IN3020/IN4020 – Database Systems Spring 2021, Week 16.2

NoSQL DBMS and Beyond

(Elmasri & Navathe, Ch. 24 + slides)

Dr. M. Naci Akkøk

CEO, In-Virtualis, Assoc. Prof. UiO/Ifi, Assoc. Prof. OsloMet/CEET

UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

Key take-away – Why/how of NoSQL DBMS

When new types of applications become main-line,
like data science or AI/ML applications
or social media applications,
they impose new requirements upon the underlying systems, like the infrastructure and the DBMS.

New DBMS´ are designed to answer these new needs.

Why/how of NoSQL DBMS´ (E&N ch. 24.1)

- Scalability
- Availability, replication (several models)
- Eventual consistency
- Sharding, partitioning (range portioning)
- High performance data access (increased and improved usage of keys)

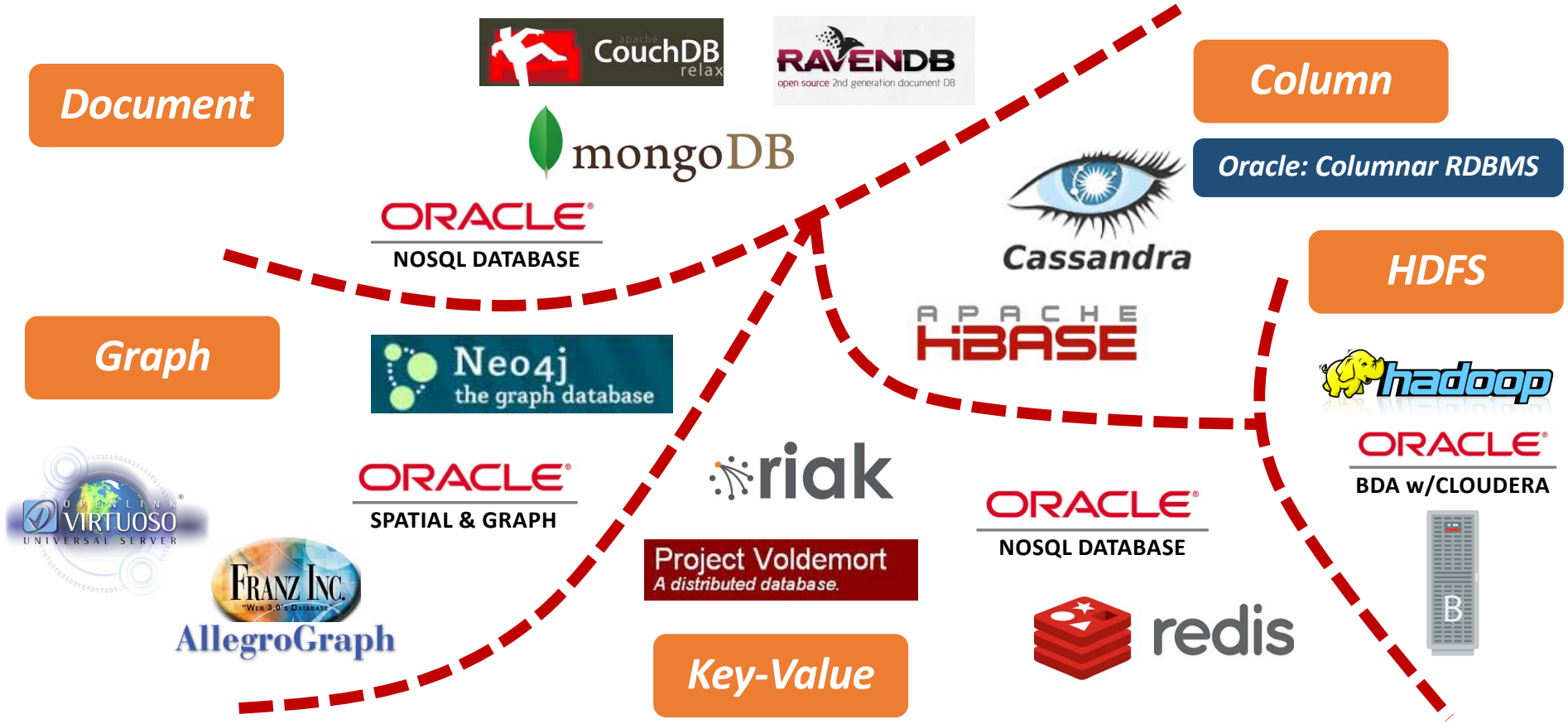
Overview of Existing NoSQL Database Management Systems

UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

Hybrid, Object DB, XML, Multi-model

NoSQL Databases and RDBMS

New (big/fast) data management, often covering several new and old/improved DBMS technologies.



CAP Theorem (E&N Ch. 24.2)

- Consistency (among replicas, NOT the same as ACID consistency)
- Availability
- Partition tolerance

Document-based NoSQL DBMS

- Collections of (similar) documents, but with varying attributes, data elements
- Self-describing
- Documents:
  - Complex objects,
  - Can be described as XML objects or JSON (or BSON, Binary JSON, in the case of Mongo-DB)

Document-based NoSQL DBMS - CRUD ops

- CRUD (Create, Read, Update, Delete) operations in a DDBMS:
  - **Create** is for creating a collection, not for documents: db. createCollection(…)
- **For documents:**
  - **Insert** creates and inserts a document (or an array of documents) into a collection
  - **Remove** deletes
  - **Find** reads (gets, fetches)

Key-value pair (KVP) NoSQL DBMS

- Key – value pairs, or key – object pairs

- High performance, almost real-time
- Good for capturing f. ex. time-series data:
  (timestamp, value)

- Used by AWS
- Also, by many others, especially in IoT, Edge devices etc.

Column-based NoSQL DBMS (CDBMS)

- For storing large amounts of data
- Google Bigtable, uses the Google File System (GFS)
- Apache Hbase is similar, and is used by the Hadoop Distributed File System (HDFS)

- The “key” here is multidimensional (can for example contain table name, row key, column info & time-stamp)

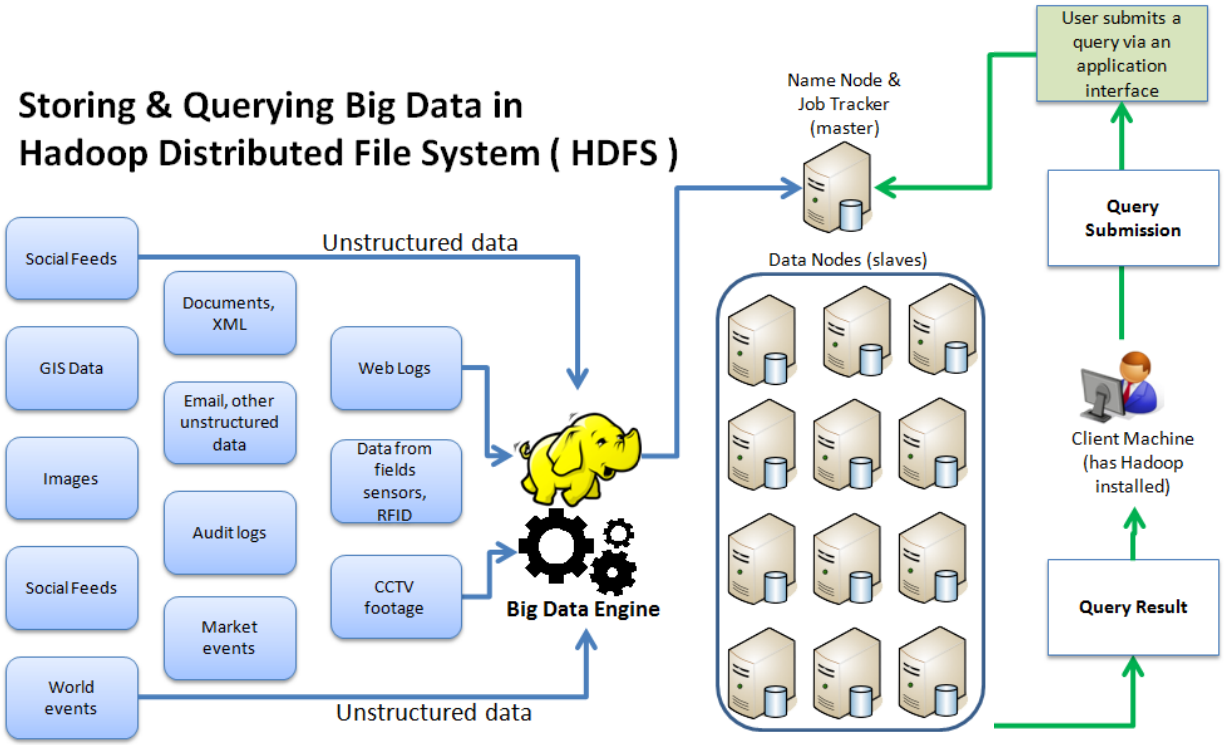
NoSQL CDBMS Data Structures (Hbase)

- Named **tables** & self-describing **rows** with row-keys (orderable lexicographically)
- Named **column families**, associated with tables, created on creation of table and cannot be changed. For grouping together related columns (attributes).
- Each column family can be associated with **column qualifiers** (making the model self-describing)
- A **column** is a combination of *ColumnFamily:ColumnQualifier*
- **Versions** of data items **time-stamps**

NoSQL CDBMS CRUD operations (Hbase)

- **Create** operation for creating a table and associating it with one or more column families
- **Put** operation for inserting data or new versions of existing data
- **Get** operation for retrieving data from a single row in a table
- **Scan** for retrieving all rows
- **Delete** command is issued through the **HBase** client & data is marked with a a tombstone marker, making **deleted** cells invisible. User Scans and Gets automatically filter **deleted** cells until they get **removed**. **HBase** periodically removes **deleted** cells during compaction

HADOOP: Scalable ”reservoir” for big & any-structured Data



Big Data Storage Technologies: The Semantic Solution

(Graph and Property Graph)

Semantic technologies have been around for quite some time.
Network databases are actually older than relational database management systems (RDBMS).
We find them more and more useful as an answer to the needs of Big Data.

UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

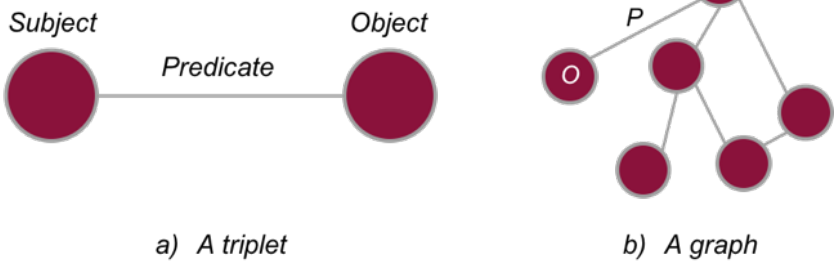
Two types of graph databases

- RDF Graph
  - Older
  - Standard (W3C standard - https://www.w3.org/RDF/)
  - Own standard query language: SPARQL (https://www.w3.org/2001/sw/wiki/SPARQL)
- Property graph
  - Newer
  - Not yet standardized (different query languages)

Property graph databases

- Not yet standardized:
  - Several vendors, two dominant (Oracle & Neo4J)
- Implies that there are at least two query languages
  - PGQL (Oracle, https://pgql-lang.org/)
  - Cypher (Neo4J, https://neo4j.com/developer/cypher/)
- And an open source one:
  - Apache Tinkerpop (https://tinkerpop.apache.org/)
- The main idea is ”pattern matching”

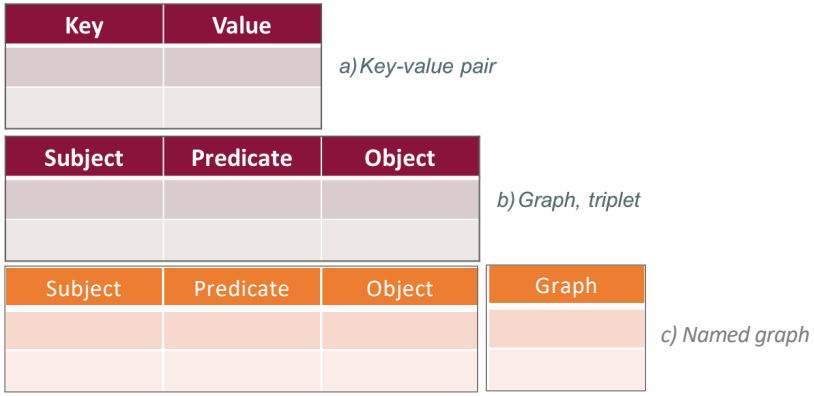
RDF Graph Databases



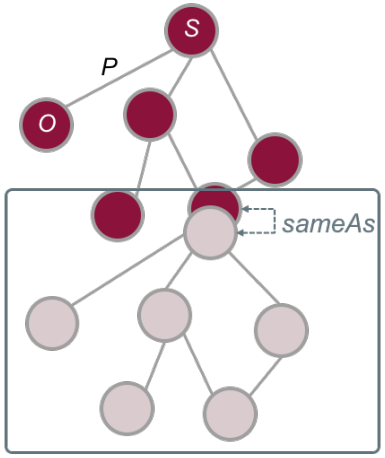
a) A triplet b) A graph

a) Graph, triplet

Subject	Predicate	Object

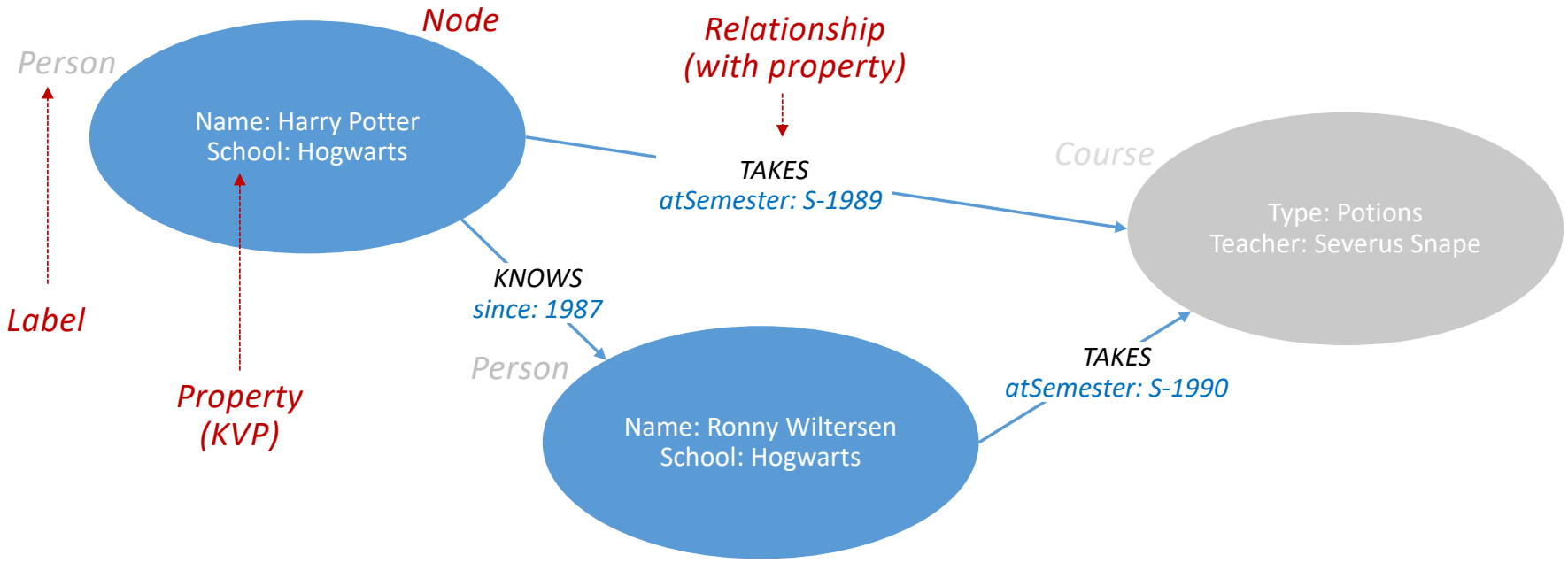


b) Linked graphs

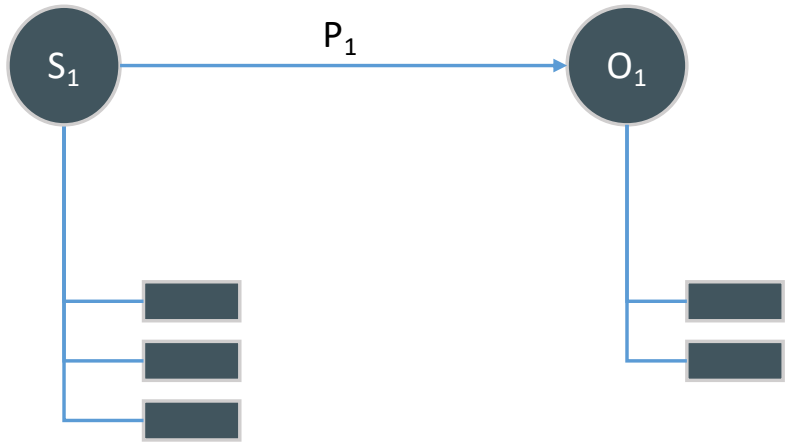


Property Graph

Apache Tinkerpop (base for Oracle Property Graph & for Neo4J)



Property Graph Databases (rom RDBS/RDF view)



Subject	Predicate	Object
S_1 w/prop. (ADT)	P_1	O_1 w/prop. (ADT)

BUT NOTE: The "predicate" can also have properties!

Cypher (Neo4j) queries – Directional!

- Create is somewhat regular:
  CREATE (d1: DEPARTMENT, {Dno: ´5´, Dname: ´Research´})
  CREATE (d2: DEPARTMENT, {Dno: ´4´, Dname: ´Admin´})
  CREATE (loc1: LOCATION, {Lname: ´Houston´})

- Example query
  MATCH (d: DEPARTMENT {Dno: ´5´}) – [: **LocatedIn**] → (loc)

Reference: https://neo4j.com/developer/cypher/intro-cypher/

PGQL (Oracle) queries, SQL-like, directional

```
CREATE PROPERTY GRAPH financial_transactions
VERTEX TABLES (
    Persons LABEL Person PROPERTIES ( name ),
    Companies LABEL Company PROPERTIES ( name ),
    Accounts LABEL Account PROPERTIES ( number ) )
EDGE TABLES (
    Transactions
        SOURCE KEY ( from_account ) REFERENCES Accounts
        DESTINATION KEY ( to_account ) REFERENCES Accounts
        LABEL transaction PROPERTIES ( amount ),
    Accounts AS PersonOwner
        SOURCE KEY ( id ) REFERENCES Accounts
        DESTINATION Persons
        LABEL owner NO PROPERTIES,
---
SELECT owner.name AS account_holder, SUM(t.amount) AS total_transacted_with_Nikita
FROM        MATCH (p:Person) <-[:owner]- (account1:Account),
            MATCH (account1) -[t:transaction]- (account2) /* match both incoming and outgoing transactions */ ,
            MATCH (account2:Account) -[:owner]-> (owner:Person|Company)
WHERE p.name = 'Nikita'
GROUP BY owner
```

Reference https://pgql-lang.org/

Other Technologies & Needs, Other DBMS or Related Solutions

UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

Date Quality Management (DQM)

- **Data Quality Management** is for ensuring the quality of the data (the information) we are pulling into the database or data we already have in the database.

- Any application that relies on data
  - Will need data of high quality (high enough with respect to its purpose)
  - And will be negatively affected by low quality data (may fail or give wrong information)

Data Quality?

Data is of high quality, if the data is **fit for the intended purpose** of use and if the data **correctly represent the real-world construct** that the data describes.

Ref. Profisee

Characteristic	How It's Measured
Accuracy	Is the information correct in every detail?
Completeness	How comprehensive is the information?
Reliability	Does the information contradict other trusted resources?
Relevance	Do you really need this information?
Timeliness	How up- to-date is information? Can it be used for real-time reporting?

Ref. Syncsort

Content & format **CORRECTNESS**:

Key fields and other relevant fields are non-empty and are in the right format, and of the right type.

Content of field makes sense with respect to its format and expected use.

Ref. practice @ Oracle

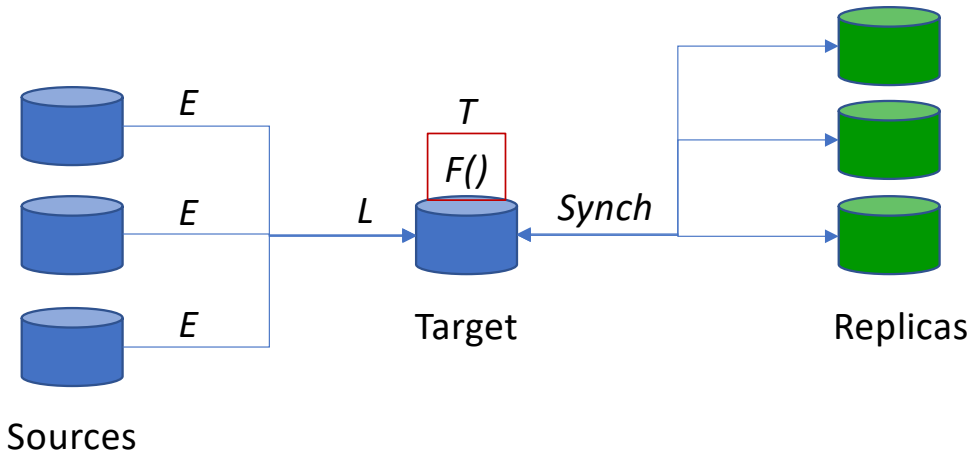
FUNCTIONS: Profiling, auditing, visualization, parsing & standardization, matching & merging , case-based clean-up, address/format verification.

Ref. EDQ sheet @ Oracle

DQM is related to ETL/ELT and DRM tools

- **Data Quality Management** is practically always related to **Extract-Transform-Load** or **Extract-Load-Transform**(*) tools and **Data Replication Management** tools.

- Typical process



(*) ELT usually faster

AI, ML and Data Science
New areas of applications that demand new types of DBMS

Yet another category of data intensive applications!

UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

AI/ML in DQM

- One main area is introducing AI/ML to automate and improve the DQM process
  - Learn and automate profiling
  - Do the necessary corrections
  - Learn and automate transformations
  - Learn and improve ingestion performance
  - …

AI/ML in DBMS

- One main area is introducing some AI/ML into the DBMS
  - Self-install
  - Self-tune (through learned and optimized indexing etc.)
  - Self-repair
  - …

A DBMS for AI/ML

- How would you best represent a deep learning (neural) network in a DBMS?
  - Suggestion: Graph Neural Networks (Explainable AI)

- How would you design your DBMS to improve performance in the case of AI/ML and Data Science applications that require large amounts of data fast?
  - Suggestion: Embed the algorithms into the DB and let them run in the database (so that you don´t have to pull the data out first)

My personal R&D focus

- Research & development in molecular modeling & simulation (MOMS) with underlying R&D in modeling molecular dynamics
- Newer storage & retrieval structures that perform better and help represent molecular structures and their interactions also in 3D and in VR (a standardization effort)
- Newer AI/ML approaches in MOMS to help identify desired interactions (drug discovery design, vaccine development, Nano-materials design/development)
- HPC for complex models/simulations and real-time rendering of visualizations

In-Virtualis

Contact: naci@in-virtualis.com
nacia@ifi.uio.no
mehmetna@oslomet.no

+47 47026879

UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet