Automated Feature Extraction with Mascheine Learning and Image Processing

PD Stefan Bosse

University of Siegen - Dept. Maschinenbau
University of Bremen - Dept. Mathematics and Computer Science

Data Storage and Access

Representation of Data

Storage of Data

Access of data

Data Storage

In general, data and their values can be stored in/on:

- **File Systems**, such as FAT32/VFAT, Ext 3, ...
- **Databases**, such as SQL, NoSQL, ...
- **Cloud Storage**, such as Seafile
- **Files**, such as HDF(5)

But: The question is not where to store the data, the question is how to organize and access the data!

File System

- A file system is composed of:
  - Files ⇒ data container (linear memory model)
  - Directories ⇒ reference tables assigning files or other directory references to names
- A file system ognaizes data by a directory (folder) tree:
  - A directory is a node in an ordered graph
  - There is a root directory
  - Children of a directory can be leafes (files) or deeper directories
  - Each file and directory is associated with a name
  - File system tree iterations are referenced by paths

Besides the organization structure, a file system can provide data structures and block level organization of data used for the storage on hardware devices

[https://www3.nd.edu/pbui/teaching/cse.30341.fa18/project06.html]

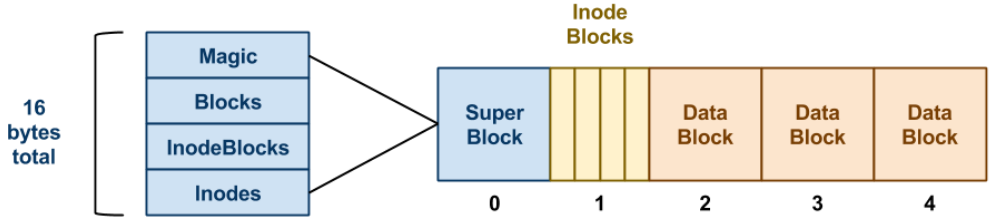


Fig. 1. Simple file system layout (linear file model)

Segments

- Magic: The first field is always the MAGIC_NUMBER or 0xf0f03410. The format routine places this number into the very first bytes of the super-block as a sort of file system "signature". When the file system is mounted, the OS looks for this magic number. If it is correct, then the disk is assumed to contain a valid file system. If some other number is present, then the mount fails, perhaps because the disk is not formatted or contains some other kind of data.
- Blocks: The second field is the total number of blocks, which should be the same as the number of blocks on the disk.
- InodeBlocks: The third field is the number of blocks set aside for storing inodes. The format routine is responsible for choosing this value, which should always be 10% of the Blocks, rounding up.
- Inodes: The fourth field is the total number of inodes in those inode blocks.

[https://www3.nd.edu/pbui/teaching/cse.30341.fa18/project06.html]

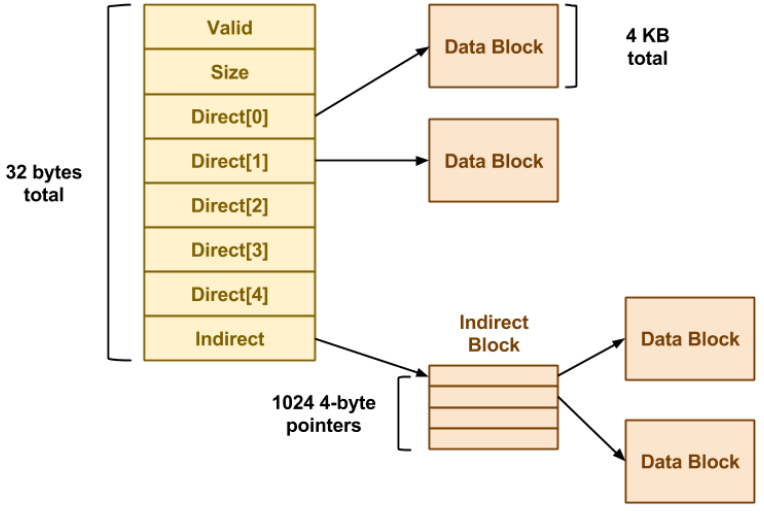


Fig. 2. I-nodes and block references

Data Types

- Raw data featuring
  - Dimension
  - Size
  - Aggregates
- Vector and time signal data
- Matrix and tensor data
- Image data (RGBA)
- Data record tables

Databases

SQL

- SQL databases organize data in tables.
  - A table is organized in rows and columns
  - A table is part of a database
  - Multiple databases can co-exist and handled by one database server
- Data types:
  - Number
  - Text
  - Binary data (blob)

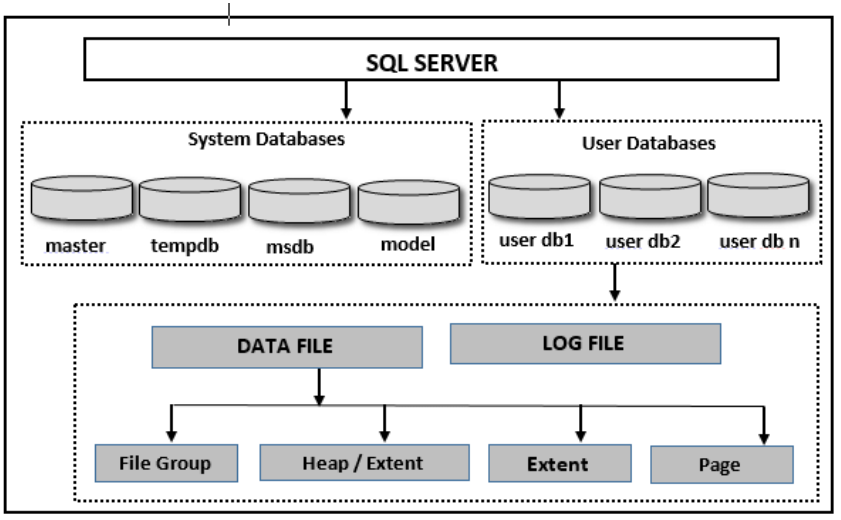


Fig. 3. Basic SQL Server architecture

Page

The page is the fundamental unit of data storage and internally, SQL server organizes and stores data in smaller units known as pages.

[Murugesan et al., International Journal of Applied Engineering Research, 2015]

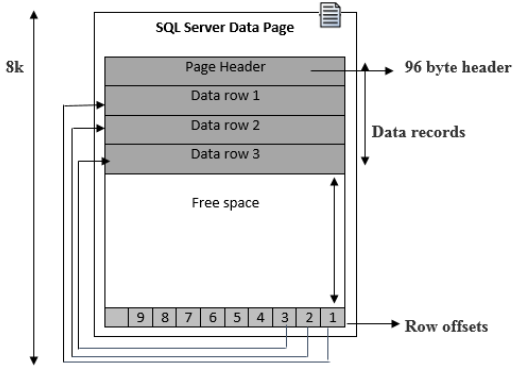


Fig. 4. Data Pages storing tables - Schematic Diagram

Tables

[]

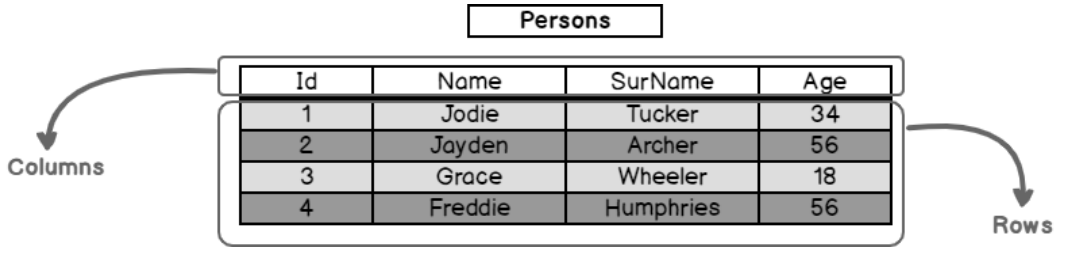


Fig. 5. SQL table structure with rows and columns

Operations

Create

Create a new empty table with a specific schema (type signature of the columns. A table is referenced by a (data base) unique name.

Insert

Insert rows into an existing table.

Update

Update fields or entire existing rows

Select

Select fields or entire rows based on patterns

Tables cannot be nested. The data base table space is flat!
But specific tables can be used to reference other tables (like I-nodes, directories in file systems, or sections in HDF structures)

Tables cannot be nested. The data base table space is flat! But specific tables can be used to reference other tables (like I-nodes, directories in file systems, or sections in HDF structures)

Meta data, arrays, or other auxiliary structures must be encoded and decoded by the user, e.g., by using JSON formats!

Tables cannot be nested. The data base table space is flat! But specific tables can be used to reference other tables (like I-nodes, directories in file systems, or sections in HDF structures)

Meta data, arrays, or other auxiliary structures must be encoded and decoded by the user, e.g., by using JSON formats!

Be aware of memory data layer hierarchy affecting performance (read/write): Data and DB Cache, Main Memory, File system, Storage Device(s).

[https://www.guru99.com/sql-server-architecture.html]

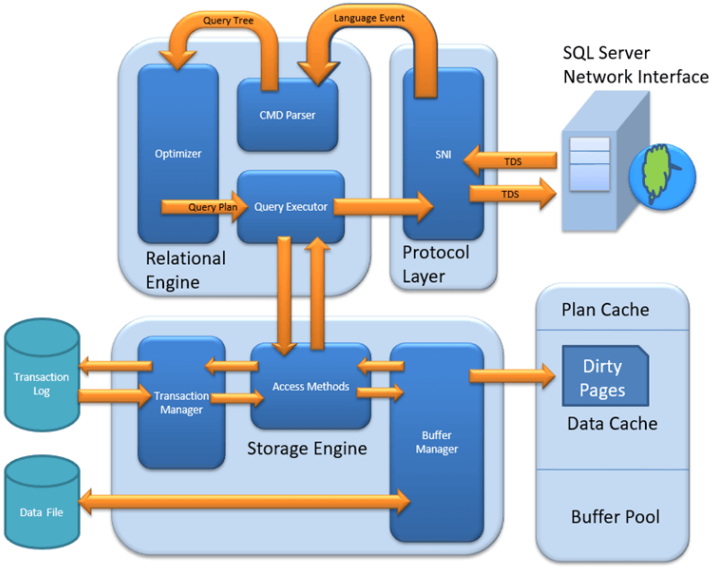


Fig. 6. Detailed and advanced SQL server architecture

MySQL

- One of the top open-source relational database management systems (RDBMS)
- Data security layers to protect sensitive data.
- Scalability for when there are large amounts of data.
- Open source RDBMS with two separate licensing models.
- Multi-master ACID transactions through MySQL Cluster ((Atomicity, Consistency, Isolation, and Durability).
- Supports both structured data (SQL) and semi-structured data (JSON).

Apache Cassandra

- Is an open-source and highly scalable NoSQL database management system
- Handles massive volumes of data.
- One of the most scalable databases with automatic sharding.
- Offers linear horizontal scaling.
- Decentralized database with multi-datacenter replication and automatic replication.
- Fault tolerant by automatically replicating data to multiple nodes.

PostgreSQL

- It extends the SQL language and combines it with various features to scale and safely store highly complicated data workloads.
- Highly secure with a robust access-control system.
- Offers ACID transactional guarantee (Atomicity, Consistency, Isolation, and Durability)
- PostgreSQL extension Citus Data offers Distributed SQL features.
- Advanced indexes such as Partial Index and Bloom Filters.
- Supports structured data (SQL), semi-structured data (JSON, XML), key-value, and spatial data.

MonogDB

- It was designed to specially handle document data
- Horizontal scaling via auto-sharding (method for distributing data across multiple machines).
- Built-in replication through primary-secondary nodes.
- Distributed multi-document ACID transactions with snapshot isolation.
- Full-text search engine and data lake built on MongoDB

MLDB

- Machine Learning Database, or MLDB, is an open-source system aimed at tackling big data machine learning tasks.
- It can be used for data collection and storage through the training of machine learning models, or to deploy real-time prediction endpoints.
- MLDB is one of the easier datasets to use, since it provides a comprehensive implementation of the SQL SELECT statement.
- It treats datasets as tables, making it easier to learn and use for data analysts already versed in an existing Relational Database Management System (RDBMS).
- Supports vertical scaling with higher efficiency.

... more information ...

https://www.unite.ai/10-best-databases-for-machine-learning-ai

Data Types JavaScript

- Core data types are:
  - number
  - boolean
  - string
  - function
  - object

That's all folks!

Data Types JavaScript

But data can be organized in:

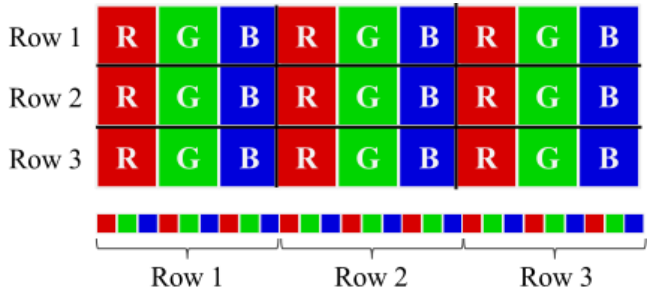
- **Arrays** (poly-sorted, i.e., not compacted with heterogeneous element types possible)
- **Typed Arrays** (mono-sorted. i.e., compacted/packed and each element with same data type):
  - Int8, Uint8
  - Int16, Uint16
  - Int32, Uint32
  - Float32, Float64
- **Records**

Image Data

Formats:

- **RGBA** ⇒ `Uint8 [Red,Green,Blue,Alpha] [col][row]`
- **RGB** ⇒ `Uint8 [Red,Green,Blue] [col][row]`
- **GRAY8** ⇒ `Uint8 [col][row]`

Data layout is relevant! Commonly, [RGB], ordered by columns, organized in rows



[DOI: 10.1109/CLEI.2015.7359995]

Fig. 7. Structure of the RGB image as a sequence of bytes in a linear memory or file model

Data Tables

Array `[[]]`

Generic arary (rows) of arrays (columns). Any dimensionality can be composed by nesting arrays.

TypedArray `TA`

Linear ordered list of values. Higher dimensions must be organized by external wrapper functions or objects knowing the value order and encoding.

Structure Arrays `[{}]`

Each row of the array is a structure of named fields.

Data Tables

Matrix `Matrix`

Wrapper object for nested arrays with extended set of methods and operations

MatrixTA `MatrixTA`

Wrapper object for typed arrays with extended set of methods and operations (two- or three-dimensional)

Volumes `Vol3`

Wrapper object for typed arrays with extended set of methods and operations (always three-dimensional), used by CNNs

File Data Formats

CSV
Comma Separated Value format (text)

JSON
JavaScript Object Notation (text)

XML
Extensible Meta Language (text)

YAML
Yet Another Meta Language (text)

NumPy
Numerical Python Format (binary)

JSON

- Array and record structure (nested, tree)

```
{
  "employee": {
    "name":       "sonoo",
    "salary":      56000,
    "married":    true,
    "awards" :    [1920,1990,2000]
  }
}
```

CSV

- Flat table (not nested) with values separated by delimiter (e.g., "," or any other delimiter like the tabulator character)
- Numbers and text are valid values
- Each line in the file is one row in the data table
- All rows should contain same number of values

```
x,y,z,class
1,2,3,"A"
1,4,2,"B"
4,5,0,"A"
```

Data Flow Architecture

The entire data processing architecture is a graph of computational nodes, data sources, and data sinks, connected by event-based channels. Nodes are connected via input and output ports.

Data Source

A data source node provides access to data, e.g., from an SQL database or from the file system. There are read and write events. Data tokens are created on request (by a read event trigger, e.g.).

Data Sink

A data sink nodes consumes data tokens and displays them. Typical display types are textual information, tables, and plots.

Computational Functions

A computational node processes data read from input ports and writes the processed data on output ports.

Examples are:

- Statistics (from data tables)
- Matrix operations (in-place or by creating new Matrix data)
- Image operations and transformations, filters
- Time-Frequency transformations, convolution, wavelet transformations
- Merging of data, splitting of data, wrapping and unwrapping of data
- Data format conversion

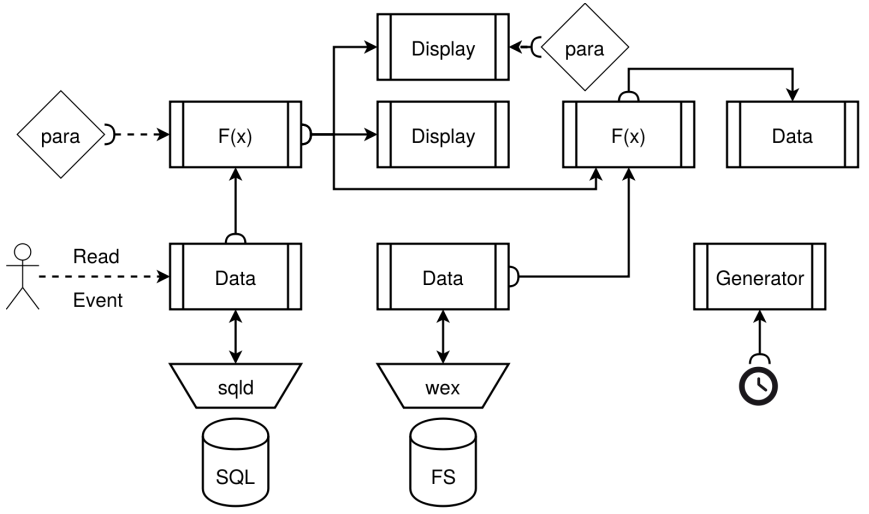


Fig. 8. Processing graph and data flow architecture with data source, processing, and sink nodes. Event-based data flow architecture and event chains. New data provided by a node is propagated to all child nodes. Parameter changes initiate a re-computation (or display), too.

Programming

```
var source1 = new Data(type,{ .. });
var procfu1 = new Transformation(type,{.. });
var procfu2 = new Filter(type,{.. });
var procfu3 = new Functional(type,{.. });
var sink1   = new Display(type,{.. });
source1.output(procfun1)
source1.output(procfun2)
procfun1.output(procfun2,0)
procfun2.output(procfun2,1)
procfun3.output(sink1)
```

Def. 1. Principle composition of data flow systems using block objects and connections

SQL-RPC API

```
var data1 = new Data('sql',{
  url:'edu-9.de:9999',
  database : 'imagesnet12',
  table : 'imageset1'
})
var result = await data1.start();
var image  = await data1.read('*','rowid=1'); // SQL select
var schema = await data1.info();
```

SQLD

- `sqld` consists of a slim native C-code implementation of the *sqlite3* server storing SQL data bases in plain binary files on the local file system.
- SQL data bases can be accessed by a Remote Procedure Call JSON interface, basically mapping SQL operations on a JSON structure (both request and reply).
- HTTP is used to access the JSON-RPC API

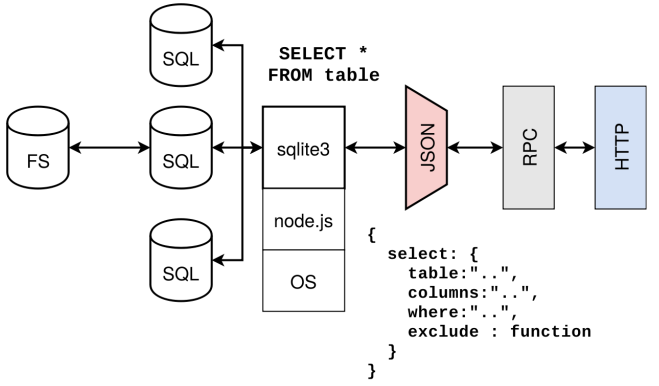


Fig. 9. SQLD architecture and JSON-RPC interface

Summary

- Data can be represented by different data types, structures, formats
- Data access is provided by SQL data bases in an unified way using a remote procedure call programming interface
- Data processing is performed by using an event-based data-flow architecture