# Modern Database Systems

Practicals. SciDB

Doc. RNDr. Irena Holubova, Ph.D.
holubova@ksi.mff.cuni.cz

# SciDB

- Provided by paradigm4
  - Co-founder: M. Stonebraker
- One of the most popular representatives
  - Wide range of functionalities
- Data model
  - Multidimensional sorted array
  - Assumption: data are not overwritten
    - Update = creating a new version of data
    - Aim: analyses of evolution/errors/corrections/… in time

If not explicitly specified

http://www.paradigm4.com/

SciDB

- **AFL (Array Functional Language)**
- **AQL (Array Query Language)**
  - Inspired by SQL
  - Instead of tables we work with arrays
    - Wider set of operations for DDL, DML
  - Compiled into AFL

```
CREATE ARRAY A <x: double, err: double> [i=0:99,10,0,
   j=0:99,10,0];
LOAD A FROM '../examples/A.scidb';
```

- Each array has:
  - At least one attribute (`x`, `err`) with a datatype (2x `double`)
  - At least one dimension (`i`, `j`)
  - Each dimension has :
    - coordinates (`0-99`)
    - size of data chunks (`10` fields) and
    - eventual overlapping (`0`)

SciDB

- SciDB distributes the chunks of data
  - Not too big, not too small
  - Recommendation: 10-20 MB
    - Depending on the datatypes
- Coordinates do not have to be limited (*)
- Overlapping is optional
  - Suitable, e.g., for faster searching nearest neighbours
    - The data would probably be otherwise stored on another cluster node

SciDB

```
// create two 1D arrays
CREATE ARRAY A <val_a:double>[i=0:9,10,0];
LOAD A FROM '../examples/exA.scidb';
CREATE ARRAY B <val_b:double>[j=0:9,10,0];
LOAD B FROM '../examples/exB.scidb';

// print values of coordinate i from array A
SELECT i FROM A;
[(0),(1),(2),(3),(4),(5),(6),(7),(8),(9)]

// print values of attribute val_a from array A and val_b from
// array B
SELECT val_a FROM A;
[(1),(2),(3),(4),(5),(6),(7),(8),(9),(10)]
SELECT val_b FROM B;
[(101),(102),(103),(104),(105),(106),(107),(108),(109),(110)]
```

SciDB

```
// usage of WHERE clause + sqrt() function
SELECT sqrt(val_b) FROM B WHERE j > 3 AND j < 7;
[(),(),(),(),(10.247),(10.2956),(10.3441),(),(),()]
```

- `SELECT` commands
  - Basic operation: inner join
    - Joined arrays must be compatible (coordinates, chunks, overlapping)
      - Amounts and datatypes of attributes can differ
      - Attributes are merged according to the given operation (condition)
  - Other joins: `MERGE`, `CROSS`, `CROSS_JOIN`, `JOIN ON` (a condition), …
  - Nested queries, aggregation (`GROUP BY`), sorting, …

SciDB

```
// joining values of arrays A and B and storing to array C
SELECT * INTO C FROM A, B;
[(1,101),(2,102),(3,103),(4,104),(5,105),(6,106),(7,107),(8,108),(9,109),(10,110)]

// joining values of arrays C and B and storing to array D
SELECT * INTO D FROM C, B;
[(1,101,101),(2,102,102),(3,103,103),(4,104,104),(5,105,105),(6,106,106),(7,107,107),
(8,108,108),(9,109,109),(10,110,110)]

// print information about array D (see attributes with the same name)
SELECT * FROM show(D);
[("D<val_a:double,val_b:double,val_b_2:double> [i=0:9,10,0]")]

// joining the values by addition
SELECT C.val_b + D.val_b FROM C, D;
[(202),(204),(206),(208),(210),(212),(214),(216),(218),(220)]

// self-joining of values
SELECT a1.val_a, a2.val_a + 2 FROM A AS a1, A AS a2;
[(1,3),(2,4),(3,5),(4,6),(5,7),(6,8),(7,9),(8,10),(9,11),(10,12)]
```

# Array Attributes

- Store individual data values in array cells
- Consist of:
  - Name
  - Data type
  - Nullability (optional)
  - Default value (optional)
    - If unspecified, the system chooses a value:
      - If the attribute is nullable: null
      - Otherwise:
        - 0 for numeric types
        - empty string "" for string type
  - Compression type (optional): zlib or bzlib

| Datatype | Default value | Description |
|---|---|---|
| bool | false | Boolean value, true (1) or false (0) |
| char | \0 | Single ASCII character |
| datetime | 1970-01-01 00:00:00 | Date and time |
| datetimetz | 1970-01-01 00:00:00 -00:00 | Date and time with timezone offset. |
| double | 0 | Double-precision floating point number |
| float | 0 | Single-precision floating-point number |
| int8 | 0 | Signed 8-bit integer |
| int16 | 0 | Signed 16-bit integer |
| int32 | 0 | Signed 32-bit integer |
| int64 | 0 | Signed 64-bit integer |
| string | '' | Variable length character string, default is the empty string |
| uint8 | 0 | Unsigned 8-bit integer |
| uint16 | 0 | Unsigned 16-bit integer |
| uint32 | 0 | Unsigned 32-bit integer |
| uint64 | 0 | Unsigned 64-bit integer |

# Assignment

- Chose your unique problem domain
- For your selected problem domain, think about an application that uses SciDB for storing and querying your data.
- Submit a script with respective commands for creating, manipulating and querying SciDB arrays + explanatory comments

# References

- Reference Guide:
  - https://paradigm4.atlassian.net/wiki/spaces/scidb/pages/3395878935/SciDB+Reference+Guide
- Operators:
  - https://paradigm4.atlassian.net/wiki/spaces/scidb/pages/3395879191/SciDB+Operators
- Functions:
  - https://paradigm4.atlassian.net/wiki/spaces/scidb/pages/3395881879/SciDB+Functions