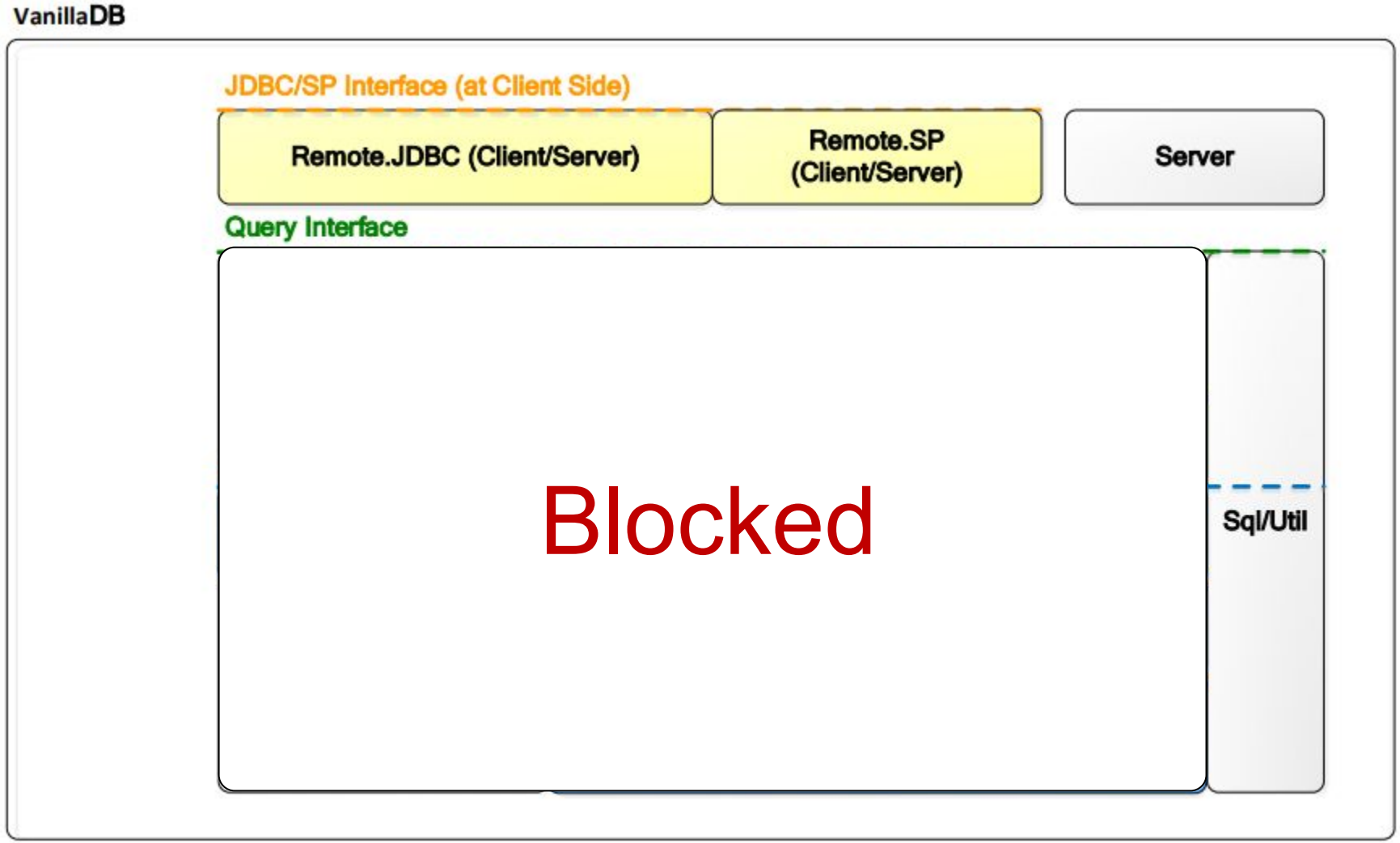
VanillaCore Walkthrough Part 2

Introduction to Databases

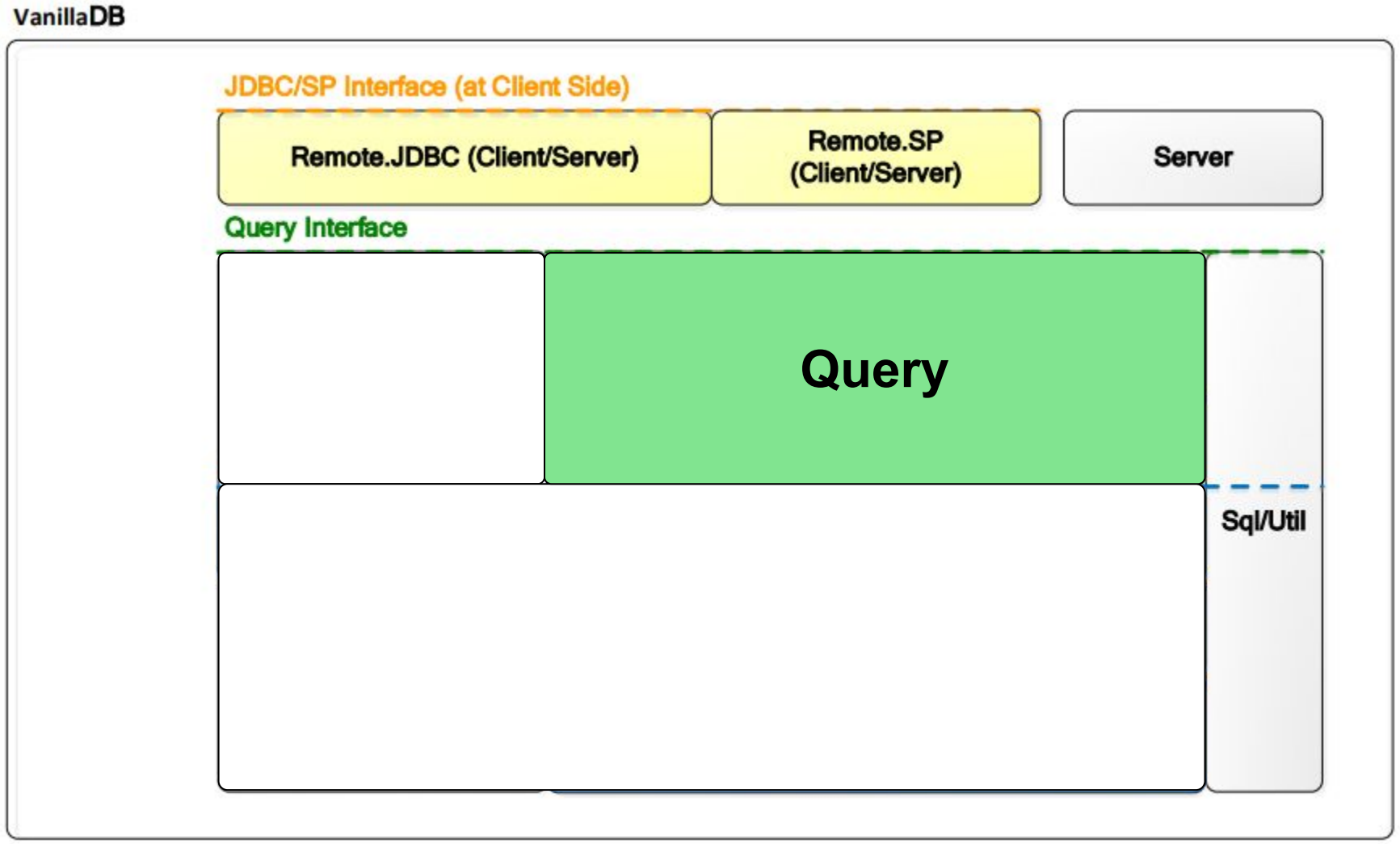
DataLab

CS, NTHU

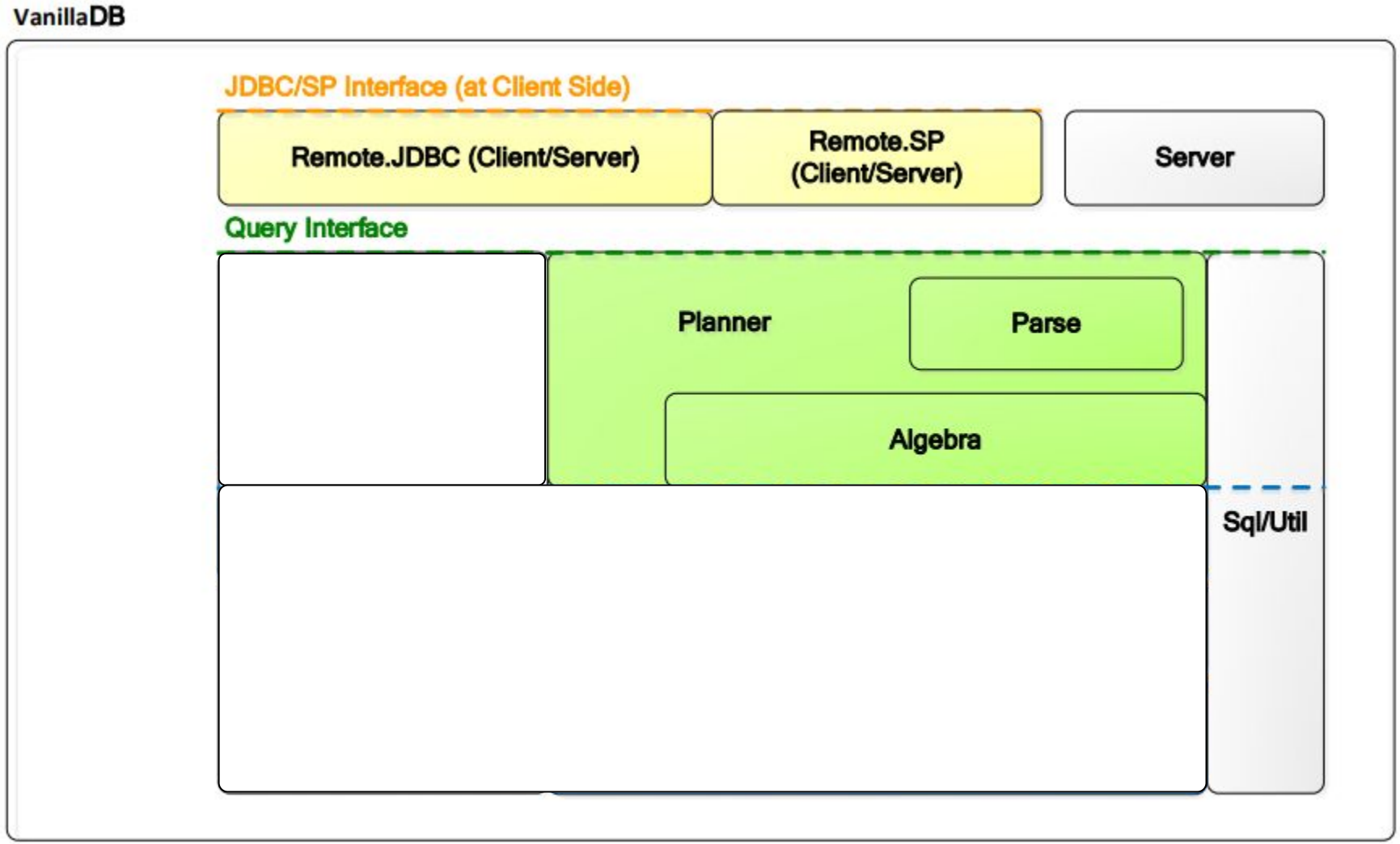
Last Time


VanillaDB
JDBC/SP Interface (at Client Side)
Remote.JDBC (Client/Server)
Remote.SP (Client/Server)
Server
Query Interface
Blocked
Sql/Util

This Time


VanillaDB
JDBC/SP Interface (at Client Side)
Remote.JDBC (Client/Server)
Remote.SP (Client/Server)
Server
Query Interface
Query
Sql/Util

This Time


VanillaDB
JDBC/SP Interface (at Client Side)
Remote.JDBC (Client/Server)
Remote.SP (Client/Server)
Server
Query Interface
Planner
Parse
Algebra
Sql/Util

Planner

1. Accepts a query string.
2. Creates a parser to parse the query string.
3. Verifies the query.
4. Generates a plan tree according to the query.

db24-assignment-3\core-patch\src\main\java\org\vanilladb\core\util\ConsoleSQLInterpreter.java

```java
public class ConsoleSQLInterpreter {
    private static Connection conn = null;

    public static void main(String[] args) {
        try {
            Driver d = new JdbcDriver();
            conn = d.connect("jdbc:vanilladb://localhost", null);

            Reader rdr = new InputStreamReader(System.in);
            BufferedReader br = new BufferedReader(rdr);

            while (true) {
                // process one line of input
                System.out.print("\nSQL> ");
                String cmd = br.readLine().trim();
                System.out.println();

                String [] str = cmd.split(" ");
                String cmdf = str[0].toUpperCase();

                if (cmd.startsWith("exit") || cmd.startsWith("EXIT"))
                    break;
                else if (cmdf.startsWith("SELECT")|| cmdf.startsWith("EXPLAIN"))
                    doQuery(cmd,cmdf);
                else
                    doUpdate(cmd);
            }
        } catch (Exception e) {
            e.printStackTrace();
```

```
private static void doQuery(String cmd,String cmdf) {
    try {
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(cmd);
        ResultSetMetaData md = rs.getMetaData();
        int numcols = md.getColumnCount();
        int totalwidth = 0;
```

db24-assignment-3\core-patch\src\main\java\org\vanilladb\core\remote\jdbc\RemoteStatementImpl.java

```java
public RemoteResultSet executeQuery(String qry) throws RemoteException {
    try {
        Transaction tx = rconn.getTransaction();
        Plan pln = VanillaDb.newPlanner().createQueryPlan(qry, tx);
        return new RemoteResultSetImpl(pln, rconn);
    } catch (RuntimeException e) {
        rconn.rollback();
        throw e;
    }
}
```

db24-assignment-3\core-patch\src\main\java\org\vanilladb\core\query\planner\Planner.java

```java
public Plan createQueryPlan(String qry, Transaction tx) {
    Parser parser = new Parser(qry);
    QueryData data = parser.queryCommand();
    Verifier.verifyQueryData(data, tx);
    return qPlanner.createPlan(data, tx);
}
```

db24-assignment-3\core-patch\src\main\java\org\vanilladb\core\query\planner\Planner.java

```
public Plan createQueryPlan(String qry, Transaction tx) {
    Parser parser = new Parser(qry);
    QueryData data = parser.queryCommand();
    Verifier.verifyQueryData(data, tx);
    return qPlanner.createPlan(data, tx);
}
```

Parser

Checking syntax.

Identifying the action and the parameters.

Lexer

Tokenizing.

Identifying keywords, IDs, values, delimiters.

db24-assignment-3\core-patch\src\main\java\org\vanilladb\core\query\parse\Parser.java

```
private Lexer lex;

public Parser(String s) {
    lex = new Lexer(s);
}
```

db24-assignment-3\core-patch\src\main\java\org\vanilladb\core\query\parse\Lexer.java

```
public Lexer(String s) {
    initKeywords();
    tok = new StreamTokenizer(new StringReader(s));
    tok.wordChars('_', '_');
    tok.ordinaryChar('.');
    /*
     * Tokens in TT_WORD type like ids and keywords are converted into lower
     * case.
     */
    tok.lowerCaseMode(true);
    nextToken();
}

private void initKeywords() {
    keywords = Arrays.asList("select", "from", "where", "and", "insert",
            "into", "values", "delete", "drop", "update", "set", "create", "table",
            "int", "double", "varchar", "view", "as", "index", "on",
            "long", "order", "by", "asc", "desc", "sum", "count", "avg",
            "min", "max", "distinct", "group", "add", "sub", "mul", "div",
            "using", "hash", "btree");
}
```

db24-assignment-3\core-patch\src\main\java\org\vanilladb\core\query\planner\Planner.java

```
public Plan createQueryPlan(String qry, Transaction tx) {
    Parser parser = new Parser(qry);
    QueryData data = parser.queryCommand();
    Verifier.verifyQueryData(data, tx);
    return qPlanner.createPlan(data, tx);
}
```

db24-assignment-3\core-patch\src\main\java\org\vanilladb\core\query\parse\Parser.java

```java
public QueryData queryCommand() {
    lex.eatKeyword("select");
    ProjectList projs = projectList();
    lex.eatKeyword("from");
    Set<String> tables = idSet();
    Predicate pred = new Predicate();
    if (lex.matchKeyword("where")) {
        lex.eatKeyword("where");
        pred = predicate();
    }
    /*
     * Non-null group-by fields (but may be empty) if "group by" appears or
     * there is an aggFn in the project list.
     */
    Set<String> groupFields = null;
    if (lex.matchKeyword("group")) {
        lex.eatKeyword("group");
        lex.eatKeyword("by");
        groupFields = idSet();
    }
    if (groupFields == null && projs.aggregationFns() != null)
        groupFields = new HashSet<String>();
    // Need to preserve the order of sort fields
    List<String> sortFields = null;
    List<Integer> sortDirs = null;
    if (lex.matchKeyword("order")) {
        lex.eatKeyword("order");
        lex.eatKeyword("by");
        // neither null nor empty if "sort by" appears
        SortList sortList = sortList();
        sortFields = sortList.fieldList();
        sortDirs = sortList.directionList();
    }
    return new QueryData(projs.asStringSet(), tables, pred,
            groupFields, projs.aggregationFns(), sortFields, sortDirs);
}
```

db24-assignment-3\core-patch\src\main\java\org\vanilladb\core\query\planner\Planner.java

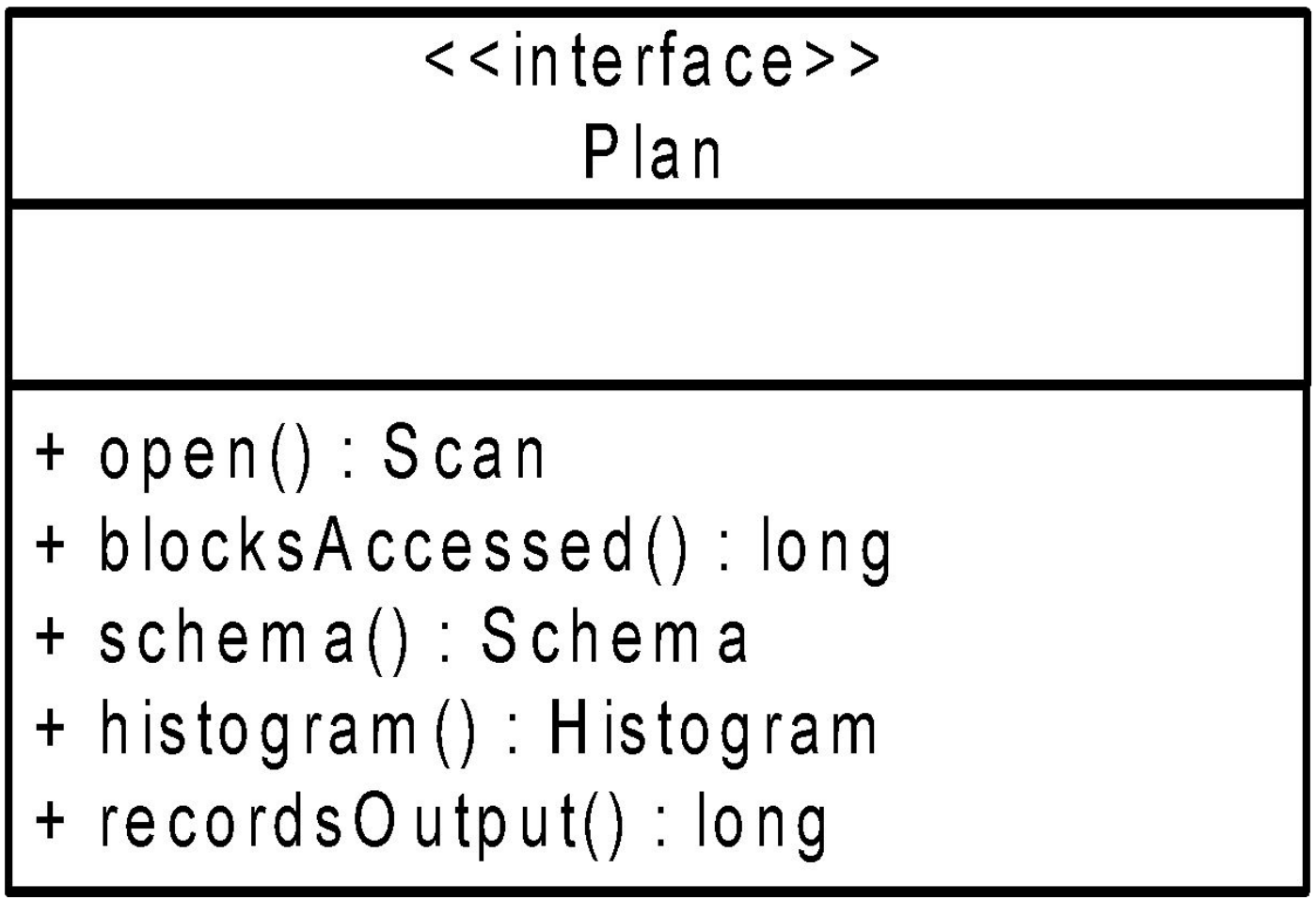
```
public Plan createQueryPlan(String qry, Transaction tx) {
    Parser parser = new Parser(qry);
    QueryData data = parser.queryCommand();
    Verifier.verifyQueryData(data, tx);
    return qPlanner.createPlan(data, tx);
}
```

db24-assignment-3\core-patch\src\main\java\org\vanilladb\core\query\planner\Verifier.java

```
// examine the table name
for (String tblName : data.tables()) {
    String viewdef = VanillaDb.catalogMgr().getViewDef(tblName, tx);
    if (viewdef == null) {
        TableInfo ti = VanillaDb.catalogMgr().getTableInfo(tblName, tx);
        if (ti == null)
            throw new BadSemanticException("table " + tblName
                    + " does not exist");
        schs.add(ti.schema());
    } else {
        Parser parser = new Parser(viewdef);
        views.add(parser.queryCommand());
    }
}
// examine the projecting field name
for (String fldName : data.projectFields()) {
    boolean isValid = verifyField(schs, views, fldName);
    if (!isValid && data.aggregationFn() != null)
        for (AggregationFn aggFn : data.aggregationFn())
            if (fldName.compareTo(aggFn.fieldName()) == 0) {
                isValid = true;
                break;
            }
    if (!isValid)
        throw new BadSemanticException("field " + fldName
                + " does not exist");
}
```

```
public Plan createQueryPlan(String qry, Transaction tx) {
    Parser parser = new Parser(qry);
    QueryData data = parser.queryCommand();
    Verifier.verifyQueryData(data, tx);
    return qPlanner.createPlan(data, tx);
}
```

Plan

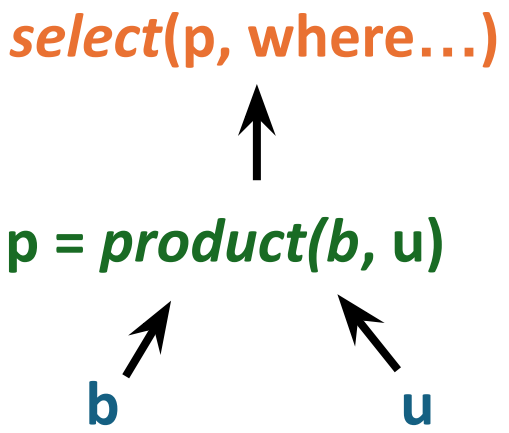

<<interface>>
Plan
+ open() : Scan
+ blocksAccessed() : long
+ schema() : Schema
+ histogram() : Histogram
+ recordsOutput() : long

db24-assignment-3\core-patch\src\main\java\org\vanilladb\core\query\planner\BasicQueryPlanner.java

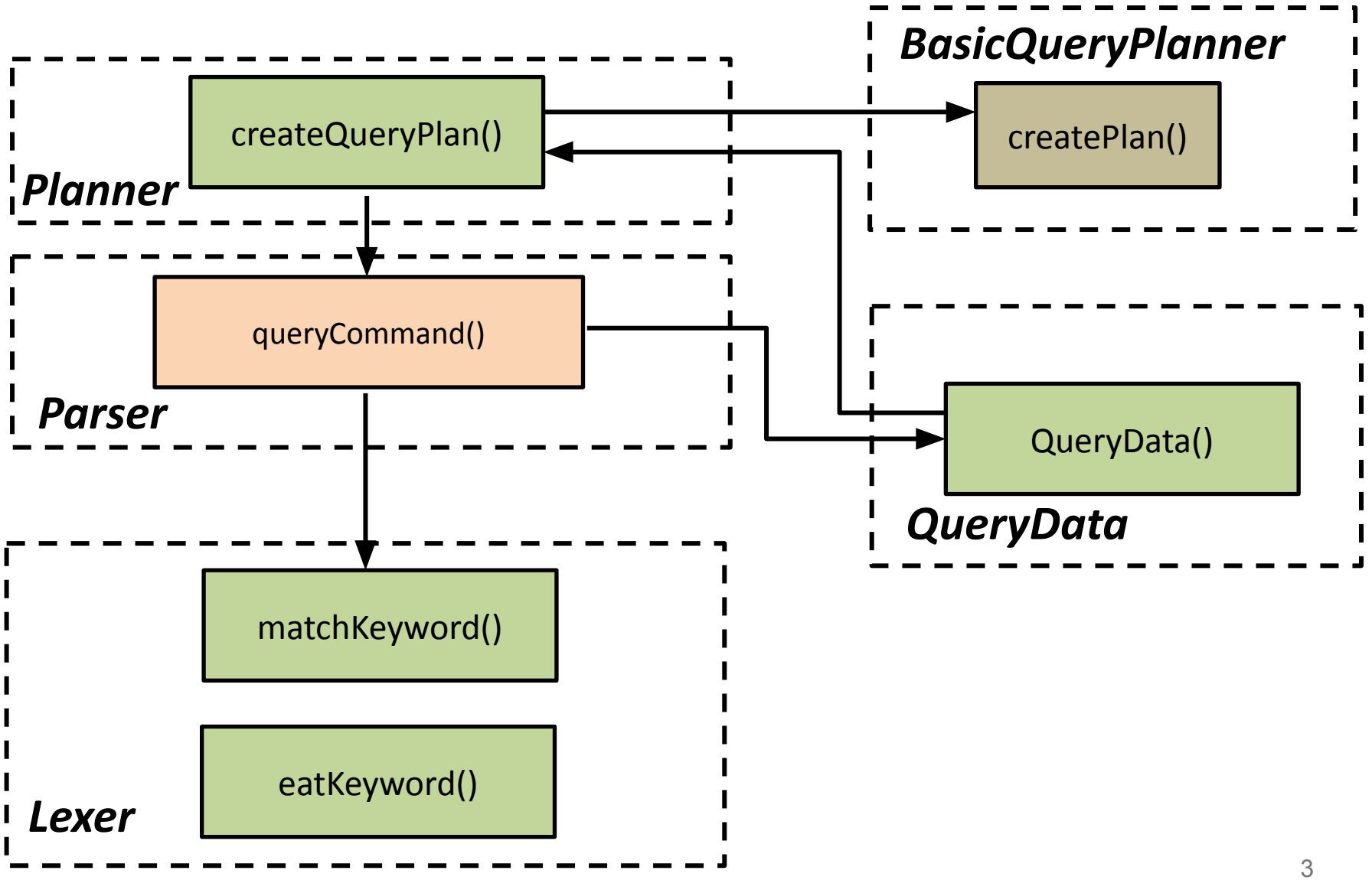
```java
public class BasicQueryPlanner implements QueryPlanner {

    /**
     * Creates a query plan as follows. It first takes the product of all tables
     * and views; it then selects on the predicate; and finally it projects on
     * the field list.
     */
    @Override
    public Plan createPlan(QueryData data, Transaction tx) {
        // Step 1: Create a plan for each mentioned table or view
        List<Plan> plans = new ArrayList<Plan>();
        for (String tblname : data.tables()) {
            String viewdef = VanillaDb.catalogMgr().getViewDef(tblname, tx);
            if (viewdef != null)
                plans.add(VanillaDb.newPlanner().createQueryPlan(viewdef, tx));
            else
                plans.add(new TablePlan(tblname, tx));
        }
        // Step 2: Create the product of all table plans
        Plan p = plans.remove(0);
        for (Plan nextplan : plans)
            p = new ProductPlan(p, nextplan);
        // Step 3: Add a selection plan for the predicate
        p = new SelectPlan(p, data.pred());
        // Step 4: Add a group-by plan if specified
        if (data.groupFields() != null) {
            p = new GroupByPlan(p, data.groupFields(), data.aggregationFn(), tx);
        }
        // Step 5: Project onto the specified fields
        p = new ProjectPlan(p, data.projectFields());
        // Step 6: Add a sort plan if specified
        if (data.sortFields() != null)
            p = new SortPlan(p, data.sortFields(), data.sortDirections(), tx);
        return p;
    }
}
```

Using a Query Plan

```
VanillaDb.init("studentdb");
Transaction tx = VanillaDb.txMgr().newTransaction(
    Connection.TRANSACTION_SERIALIZABLE, true);

Plan pb = new TablePlan("b", tx);
Plan pu = new TablePlan("u", tx);
Plan pp = new ProductPlan(pb, pu);
Predicate pred = new Predicate("...");
Plan sp = new SelectPlan(pp, pred);

sp.blockAccessed(); // estimate #blocks accessed
```



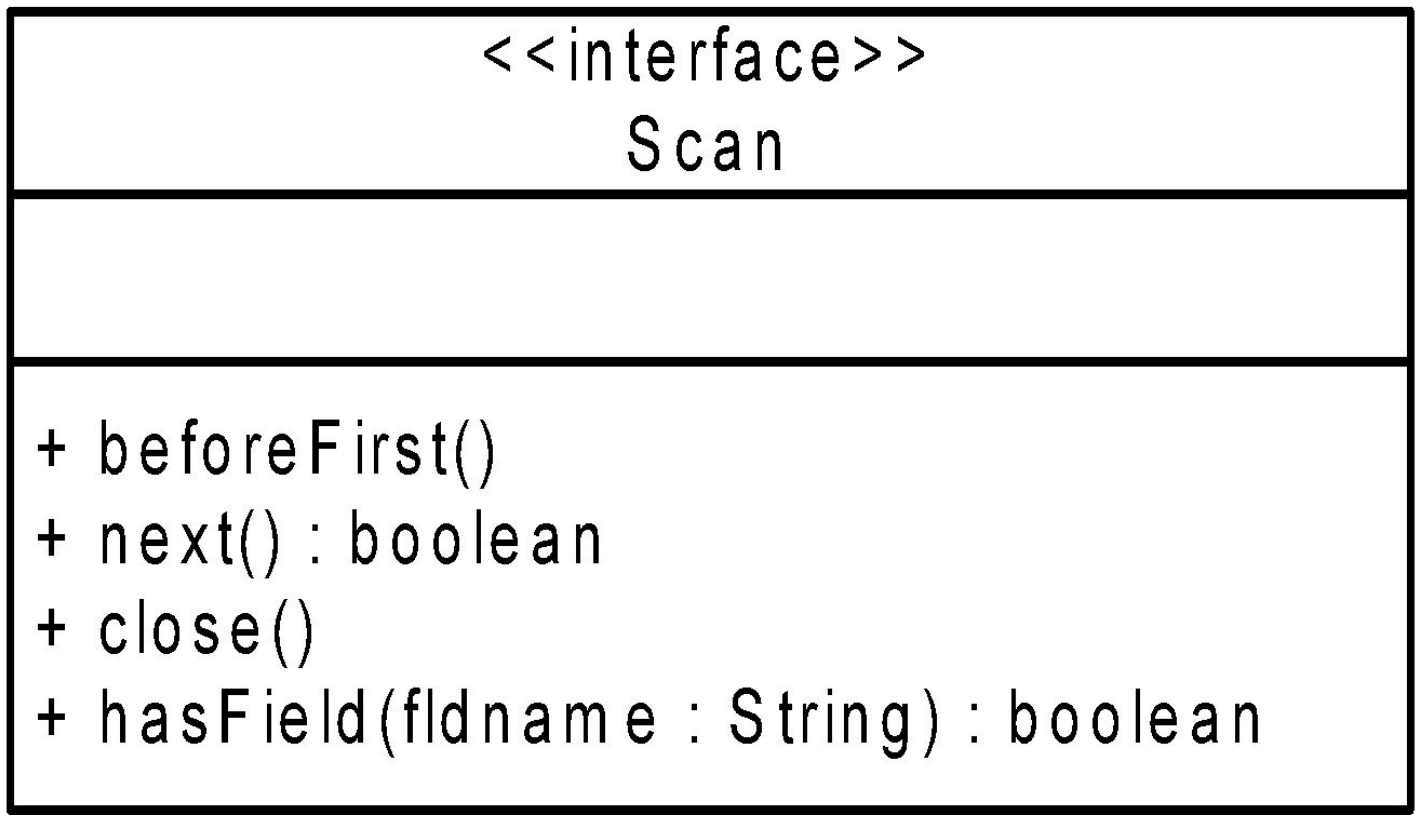
Overview


BasicQueryPlanner
createQueryPlan()
createPlan()
Planner
queryCommand()
Parser
QueryData()
QueryData
matchKeyword()
eatKeyword()
Lexer

db24-assignment-3\core-patch\src\main\java\org\vanilladb\core\remote\jdbc\RemoteStatementImpl.java

```java
public RemoteResultSet executeQuery(String qry) throws RemoteException {
    try {
        Transaction tx = rconn.getTransaction();
        Plan pln = VanillaDb.newPlanner().createQueryPlan(qry, tx);
        return new RemoteResultSetImpl(pln, rconn);
    } catch (RuntimeException e) {
        rconn.rollback();
        throw e;
    }
}
```

Scan


<<interface>>
Scan
+ beforeFirst()
+ next() : boolean
+ close()
+ hasField(fldname : String) : boolean

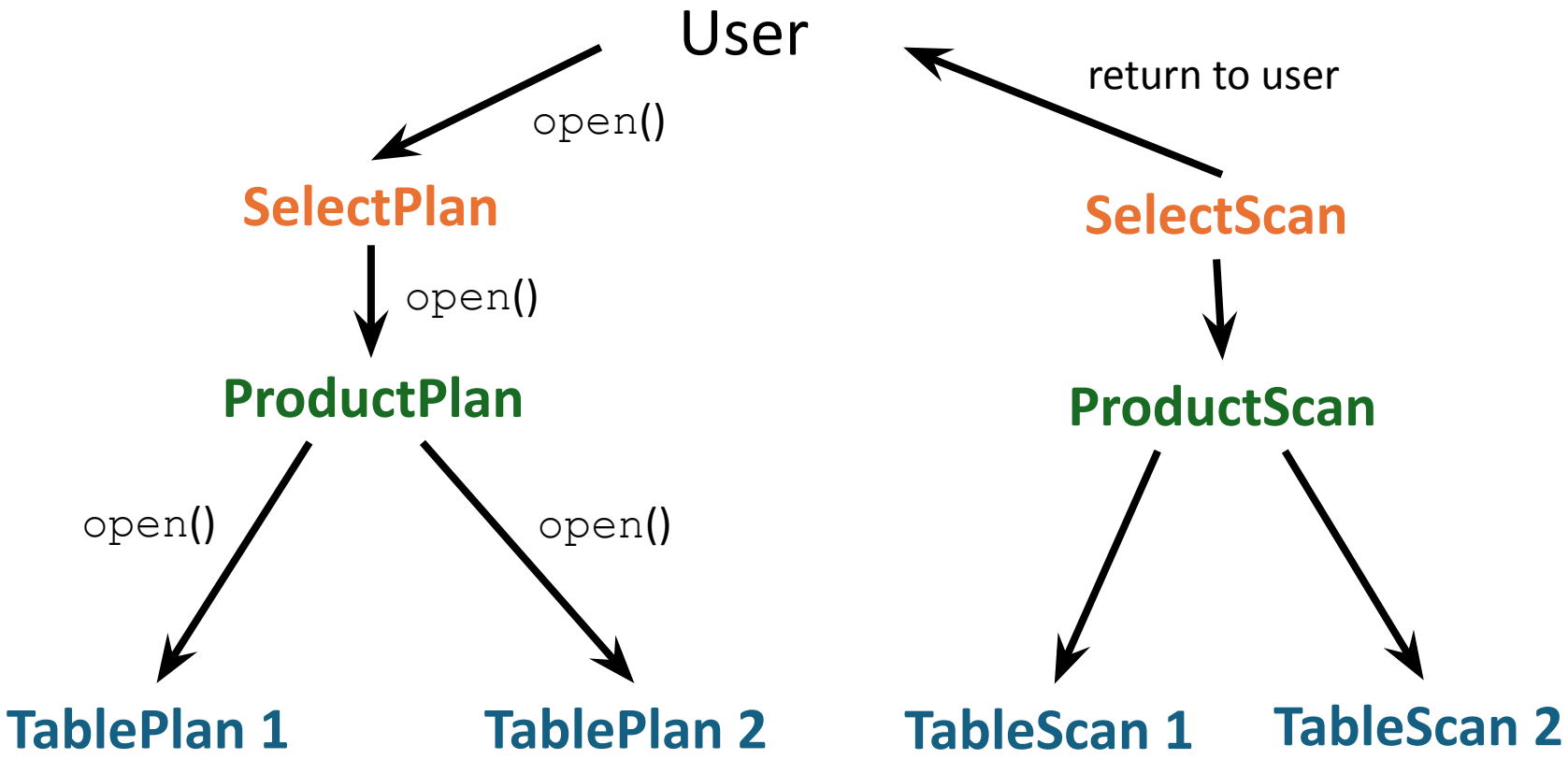
db24-assignment-3\core-patch\src\main\java\org\vanilladb\core\remote\jdbc\RemoteResultSetImpl.java

```
class RemoteResultSetImpl extends UnicastRemoteObject implements
        RemoteResultSet {
    private Scan s;
    private Schema schema;
    private RemoteConnectionImpl rconn;

    /**
     * Creates a RemoteResultSet object. The specified plan is opened, and the
     * scan is saved.
     *
     * @param plan
     *            the query plan
     * @param rconn
     *
     * @throws RemoteException
     */
    public RemoteResultSetImpl(Plan plan, RemoteConnectionImpl rconn)
            throws RemoteException {
        s = plan.open();
        schema = plan.schema();
        this.rconn = rconn;
    }
```

db24-assignment-3\core-patch\src\main\java\org\vanilladb\core\util\ConsoleSQLInterpreter.java

```
rs.beforeFirst();
// print records
while (rs.next()) {
    for (int i = 1; i <= numcols; i++) {
        String fldname = md.getColumnName(i);
        int fldtype = md.getColumnType(i);
        String fmt = "%" + md.getColumnDisplaySize(i);
        if (fldtype == Types.INTEGER)
            System.out.format(fmt + "d", rs.getInt(fldname));
        else if (fldtype == Types.BIGINT)
            System.out.format(fmt + "d", rs.getLong(fldname));
        else if (fldtype == Types.DOUBLE)
            System.out.format(fmt + "f", rs.getDouble(fldname));
        else
            System.out.format(fmt + "s", rs.getString(fldname));
    }
    System.out.println();
}
```

open()


User
open()
return to user
SelectPlan
SelectScan
open()
ProductPlan
ProductScan
open()
open()
TablePlan 1
TablePlan 2
TableScan 1
TableScan 2

Example

project(s, select blog_id)

↓ beforeFirst()

select(p, where name = 'Picachu'
and author_id = user_id)

↓ beforeFirst()

product(b, u)

↓ beforeFirst()

```
SELECT blog_id FROM b, u
        WHERE name = "Picachu"
        AND author_id = user_id;
```

b

blog_id	url	created	author_id
33981	…	2009/10/31	729
33982	…	2012/11/15	730
41770	…	2012/10/20	729

u

user_id	name	balance
729	Steven Sinofsky	10,235
730	Picachu	NULL

Example

project(s, select blog_id)

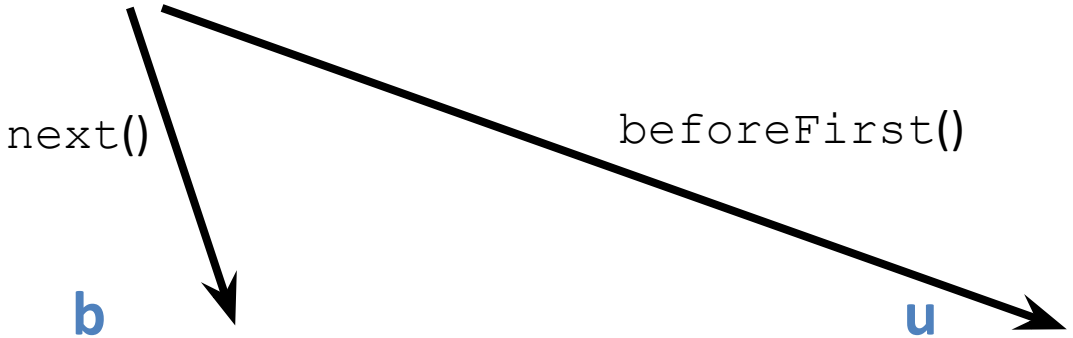
↓ beforeFirst()

select(p, where name = 'Picachu'
and author_id = user_id)

↓ beforeFirst()

product(b, u)

```
SELECT blog_id FROM b, u
        WHERE name = "Picachu"
        AND author_id = user_id;
```



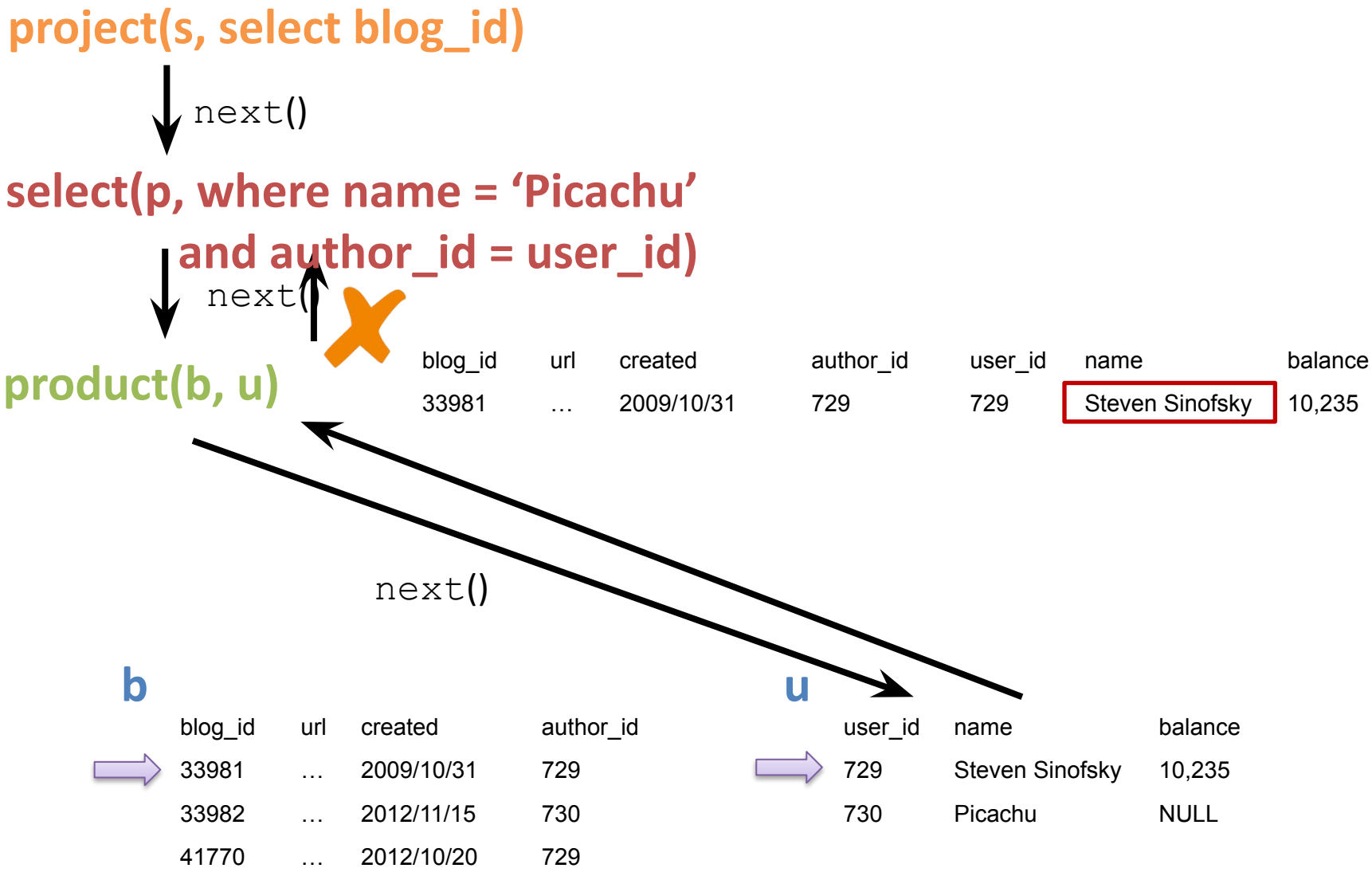
b

blog_id	url	created	author_id
33981	…	2009/10/31	729
33982	…	2012/11/15	730
41770	…	2012/10/20	729

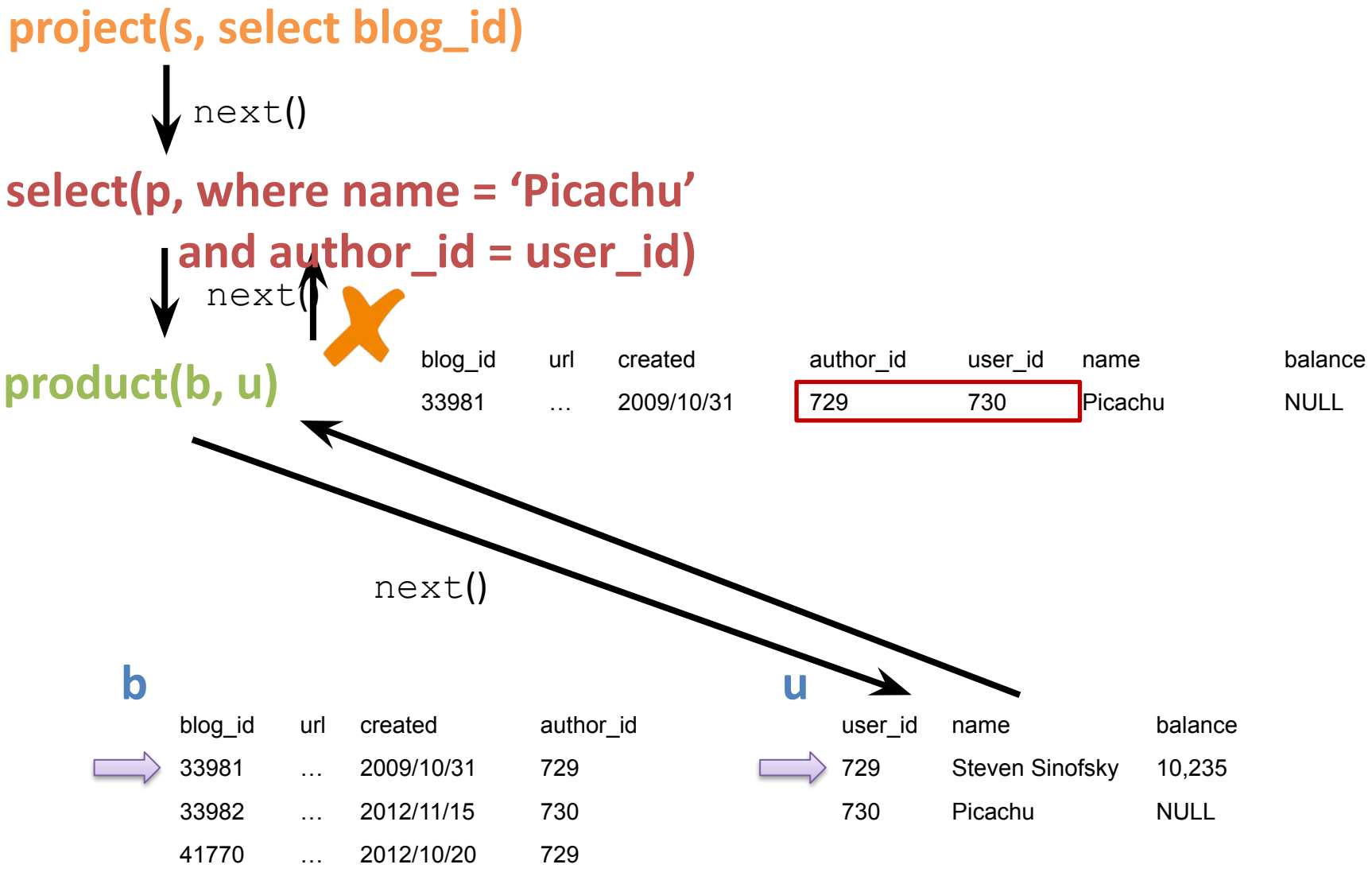
u

user_id	name	balance
729	Steven Sinofsky	10,235
730	Picachu	NULL

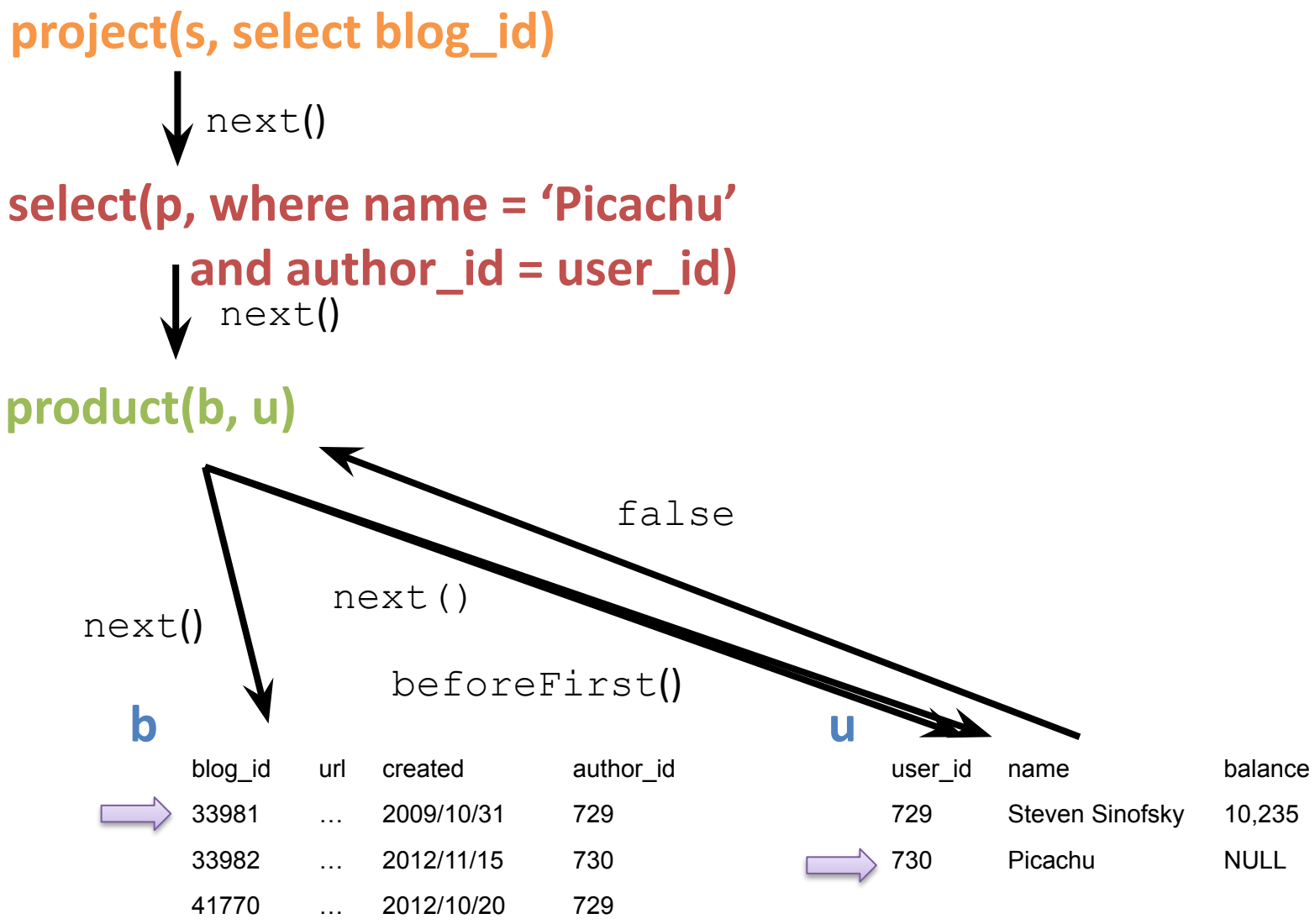
Example


project(s, select blog_id)
next()
select(p, where name = 'Picachu'
and author_id = user_id)
next()
product(b, u)
blog_id url created author_id user_id name balance
33981 … 2009/10/31 729 729 Steven Sinofsky 10,235
next()
b
blog_id url created author_id
33981 … 2009/10/31 729
33982 … 2012/11/15 730
41770 … 2012/10/20 729
u
user_id name balance
729 Steven Sinofsky 10,235
730 Picachu NULL

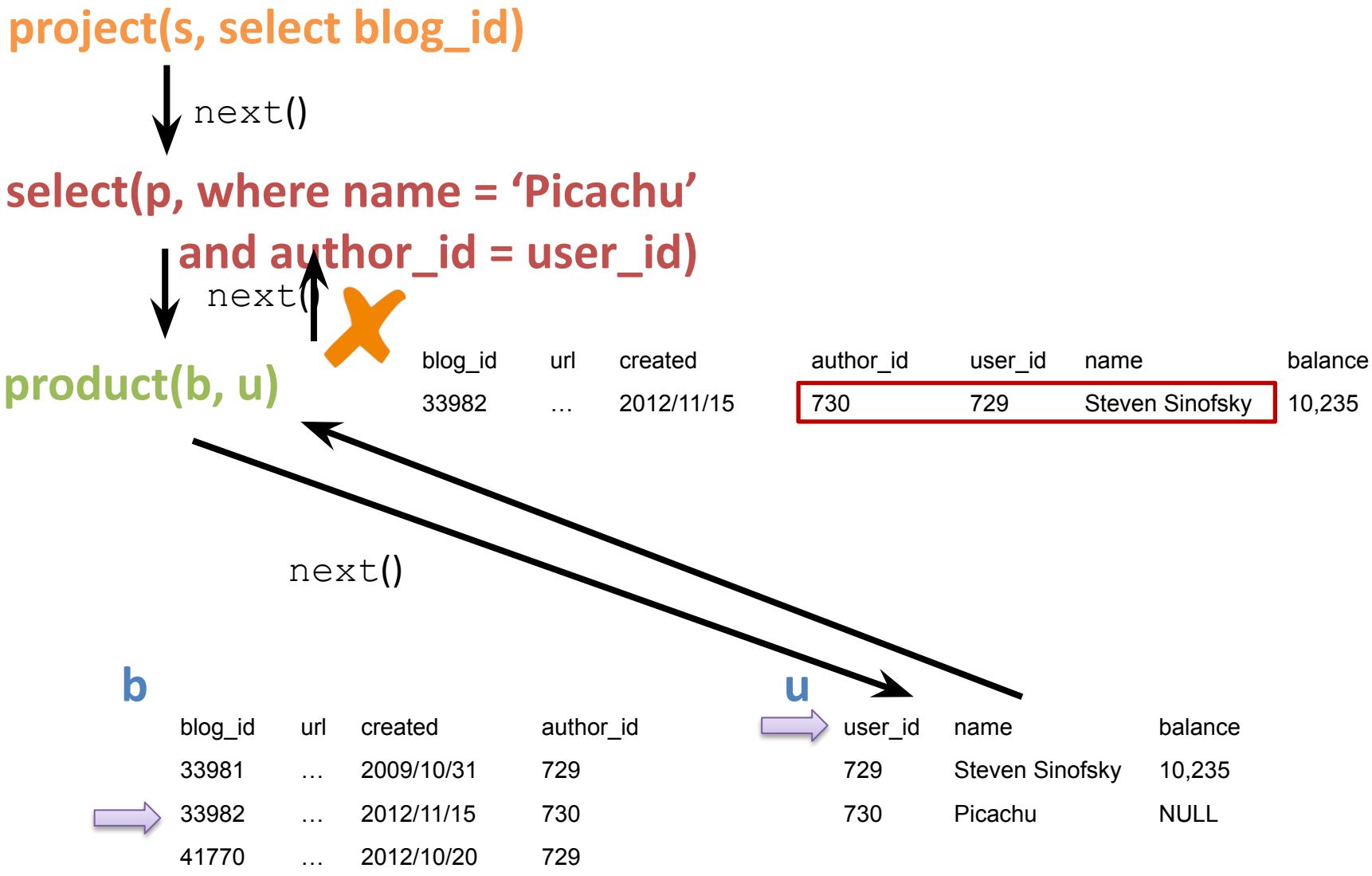
Example


project(s, select blog_id)
next()
select(p, where name = 'Picachu'
and author_id = user_id)
next()
product(b, u)
blog_id url created author_id user_id name balance
33981 … 2009/10/31 729 730 Picachu NULL
next()
b
blog_id url created author_id
33981 … 2009/10/31 729
33982 … 2012/11/15 730
41770 … 2012/10/20 729
u
user_id name balance
729 Steven Sinofsky 10,235
730 Picachu NULL

Example


project(s, select blog_id)
next()
select(p, where name = 'Picachu'
and author_id = user_id)
next()
product(b, u)
false
next()
next()
beforeFirst()
b
u
blog_id url created author_id
33981 … 2009/10/31 729
33982 … 2012/11/15 730
41770 … 2012/10/20 729
user_id name balance
729 Steven Sinofsky 10,235
730 Picachu NULL

Example

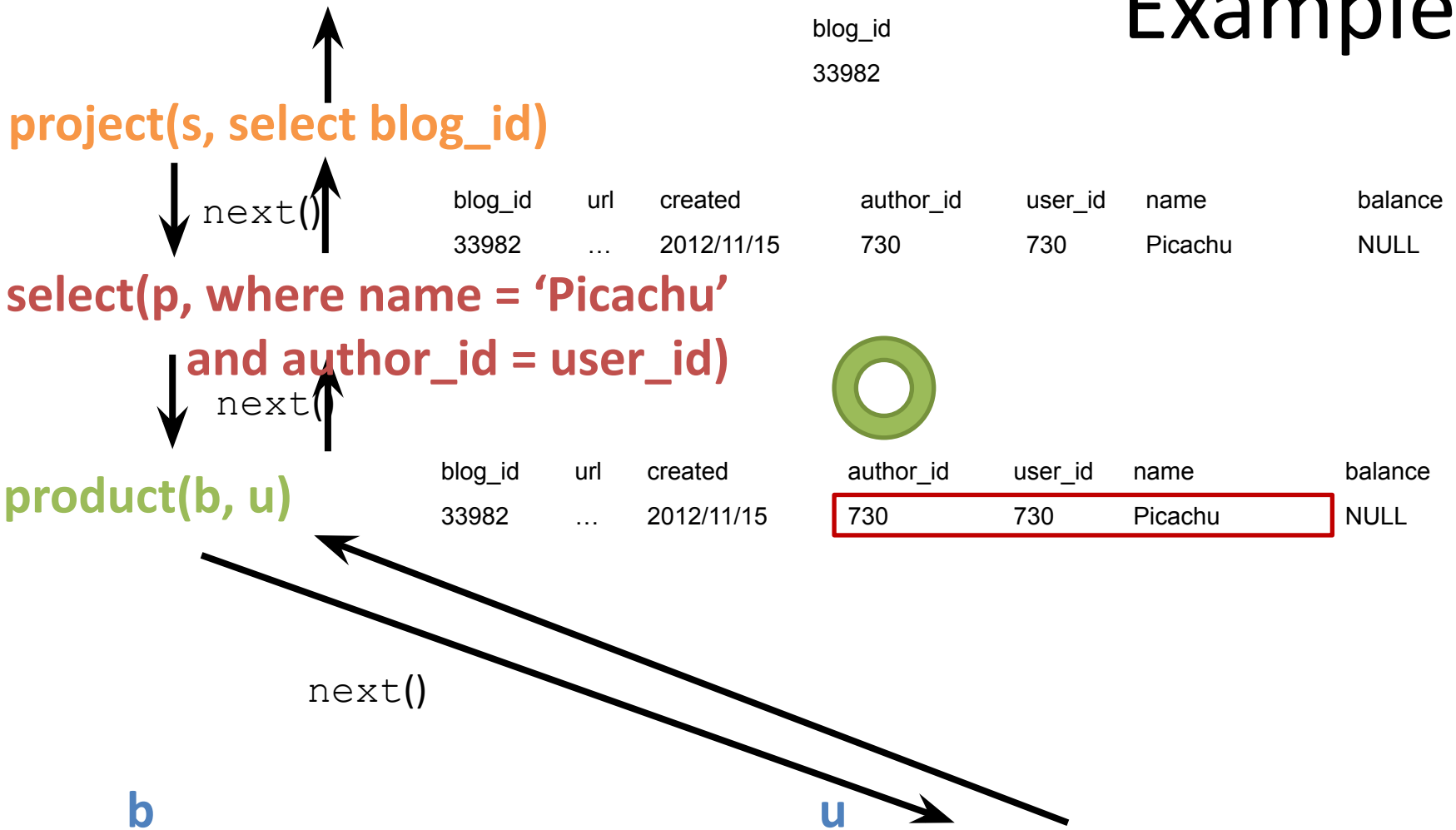


blog_id	url	created	author_id	user_id	name	balance
33982	…	2012/11/15	730	729	Steven Sinofsky	10,235

blog_id	url	created	author_id
33981	…	2009/10/31	729
33982	…	2012/11/15	730
41770	…	2012/10/20	729

user_id	name	balance
729	Steven Sinofsky	10,235
730	Picachu	NULL

Example


project(s, select blog_id)
next()
select(p, where name = 'Picachu'
and author_id = user_id)
next()
product(b, u)
next()
blog_id
33982
blog_id url created author_id user_id name balance
33982 … 2012/11/15 730 730 Picachu NULL
blog_id url created author_id user_id name balance
33982 … 2012/11/15 730 730 Picachu NULL
b
blog_id url created author_id
33981 … 2009/10/31 729
33982 … 2012/11/15 730
41770 … 2012/10/20 729
u
user_id name balance
729 Steven Sinofsky 10,235
730 Picachu NULL