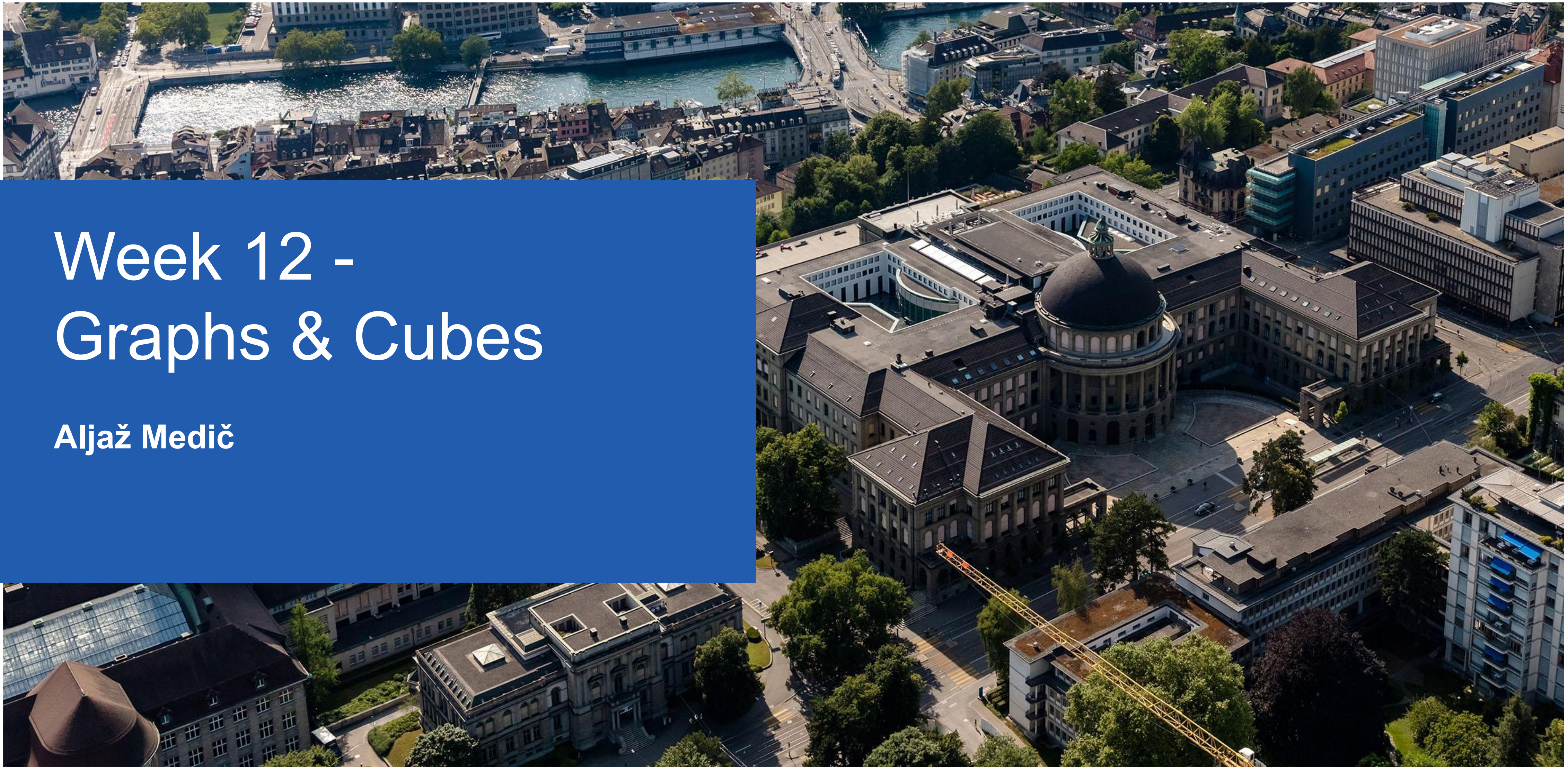
ETH zürich



Week 12 - Graphs & Cubes

Aljaž Medič

Plan for today

1. Questions about the previous Exercise Sheet?
2. Quiz
3. Neo4j, CYPHER
4. Data Cubes
5. Old exam questions

QUIZ

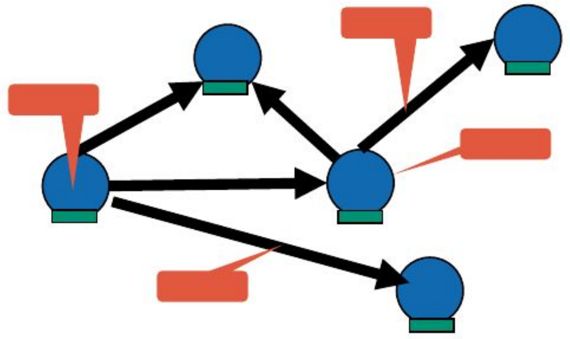
What is the main selling point for graph databases?

Graph databases excel at queries involving many **relationships** and patterns, which can be challenging for relational databases.

Which are the two main graph data models?

Labeled Property Graphs and Triple Stores (RDFs).

(Resource Description Framework)


Labeled property Graph

What are the components of the Labeled Property Graph model?

Nodes, Relationships, Properties and Labels.

Triple stores (RDF)

QUIZ

A native graph database is usually built on top of an existing relational or document database.

False. (They were built from scratch, specifically for graph-based data.)

"Index-free adjacency" in graph databases means each node directly references its neighbors, avoiding global index lookups during traversals.

True.

There is no way to shard index-free adjacency graph databases across different machines, due to their high interconnectivity.

False. (Recent (y. 2024) graph databases support partitioning across multiple machines.)

QUIZ

Which 3 ways of representing graphs in memory were mentioned at the lectures?

Adjacency list, Adjacency matrix, Incidence matrix.

In labeled property graphs …

… only nodes are allowed to have properties.

False. (Relationships may also have properties. Both may have **0..***, need unique keys.)

… a node can have multiple labels.

True.

… a relationship cannot have multiple labels.

True. (Exactly **1**)

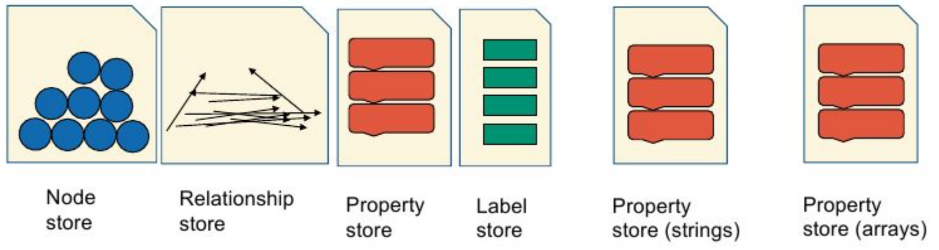
QUIZ

Both labeled property graphs and triple stores are capable of storing data with index-free adjacency.

False. (Triple-stores rely on indices in RDBMS to find connections between entities.)

Neo4j stores graph data in several different store files on disk.

True.



Relationship records in Neo4j can have variable size.

False. (Both nodes and relationships are stored in **fixed-size** records, which gives us faster lookups in the store files.)

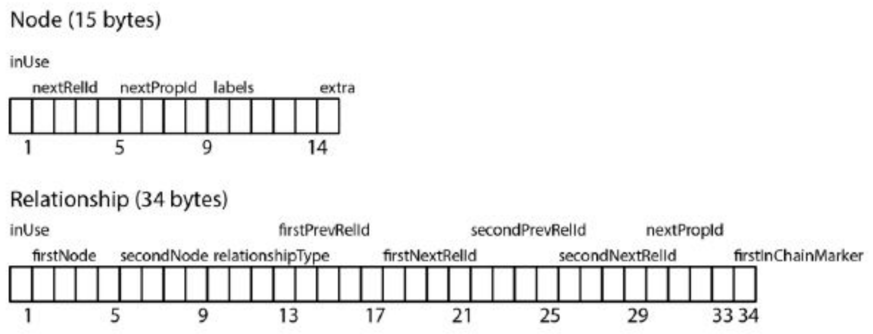
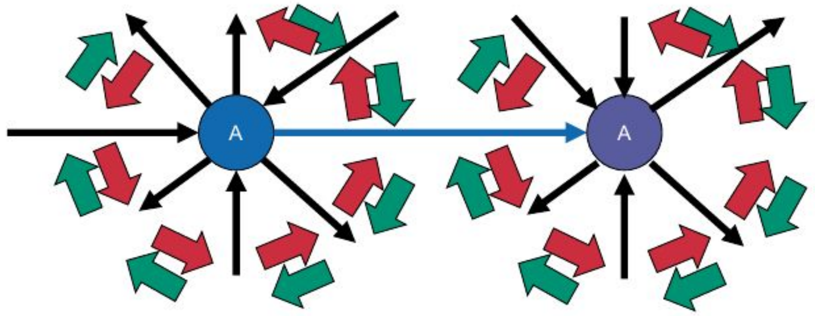


Figure 6-4. Neo4j node and relationship store file record structure



QUIZ

What are the components of triple stores?

Subject-property-object triplets.

In triple stores, …

… URI/IRI can appear in any of the three components.

True.

… a literal may appear under subject or object.

False. (Only under object.)

… a blank may not appear under property.

True.

	Subject	Property	Object
IRI	YES	YES	YES
Literal	NO	NO	YES
Blank node	YES	NO	YES

QUIZ

OLTP systems are optimized for analytical queries over large historical datasets.

False. (T - **Transactional**, the description matches OLAP. A - **Analytical**)

Once data is loaded into warehouse it remains fixed (read-only).

True.

What does ETL stand for?

Extract-Transform-Load.

QUIZ

List the four terms that we mentioned for manipulating cubes:

Slice, Dice, Roll-up, Drill-down.

An OLAP data cube consists of facts (values) defined by specific combination of dimension values.

True.

Slicing a data cube means focusing on a single value for one dimension, reducing its dimensionality by one.

True.

QUIZ

Dicing the data cube means aggregating the data alongside axis.

False. (It is focusing on a specific sub-cube, and the amount of dimensions stays the same.)

Rolling up corresponds to aggregating on a dimension, and drilling down refers to breaking the aggregated data into finer details. (They are opposites.)

True.

What is the difference between MOLAP and ROLAP?

Multidimensional OLAP uses specialized multidimensional storage for cubes, Relational OLAP uses a relational database underneath to implement the cubes.

Cypher

- Supports graph pattern matching, where the syntax looks like ASCII art.
- MATCH clause specifies the pattern.
- RETURN clause specifies what data to retrieve.

```
MATCH (alpha {name: 'Einstein' })-[:A]->(beta)-[:B]->(gamma)
RETURN gamma
```

- UNWIND functions as `for` in JSONiq.

```
WITH [ { foo:1 }, { foo:2 }, { foo:3 } ] AS a
UNWIND a as b
RETURN b
```

- WITH clause computes intermediate results (think of `let` in JSONiq*).

```
WITH 1 AS a
WITH a+1 AS b
RETURN a
```
✗

```
WITH 1 AS a
WITH a, a+1 AS
RETURN a
```
✓

```
MATCH (george {name: 'George'})<--(otherPerson)
WITH otherPerson, toUpper(otherPerson.name) AS upperCaseName
WHERE upperCaseName STARTS WITH 'C'
RETURN otherPerson.name
```

Turtle

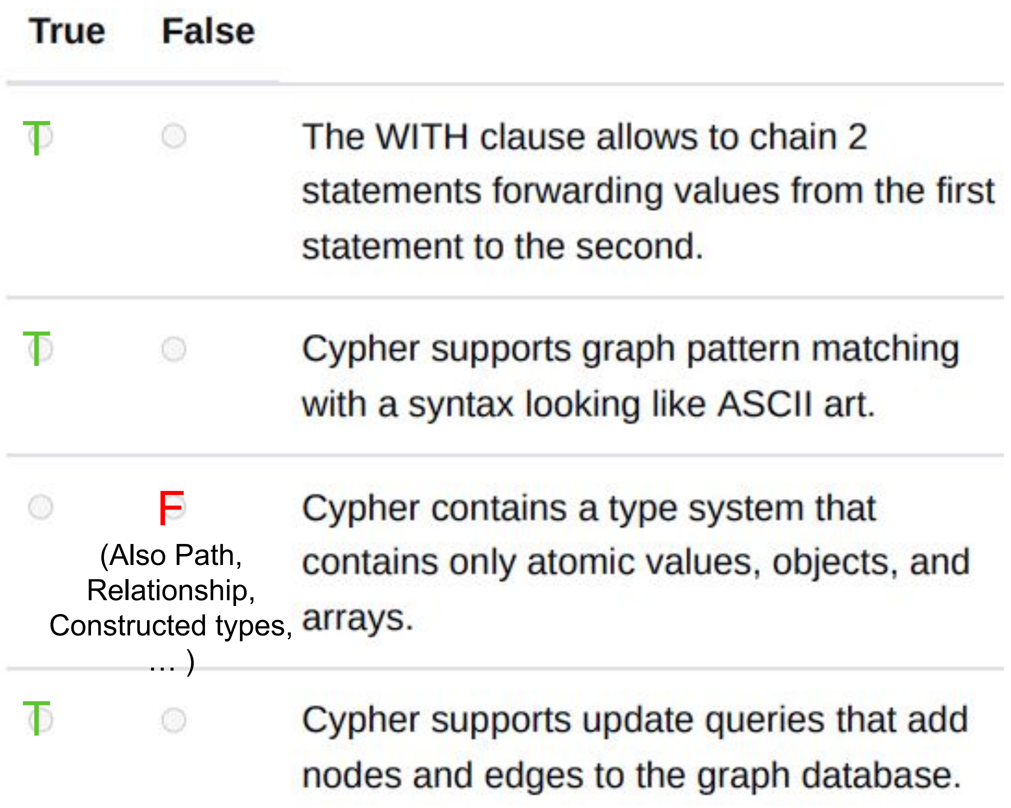
- Comma (,) for listing objects,
- Semicolon (;) for listing properties+objects,
- Dot (.) to stop.

	Subject	Property	Object
IRI	YES	YES	YES
Literal	NO	NO	YES
Blank node	YES	NO	YES

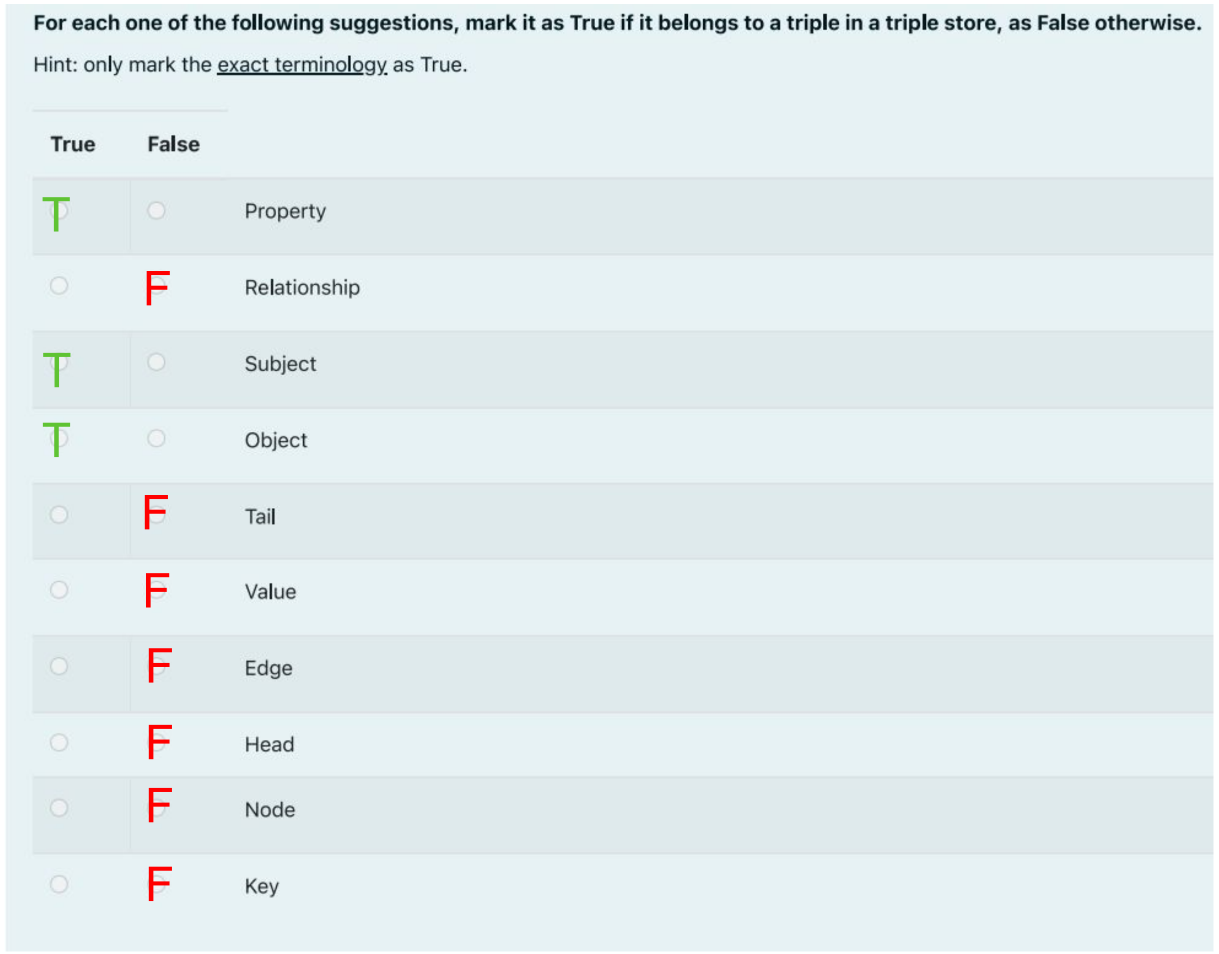
```
@prefix geo: <http://www.example.com/geography#> .
@prefix countries: <http://www.example.com/countries#> .
@prefix eth: <http://www.ethz.ch/#> .

eth:self geo:isLocated countries:Switzerland,
                       countries:Europe ;
         geo:population 25000 .
```

HS22; Q67 - Neo4J; HS23; Q67 - Triple stores



True	False	
T	○	The WITH clause allows to chain 2 statements forwarding values from the first statement to the second.
T	○	Cypher supports graph pattern matching with a syntax looking like ASCII art.
○	F (Also Path, Relationship, Constructed types, …)	Cypher contains a type system that contains only atomic values, objects, and arrays.
T	○	Cypher supports update queries that add nodes and edges to the graph database.



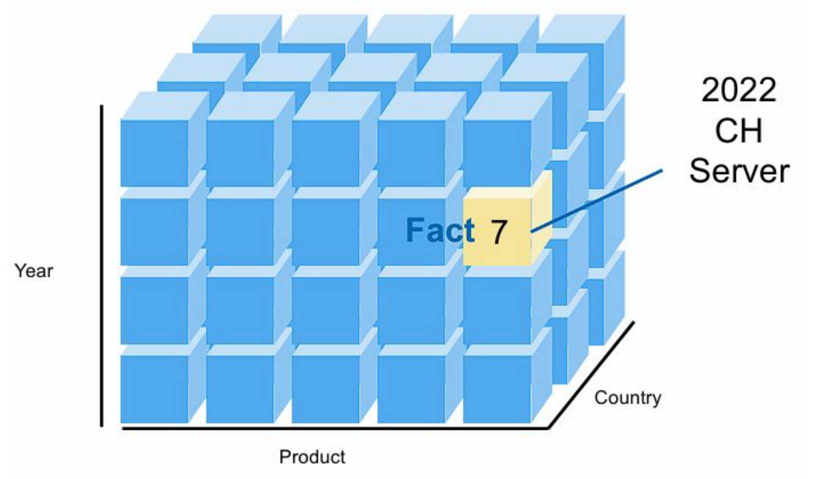
For each one of the following suggestions, mark it as True if it belongs to a triple in a triple store, as False otherwise.

Hint: only mark the exact terminology as True.

True	False	
T	○	Property
○	F	Relationship
T	○	Subject
T	○	Object
○	F	Tail
○	F	Value
○	F	Edge
○	F	Head
○	F	Node
○	F	Key

Data Cubes

Store multi-dimensional data



Location [Axis]	Period [Axis]	Salesperson [Axis]	Currency [Axis]	Value
Germany [Member]	2022 [Member]	Peter [Member]	USD [Member]	1,000
Germany [Member]	2022 [Member]	Mary [Member]	USD [Member]	2,000
Germany [Member]	2021 [Member]	Peter [Member]	USD [Member]	3,000
Germany [Member]	2021 [Member]	Mary [Member]	USD [Member]	4,000
Switzerland [Member]	2022 [Member]	Peter [Member]	USD [Member]	5,000
Switzerland [Member]	2022 [Member]	Mary [Member]	USD [Member]	6,000
Switzerland [Member]	2021 [Member]	Peter [Member]	USD [Member]	7,000
Switzerland [Member]	2021 [Member]	Mary [Member]	USD [Member]	8,000

Period [Axis]	All times [Domain]
Currency [Axis]	1,000 USD [Member]

	Location [Axis]													
	World [Domain]													
	Europe [Member]								Asia [Member]					
	Switzerland [Member]				Germany [Member]				India [Member]		China [Member]			
Salesperson [Axis]	Zurich [Member]	Geneva [Member]	Interlaken [Member]		Berlin [Member]	Frankfurt [Member]			Delhi [Member]		Beijing [Member]			
Peter [Member]	1	2	3	6	1	2	3	9	3	3	4	4	7	16
Mary [Member]	1	2	3	6	1	2	3	9	3	3	4	4	7	16
All salespersons [Domain]	2	4	6	12	2	4	6	18	6	6	8	8	14	32

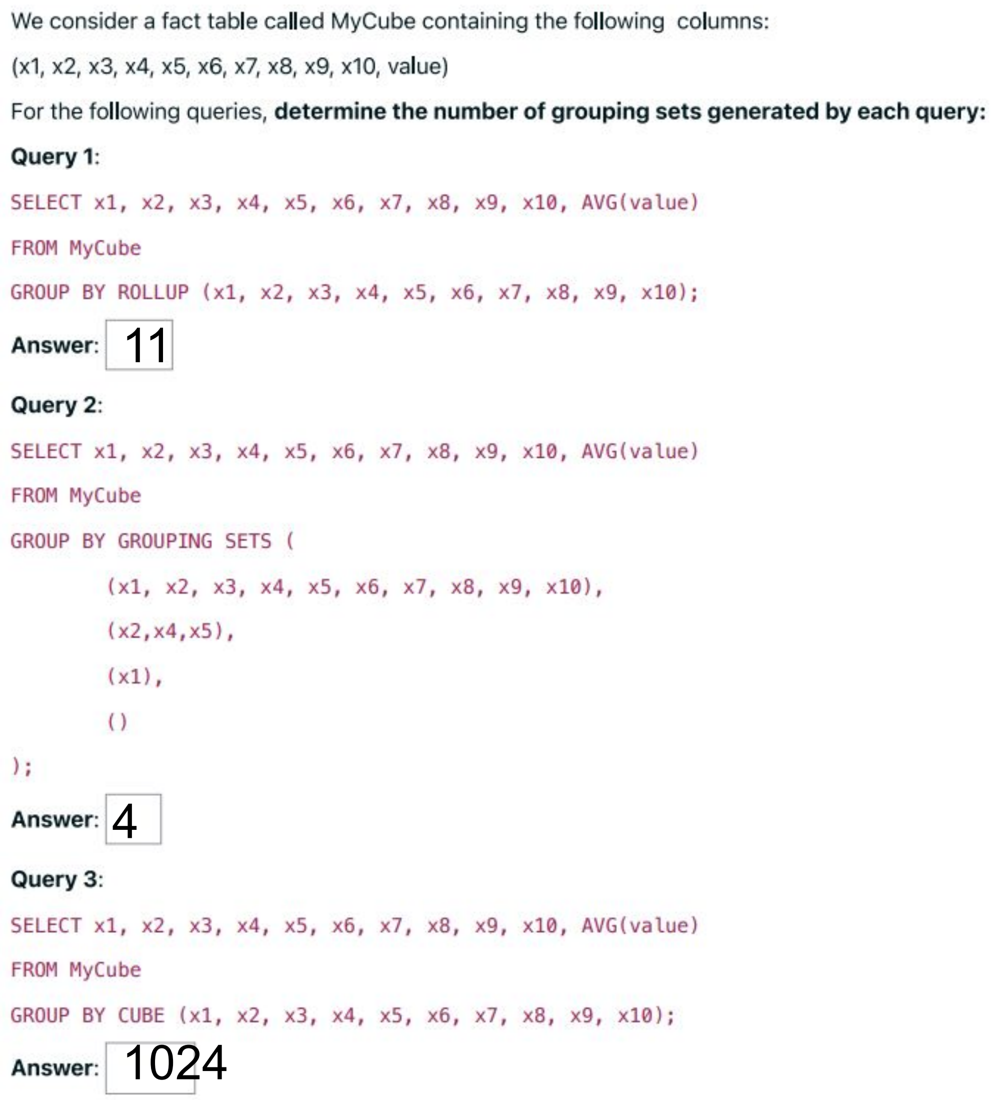
Location [Axis]	Switzerland [Member]
Currency [Axis]	USD [Member]

Slicers

Dicers

Salesperson [Axis]	2021 [Member]	2022 [Member]
Peter [Member]	7,000	5,000
Mary [Member]	8,000	6,000

HS24; Q71 - Grouping sets in Data cubes



We consider a fact table called MyCube containing the following columns:

(x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, value)

For the following queries, **determine the number of grouping sets generated by each query:**

Query 1:

```
SELECT x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, AVG(value)
FROM MyCube
GROUP BY ROLLUP (x1, x2, x3, x4, x5, x6, x7, x8, x9, x10);
```

Answer: 11

Query 2:

```
SELECT x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, AVG(value)
FROM MyCube
GROUP BY GROUPING SETS (
        (x1, x2, x3, x4, x5, x6, x7, x8, x9, x10),
        (x2,x4,x5),
        (x1),
        ()
);
```

Answer: 4

Query 3:

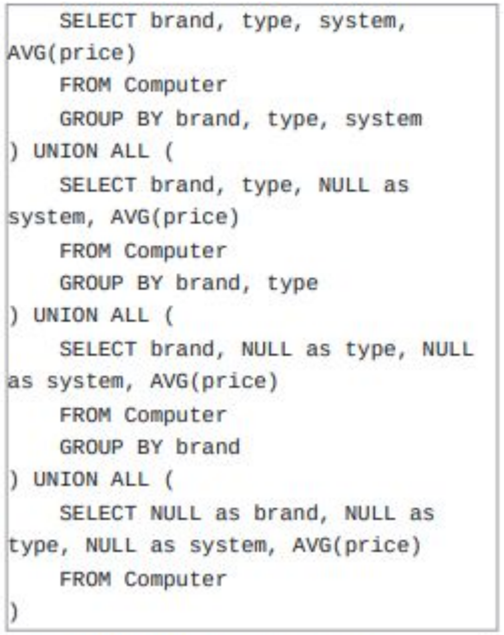
```
SELECT x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, AVG(value)
FROM MyCube
GROUP BY CUBE (x1, x2, x3, x4, x5, x6, x7, x8, x9, x10);
```

Answer: 1024

Shorthands for bunch of GROUP BY + UNION statements!

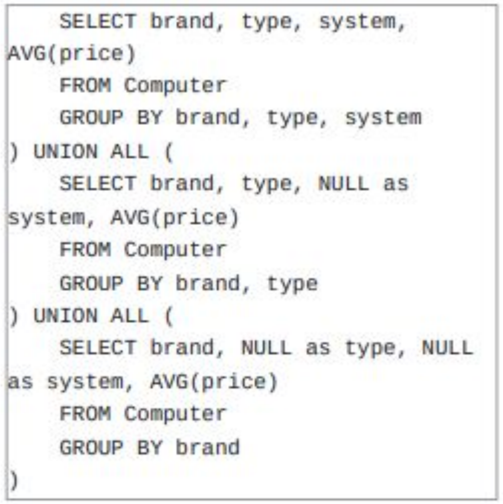
HS22; Q71 - SQL Matching

1C



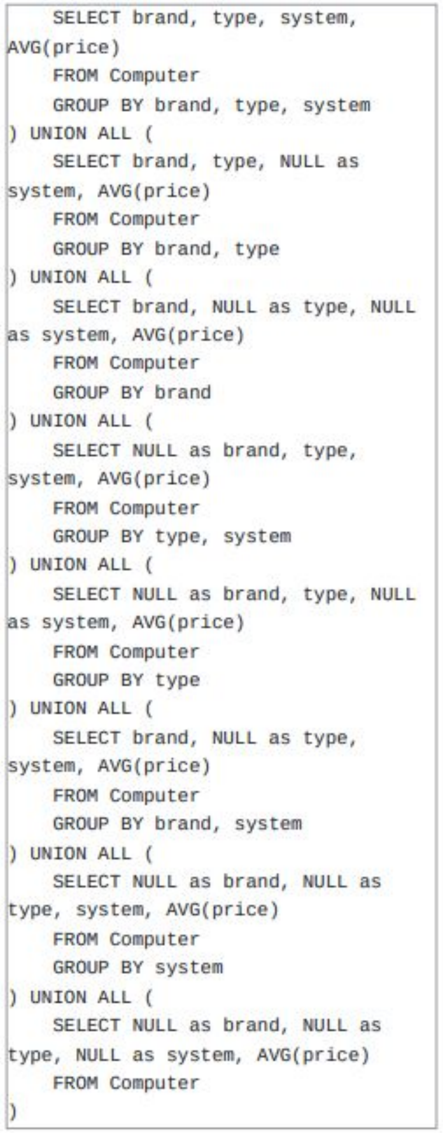
```
    SELECT brand, type, system,
AVG(price)
    FROM Computer
    GROUP BY brand, type, system
) UNION ALL (
    SELECT brand, type, NULL as
system, AVG(price)
    FROM Computer
    GROUP BY brand, type
) UNION ALL (
    SELECT brand, NULL as type, NULL
as system, AVG(price)
    FROM Computer
    GROUP BY brand
) UNION ALL (
    SELECT NULL as brand, NULL as
type, NULL as system, AVG(price)
    FROM Computer
)
```

2A



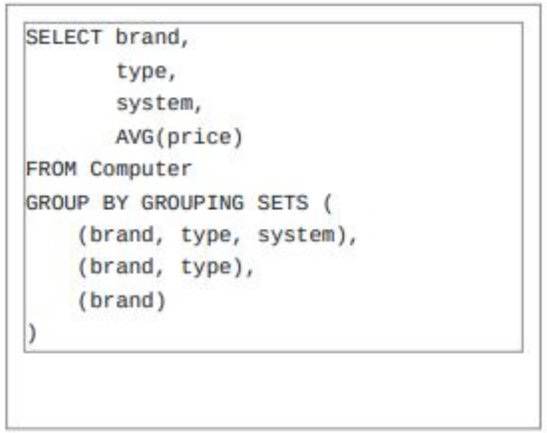
```
    SELECT brand, type, system,
AVG(price)
    FROM Computer
    GROUP BY brand, type, system
) UNION ALL (
    SELECT brand, type, NULL as
system, AVG(price)
    FROM Computer
    GROUP BY brand, type
) UNION ALL (
    SELECT brand, NULL as type, NULL
as system, AVG(price)
    FROM Computer
    GROUP BY brand
)
```

3B



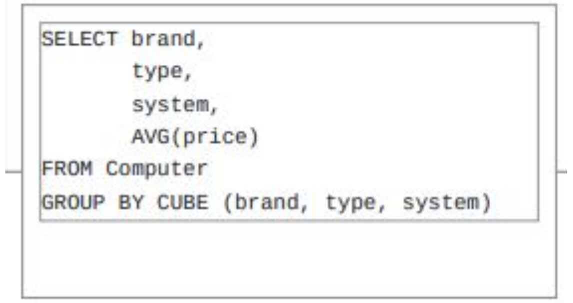
```
    SELECT brand, type, system,
AVG(price)
    FROM Computer
    GROUP BY brand, type, system
) UNION ALL (
    SELECT brand, type, NULL as
system, AVG(price)
    FROM Computer
    GROUP BY brand, type
) UNION ALL (
    SELECT brand, NULL as type, NULL
as system, AVG(price)
    FROM Computer
    GROUP BY brand
) UNION ALL (
    SELECT NULL as brand, type,
system, AVG(price)
    FROM Computer
    GROUP BY type, system
) UNION ALL (
    SELECT NULL as brand, type, NULL
as system, AVG(price)
    FROM Computer
    GROUP BY type
) UNION ALL (
    SELECT brand, NULL as type,
system, AVG(price)
    FROM Computer
    GROUP BY brand, system
) UNION ALL (
    SELECT NULL as brand, NULL as
type, system, AVG(price)
    FROM Computer
    GROUP BY system
) UNION ALL (
    SELECT NULL as brand, NULL as
type, NULL as system, AVG(price)
    FROM Computer
)
```

A



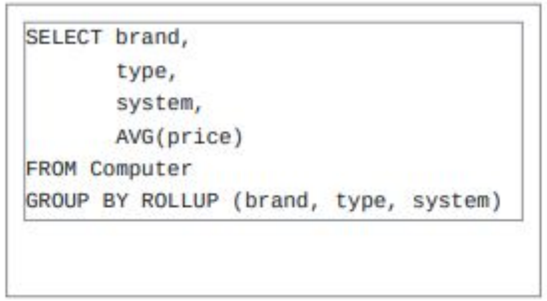
```
SELECT brand,
       type,
       system,
       AVG(price)
FROM Computer
GROUP BY GROUPING SETS (
    (brand, type, system),
    (brand, type),
    (brand)
)
```

B



```
SELECT brand,
       type,
       system,
       AVG(price)
FROM Computer
GROUP BY CUBE (brand, type, system)
```

C



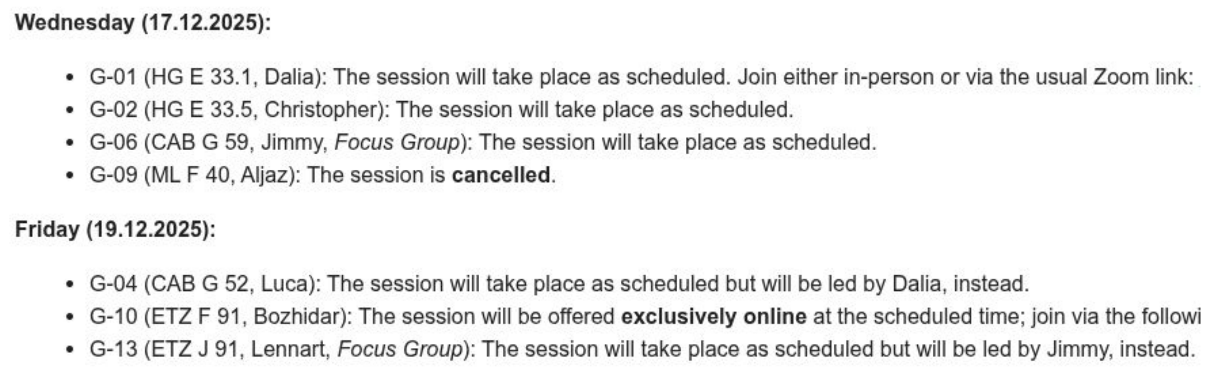
```
SELECT brand,
       type,
       system,
       AVG(price)
FROM Computer
GROUP BY ROLLUP (brand, type, system)
```

ETH zürich

Thanks for your attention!

Aljaž Medič
amedic@ethz.ch

Next week I will not hold an exercise session, you can seek to join one of the other ones below:



Wednesday (17.12.2025):

- G-01 (HG E 33.1, Dalia): The session will take place as scheduled. Join either in-person or via the usual Zoom link:
- G-02 (HG E 33.5, Christopher): The session will take place as scheduled.
- G-06 (CAB G 59, Jimmy, *Focus Group*): The session will take place as scheduled.
- G-09 (ML F 40, Aljaz): The session is **cancelled**.

Friday (19.12.2025):

- G-04 (CAB G 52, Luca): The session will take place as scheduled but will be led by Dalia, instead.
- G-10 (ETZ F 91, Bozhidar): The session will be offered **exclusively online** at the scheduled time; join via the followi
- G-13 (ETZ J 91, Lennart, *Focus Group*): The session will take place as scheduled but will be led by Jimmy, instead.

Slides

Wish you a Happy December 🎅🎄,
and best of luck with all your exams!!