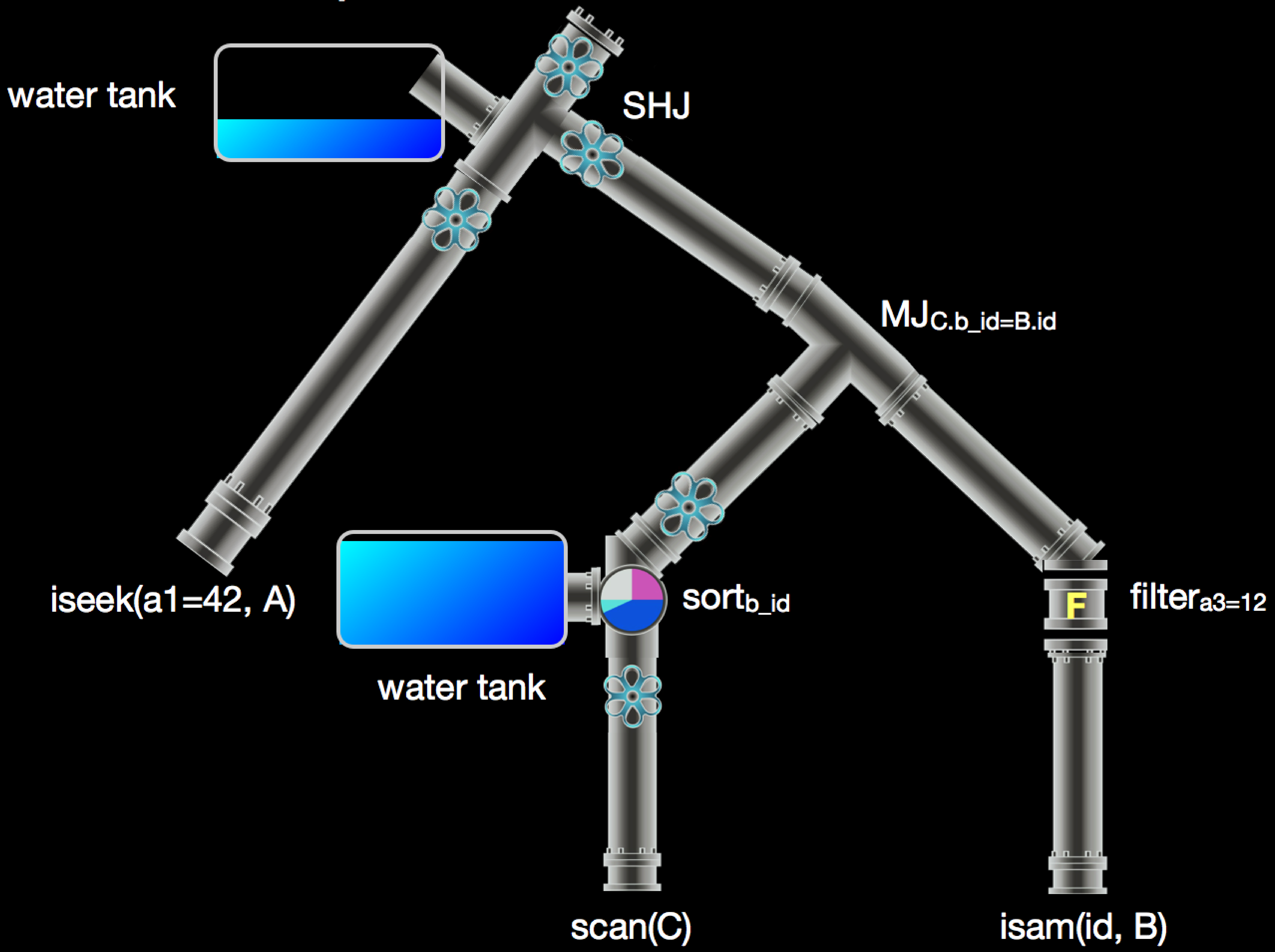
How to implement the Pipeline?



Operator Interface

```
interface Operator<Chunk>{

      void open();                     //initializes the operator

      Chunk next();                    //returns the next chunk of data

      void close();                    //performs cleanup work (if necessary)
}
```

~ constructor

~ destructor

C++: ~<class name>

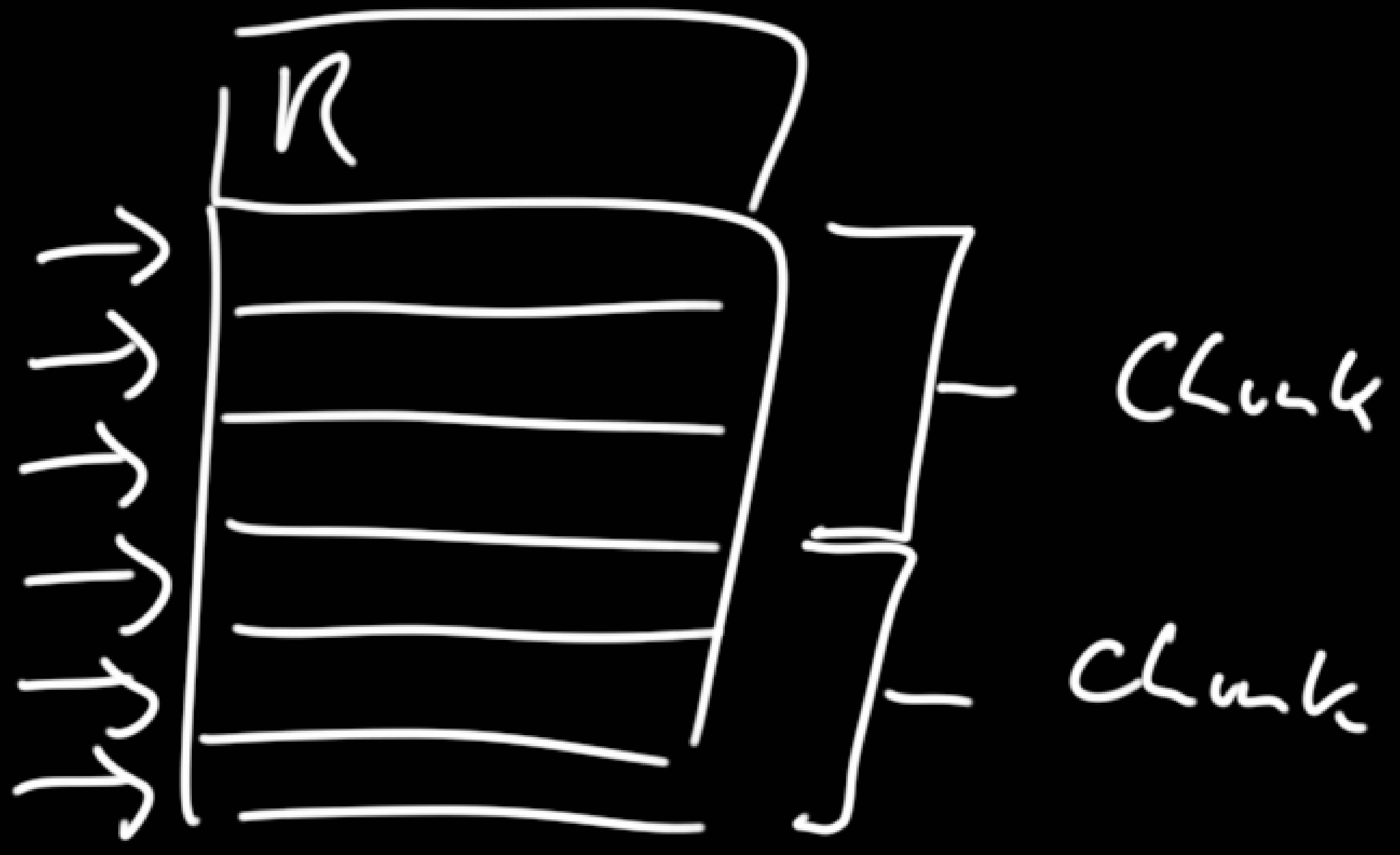
Java

Finalize()

What is a chunk?

disk-based DBMS -> Rows

main-memory DBMS -> Chunks

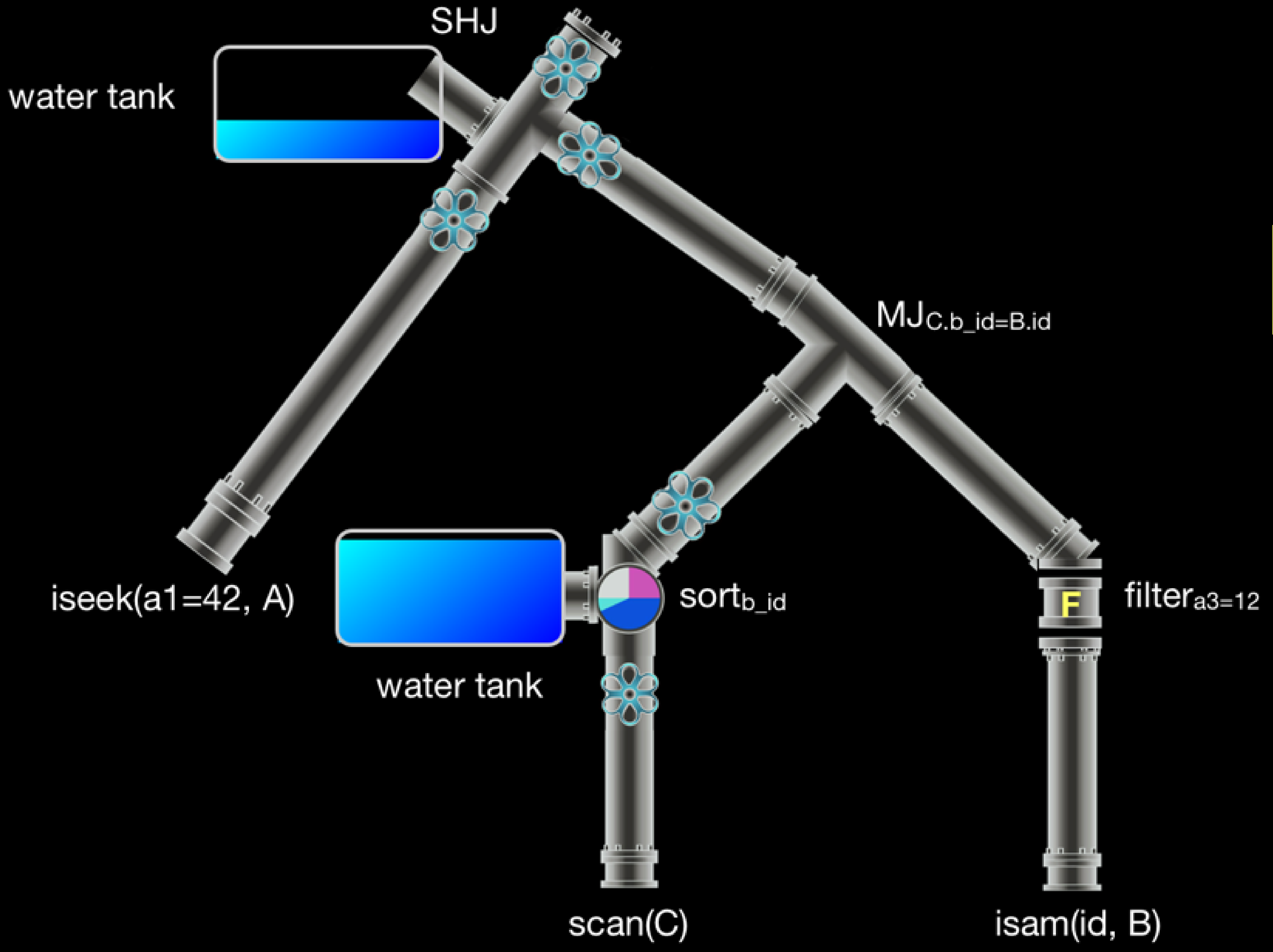


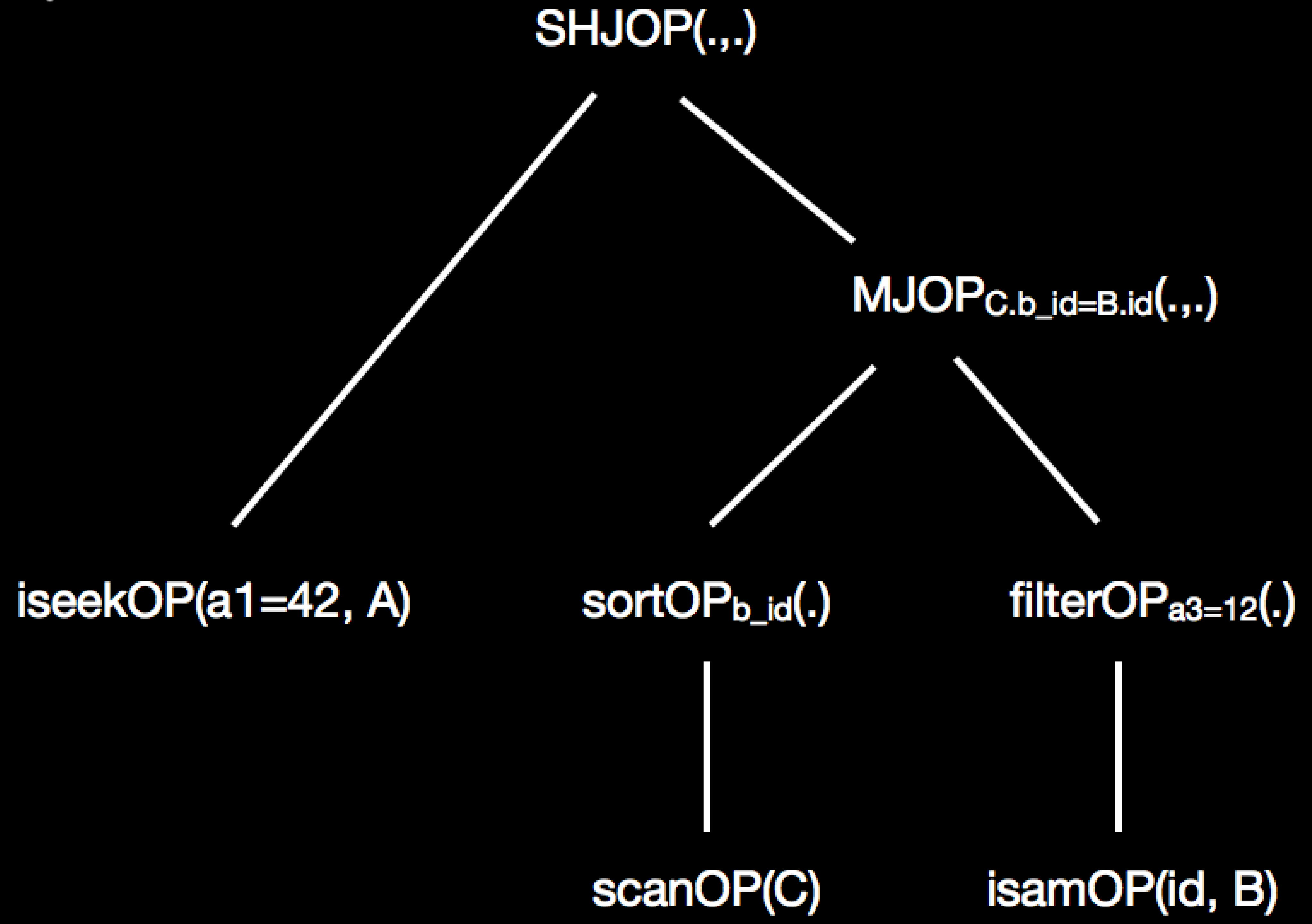
java.util.Iterator

```
public interface Iterator<Chunk>{
        //not available, could be done in constructor:
        //open();                                               //initializes the operator

        //additional method, does this make sense?:
        boolean hasNext();                                      //returns true if next chunk of data exists

        //OK:
        Chunk next();                                           //returns the next chunk of data

        //not available, could be done in destructor (if it existed in Java):
        //close();                                              //performs cleanup work (if necessary)

        //additional method, does this make sense?:
        void remove();                                          //removes last element returned from
                                                                //underlying collection
}
```

Example Translation with Operators

```
class Filter implements Operator {

}
```



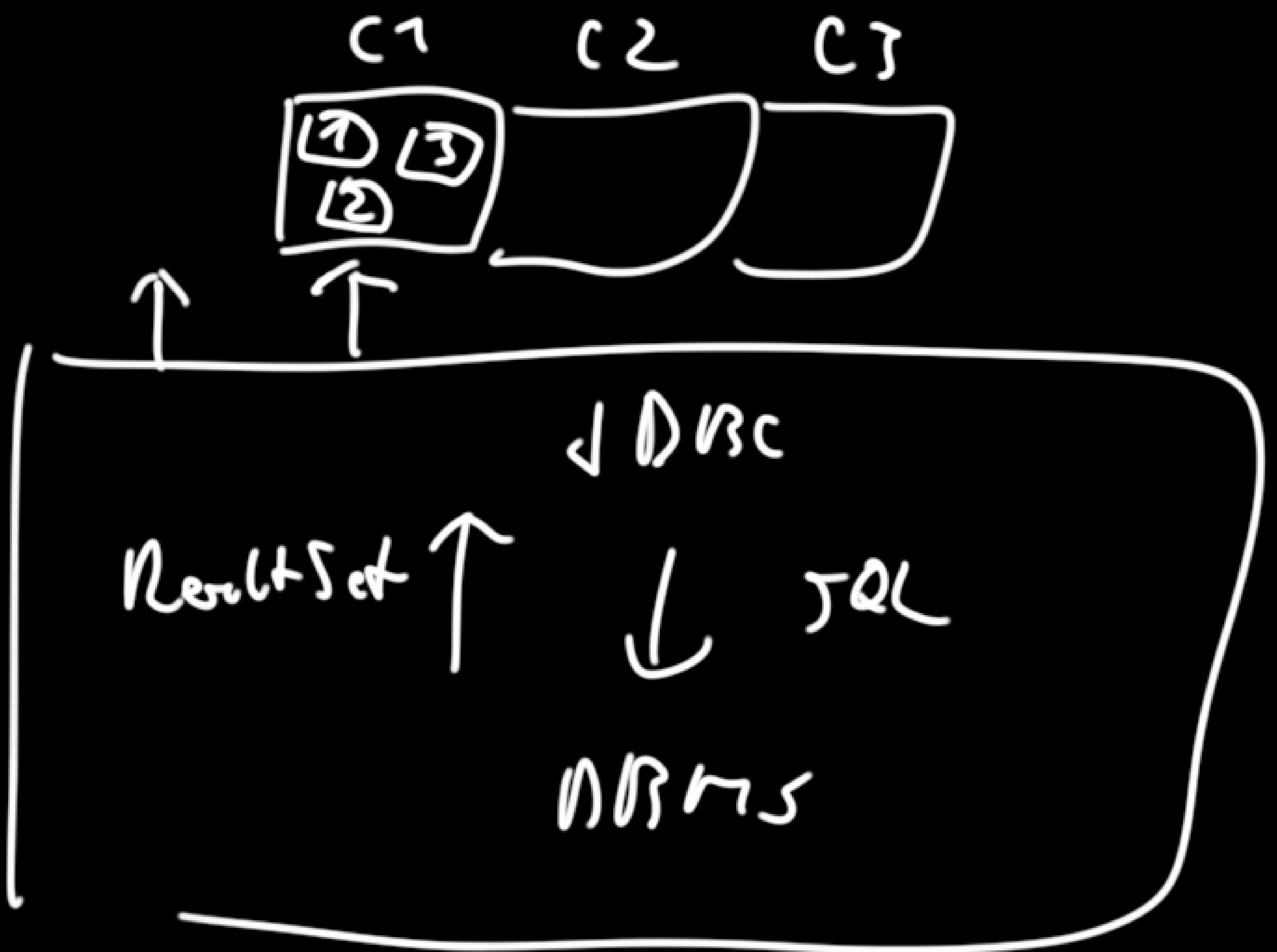


= textbook translation

General ResultSet-style Interface

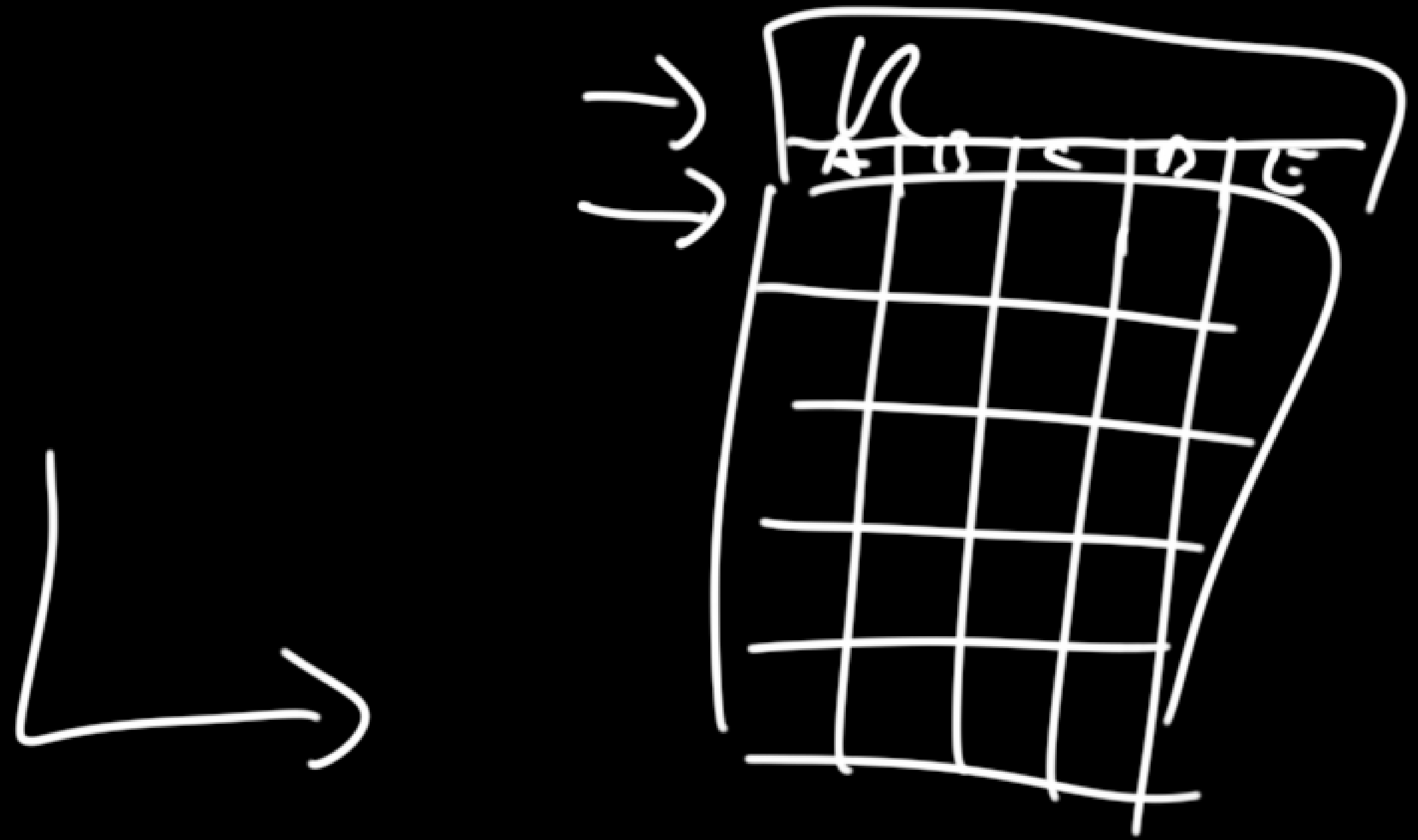
```
interface ResultSet<Chunk>{

    void open();                                  //initializes the operator

    boolean next();                               //moves pointer to next chunk
                                                  //returns true if valid pointer position

    void close();                                 //performs cleanup work (if necessary)

    SUB_CHUNK_1 getSUB_CHUNK_1(Key key);

    ...

    SUB_CHUNK_N getSUB_CHUNK_N(Key key);

}
```

e.g. JDBC



ResultSet-style Interface with Rows

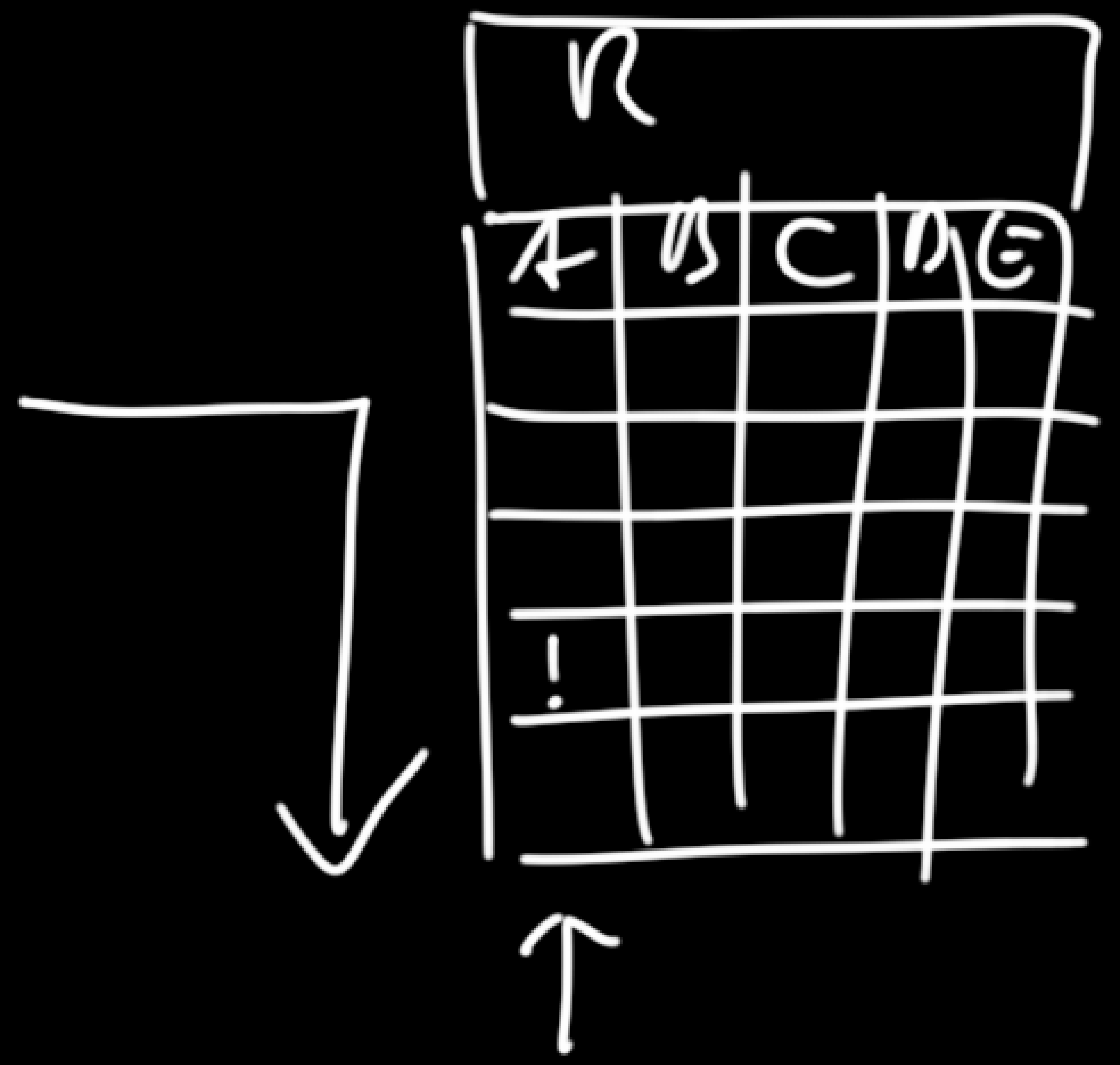
```
interface ResultSet<Row>{

    void open();                                   //initializes the operator

    boolean next();                                //moves pointer to next chunk
                                                   //returns true if valid pointer position

    void close();                                  //performs cleanup work (if necessary)

    String getString(int attributeIndex);

    ...

    int getInt(int attributeIndex);

}
```



getString(3);

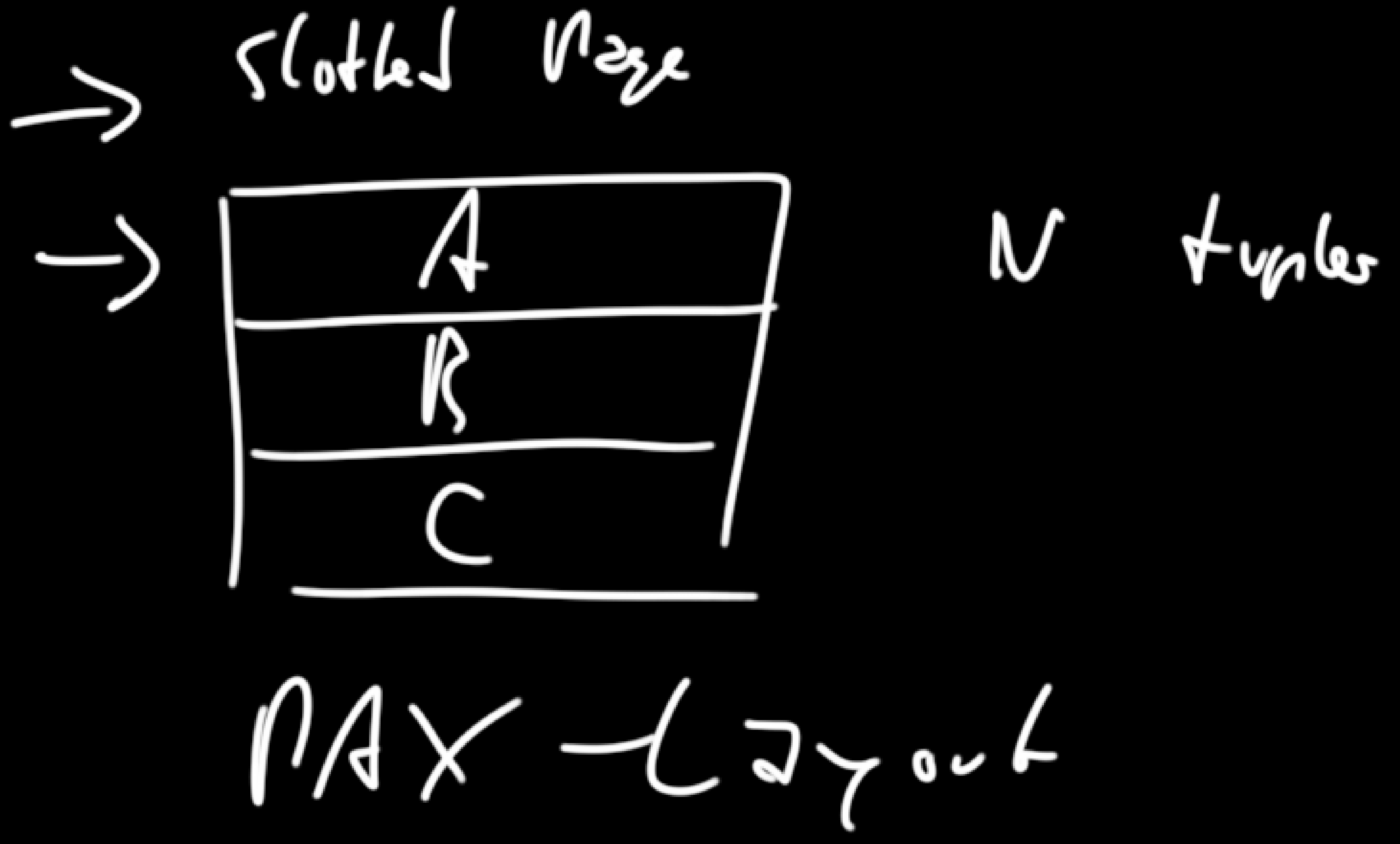
ResultSet-style Interface with Columns

```
interface ResultSet<Column>{

        void open();                                  //initializes the operator

        boolean next();                               //moves pointer to next chunk
                                                      //returns true if valid pointer position

        void close();                                 //performs cleanup work (if necessary)

        String getString(int rowIndex);

        ...

        int getInt(int rowIndex);

}
```



ResultSet-style Interface with Pages

```
interface ResultSet<Page>{

        void open();                          //initializes the operator

        boolean next();                       //moves pointer to next chunk
                                              //returns true if valid pointer position

        void close();                         //performs cleanup work (if necessary)

        Row getRow(int slot);
        Column getColumn(int attributeIndex)
        //i.e. caller can handle or process complete rows
        //OR:
        //gets a ResultSet for each row again

}
```



	Row	Column	Page	HP	VP
Operator	disk-based operator				
Iterator	disk-based operator				
ResultSet-style	JDBC				

↑

Copyrights and Credits

© iStock.com:

Horned_Rat