

User Story - 1

During these unprecedented pandemic situations where it is impossible to meet near ones, friends and relatives, Mr. Ralf finds it difficult to meet them for any of the occasions and exchange the best wishes or gifts, so, he wants to at least surprise them with best wishes or gifts delivered at their doorstep.

Identified Use Cases – Implemented in Neo4j

1. Search for gift items based on the occasion category (example: Birthday, Anniversary etc..).
2. Search for gift products based on the occasion sub-category (example: Greeting cards, chocolates etc... for birthday) directly and products are viewed.
3. Search gift stores at a particular location.
4. Search gift items based on gift stores.
5. Search for gift items and the gift items based on locations, occasions and gift stores are displayed.

Actors - Users

Detailed description

1. The first use case Interaction of the user and the database searches for the different occasions for which gifts are available. When the user opens the website and clicks on view all occasions, then the occasion list is displayed.
(User → Birthday, Anniversary, Wedding, Parties etc...)
2. The second use case Interaction of the user and the database searches for the gift products based on a particular occasion selected by the user and displays the gift products based on the occasion selected by the user.
(User → Birthday → Coffee Mugs, Greeting Cards, Chocolates, Photo Frames)
3. The third use case Interaction of the user and the database searches for the gift stores based on a particular location

desired and typed by the user and displays the gift stores based on the location.

(User → Mannheim → Tedi, Muller, Rossmann etc...)

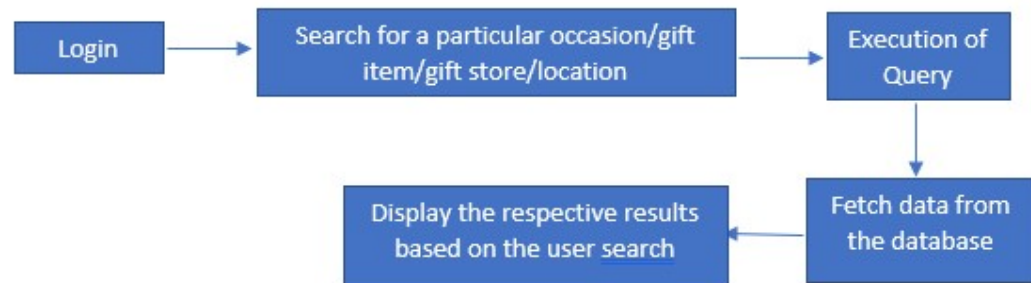
4. The fourth use case Interaction of the user and the database displays the gift items based on a particular store desired by the user and displays the gift products.

(User → Tedi, Muller, Rossmann → Coffee Mugs, Greeting Cards etc...)

5. The fifth use case Interaction of the user and the database displays the gift items which the user searches and wants with the gift stores and the location it is based.

(User → Coffee Mugs → Tedi → Heidelberg)

Data Flow



Databases – Neo4j

Databases used and why?

- Neo4j is a scalable, fast graph database, and we get high speeds results with the graph traversals from the nodes and the relationships connected. Hence, Neo4j can be easily used for searching and displaying the results.

Expressions used:

- **Use Case 1:**

Create (n:Type{name:'Occassions'})

Create (n:Type{name:'Birthday'})

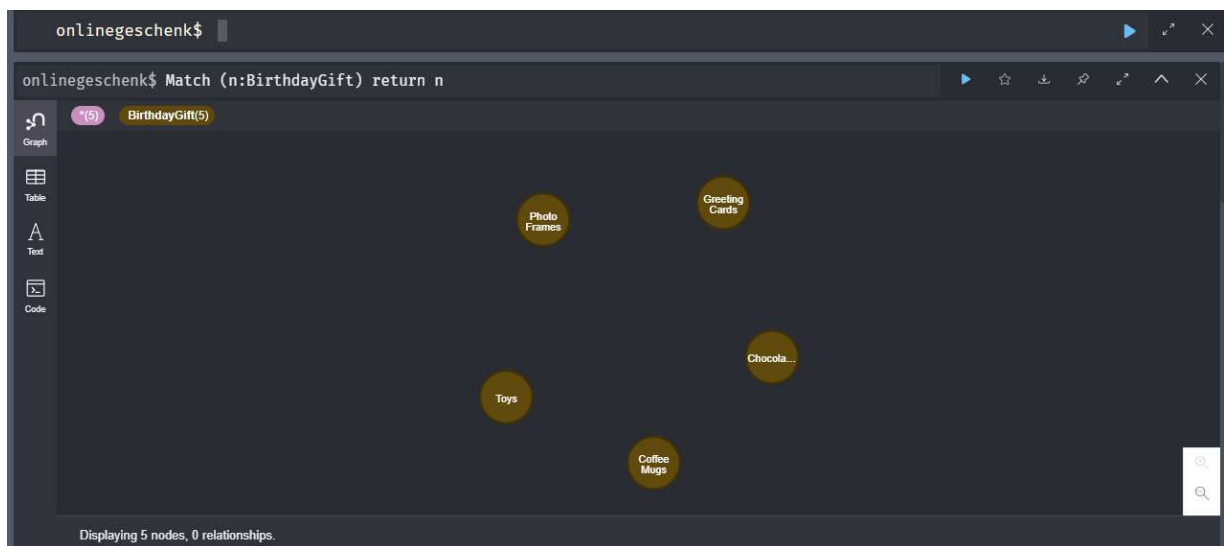
Match(o:Type) return o



- **Use Case 2:**

Create (n:BirthdayGift{name:'Coffee Mugs'})

Match (n:BirthdayGift) return n



- **Use Case 3:**

```
MATCH (n:Store) <-[:Store]-(a:Loc)
Where a.name = 'Mannheim'
RETURN n.name AS Gift_Stores, collect(a.name) AS Location
```

onlinegeschenk\$

```
1 MATCH (n:Store) <-[:Store]-(a:Loc)
2 Where a.name = 'Mannheim'
3 RETURN n.name AS Gift_Stores, collect(a.name) AS Location
4
```

	Gift_Stores	Location
1	"Tedi"	["Mannheim"]
2	"Rossman"	["Mannheim"]
3	"Bauhaus"	["Mannheim"]

- **Use Case 4:**

```
MATCH (b: BirthdayGift) <-[:GIFT_TYPE]-(t:Type) <-[:Birthday]-
(n:Store) <-[:Store]-(a:Loc) Where a.name = 'Heidelberg'
RETURN a.name AS Location, n.name AS Gift_Stores, t.name AS
Occassion, collect(b.name) AS Gift_Products
```

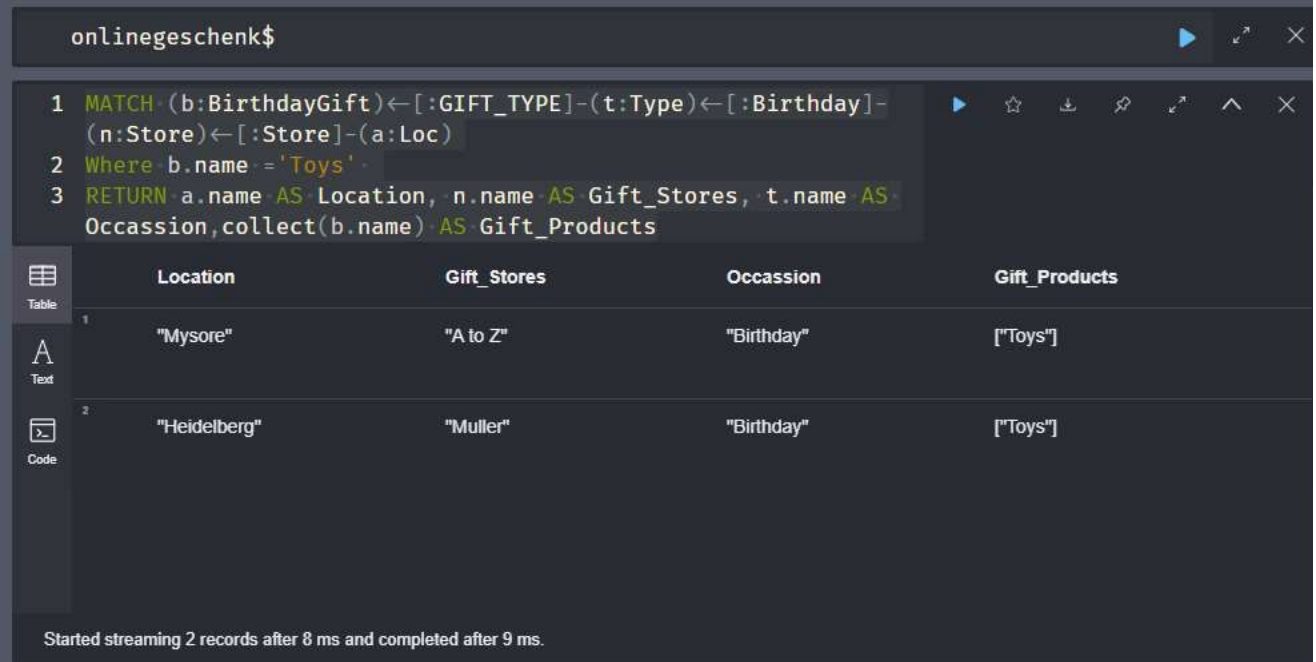
onlinegeschenk\$

```
1 MATCH (b: BirthdayGift) <-[:GIFT_TYPE]-(t:Type) <-[:Birthday]-
2 (n:Store) <-[:Store]-(a:Loc)
3 Where a.name = 'Heidelberg'
4 RETURN a.name AS Location, n.name AS Gift_Stores, t.name AS
5 Occassion, collect(b.name) AS Gift_Products
```

	Location	Gift_Stores	Occassion	Gift_Products
1	"Heidelberg"	"Muller"	"Birthday"	["Toys", "Photo Frames", "Chocolates", "Greeting Cards", "Coffee Mugs"]

- **Use Case 5:**

```
MATCH (b:BirthdayGift)-[:GIFT_TYPE]-(t:Type)-[:Birthday]-(n:Store)-[:Store]-(a:Loc)
Where b.name ='Toys'
RETURN a.name AS Location, n.name AS Gift_Stores, t.name AS Occassion, collect(b.name) AS Gift_Products
```



The screenshot shows a Neo4j Cypher query execution interface. The query is as follows:

```
1 MATCH (b:BirthdayGift)←[:GIFT_TYPE]-(t:Type)←[:Birthday]-(n:Store)←[:Store]-(a:Loc)
2 Where b.name = 'Toys'
3 RETURN a.name AS Location, n.name AS Gift_Stores, t.name AS Occassion, collect(b.name) AS Gift_Products
```

The results are displayed in a table with the following columns: Location, Gift_Stores, Occassion, and Gift_Products. There are two rows of data:

	Location	Gift_Stores	Occassion	Gift_Products
1	"Mysore"	"A to Z"	"Birthday"	["Toys"]
2	"Heidelberg"	"Muller"	"Birthday"	["Toys"]

At the bottom, a status message reads: "Started streaming 2 records after 8 ms and completed after 9 ms."

Outcome (what did you learn?)

Learning the largest and most vibrant community of graph database – Neo4j to deep search and get results for use cases.

User Story - 2

Ralf has been invited to his colleague Mathias farewell party and he has to gift him something. So, he needs some ideas to gift his colleague on his farewell day.

Identified Use Cases – Implemented in Neo4j

1. Searching for gift items based on Occasions and gift items are displayed.

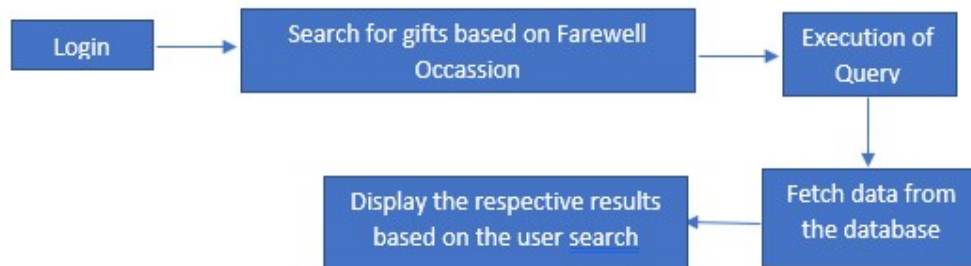
Actors - Users

Detailed description

1. The first use case Interaction of the user and the database searches for the gift products based on the occasion selected 'farewell' by the user and displays the gift products based for Farewell.

(User → Farewell → Bags)

Data Flow



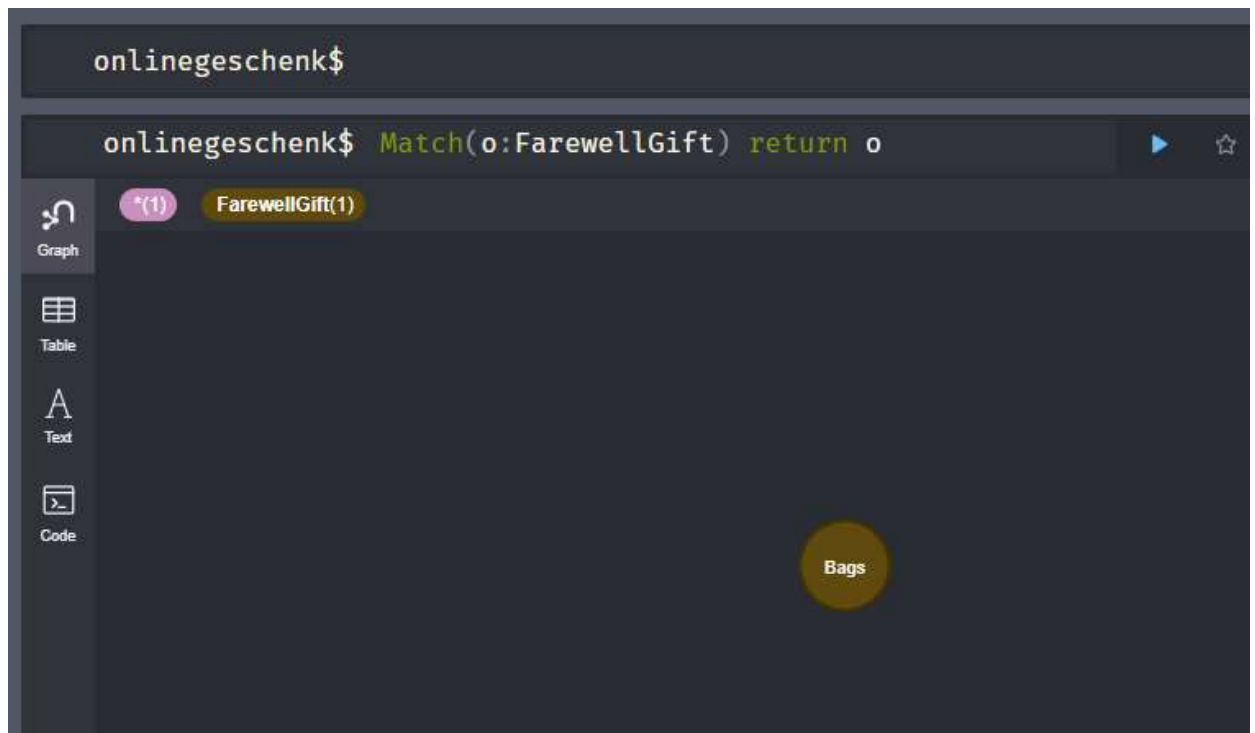
Databases – Neo4j

Databases used and why?

- Neo4j is a scalable, fast graph database, and we get high speeds results with the graph traversals from the nodes and the relationships connected. Hence, Neo4j can be easily used for searching and displaying the results.

Expressions used:

- Usecase1:
Match (o: FarewellGift) return o



Outcome (what did you learn?)

Learning the largest and most vibrant community of graph database – Neo4j to deep search and get results for use cases.

User Story - 3

I, as a user now would like to get some recommendations for gifts for a birthday party occasion for my sister's kid who is 2years old and also for my sister on her farewell day of age 30years – Used Neo4j.

Identified Use Cases – Implemented in Neo4j

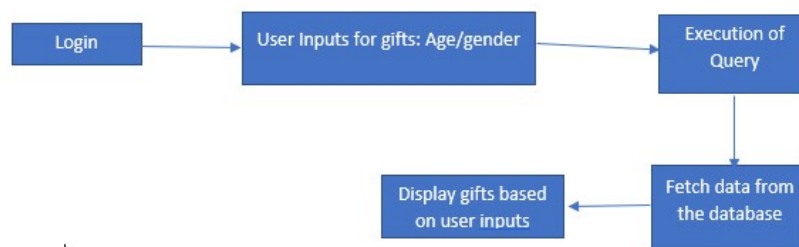
1. Users can get recommendations for gifts for any occasion based on age.
2. Users can get recommendations for gifts for any occasion based on gender and also can get recommendations on age and gender together.

Actors - Users

Detailed description

1. The first use case Interaction of the database and the user describes the user Interaction with the online website, here in this use case the user provides his/her desired occasion: birthday type and inputs the age as '2' and the gifts for the birthday occasion for a 2 year kid are displayed.
2. The second use case Interaction of the database and the user describes the user Interaction with the online website, here in this use case the user provides his/her desired occasion: farewell type and inputs the gender as 'female' and age as '30' and the gifts for the farewell occasion for a 30-year-old female are displayed.

Data Flow



Databases – Neo4j

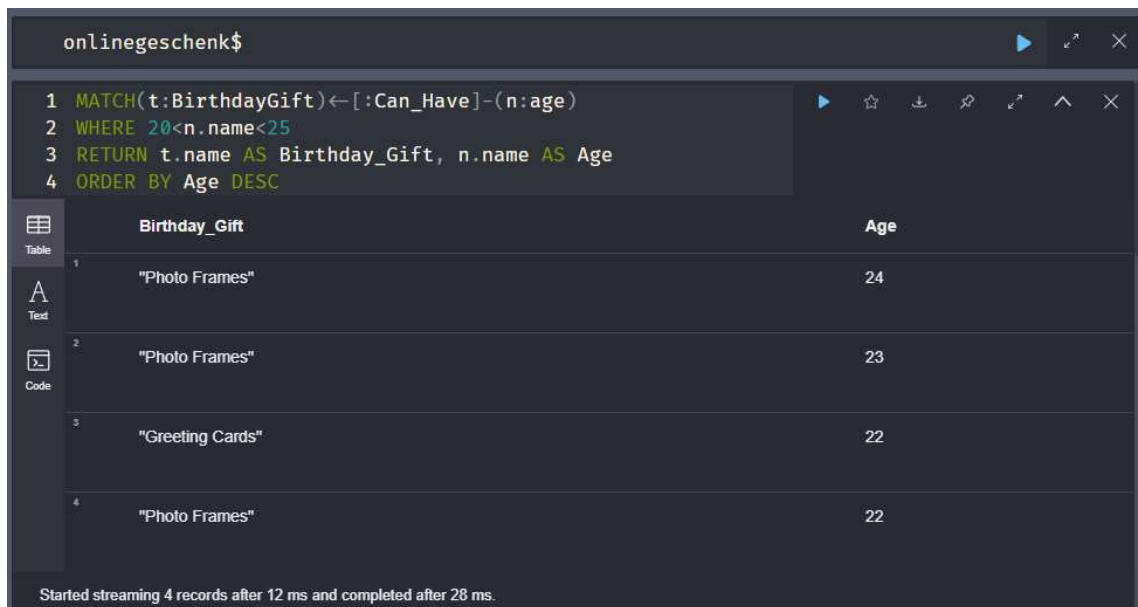
Databases used and why?

- Index-free adjacency shortened our read time and got better as the data complexity grew. Got reliably fast transactions with high throughput.
- To get recommendations Neo4j was the best as it could handle the data complexity and provide us with the recommendation query results.

Expressions used:

- **Usecase 1:**

```
MATCH(t:BirthdayGift)←[:Can_Have]-(n:age)
WHERE 20<n.name<25
RETURN t.name AS Birthday_Gift, n.name AS Age
ORDER BY Age DESC
```



The screenshot shows the Neo4j Cypher console interface. At the top, the prompt is 'onlinegeschenk\$'. Below it, a query is entered and executed. The query is:
1 MATCH(t:BirthdayGift)←[:Can_Have]-(n:age)
2 WHERE 20<n.name<25
3 RETURN t.name AS Birthday_Gift, n.name AS Age
4 ORDER BY Age DESC
The results are displayed in a table with two columns: 'Birthday_Gift' and 'Age'. There are four rows of data. The first two rows have 'Photo Frames' as the gift and ages 24 and 23 respectively. The next two rows have 'Greeting Cards' and 'Photo Frames' as gifts with ages 22 and 22 respectively. At the bottom, a status message reads: 'Started streaming 4 records after 12 ms and completed after 28 ms.'

	Birthday_Gift	Age
1	"Photo Frames"	24
2	"Photo Frames"	23
3	"Greeting Cards"	22
4	"Photo Frames"	22

Started streaming 4 records after 12 ms and completed after 28 ms.

- **Usecase 2:**

```
MATCH(t:FarewellGift)←[:loves]-(n:gender) WHERE n.name
='female'
RETURN t.name AS Farewell_Gift, n.name AS PersonType
```

```

onlinegeschenk$
1 MATCH(t:FarewellGift)←[:loves]-(n:gender)
2 WHERE n.name = 'female'
3 RETURN t.name AS Farewell_Gift, n.name AS PersonType
4

```

	Farewell_Gift	PersonType
1	"Bags"	"female"

Started streaming 1 records after 3 ms and completed after 4 ms.

- **Usecase 3:**

```

MATCH(t:BirthdayGift)←[:Can_Have]-(n:age)←[:AGED]-(s:gender)
WHERE s.name = 'female' RETURN t.name AS Birthday_Gift, n.name
AS Age, s.name AS gender

```

```

onlinegeschenk$
1 MATCH(t:BirthdayGift)←[:Can_Have]-(n:age)←[:AGED]-(s:gender)
2 WHERE s.name = 'female'
3 RETURN t.name AS Birthday_Gift, n.name AS Age, s.name AS gender

```

	Birthday_Gift	Age	gender
1	"Coffee Mugs"	30	"female"
2	"Photo Frames"	30	"female"

Started streaming 2 records after 2 ms and completed after 4 ms.

Outcome (what did you learn?)

- Learnt Cypher – Neo4j's graph query language.
- ACID compliance to ensure predictability of relationship-based queries

User Story - 4

I would like to only see gift stores which is highest popular.

Identified Use Cases – Implemented in Neo4j

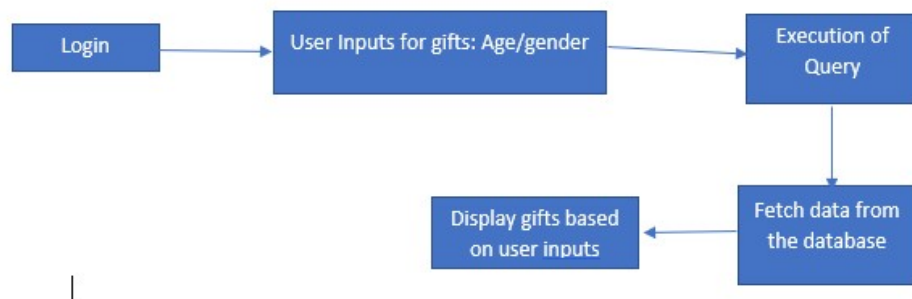
1. Users can get stores which are highest popular.

Actors - Users

Detailed description

1. The first use case Interaction of the database and the user displays the results for the user of the highest popular store that is it displays the average highest ratings given by the user to the stores. Here, the user can also see and set the store and the location.

Data Flow



Databases – Neo4j

Databases used and why?

- Neo4j is a scalable, fast graph database, and we get high speeds results with the graph traversals from the nodes and the relationships connected. Hence, Neo4j can be easily used for searching and displaying the results.
- To get recommendations Neo4j was the best as it could handle the data complexity and provide us with the recommendation query results.

Expressions used:

- Usecase 1:

```
MATCH(t:Type)<-[:Birthday]-(n:Store)<-[:rated]-(a:Ratings)
RETURN n.name AS Gift_Store, avg(a.name) AS Ratings, t.name AS
Gift_Occassions Order By Ratings desc
```



The screenshot shows a Neo4j Cypher query interface. The query is as follows:

```
1 MATCH(t:Type)<-[:Birthday]-(n:Store)<-[:rated]-(a:Ratings)
2 RETURN n.name AS Gift_Store, avg(a.name) AS Ratings,
   t.name AS Gift_Occassions
3 Order By Ratings desc
```

The results are displayed in a table with three columns: Gift_Store, Ratings, and Gift_Occassions. There are two rows of data.

	Gift_Store	Ratings	Gift_Occassions
1	"Muller"	3.0	"Birthday"
2	"A to Z"	2.0	"Birthday"

At the bottom, a status message reads: "Started streaming 2 records after 1 ms and completed after 17 ms."

Outcome (what did you learn?)

- Learnt Cypher – Neo4j's graph query language.
- ACID compliance to ensure predictability of relationship-based queries.