Introduction to Data Management

Konstantin Tretyakov
http://kt.era.ee

AACIMP Summer School 2014, Kiev

What is “Data”?

What is “Data”?

- Data as an *abstract concept* can be viewed as the lowest level of abstraction, from which *information* and then *knowledge* are derived. [Wikipedia]

What is "Data"?

- Data as an *abstract concept* can be viewed as the lowest level of abstraction, from which *information* and then *knowledge* are derived. [Wikipedia]

- For our purposes, data is any digital entity (or a set of them) that you can meaningfully
  - Store
  - Transfer or Process
  - Retrieve or "Query"

Abstract data models

Unstructured

010010100111
011101011011
0011001110…

"Flat file", "BLOB"

Structured

Name: John
Surname: Smith
Age: 40
Sex: M

"Record", "Object"

Abstract data models

Unstructured

0100101001 1 1
01110101 1011
001100111 0…

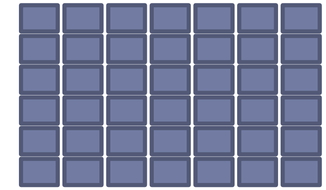
"Flat file", "BLOB"

Structured

Name: John
Surname: Smith
Age: 40
Sex: M

"Record", "Object"

Multidimensional



"Array", "Matrix"

Table / Relation

Na	Su	Age
John	Smith	40
Ann	Smith	35

"List / Set of Records"

Abstract data models

Unstructured

0100101001111
0111010110111
00111001110…

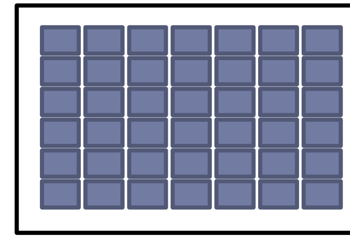
"Flat file", "BLOB"

Structured

Name: John
Surname: Smith
Age: 40
Sex: M

"Record", "Object"

Multidimensional



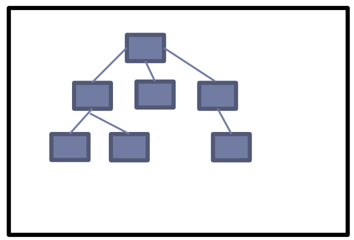
"Array", "Matrix"

Table / Relation

Na	Su	Age
John	Smith	40
Ann	Smith	35

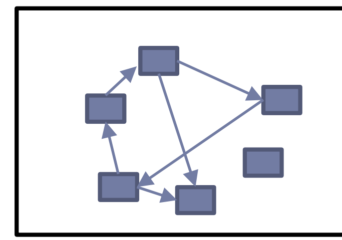
"List / Set of Records"

Tree / Hierarchy



"DOM"

Graph



"Networks"

Abstract data models

Unstructured

0100101001111
0111010110111
001100111 0...

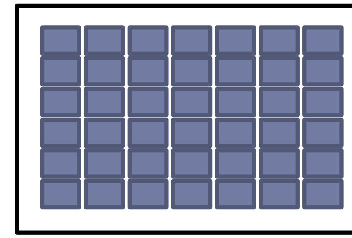
"Flat file", "BLOB"

Structured

Name: John
Surname: Smith
Age: 40
Sex: M

"Record", "Object"

Multidimensional



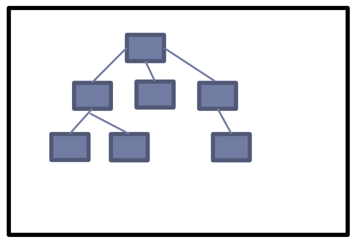
"Array", "Matrix"

Table / Relation

Na	Su	Age
John	Smith	40
Ann	Smith	35

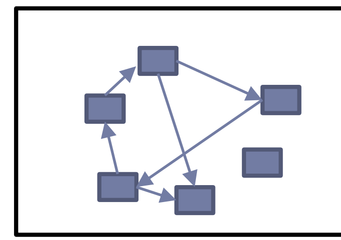
"List / Set of Records"

Tree / Hierarchy



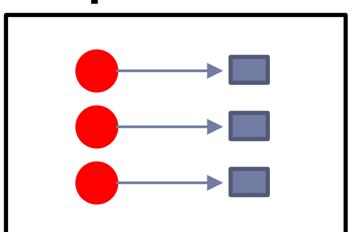
"DOM"

Graph



"Networks"

Map



"Key-value store"

+ Any combination of those

Abstract data **structures**

Abstract data **structures**

Unstructured

0100101001110111010110110011001110…

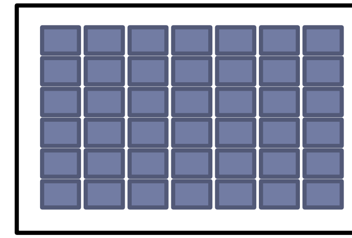
"Memory"

Structured

Name: John
Surname: Smith
Age: 40
Sex: M

"Record", "Object"

Multidimensional



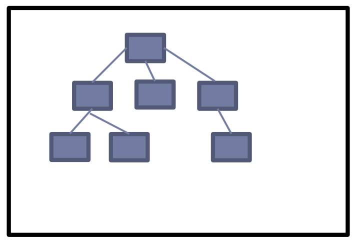
"Array", "Matrix"

Table / Relation

Na	Su	Age
John	Smith	40
Ann	Smith	35

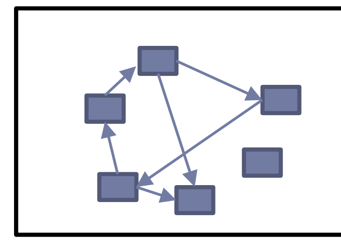
"List / Set of Records"

Tree / Hierarchy



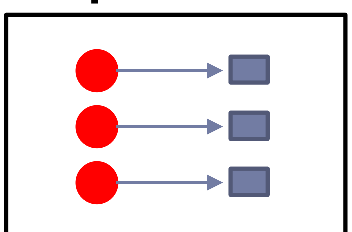
"DOM"

Graph



"Networks"

Map



"Key-value store"

+ Any combination of those

Data model vs data structure

- **Data model** – how you **interpret** your data
- **Data structure** – how you actually **store**, **transfer** and **query** it.

- Example:

Data model vs data structure

- **Data model** – how you **interpret** your data
- **Data structure** – how you actually **store**, **transfer** and **query** it.

- Example:
  - You can use a relational database to store a graph.
  - You can use a map to store a relational table.

Data structure implementations

- Conventional programming languages provide in-memory implementations of various data structures.
- E.g. in Python:

```
my_map = {'key1' : 10.0,
          'key2' : ['List','of','strings'],
          'key3' : numpy.array([3,4,5]),
          'key4' : SomeRecord(x=2, y=3),
          'key5' : networkx.Graph() }
```

Understanding Data

Data model

Abstract data structure(s)

Data structure implementation

Data stored in memory or on disk

Understanding Data

Data model

Blog Post (Author, Date, Text)

Comment (Author, Date, Text)

Abstract data structure(s)

Data structure implementation

Data stored in memory or on disk

Understanding Data

Data model

Blog Post (Author, Date, Text)

Comment (Author, Date, Text)

Abstract data structure(s)

Id	Parent_id	Author	Date	Text
1	0	John	1.2.3	Hey
2	1	Ann	2.2.3	Ho

Data structure implementation

Data stored in memory or on disk

Understanding Data

Data model

Blog Post (Author, Date, Text)

→ Comment (Author, Date, Text)

Abstract data structure(s)

Id	Parent_id	Author	Date	Text
1	0	John	1.2.3	Hey
2	1	Ann	2.2.3	Ho

Data structure implementation

MySQL

Data stored in memory or on disk

InnoDB

Understanding Data

Data model

Blog Post (Author, Date, Text)

→ Comment (Author, Date, Text)

Abstract data structure(s)

Id	Parent_id	Author	Date	Text
1	0	John	1.2.3	Hey
2	1	Ann	2.2.3	Ho

Data structure implementation

CSV file

Data stored in memory or on disk

Understanding Data

Data model

Blog Post (Author, Date, Text)

Comment (Author, Date, Text)

Abstract data structure(s)

Author: John
Date: 1.2.3
Text: Hey
Comments:

Data structure implementation

CouchDB

Data stored in memory or on disk

Understanding Data

Data model

Blog Post (Author, Date, Text)

Comment (Author, Date, Text)

Abstract data structure(s)

Author: John
Date: 1.2.3
Text: Hey
Comments:

Data structure implementation

MongoDB

Data stored in memory or on disk

Understanding Data

Data model

Blog Post (Author, Date, Text)

Comment (Author, Date, Text)

Abstract data structure(s)

Author: John
Date: 1.2.3
Text: Hey
Comments:

Data structure implementation

MongoDB

Data stored in memory or on disk

Memcached + MongoDB

Understanding Data

Data model

Blog Post (Author, Date, Text)

Comment (Author, Date, Text)

Abstract data structure(s)

Author: John
Date: 1.2.3
Text: Hey
Comments:

Data structure implementation

JSON File

Data stored in memory or on disk

Understanding Data

Data model	**Applications**
Abstract data structure(s)	**Interfaces, protocols and query languages**
Data structure implementation	**Database engines, libraries & tools**
Data stored in memory or on disk	**Data formats**

Understanding Data

Data model	Social, Business, Science, Geo, Bio, Image, Sound
Abstract data structure(s)	SQL, MDX, DOM, ORM, OpeNDAP, Graph
Data structure implementation	MySQL, Postgres, Rasdaman, MongoDB, Neo4J, Mondrian
Data stored in memory or on disk	CSV, JSON, XML, Sqlite, HDF5, BSD, Microformats

Overview

- Relational databases & SQL
- Multidimensional / Array databases & MDX
- Hierarchical data & DOM
- Key-value stores & NoSQL

Relational databases

Core concept: a “relation”, i.e. a table:

Column A	Column B	Column C
Paul	McCartney	1
John	Lennon	2
Ringo	Starr	3
George	Harrison	4

A **relation** is a **set of records** each with a fixed set of fields.

Relational algebra

- We can define a set of operations that take relations as input and produce relations as output:
  - Set operations (union, intersection, difference)
  - Projection
  - Filtering
  - Joins

Relational algebra

- **Union, intersection** and **difference**

A	B	C
Paul	McCartney	1
John	Lennon	2
Ringo	Starr	3

U

A	B	C
Paul	McCartney	1
George	Harrison	4

Relational algebra

- **Union**, **intersection** and **difference**

A	B	C
Paul	McCartney	1
John	Lennon	2
Ringo	Starr	3

$\cap$

A	B	C
Paul	McCartney	1
George	Harrison	4

Relational algebra

- **Union**, **intersection** and **difference**

A	B	C
Paul	McCartney	1
John	Lennon	2
Ringo	Starr	3

-

A	B	C
Paul	McCartney	1
George	Harrison	4

Relational algebra

- **Filtering**

$$\sigma_{[A=Paul]}\left(\begin{array}{|l|l|l|}\hline A & B & C \\ \hline Paul & McCartney & 1 \\ John & Lennon & 2 \\ Ringo & Starr & 3 \\ \hline\end{array}\right)$$

A	B	C
Paul	McCartney	1
John	Lennon	2
Ringo	Starr	3

Relational algebra

- **Filtering**

$$\sigma_{[A=Paul]}\left(\begin{array}{|l|l|l|}\hline A & B & C \\ \hline Paul & McCartney & 1 \\ John & Lennon & 2 \\ Ringo & Starr & 3 \\ \hline\end{array}\right)$$

A	B	C
Paul	McCartney	1

Relational algebra

- **Projection**

$$\Pi_{[A,C]}\left(\begin{array}{|l|l|l|}\hline A & B & C \\ \hline Paul & McCartney & 1 \\ John & Lennon & 2 \\ Ringo & Starr & 3 \\ \hline\end{array}\right)$$

A	B	C
Paul	McCartney	1
John	Lennon	2
Ringo	Starr	3

Relational algebra

- **Projection**

$$\Pi_{[A,C]}\left(\begin{array}{|l|l|l|}\hline A & B & C \\ \hline Paul & McCartney & 1 \\ John & Lennon & 2 \\ Ringo & Starr & 3 \\ \hline\end{array}\right)$$

A	C
Paul	1
John	2
Ringo	3

Relational algebra

- **Cross-Join**

A	B	C
Paul	McCartney	1
John	Lennon	2
Ringo	Starr	3

$\times$

D	E
2	X
3	Y

Relational algebra

- **Cross-Join**

A	B	C
Paul	McCartney	1
John	Lennon	2
Ringo	Starr	3

×

D	E
2	X
3	Y

A	B	C	D	E
Paul	McCartney	1	2	X
John	Lennon	2	2	X
Ringo	Starr	3	2	X
Paul	McCartney	1	3	Y
John	Lennon	2	3	Y
Ringo	Starr	3	3	Y

Relational algebra

- **(Natural / Equi) Join**

A	B	C
Paul	McCartney	1
John	Lennon	2
Ringo	Starr	3

$$\bowtie_{[C=D]}$$

D	E
2	X
3	Y

Relational algebra

- **(Natural / Equi) Join**

A	B	C
Paul	McCartney	1
John	Lennon	2
Ringo	Starr	3

$\bowtie_{[C=D]}$

D	E
2	X
3	Y

A	B	C	D	E
John	Lennon	2	2	X
Ringo	Starr	3	3	Y

Relational algebra

- **Left outer join**

A	B	C
Paul	McCartney	1
John	Lennon	2
Ringo	Starr	3

$$⟕_{[C=D]}$$

D	E
2	X
3	Y

A	B	C	D	E
Paul	McCartney	-	-	-
John	Lennon	2	2	X
Ringo	Starr	3	3	Y

Relational algebra

A combination of relational algebra operators formulates a *query*, e.g.:

$$\Pi_{E}(\sigma_{[A=\text{Paul}]}(R_1 \sqsupset\!\bowtie_{[C=D]} R_2))$$

Relational algebra

RA operations satisfy a number of useful algebraic properties, which can be used for **query optimization**:

$$\sigma_A(R \times P) = \sigma_{B \wedge C \wedge D}(R \times P) = \sigma_D(\sigma_B(R) \times \sigma_C(P))$$

$$\sigma_{A \wedge B}(R) = \sigma_A(\sigma_B(R)) = \sigma_B(\sigma_A(R))$$

$$\sigma_A(R \cap P) = \sigma_A(R) \cap \sigma_A(P) = \sigma_A(R) \cap P = R \cap \sigma_A(P)$$

… etc

SQL

- In practice we formulate relational queries using Structured Query Language (SQL).

```
select A, B, C
from Rel1
```

SQL

- In practice we formulate relational queries using Structured Query Language (SQL).

```
select A, B, C
from Rel1, Rel2
```

SQL

- In practice we formulate relational queries using Structured Query Language (SQL).

```
select A, B, C
from Rel1, Rel2
where
    Rel1.C = Rel2.D and Rel1.A = 'Paul'
```

SQL

- In practice we formulate relational queries using Structured Query Language (SQL).

```
select A, B, C
from Rel1
    left join Rel2 on (Rel1.C = Rel2.D)
where
    Rel1.A = ‘Paul’
```

SQL

- In practice we formulate relational queries using Structured Query Language (SQL).

```
select C
from Rel1

union

select D
from Rel2
```

SQL

- In practice we formulate relational queries using Structured Query Language (SQL).

```
select C
from Rel1
order by A desc
```

SQL

- In practice we formulate relational queries using Structured Query Language (SQL).

Rel1

A	B
Z	1
Z	2
Z	3
Y	4
X	5
X	6

```
select A, sum(B) as S
from Rel1
group by A
```

SQL

- In practice we formulate relational queries using Structured Query Language (SQL).

Rel1

A	B
Z	1
Z	2
Z	3
Y	4
X	5
X	6

```
select A, sum(B) as S
from Rel1
group by A
```

result

A	S
Z	6
Y	4
X	11

SQL

- In practice we formulate relational queries using Structured Query Language (SQL).

Rel1

A	B
Z	1
Z	2
Z	3
Y	4
X	5
X	6

```
select * from
    (select A, sum(B) as S
     from Rel1
     group by A)
where S > 5
```

result

A	S
Z	6
X	11

SQL

- SQL is declarative, the actual execution of the query is determined by the database engine.

Rel1

id	x
1	A
2	B
3	C

Rel2

id	y
1	D
2	E
4	F

```
select *
from Rel1, Rel2
where
    Rel1.id = Rel2.id
    and Rel1.x = 'A'
```

SQL

- SQL is declarative, the actual execution of the query is determined by the database engine.

Rel1

id	x
1	A
2	B
3	C

Rel2

id	y
1	D
2	E
4	F

```
select *
from Rel1, Rel2
where
    Rel1.id = Rel2.id
    and Rel1.x = 'A'
```

```
Execution plan:
1. Create cross join
2. Perform filtering
   on
   Rel1.id = Rel2.id
   and Rel1.x = 'A'
```

SQL

- SQL is declarative, the actual execution of the query is determined by the database engine.

Rel1

id	x
1	A
2	B
3	C

Rel2

id	y
1	D
2	E
4	F

```
select *
from Rel1, Rel2
where
    Rel1.id = Rel2.id
    and Rel1.x = 'A'
```

```
Execution plan:
1. Perform filtering
   on Rel1.x = 'A'
2. For each resulting
   row of Rel1,
   scan Rel2,
   searching for
   matches on id.
```

SQL

- SQL is declarative, the actual execution of the query is determined by the database engine.

Rel1

id	x
1	A
2	B
3	C

Rel2

id	y
1	D
2	E
4	F

```
select *
from Rel1, Rel2
where
    Rel1.id = Rel2.id
    and Rel1.x = 'A'
```

```
Execution plan:
1. Create an index
   on Rel2.id.
2. Perform filtering
   on Rel1.x = 'A'
3. For each resulting
   row of Rel1,
   use index from 1.
   to find matches.
```

SQL

- If you know tables may often need to be searched by a certain field, you can explicitly create an index on a given field.

```
create index ix_rel2_id on Rel2 (id)
```

SQL

- Most relational databases are created with **concurrent transactional processing** in mind.

```
begin transaction;
insert into Rel1 values (1, 'A');
update Rel2 set y = 'B' where id = 1;
delete from Rel1 where x = 'C';
commit;    -- (or rollback);
```

SQL

- Most relational databases are created with **concurrent transactional processing** in mind.

```
begin transaction;
insert into Rel1 values (1, 'A');
update Rel2 set y = 'B' where id = 1;
delete from Rel1 where x = 'C';
commit;     -- (or rollback);
```

ACID = Atomicity, Consistency, Isolation, Durability

ORM

Post

id	author_id	text
1	1	Hey
2	1	Jude
3	2	Don't

Author

id	name
1	Paul
2	John
3	George
4	Ringo

- In software we may access relational databases using standard SQL queries.

```
select Author.name
from Post
   left join Author on (author_id = Author.id)
where Post.id = 1
```

ORM

Post

id	author_id	text
1	1	Hey
2	1	Jude
3	2	Don't

Author

id	name
1	Paul
2	John
3	George
4	Ringo

- In practice, however, it may often be convenient to use a higher-level abstraction: *Object-Relational Mapping* (ORM).

ORM

Post

id	author_id	text
1	1	Hey
2	1	Jude
3	2	Don't

Author

id	name
1	Paul
2	John
3	George
4	Ringo

```
class Post:
  __tablename__ = 'Post'
  id          = Column(Integer, primary_key=True)
  author_id = Column(Integer, ForeignKey(Author))
  text        = Column(Unicode)
```

```
class Author:
   __tablename__ = 'Author'
   id = Column(Integer, primary_key = True)
   name = Column(String)
```

ORM

Post

id	author_id	text
1	1	Hey
2	1	Jude
3	2	Don't

Author

id	name
1	Paul
2	John
3	George
4	Ringo

```
connection.query("select Author.name
from Post
   left join Author on (author_id = Author.id)
where Post.id = 1")
```

vs

```
Post.get(1).author.name
```

ORM

- Although SQL is largely standardized, various database engines have slightly different *dialects*.

- ORM systems often let you abstract from those differences. In this case, you can transparently switch database implementations.

```
e = create_engine('mysql://user:pass@localhost/my_db')
```

```
e = create_engine('postgresql://user:pass@localhost/my_db')
```

```
e = create_engine('sqlite://:memory:')
```

SQL Summary

- Relational databases are the lingua franca of data management.
- Knowledge of SQL is mandatory for you.
- ORM is a nice-to-have addition sometimes (in software development often a must).

Quiz

- ACID = __________
- Table `Rel1` has 4 rows, Table `Rel2` has 8 rows. How many rows does a cross-join of `Rel1` and `Rel2` have?

Quiz

- The table `Post` has a field `author_id`, which is a pointer (foreign key) to the `Author.id` field. Should we create an index on `Post.author_id`?

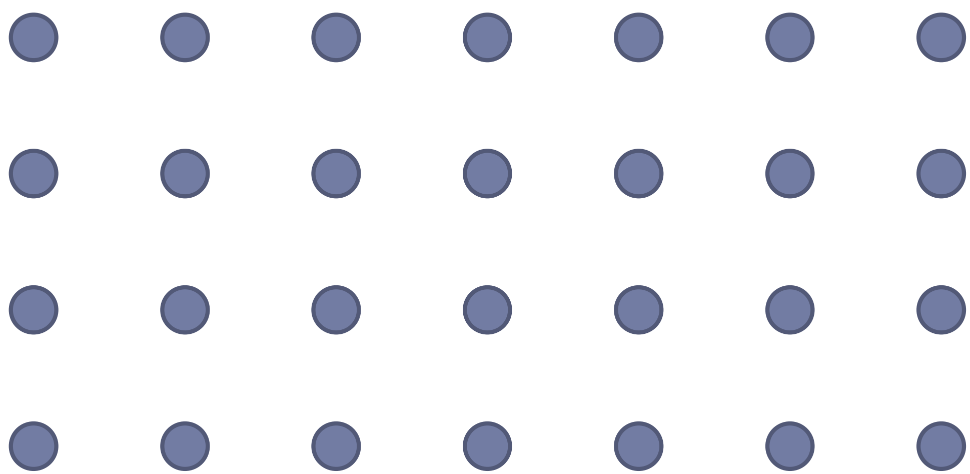
- Why use a relational database server, when you can use data structures built in your programming language?

Overview

- ~~Relational databases & SQL~~
- Multidimensional / Array databases & MDX
- Hierarchical data & DOM
- Key-value stores & NoSQL

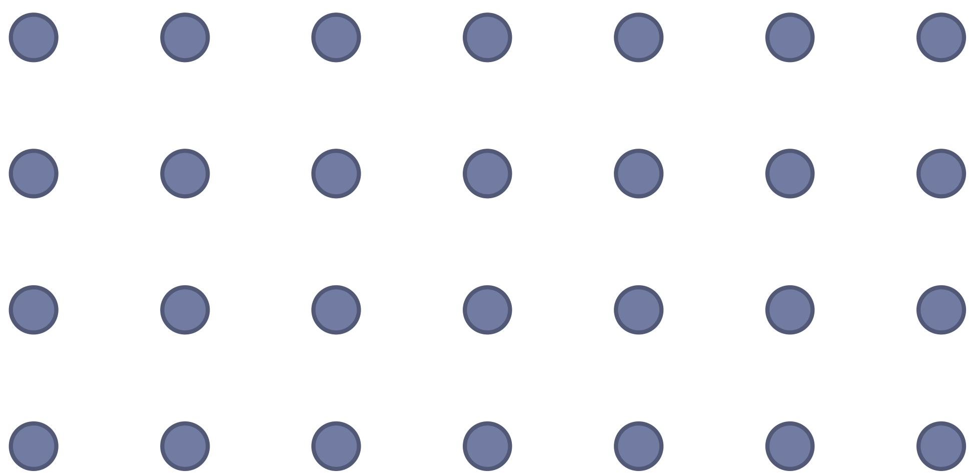
Multidimensional databases

- Consider a dataset of atmospheric indicators (pressure, humidity, etc), taken over a regular 2D grid of points spread over the atmosphere, with one measurement per hour.



Multidimensional databases

- It is not unusual to have a grid sized1000x1000. If every point produces two measurements (pressure and humidity), the whole grid produces 24x2x1000x1000 measurements per day.



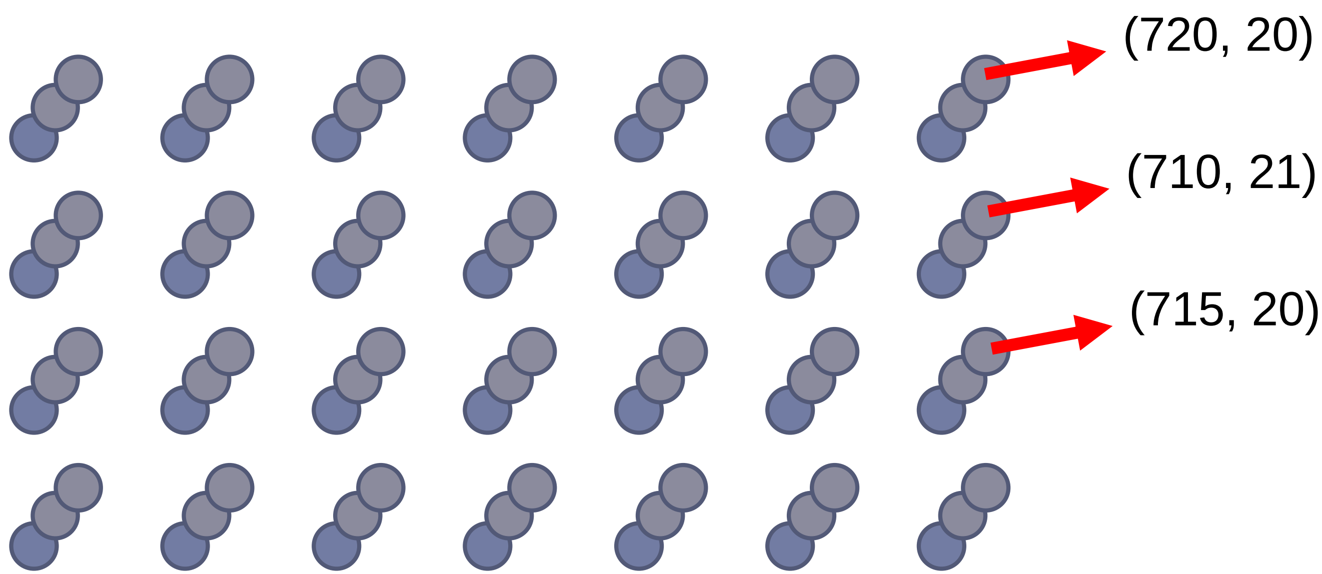
Multidimensional databases

You can store such measurements in a relational database like that:

Grid X	Grid Y	Timepoint	Pressure	Humidity
1	1	1	700	20
1	2	1	710	21
1	3	1	705	20

Multidimensional databases

However, it is more reasonable to represent the data (both internally and conceptually) like a multidimensional array ("cube"):



Multidimensional databases

However, it is more reasonable to represent the data (both internally and conceptually) like a multidimensional array ("cube"):

720, 20	722, 20	721, 20	722, 20	720, 20
710, 21	711, 20	712, 21	711, 20	710, 21
715, 20	713, 21	701, 20	713, 21	715, 20
711, 22	714, 23	712, 22	714, 23	711, 22
714, 23	709, 21	701, 23	709, 21	714, 23

Multidimensional databases

- The most common types of operations applied with multidimensional data are:
  - Slicing
  - Dicing
  - Drill Up/Drill Down/Roll Up

Multidimensional databases

- Slicing
  - “Pick all pressure values for given timepoint and grid column”

720	722, 20	721, 20	722, 20	720, 20
710	711, 20	712, 21	711, 20	710, 21
715	713, 21	701, 20	713, 21	715, 20
711	714, 23	712, 22	714, 23	711, 22
714	709, 21	701, 23	709, 21	714, 23

```
pressure[:,1,1]
```

Multidimensional databases

- Slicing
  - “Pick all pressure values for given grid coordinates”

720, 20	722	721, 20	722, 20	720, 20
710, 21	711, 20	712, 21	711, 20	710, 21
715, 20	713, 21	701, 20	713, 21	715, 20
711, 22	714, 23	712, 22	714, 23	711, 22
714, 23	709, 21	701, 23	709, 21	714, 23

```
pressure[1,2,:]
```

Multidimensional databases

- Dicing
  - "Select a subcube limited by time 1..2, grid coordinates 1..3, 1..2"

720, 20	722, 20	721, 20	722, 20	720, 20
710, 21	711, 20	712, 21	711, 20	710, 21
715, 20	713, 21	701, 20	713, 21	715, 20
711, 22	714, 23	712, 22	714, 23	711, 22
714, 23	709, 21	701, 23	709, 21	714, 23

```
data[1:3,1:2,1:2]
```

Multidimensional databases

- Drill-Up
  - "Average values over all timepoints"

720, 20	722, 20	721, 20	722, 20	720, 20
710, 21	711, 20	712, 21	711, 20	710, 21
715, 20	713, 21	701, 20	713, 21	715, 20
711, 22	714, 23	712, 22	714, 23	711, 22
714, 23	709, 21	701, 23	709, 21	714, 23

```
mean(data[:,:,:], axis=2)
```

Multidimensional databases

- Drill-Up
  - "Average values over all timepoints and grid rows"

720, 20	722, 20	721, 20	722, 20	720, 20

```
mean(mean(data[:,:,:], axis=2), axis=0)
```

Multidimensional databases

- Drill-Up
  - "Average values over all timepoints/grid rows/columns"

720, 20

```
mean(data)
```

Multidimensional databases

- Roll-Up with formulas
  - "Average (pressure – 2*humidity) over all data"

680

```
mean(pressure - 2*humidity)
```

Multidimensional databases

- Similarly to Relational Algebra, the set slice/dice/drill-up/drill-down operations makes an algebra, and any query can be represented by an algebraic expression.
- There is no de-facto standard query language – each vendor has its own.
  - Rasdaman – rasql
  - SciDB – AQL
  - Microsoft SQL Server / Mondrian - MDX

Multidimensional databases

- RaSQL:

```
select mr[100:150,40:80] / 2
from mr
where some_cells( mr[120:160, 55:75] > 250 )
```

- AQL:

```
select sqrt(pressure)
from data
where i >= 1 and i < 5
```

- MDX:

```
select { Time.1, Time.2 } on columns
         Rows.Children         on rows
from Data
where  (MeasureType.Pressure)
```

Quiz

- Who are the primary users of multidimensional / array databases?
- When does it make sense to store multidimensional data in a relational table?

Summary

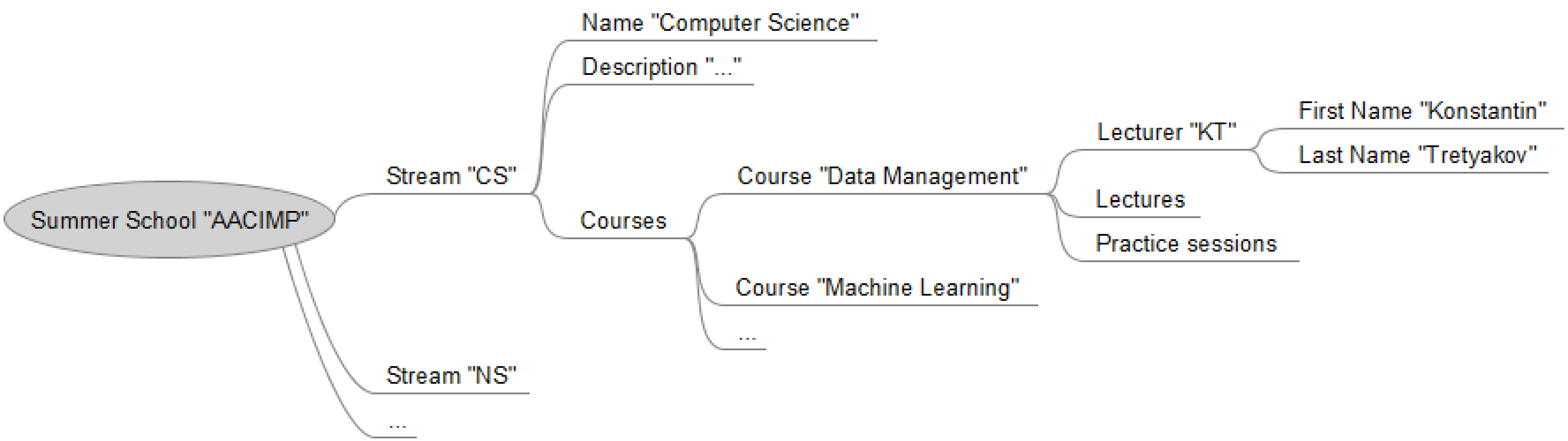
- If you do science, you need to know how to work with large multidimensional arrays.
- Even if your data is not inherently an array, but you want to query a lot of summary statistics (means, sums, trends, etc), you better regard it as a *cube*.
- There are systems and query languages for multidimensional data.
- Learning MDX may positively change the way you think about data.

Overview

- ~~Relational databases & SQL~~
- ~~Multidimensional / Array databases & MDX~~
- Hierarchical data & DOM
- Key-value stores & NoSQL

Tree data model

- A tree model is a very natural way of representing data about various objects:



Tree data model

- In fact, it is perhaps the most commonly used type of data representation.

Tree data model

- Files on your computer and on the web are organized as a tree

```
/usr/lib/share/whatever/etc
```

Tree data model

- Web domain names are organized as a tree:

```
ua
   org.ua
      ssa.org.ua
          summerschool.ssa.org.ua
```

Tree data model

- Most data served over the web is a tree:

```
▼<html lang="en" dir="ltr" class="client-js ve-not-available">
  ▼<head>
      <meta charset="UTF-8">
      <title>Data model - Wikipedia, the free encyclopedia</title>
      <meta name="generator" content="MediaWiki 1.24wmf14">
      <link rel="alternate" type="application/x-wiki" title="Edit this page" href="/w/index.p
      <link rel="edit" title="Edit this page" href="/w/index.php?title=Data_model&action=edit
    </head>
  ▼<body class="mediawiki ltr sitedir-ltr ns-0 ns-subject page-Data_model skin-vector actio
      <div id="mw-page-base" class="noprint"></div>
      <div id="mw-head-base" class="noprint"></div>
    ▶<div id="content" class="mw-body" role="main">…</div>
    ▶<div id="mw-navigation">…</div>
    ▶<script>…</script>
    ▶<div class="suggestions" style="display: none; font-size: 12.727272033691406px;">…</di
    </body>
</html>
```

Tree data model

- Most data served over the web is a tree:

```
{ tagName: "HTML",
  lang: "en",
  childNodes: [
        { tagName: "HEAD",
          childNodes : [
                { tagName: "META",
                  charset: "UTF-8" },
                { tagName: "TITLE",
                  textContent: "Data Model" }
                ...
          ]
        },
        { tagName: "BODY",
          childNodes: [
                ...
          ]
        }
  ]
}
```

Tree data model

- A typical programming interface for working with a tree data model consists of a "Node" object with methods to get/set attributes, access/modify children, access the parent node:

```
> document
  ▶ #document
> document.childNodes
  [<!DOCTYPE html>, ▶ <html lang="en" dir="ltr" class="client-js ve-not-available">…</html>]
> document.childNodes[1].tagName
  "HTML"
> document.childNodes[1].getAttribute("lang")
  "en"

> document.childNodes[1].appendChild(document.createElement("my-new-node"))
  <my-new-node></my-new-node>
```

Quiz

- How to store a tree in a relational table?

Quiz

- How to store a tree in a relational table?
  - “Parent pointers”

Node ID	Parent node ID	Node data
1	0	Root node
2	1	---First child
3	2	------First grandchild
4	1	---Second child
5	4	------Second grandchild
6	4	------Third grandchild

Quiz

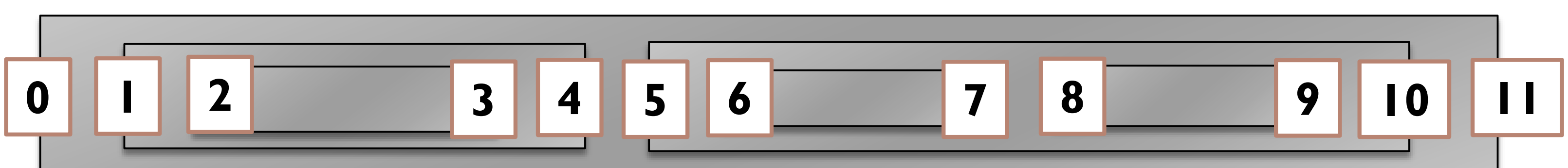
- How to store a tree in a relational table?
  - "Nested Sets"

Node ID	Left	Right	Node data
1	0	11	Root node
2	1	4	---First child
3	2	3	------First grandchild
4	5	10	---Second child
5	6	7	------Second grandchild
6	8	9	------Third grandchild

Quiz

- How to store a tree in a relational table?
  - "Nested Sets"

Node ID	Left	Right	Node data
1	0	11	Root node
2	1	4	---First child
3	2	3	------First grandchild
4	5	10	---Second child
5	6	7	------Second grandchild
6	8	9	------Third grandchild



Quiz

- How to store a tree in a relational table?
  - "Flat table"

Node ID	Level	Node data
1	0	Root node
2	1	---First child
3	2	------First grandchild
4	1	---Second child
5	2	------Second grandchild
6	2	------Third grandchild

Tree query languages

- There are several query languages for tree data. The two most important ones are **XPath** and **CSS Selectors.**
  - XPath:

```
/html/head/title
```

```
//div[@class="text"]/p//*
```

  - CSS Selectors:

```
html > head > title
```

```
div.text > p *
```

Summary

Data type	Operations	Query languages
Relational	Relational algebra	SQL
Multidimensional	Slice / Dice / Drill Up-Down	MDX, RaSQL, AQL, …
Tree	Tree node operations	XPath, CSS

Overview

- ~~Relational databases & SQL~~
- ~~Multidimensional / Array databases & MDX~~
- ~~Hierarchical data & DOM~~
- Key-value stores & NoSQL

Introduction to Data Management

Part II

Konstantin Tretyakov

http://kt.era.ee

AACIMP Summer School 2014, Kiev

Quiz

- Yesterday we discussed which three major data models / structures?
- What operations are supported for each of those data models?
- What query languages are used with them?

Abstract data models

Unstructured

0100101001111
0111010110111
0011001110…

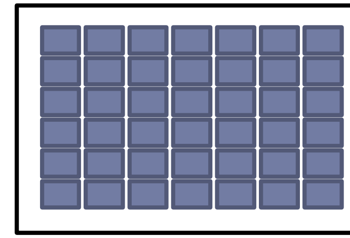
“Flat file”, “BLOB”

Structured

Name: John
Surname: Smith
Age: 40
Sex: M

“Record”, “Object”

Multidimensional



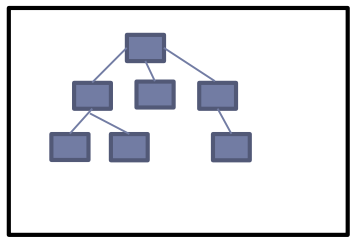
“Array”, “Matrix”

Table / Relation

Na	Su	Age
John	Smith	40
Ann	Smith	35

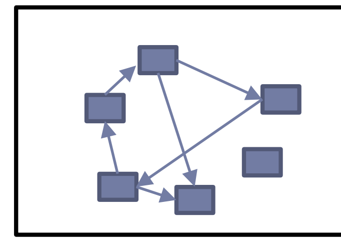
“List / Set of Records”

Tree / Hierarchy



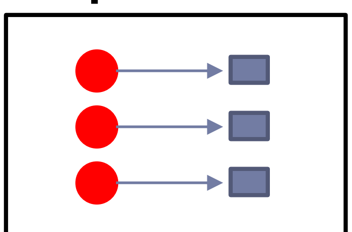
“DOM”

Graph



“Networks”

Map



“Key-value store”

+ Any combination of those

Abstract data models

Unstructured

0100101001 1 1
0111010110 1 1
0011001110…

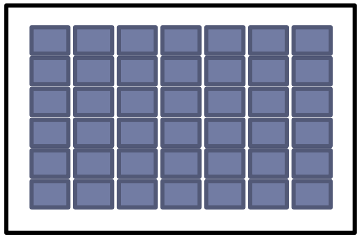
"Flat file", "BLOB"

Structured

Name: John
Surname: Smith
Age: 40
Sex: M

"Record", "Object"

Multidimensional



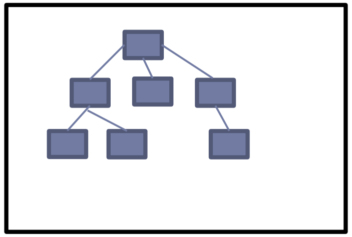
"Array", "Matrix"

Table / Relation

Na	Su	Age
John	Smith	40
Ann	Smith	35

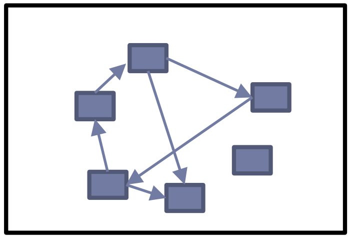
"List / Set of Records"

Tree / Hierarchy



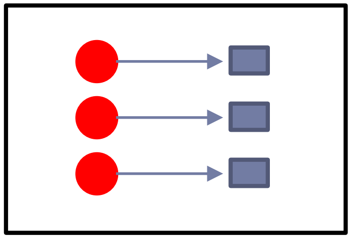
"DOM"

Graph



"Networks"

Map



"Key-value store"

+ Any combination of those

Overview

- Relational databases & SQL
- Multidimensional / Array databases & MDX
- Hierarchical data & DOM
- Key-value stores & NoSQL

Partitioning

- How can you partition:
  - A relational database?
  - A multidimensional database?
  - A tree?

Partitioning

- How can you partition:
  - A relational database?
    - Vertically (“Normalization”)
    - Horizontally (“Sharding”)
  - A multidimensional database?
  - A tree?

Partitioning

- How can you partition:
  - A relational database?
    - Vertically ("Normalization")
    - Horizontally ("Sharding")
  - A multidimensional database?
    - "Tiling"
  - A tree?

Partitioning

- How can you partition:
  - A relational database?
    - Vertically (“Normalization”)
    - Horizontally (“Sharding”)
  - A multidimensional database?
    - “Tiling”
  - A tree?
    - By subtrees

Partitioning

- When your database is partitioned you have to choose:
  - Either it may become inconsistent at times (i.e. different users will get different results)
  - Or your database may become unresponsive when the links between the partitions go down.

“CAP theorem”

Consistency

Availability

Partition-tolerance

You can only choose two

Consistency

- So far we have taken **data consistency** as something natural and implied.
- For very large operational systems availability and partition-tolerance becomes more important than consistency.
- This is where "NoSQL" databases come into play.

NoSQL Operations

- A modern-day "NoSQL" database is a simple key-value store, which primarily supports two operations:
  - set_value(key, value)
    ```
    data[key] = value
    ```
  - get_value(key)
    ```
    data[key]
    ```

NoSQL tradeoffs

- This trivial interface is often sufficient for practical purposes (e.g. Facebook).

- Pros:
  - Trivial scalability
  - Basic availability
- Cons:
  - "Soft state"
  - "Eventual consistency"

NoSQL engines

- There are multiple vendors of NoSQL databases
  - Memcached
  - MongoDB
  - CouchDB
  - Redis
  - Cassandra, etc
- Each engine has a different set of features. Some allow queries by filtering on attributes, (pre)indexing and rudimental "joins".

Example: MongoDB Query Language

```
db.inventory.find(
    {
      type: 'food',
      $or: [ { qty:    { $gt: 100 } },
             { price: { $lt: 9.95 } } ]
    }
)
```

NoSQL is a bad name

- Note that the name "NoSQL" does not mean the database could not be queried with SQL.
- A better name would be "NoACID" to signify that the focus is on scalability at the price of atomicity / consistency / isolation / durability.
- In fact, there is a term (not commonly used) BASE (Basic Availability Soft state Eventual consistency) to refer to NoSQL databases.

Quiz

- What type of operational systems may not use a NoSQL database at the backend?

Summary

Data type	Operations	Query languages
Relational	Relational algebra	SQL
Multidimensional	Slice / Dice / Drill Up-Down	MDX, RaSQL, AQL, …
Tree	Tree node operations	XPath, CSS
Key-value store	Set / Get / Filter	Vendor-specific

Summary

Data type	Operations	Query languages
Relational	Relational algebra	SQL
Multidimensional	Slice / Dice / Drill Up-Down	MDX, RaSQL, AQL, …
Tree	Tree node operations	XPath, CSS
Key-value store	Set / Get / Filter	Vendor-specific
Graph database	Graph operations	Vendor-specific (e.g. Cypher)

Database Trade-offs

- Strict Schema – No Strict Schema
- Efficient write – Efficient read
  [complex queries]
- Consistency – No Consistency
  [eventual consistency]
- Partitioning – No Partitioning

Quiz

Case	Data model	Strict schema	Efficient write	Strict consistency	Partitioned
Supermarket operational point of sale database					
Supermarket data warehouse for business analytics and data mining					
A large-scale internet social network					
The Human Genome					
Medical patient records in a hospital					
Medical patient records for research					

Quiz

Case	Data model	Strict schema	Efficient write	Strict consistency	Partitioned
Supermarket operational point of sale database	SQL	Y	Y	Y	Y/N
Supermarket data warehouse for business analytics and data mining					
A large-scale internet social network					
The Human Genome					
Medical patient records in a hospital					
Medical patient records for research					

Quiz

Case	Data model	Strict schema	Efficient write	Strict consistency	Partitioned
Supermarket operational point of sale database	SQL	Y	Y	Y	Y/N
Supermarket data warehouse for business analytics and data mining	SQL/MD	Y	N	Y/N	N
A large-scale internet social network					
The Human Genome					
Medical patient records in a hospital					
Medical patient records for research					

Quiz

Case	Data model	Strict schema	Efficient write	Strict consistency	Partitioned
Supermarket operational point of sale database	SQL	Y	Y	Y	Y/N
Supermarket data warehouse for business analytics and data mining	SQL/MD	Y	N	Y/N	N
A large-scale internet social network	SQL/Tree /NoSQL	Y/N	Y	N	Y
The Human Genome					
Medical patient records in a hospital					
Medical patient records for research					

Quiz

Case	Data model	Strict schema	Efficient write	Strict consistency	Partitioned
Supermarket operational point of sale database	SQL	Y	Y	Y	Y/N
Supermarket data warehouse for business analytics and data mining	SQL/MD	Y	N	Y/N	N
A large-scale internet social network	SQL/Tree /NoSQL	Y/N	Y	N	Y
The Human Genome	Array	Y	N	Y	N
Medical patient records in a hospital					
Medical patient records for research					

Quiz

Case	Data model	Strict schema	Efficient write	Strict consistency	Partitioned
Supermarket operational point of sale database	SQL	Y	Y	Y	Y/N
Supermarket data warehouse for business analytics and data mining	SQL/MD	Y	N	Y/N	N
A large-scale internet social network	SQL/Tree /NoSQL	Y/N	Y	N	Y
The Human Genome	Array	Y	N	Y	N
Medical patient records in a hospital	SQL/Tree	Y/N	Y	Y	Y/N
Medical patient records for research					

Quiz

Case	Data model	Strict schema	Efficient write	Strict consistency	Partitioned
Supermarket operational point of sale database	SQL	Y	Y	Y	Y/N
Supermarket data warehouse for business analytics and data mining	SQL/MD	Y	N	Y/N	N
A large-scale internet social network	SQL/Tree /NoSQL	Y/N	Y	N	Y
The Human Genome	Array	Y	N	Y	N
Medical patient records in a hospital	SQL/Tree	Y/N	Y	Y	Y/N
Medical patient records for research	SQL/Tree /NoSQL	Y/N	N	Y/N	N

Quiz

- What are the main data model types you should know?
- What are the main tradeoffs when choosing a data structure or a database engine?

Summary

Unstructured

0100101001 1 1
0111010110 1 1
0011001110...

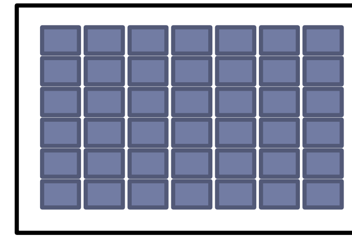
"Memory"

Structured

Name: John
Surname: Smith
Age: 40
Sex: M

"Record", "Object"

Multidimensional



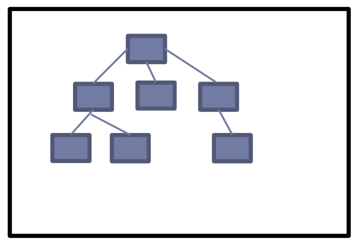
"Array", "Matrix"

Table / Relation

Na	Su	Age
John	Smith	40
Ann	Smith	35

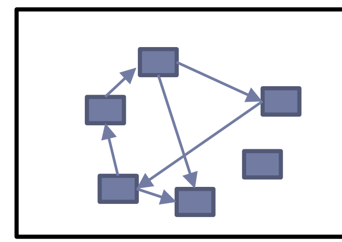
"List / Set of Records"

Tree / Hierarchy



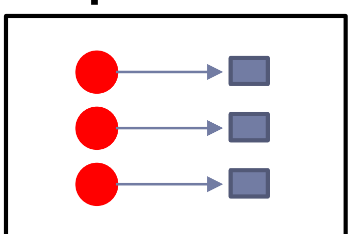
"DOM"

Graph



"Networks"

Map



"Key-value store"

+ Any combination of those

Summary

Data type	Operations	Query languages
Relational	Relational algebra	SQL
Multidimensional	Slice / Dice / Drill Up-Down	MDX, RaSQL, AQL, …
Tree	Tree node operations	XPath, CSS
Key-value store	Set / Get / Filter	Vendor-specific
Graph database	Graph operations	Vendor-specific