

Intermediate Cypher Queries

- Bu raporda oluşturulan aracı kurum data modeli üzerinde intermediate sorgu örnekleri gösterilmiştir.
- Intermediate Cypher Sorguları genel olarak şu şekilde sınıflandırılabilir;

1-Sorgu Filtreleme

Equality and Comparison: =, <, >, <=, >=, <>

NULL Check: IS NULL, IS NOT NULL

Value in List: IN, ANY, ALL

Range Queries: x >= min AND x <= max

2- Metin (String) İşlemleri

String Contains: CONTAINS, STARTS WITH, ENDS WITH

Case-insensitive Search: toLower(), toUpper()

Conditional Return: CASE WHEN ... THEN ... END

3- Pattern Tespiti ve Traversal

Pattern Matching: (a)-[:REL]->(b)

Graph Traversal: MATCH, length(p), nodes(p), relationships(p)

Varying Length Traversal: [:REL*], shortestPath()

4- Veri Biçimlendirme

Map Projection: n { .title, .year }

Dates and Durations: date(), duration(), duration.between()

UNWIND: Listeyi satırlara açmak için

DISTINCT: Yinelenen verileri kaldırmak için

5- Toplama ve Gruplama Fonksiyonları

Aggregation with count(): count(), avg(), sum(), min(), max()

collect() Usage: collect(), size(collect(...))

ORDER BY, LIMIT, SKIP: sıralama ve sonuç sayısını sınırlama

6- Sorgu Yapılandırma

WITH Clause: scope yönetimi, sıralama, limitleme, değişken aktarma

Scoping Variables: WITH x AS y, WITH a, b ile yeniden kapsam tanımlama

Subqueries (CALL {}): alt sorgularla ara hesaplama

Subqueries and UNION: UNION, UNION ALL, birleştirilmiş alt sorgular

7- Parametre Kullanımı

Parameter Usage: \$param, :param, :params

1- ORDER BY - LIMIT

- Bir kullanıcının son giriş yaptığı IP adresi

```
1 MATCH (u:User {user_id: "u001"})-[:LOGGED_IN_VIA]->(c:AccessContext)
2 RETURN c.ip, c.device_id, c.timestamp
3 ORDER BY c.timestamp DESC
4 LIMIT 1
5 |
```

Table RAW

	c.ip	c.device_id	c.timestamp
1	"192.168.1.1"	"device1"	"2023-07-03T00:00:00"

- En yüksek hisse alım işlem hacmine sahip 3 kullanıcı

```
1 MATCH (u:User)-[:HAS_INVESTMENT_ACCOUNT]->(inv:InvestmentAccount)<-[:RECORDED_IN]-(t:Trade)
2 WHERE t.type = "buy"
3 RETURN u.user_id,
4       round(sum(t.quantity * t.price_per_unit)) AS total_buy_volume
5 ORDER BY total_buy_volume DESC
6 LIMIT 3
```

Table RAW

	u.user_id	total_buy_volume
1	"u009"	15657.0
2	"u008"	7588.0
3	"u004"	4096.0

2- WHERE

- Bekleyen işlemler (pending)

```
1 MATCH (t:Transaction)
2 WHERE t.type = "withdrawal" AND t.status IN ["failed", "pending"]
3 RETURN t.transaction_id, t.amount, t.status, t.timestamp
4
```

Table RAW

	t.transaction_id	t.amount	t.status	t.timestamp
1	"txn003"	2760	"pending"	2023-04-19T21:00:00Z
2	"txn007"	4033	"pending"	2022-01-14T21:00:00Z
3	"txn009"	10204	"pending"	2022-08-04T21:00:00Z

3- CASE

- İşlemlerin durumunu başarılı, başarısız veya beklemede olarak etiketler.

```
1 MATCH (t:Transaction)
2 RETURN t.transaction_id,
3       CASE
4         WHEN t.status = "success" THEN "Başarılı"
5         WHEN t.status = "pending" THEN "Beklemede"
6         ELSE "Başarısız"
7       END AS durum_etiketi
8
```

Table RAW

	t.transaction_id	durum_etiketi
1	"txn003"	"Beklemede"
2	"txn004"	"Başarısız"
3	"txn005"	"Başarısız"
4	"txn006"	"Başarılı"
5	"txn007"	"Beklemede"
6	"txn001"	"Başarılı"

4- CALL - WITH

- Kullanıcının ait olduğu segmentteki toplam kişi sayısı

```
1 MATCH (u:User {user_id: "u001"})-[:BELONGS_TO]->(s:Segment)
2 CALL {
3   WITH s
4   MATCH (other:User)-[:BELONGS_TO]->(s)
5   RETURN count(other) AS user_count
6 }
7 RETURN s.name, user_count
8
```

Table RAW

	s.name	user_count
1	"low"	4

5- COUNT ()

- Kullanıcıların toplam favorilediği hisse sayısı

```
1 MATCH (u:User)-[:FAVORITED]->(s:Stock)
2 RETURN u.user_id AS user, count(s) AS favorited_count
3 ORDER BY favorited_count DESC;
4
```

Table RAW

	user	favorited_count
1	"u001"	4
2	"u002"	3
3	"u006"	2
4	"u003"	2
5	"u007"	2
6	"u008"	2

6-UNWIND

- Bir kullanıcının belirlenen bir listedeki konumlardan kaç kez giriş yaptığı

```
1 MATCH (u:User {user_id: "u004"})-[:LOGGED_IN_VIA]->(a:AccessContext)
2 UNWIND ["TR", "NL", "DE", "SE"] AS country
3 WITH country, count(CASE WHEN a.location_text CONTAINS country THEN 1 END) AS login_count
4 RETURN country, login_count
5
```

Table RAW

	country	login_count
1	"TR"	1
2	"NL"	0
3	"DE"	0
4	"SE"	0

7- OPTIONAL MATCH – UNWIND

```
1 UNWIND ["High School", "Bachelor", "Master", "PhD"] AS level
2 OPTIONAL MATCH (u:User)-[:HAS_EDUCATION_LEVEL]->(e:Education {level: level})
3 RETURN level, count(u) AS user_count
4 ORDER BY user_count DESC
5
```

Table RAW

	level	user_count
1	"High School"	3
2	"Bachelor"	3
3	"Master"	2
4	"PhD"	2

8 - GRAPH TRAVERSAL

- Bir kullanıcının favorilediği hisseleri aynı anda favorileyen diğer kullanıcıları ve kaç adımda ulaşıldığını bulmak.

```
1 MATCH path = (u1:User {user_id: "u001"})-[:FAVORITED]->(s:Stock)<-[:FAVORITED]-(u2:User)
2 WHERE u1 <> u2
3 RETURN u2.user_id AS other_user, length(path) AS steps
4 ORDER BY steps
5 |
```

Table RAW

	other_user	steps
1	"u006"	2
2	"u006"	2

9-SUBQUERIES - CALL() and UNION

- Bir kullanıcının favoriye eklediği veya alım/satım işlemi gerçekleştirdiği hisseleri listesi.

```
1 MATCH (u:User)
2 WITH u LIMIT 10
3 CALL {
4   WITH u
5   OPTIONAL MATCH (u)-[:FAVORITED]->(s:Stock)
6   RETURN s.ticker + ' (Favorited)' AS interaction
7   UNION
8   WITH u
9   OPTIONAL MATCH (u)-[:HAS_CASH_ACCOUNT]->(c:CashAccount)-[:USED_FOR]->(t:Trade)-[:INVOLVES]->(s:Stock)
10  RETURN s.ticker + ' (Traded)' AS interaction
11 }
12 RETURN u.user_id, collect(interaction) AS stock_interactions
13
```

Table RAW

	u.user_id	stock_interactions
1	"u001"	["ASELS (Favorited)", "ASELS (Traded)"]
2	"u002"	["THYAO (Favorited)", "GARAN (Favorited)", "THYAO (Trade d)"]
3	"u003"	["THYAO (Favorited)", "GARAN (Favorited)", "GARAN (Trade d)"]
4	"u004"	["BIMAS (Favorited)", "BIMAS (Traded)"]

10 - DATETIME

- Kullanıcıların sistemde oluşturdukları hesapların yaşı.

```
1 MATCH (c:CashAccount)
2 WHERE c.created_at IS NOT NULL
3 RETURN c.account_id,
4         duration.between(date(c.created_at), date()).years AS account_age
5 ORDER BY account_age DESC
6
```

Table

RAW

	c.account_id	account_age
1	"cash001"	3
2	"cash004"	3
3	"cash010"	3
4	"cash005"	2
5	"cash009"	2
6	"cash003"	1
7	"cash007"	1

11- MAP PROJECTION

- Her kullanıcının adı, doğum yılı, segmenti ve risk profilini JSON benzeri bir yapı içinde gösterir

```
1 MATCH (u:User)-[:BELONGS_TO]->(s:Segment)
2 RETURN u { .user_id, .name, .birth_year, .risk_profile, segment: s.name } AS
userInfo
3
```

Table

RAW

{ }

userInfo

```
1 {
  segment: "low",
  name: "User1",
  risk_profile: "medium",
  user_id: "u001",
  birth_year: 1994
}

2 {
  segment: "low",
  name: "User4",
  risk_profile: "low",
  user_id: "u004",
  birth_year: 1990
}
```

12- DISTINCT

- Kullanıcıların kaç farklı sektörden hisse favorilediğini tespit eder.

```
1 MATCH (u:User)-[:FAVORITED]->(s:Stock)
2 RETURN u.user_id, count(DISTINCT s.sector) AS distinct_favorite_sectors
3 ORDER BY distinct_favorite_sectors DESC
4
```

Table

RAW

	u.user_id	distinct_favorite_s
1	"u002"	2
2	"u003"	2
3	"u007"	2
4	"u008"	2
5	"u001"	1

13- AVG()

- Kullanıcıların sistemde yaptıkları ortalama hesaplar arası nakit akışı tutarı.

```
1 MATCH (u:User)-[:HAS_CASH_ACCOUNT]->(c:CashAccount)
2 MATCH (t:Transaction)-[:FROM_ACCOUNT]->(c)
3 WHERE t.type = "internal_transfer"
4 RETURN u.user_id AS userId, avg(t.amount) AS avgTransferAmount
5 ORDER BY avgTransferAmount DESC
6
```

Table

RAW

	userId	avgTransferAmou
1	"u001"	17592.0
2	"u003"	1089.0

14 - STRING CONTAINS

- "location_text" alanında "İstanbul" geçen erişim (AccessContext) kayıtlarının listesi.

```
1 MATCH (a:AccessContext)
2 WHERE a.location_text CONTAINS "İstanbul"
3 RETURN a.context_id, a.location_text, a.timestamp
4 |
```

Table RAW

	a.context_id	a.location_text	a.timestamp
1	"ctx001"	"İstanbul, TR"	"2023-07-03T00:00:00"
2	"ctx002"	"İstanbul, TR"	"2023-10-26T00:00:00"
3	"ctx003"	"İstanbul, TR"	"2025-10-13T00:00:00"
4	"ctx004"	"İstanbul, TR"	"2025-10-24T00:00:00"
5	"ctx006"	"İstanbul, TR"	"2023-12-19T00:00:00"

15 - PARAMETER

- Parametre tanımlama ve kullanımı

```
neo4j$ :param city => "Rotterdam, NL";
```

```
{
  city: "Rotterdam, NL"
}
```

✓ 1 new parameter successfully set

```
1 MATCH (a:AccessContext)
2 WHERE toLower(a.location_text) CONTAINS toLower($city)
3 RETURN a.context_id, a.location_text, a.timestamp
4
```

Table RAW

a.context_id	a.location_text	a.timestamp
"ctx005"	"Rotterdam, NL"	"2024-10-21T00:00:00"

16 - RANGE QUERIES

- Belirli bir tarih aralığına göre AccessContext node'larını filtreler.

```
1 WITH datetime("2023-01-01T00:00:00") AS startDate,  
2 datetime("2024-01-01T00:00:00") AS endDate  
3 MATCH (a:AccessContext)  
4 WHERE datetime(a.timestamp) >= startDate AND datetime(a.timestamp) <= endDate  
5 RETURN a.context_id, a.timestamp  
6 ORDER BY a.timestamp  
7
```

Table RAW

	a.context_id	a.timestamp
1	"ctx009"	"2023-03-10T00:00:00"
2	"ctx001"	"2023-07-03T00:00:00"
3	"ctx002"	"2023-10-26T00:00:00"
4	"ctx006"	"2023-12-19T00:00:00"

17- EXPLAIN

- Bu komut sorgu çalıştırmaz ve sadece sorgunun yürütme planını gösterir. Veri üzerinde işlem yapmadan ne olacağını analiz etmek için kullanılır.
- Bu örnek, 50.000₺'den fazla bakiyesi olan kullanıcıları sorgularken hangi yolların taranacağını, index kullanılıp kullanılmadığını, kaç düğümün dokunulacağını görülmesini sağlayacaktır.

```
1 EXPLAIN  
2 MATCH (u:User)-[:HAS_CASH_ACCOUNT]->(c:CashAccount)  
3 WHERE c.balance > 50000  
4 RETURN u.user_id, c.balance  
5
```

18 - PROFILE

- Bu komut sorguyu çalıştırır ve gerçek yürütme istatistiklerini verir.
- Bu örnek, "Finance" sektöründeki hisseleri favorileyen kullanıcıları getirir. PROFILE komutu sayesinde sorgunun kaç ms sürdüğünü, hangi adımda kaç kayıt işlendiğini görebiliriz.

```
1 PROFILE
2 MATCH (u:User)-[:FAVORITED]->(s:Stock)
3 WHERE s.sector = 'Finance'
4 RETURN u.user_id, s.ticker
```

Plan Table RAW

▼ NodeByLabelScan@neo4j
s
s:Stock
1 pagecache hits
0 pagecache misses
10 estimated rows
6 db hits

5 rows

▼ Filter@neo4j
s
s.sector = \$autostring_0
2 pagecache hits
0 pagecache misses
1 estimated rows
5 db hits

1 row

Cypher version: 5
Planner: COST
Runtime: SLOTTED
25 total db hits in 67 ms.

▼ CacheProperties@neo4j
s
cache[s.ticker]
0 pagecache hits