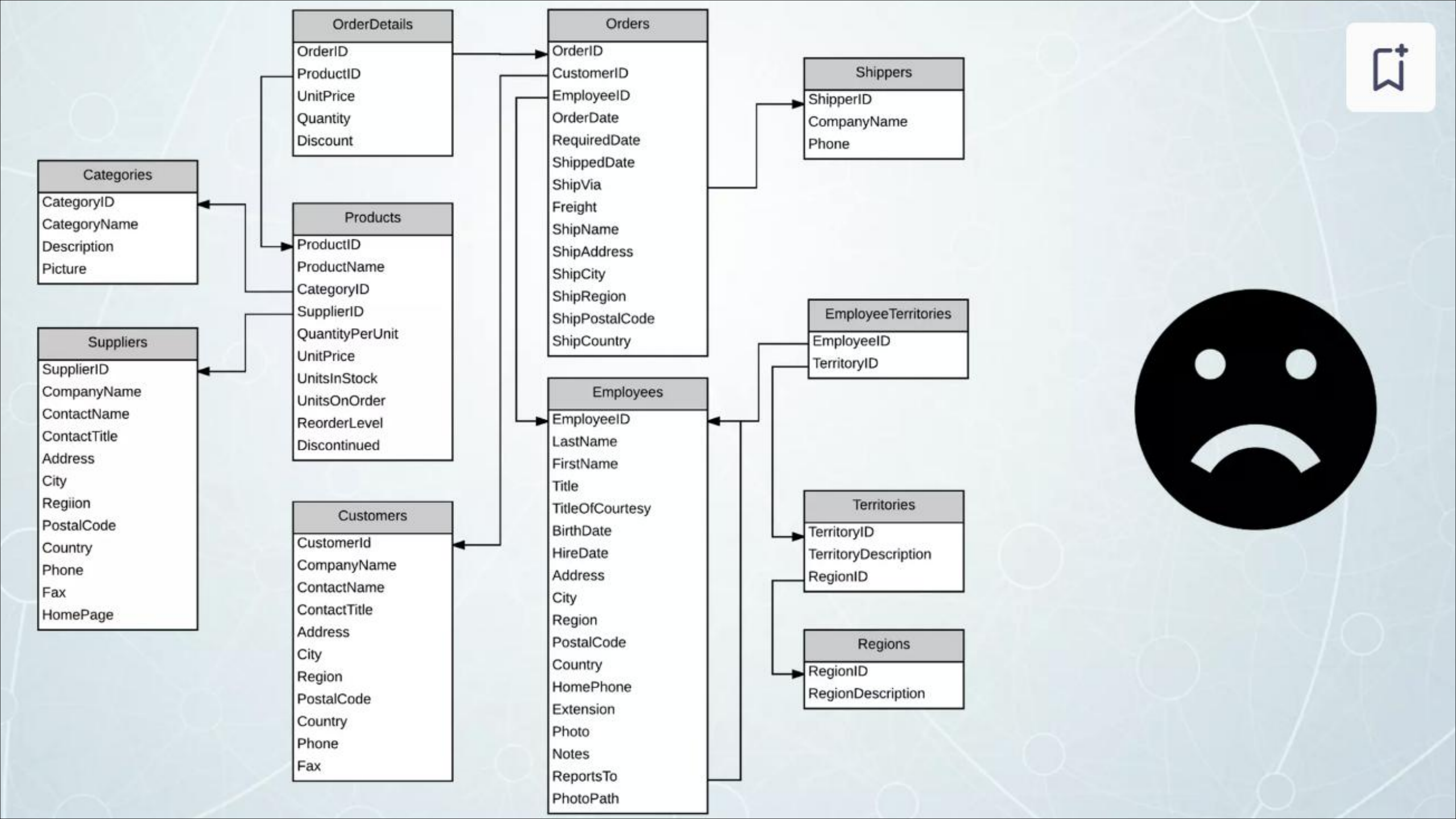
SQL Pains

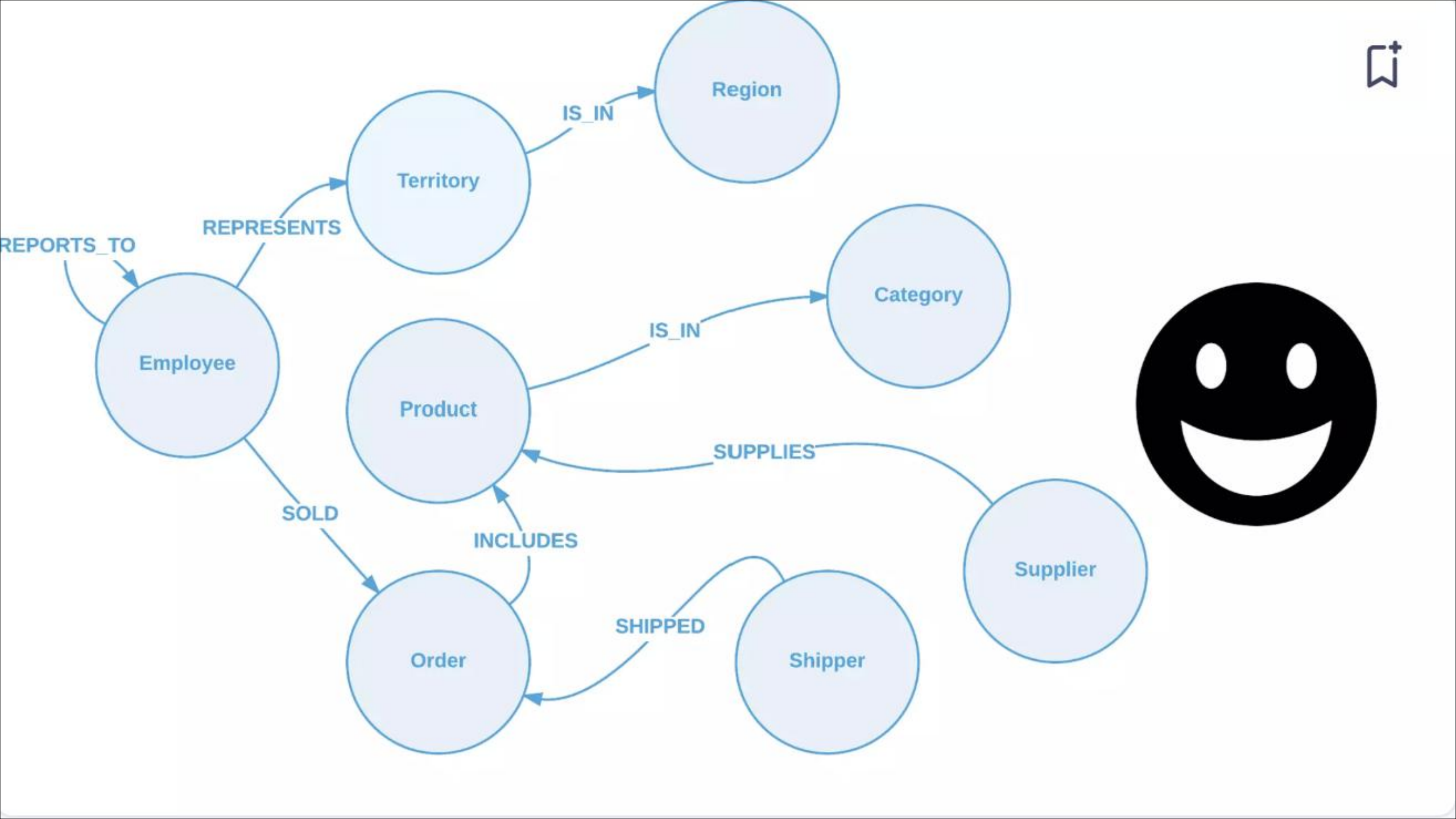
- Complex to model and store relationships
- Performance degrades with increases in data
- Queries get long and complex
- Maintenance is painful

Graph Gains

- Easy to model and store relationships
- Performance of relationship traversal remains constant with growth in data size
- Queries are shortened and more readable
- Adding additional properties and relationships can be done on the fly - no migrations

FROM RDBMS TO GRAPHS


OrderDetails
OrderID
ProductID
UnitPrice
Quantity
Discount
Orders
OrderID
CustomerID
EmployeeID
OrderDate
RequiredDate
ShippedDate
ShipVia
Freight
ShipName
ShipAddress
ShipCity
ShipRegion
ShipPostalCode
ShipCountry
Shippers
ShipperID
CompanyName
Phone
Categories
CategoryID
CategoryName
Description
Picture
Products
ProductID
ProductName
CategoryID
SupplierID
QuantityPerUnit
UnitPrice
UnitsInStock
UnitsOnOrder
ReorderLevel
Discontinued
Suppliers
SupplierID
CompanyName
ContactName
ContactTitle
Address
City
Regiion
PostalCode
Country
Phone
Fax
HomePage
Customers
CustomerId
CompanyName
ContactName
ContactTitle
Address
City
Region
PostalCode
Country
Phone
Fax
Employees
EmployeeID
LastName
FirstName
Title
TitleOfCourtesy
BirthDate
HireDate
Address
City
Region
PostalCode
Country
HomePhone
Extension
Photo
Notes
ReportsTo
PhotoPath
EmployeeTerritories
EmployeeID
TerritoryID
Territories
TerritoryID
TerritoryDescription
RegionID
Regions
RegionID
RegionDescription


Region
IS_IN
Territory
REPRESENTS
REPORTS_TO
Employee
Category
IS_IN
Product
SUPPLIES
SOLD
INCLUDES
Supplier
SHIPPED
Order
Shipper

Thank You

By,
Dr. Bhargava R