# ML and Numerical Software Development

## Data&Databases

Organon Analytics

November 28, 2019

## Data: Structured, Semi-structured, Unstructured

Structured data:

- Composes of structured data fields
- The type of each field is one of the following basic types: Bytes, Integers, Floating-Point types, Strings, Dates
- The type of each field is known in advance (Data Schema is known)
- Data You can keep in Excel or RDBMS tables where you can query without any pre-processing

Examples:

- **Customer** data fields: Date-of-birth(Date), Profession(String), Address(String), Number-of-Children(Integer)
- **Product** data fields: Product Launch Date(Date), Product Id(Integer), Current Price(Double)
- **Transaction** data fields: Transaction type(String), Transaction Id(Integer), Transaction date(Date), Transaction amount(Decimal)

## Semi-structured data

- Data that is stored in HTML, XML, JSON formats
- No fixed schema. Schema might change over time
- You can keep hierarchical data, object data in semi-structured format

Examples:

- Web page data
- Log data

You can keep semi-structured data in RDBMS databases but direct querying may not be possible

## Unstructured data

- Text: Strings of arbitrary length.
- Picture
- Video (Surveillance data)
- Audio (Call center recordings)

Examples:

- E-mails, documents
- Pictures on your iPhone
- Think of Youtube database
- Think of Spotify database, Call center recordings

# Structured Query Language (SQL)

- A declarative language for accessing and manipulating structured data in RDBMSs
- Data is logically represented as tables: A matrix structure with rows and columns
- Each column corresponds to a fixed data field with a fixed data type
- Created by Edgar F. Codd at IBM. DB2 is the first commercial SQL database
- The primary purpose: Online Transactions Processing. Designed to do it well
- People started using them for analytical processing (reporting and statistical modelling) in late 1990's and early 2000's: The birth of the datawarehouse
- Not optimized for analytical processing but still used
- Analytical extensions appeared in 2000's
- Mission critical applications use SQL databases for real-time business operations on specialized hardware

## Major SQL databases

- Oracle: Market leader. Exadata for mission critical, high-performance apps
- IBM DB2: Run on IBM mainframes
- Microsoft SQL Server: Most small, medium businesses use it
- Teradata: Mostly datawarehousing applications
- PostgreSQL: Open source adoption increasing. Small, medium business
- Hive: Runs on Hadoop file-systems. Performance has not matured yet

# The structure of an SQL statement

## SELECT

- Raw or Transformed Field Values
- Row functions: Algebraic operations, logical operations, conditional operations
- Column functions: Agregated Statistics

## FROM

- Specifies the dataset(s)
- It might be a result of join operations on more than 1 data-sets

## WHERE

- Specifies a subset of dataset defined by conditions
- Composes of logical, set, and comparison operations: AND, OR, BETWEEN, <=, LIKE, etc.

## GROUP BY

- Divides the dataset into groups
- Aggregated statistics might be obtained over each group

## Database objects, concepts

- Schemas: A logical grouping of database objects. Think of a folder. (e.g. data for a specific application, data for a department, data per use.)
- Tables, Views: Tables store the data and are physical. Views are logical views of tables: A subset of rows, a subset of columns, the result of query over one/multiple tables
- Indexes: Govern the access to a table. Consists of one or more data fields. There might be more than one index. Useful for fast access to the data
- Keys: Corresponds to the identifiers for different entities (Customer id, Product id, transaction id). They represent the relations between different entities
- Functions, stored procedures: User defined functions that are used frequently. Provides modularity, encapsulation, re-use

## Important SQL Commands on DB objects

| Data Definition | Data Manipulation | Data Query | Data Control |
|---|---|---|---|
| CREATE | INSERT | SELECT | GRANT |
| ALTER | UPDATE | | REVOKE |
| DROP | DELETE | | |
| RENAME | MERGE | | |
| TRUNCATE | CALL | | |
| COMMENT | EXPLAIN PLAN | | |

## Transaction Processing

- Transactions: Drawing cash from ATM, Any credit card transaction, Entering a loan contract into the system, Viewing your account balance at your bank's online branch, login into an e-commerce site
- Number of requests per second: thousands, possibly hundreds of thousands
- Required to be highly available: All the time
- No margin for the error (think of your account balance)
- Each query consumes a small amount of CPU but there are many of them
- Each query processes a few rows of database table
- Specialized hardware for mission critical ops (mainframes, high-end storage networks, high-end chips)
- Latency requirement: sub-seconds

## Analytical Processing

- Queries: Number of customer visiting stores per district/city over the last day/week/month, Sum of credit card transaction amounts on Mother's Day, Product sales in each e-commerce category last month, Running a ML model over aggregated customer datamart, computing PageRank at Google
- Main use cases: Reporting, Machine Learning
- Each query processes many (thousands, millions, possibly billions) rows
- Some queries may need joining many tables (possibly tens of them) together
- Number of users small: Tens, hundreds, possibly thousands at a large corporation
- Latency requirement: seconds, minutes, possibly hours for large computations
- Runs on commodity hardware. Compute clusters composed of many commodity machines

## Stream Processing

- Queries: The number of connections from China in the last 15min, the amount of bytes downloaded per IP in the last half hour
- Main use cases: Reporting, Machine Learning
- Applications: Cyber Security, Fraud Systems, Real-Time Marketing
- Processing while data is still in transit: Data is not on disk
- Number of users: Small, mostly applications
- Specialized software: Message brokers (Kafka), stream processors (Spark, Storm, Flink)

# Transaction vs. Analytical Processing

| Feature | TP | AP |
|---|---|---|
| Users | Front-end&IT personnel | Data Scientists&ML Engineers |
| Function | Operations | Decision Support |
| Data | Current | Historical |
| | Atomic | Integrated |
| Access | Read/Write | Table scans |
| | Indexed access | |
| Unit-Of-Work | Transactions | Complex query |
| Records accessed | Tens | Millions |
| # of users | thousands | hundreds |
| DB size | $\sim$GB | $\sim$TB, PB |
| Latency | Low | High |

## Why we need RDBMs, and use SQL?

- Market dynamics: Our clients' data is till mostly in RDBMS databases
- Data Investigation: Data Scientists use SQL for initial understanding of data
- Data Preparation: Platform creates SQL statements that summarizes and joins multiple databases (sometimes hundreds of them) to create data for Machine Learning computations
- Scoring with SQL: We export ML models as SQL statements for scoring. In-database SQL-scoring is much faster than in-memory scoring (for batch scoring)
- Troubleshooting: Investigating a problem through log files

## SQL Epilogue

- Learn SQL: Both basics, and the analytical functions
- Leading implementations conform with the ANSI-SQL standard
- BUT, do not forget that there are dialect differences between different implementations
- A tutorial for Oracle: `https://www.oracletutorial.com/`
- SQL is still the primary language for transactions, analytical and stream processing. It is here to stay
- You can impress your recruiters with good SQL skills
- Easy to learn&apply, hard to optimize

## When SQL is not enough?

- Unstructured data: Text, Picture, Video, audio
- BIG DATA: Lots of it
- Think of the data Google stores: BIG and UNSTRUCTURED
- Lots of columns
- SQL is successfull when data has the following properties:
  - *i* Structured AND
  - *ii* Number of rows at most billions AND
  - *iii* Number of rows in the hundreds AND
  - *iii* Volume of the data is in a fews terabytes AND

  It FAILS otherwise. It could not scale. Becomes very expensive

## A new era: Unstructured and Big Data

Sensors and storage technology got cheaper. Disk access rates got faster.

- Image data (Think of the surveillance data stored)
- Audio data (Think of the call centers)
- Text data (Think of all the scanned/digitized documents, text as a result of speech-to-text)
- Log data: Web logs, app-logs, firewall logs

Facts:

- Storage/access must be affordable
- High-end HW chip/RAM/disk/switch very expensive. You could not scale Exadata or IBM mainframe (only large enterprises could afford it)
- But data is abundant
- Distribute the data over a set of commodity clusters
- Do local computation with minimal network transfer

A new paradigm required

## No-SQL, Hadoop, Map Reduce, Spark

- Web search (Google)
- Social media data storage, access, analytics (Twitter, Facebook, Instagram on Hadoop, No-SQL)
- Computer Vision apps (Facial recognition, driverless car)
- Speech understanding apps (Facial recognition, driverless car)
- Natural Language Processing

A new eco-system of storage/processing tools and paradigms:

- Google File System $\rightarrow$ Hadoop File System
- Map-Reduce for batch computations (Slow at first)
- Spark has arrived: Distributed computations over distributed file-systems. Fault-tolerant. Horizontally scalable
- Processes PBs of data
- Big volumes, high variety, high velocity

RDBMS vs. Hadoop(HDFS+MapReduce+Spark)

| | RDBMS | Hadoop |
|---|---|---|
| **Data Size** | Gigabytes | Petabytes |
| **Access** | Interactive&Batch | Batch |
| **Updates** | R/W many times | Write once, read many |
| **Transactions** | ACID | None |
| **Structured** | Structured | Everything |
| **Integrity** | High | Low |
| **Scalability** | Non-linear | Linear |
| **Cost** | High | Low |

**ACID**: Atomicity, Consistency, Isolation, Durability

# Our example

Databases:

- Databases: Oracle, SQL Server, Teradata, PostgreSQL, Hive, Big Query
- Distributed Filesystems: HDFS, S3
- Stream Processing: Kafka, Spark Structured Streaming
- In-memory databases: Redis

Some examples:

- A single project uses hundred of database tables
- Some transactional tables consist billions of records
- Cyber security example: 50K log records per sec (each record $\sim$ 2K)
- Millions of new features created in memory
- Number of scores created in 2019 $\sim$trillions
- MR, X-Ray data
- Call center speech data