

VanillaDB

A Tutorial-Oriented RDBMS

Yu-Shan Lin @ COSCUP 2017

About Me

- Yu-Shan Lin (林玉山)
- Net name: SLMT
- PhD student in Datalab, CS, NTHU
- Research topic: DBMS
- <http://www.slmt.tw>



Why Should You Learn The Internal of Databases Systems ?

10 Richest People in the World

Rank	Name	Owned Company	Net Worth
1	Bill Gates		\$ 85.2 billion
2	Warren Buffett		\$ 77.2 billion
3	Jeff Bezos		\$ 73.1 billion
4	Amancio Ortega		\$ 68.5 billion
5	Mark Zuckerberg		\$ 58.5 billion
6	Carlos Slim Helú		\$ 50.7 billion
7	Charles Koch		\$ 47.9 billion
7	David Koch		\$ 47.9 billion
9	Larry Ellison		\$ 45.3 billion
10	Ingvar Kamprad		\$ 43 billion

Source: Business Insider

10 Richest People in the World

Rank	Name	Owned Company	Net Worth
1	Bill Gates	Microsoft	\$ 85.2 billion
2	Warren Buffett	Berkshire Hathaway	\$ 77.2 billion
3	Jeff Bezos	Amazon.com	\$ 73.1 billion
4	Amancio Ortega	Inditex	\$ 68.5 billion
5	Mark Zuckerberg	Facebook	\$ 58.5 billion
6	Carlos Slim Helú	Grupo Carso	\$ 50.7 billion
7	Charles Koch	Koch Industries	\$ 47.9 billion
7	David Koch	Koch Industries	\$ 47.9 billion
9	Larry Ellison	Oracle	\$ 45.3 billion
10	Ingvar Kamprad	IKEA	\$ 43 billion

Source: Business Insider

10 Richest People in the World

Rank	Name	Owned Company	Net Worth
1	Bill Gates	Microsoft	\$ 85.2 billion
2	Warren Buffett	Berkshire Hathaway	\$ 77.2 billion
3	Jeff Bezos	Amazon.com	\$ 73.1 billion
4	Amancio Ortega	Inditex	\$ 68.5 billion
5	Mark Zuckerberg	Facebook	\$ 58.5 billion
6	Carlos Slim Helú	Grupo Carso	\$ 50.7 billion
7	Charles Koch	Koch Industries	\$ 47.9 billion
7	David Koch	Koch Industries	\$ 47.9 billion
9	Larry Ellison	Oracle	\$ 45.3 billion
10	Ingvar Kamprad	IKEA	\$ 43 billion

Source: Business Insider

10 Richest People in the World

Rank	Name	Owned Company	Net Worth
1	Bill Gates	Microsoft	\$ 85.2 billion
2	Warren Buffett	Berkshire Hathaway	\$ 77.2 billion
Database systems play very important roles in these companies !			
6	Carlos Slim Helu	Grupo Carso	\$ 50.7 billion
7	Charles Koch	Koch Industries	\$ 47.9 billion
7	David Koch	Koch Industries	\$ 47.9 billion
9	Larry Ellison	Oracle	\$ 45.3 billion
10	Ingvar Kamprad	IKEA	\$ 43 billion

Source: Business Insider

**“I don’t own a company.
Why should I care ? ”**

How About Being A **Database Administrator (DBA)** ?

**The Median Pay for a DBA
is 84,950 USD / year in 2016 !**

Source: Bureau of Labor Statistics

84,950 USD \doteq 2,564,528 TWD
(2017/8/4)

How Does It Help DBA ?

- Understanding how a DBMS works helps a DBA know what he/she needs to consider while tuning it.
 - Buffer Pool ?
 - Join Buffer & Sort Buffer ?
 - Locks ?

“I have already had a coding job.”

Well... learning this can also help you in other fields, too !

How Does Learning DB Help You in Other Fields ?

- A database management system (DBMS) is a extremely **complicated** and **highly optimized** system.
- Learning the internal of such systems help you know...
 - how to read the code of such systems.
 - what you need to consider while altering such systems.
 - optimization techniques.

**“If you are good enough to write code
for a DBMS, then you can write code on
almost anything else.”**

- Andy Pavlo @ CMU 15-721

**Or, You May Be Just a Person
Who Wants to Know Everything**

Like Me !!

You came to the right place !!

Just Curious

- How is a SQL processed in a DBMS ? (explained later)
- Why the data are still correct even when lots of user accesses the same data at the same time ?
- Why can a DB recover to a normal state after it crashes ?

Outline

- Motivations
- Introduction to RDBMS
- A Day of a Query in VanillaDB
- Some Challenges of Developing a RDBMS
- VanillaDB Project
- Our Next Step ?

Outline

- Motivations
- Introduction to RDBMS
- A Day of a Query in VanillaDB
- Some Challenges of Developing a RDBMS
- VanillaDB Project
- Our Next Step ?

**What Is Difference Between
a File System and a DBMS ?**

Advantages of a Database System

- It answers queries fast.
- Queries (from multiple users) can execute concurrently without affecting each other.
- It recovers from crash.

What Is a RDBMS ?

- RDBMS => Relational Database Management System
 - Not just a database.
 - Including a “management system”.
- So, what is “relational” ?

Relational Models

Attribute, Field

id	name	balance
1	Red	3300
2	Blue	2200
3	Green	4500

Schema

Row,
Record,
Tuple

Relation

Why Using Relations ?

- Easy to manage on disks.
- Easy to understand.
- Can be applied very complex queries (SQL).

SQL

id	name	balance
1	Red	3300
2	Blue	2200
3	Green	4500

```
SELECT balance FROM account WHERE name = "Red";
```

balance
3000

Transactions



```
BEGIN TRANSACTION;
```

```
UPDATE account SET balance = balance - 100 WHERE name = "Red";
```

```
UPDATE account SET balance = balance + 100 WHERE name = "Blue";
```

```
COMMIT TRANSACTION;
```

ACID 

ACID

- Atomicity
- Consistency
- Isolation
 - Isolation Levels
- Durability

A - Atomicity

- All or nothing

```
BEGIN TRANSACTION;  
UPDATE account ...  
UPDATE account ...  
COMMIT TRANSACTION;
```

None



```
BEGIN TRANSACTION;  
UPDATE account ...  
UPDATE account ...  
COMMIT TRANSACTION;
```

Half



```
BEGIN TRANSACTION;  
UPDATE account ...  
UPDATE account ...  
COMMIT TRANSACTION;
```

All



C - Consistency

- The database must be consistent after transactions committed.

User Specified Rule: $\text{Sum}(\text{balance}) = 10000$

```
BEGIN TRANSACTION;  
UPDATE account ...  
UPDATE account ...  
COMMIT TRANSACTION;
```

In Progress

$\text{Sum}(\text{balance}) = 9900$



Inconsistent !!

All other transactions should not see this.

I - Isolation

The result of concurrently executing {T1, T2, T3}.

equals to

The result of executing T1 -> T2 -> T3.

(or in other orders)

It is called **Serializable** Isolation.

Isolation Levels

Level	Dirty Reads	Non-Repeatable Reads	Phantoms
Read Uncommitted	May Happen	May Happen	May Happen
Read Committed	Safe	May Happen	May Happen
Repeatable Read	Safe	Safe	May Happen
Serializable	Safe	Safe	Safe

MySQL's InnoDB uses **Repeatable Read** as default.

BTW, Do We Really Need Serializable ?

Actually, most people only use **READ COMMITTED** ! [1]

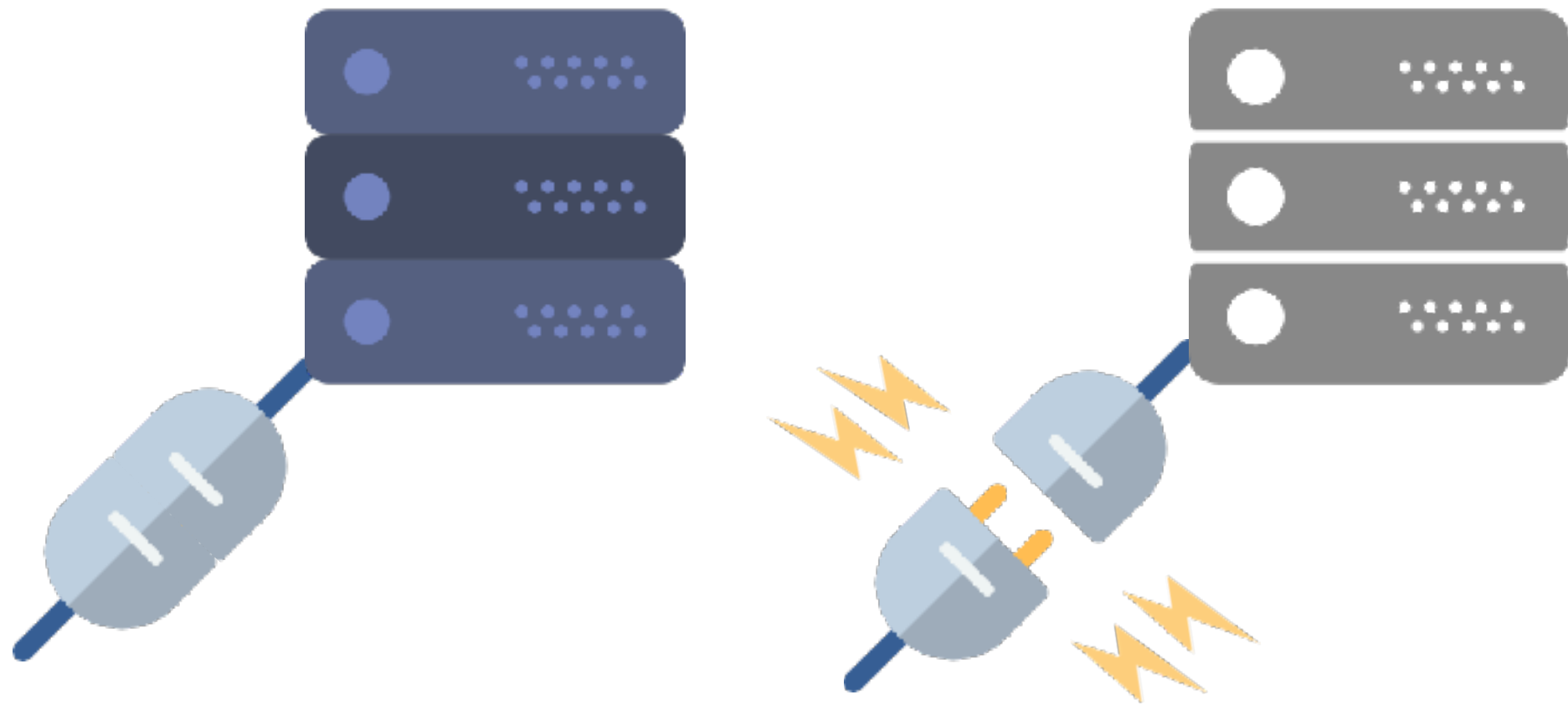
But, low isolation levels have **security risks**. [2]

[1] “What Are We Doing With Our Lives? Nobody Cares About Our Research on Concurrency Control” in SIGMOD’17

[2] “ACIDRain: Concurrency-Related Attacks on Database-Backed Web Applications” in SIGMOD’17

D - Durability

- The committed results must be saved.



The data must be persistent even the system crashes !!

Outline

- Motivations
- Introduction to RDBMS
- **A Day of a Query in VanillaDB**
- Some Challenges of Developing a RDBMS
- VanillaDB Project
- Our Next Step ?

Example: Stock Accounts

id	name	balance
1	Red	3300
2	Blue	2200
3	Green	4500

account

buyer	stock_id	amount	time
1	103	50	7/19
1	297	300	8/1
1	31	230	8/5
2	45	40	8/7
3	24	100	9/2

stock_history

Query: Find a guy with money > 3000 and buying at least one stock recently ($\geq 9/1$).

Example Query

id	name	balance
1	Red	3300
2	Blue	2200
3	Green	4500

account

buyer	stock_id	amount	time
1	103	50	7/19
1	297	300	8/1
1	31	230	8/5
2	45	40	8/7
3	24	100	9/2

stock_history

```
SELECT name FROM account, stock_history WHERE
  id = buyer AND balance > 3000 AND time >= 9/1;
```

```

Connection conn = null;
try {
    // Connect to the database server
    Driver d = new JdbcDriver();
    conn = d.connect("jdbc:vanilladb://localhost", null);
    conn.setAutoCommit(false); // Using transaction

    // Execute the query
    Statement stmt = conn.createStatement();
    String qry = "SELECT name FROM account, stock_history WHERE"
        + "id = buyer AND balance > 3000 AND time >= 9/1;";
    ResultSet rs = stmt.executeQuery(qry);

    // Loop through the result set
    rs.beforeFirst();
    while (rs.next()) {
        String sName = rs.getString("name");
        System.out.println(sName);
    }
    rs.close();

    // Commit the transaction
    conn.commit();

} catch (SQLException e) {
    e.printStackTrace();
} finally {
    // Closes the connection
    ...
}

```

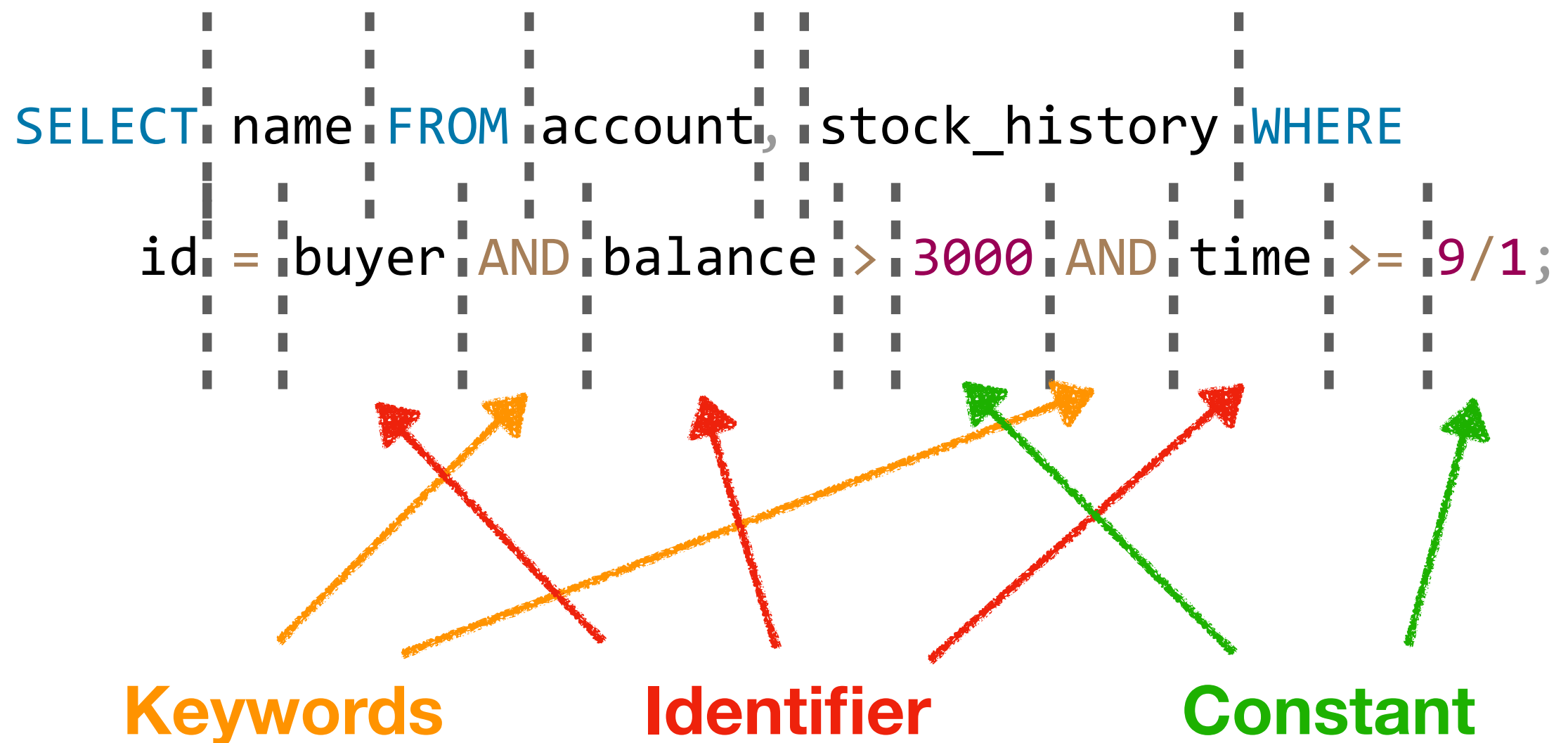
A day of a query

A Day of A Query

1. Tokenizing & analyzing the SQL.
2. Parsing the SQL.
3. Planning (selecting a plan tree).
4. Creating a record scan.
5. Retrieving and returning records one by one.
6. Close the scan.

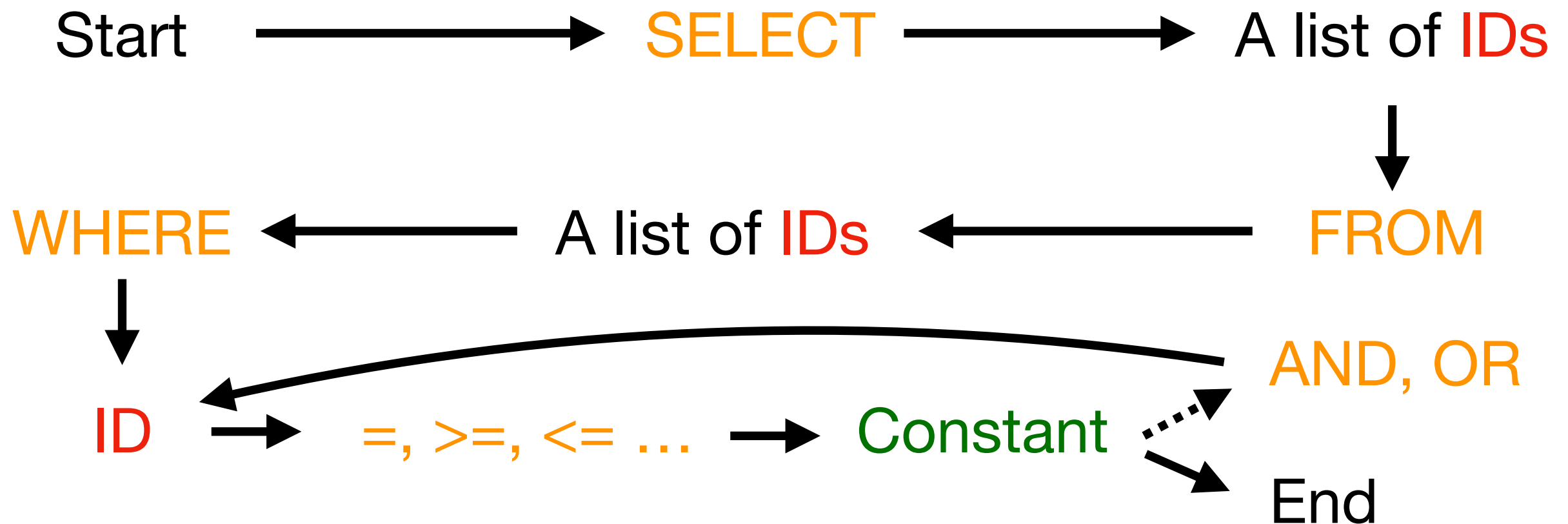
Lexical Analysis

Tokenization



Parsing

Continuous checks using predefined rules



Plan Trees

```
SELECT name FROM account, stock_history WHERE  
id = buyer AND balance > 3000 AND time >= 9/1;
```

π Projection {'name'}



σ Selection {id = buyer & balance > 3000 & time > 9/1}



\times Cross Product

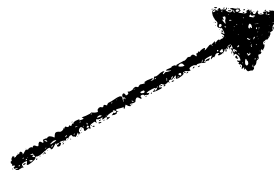


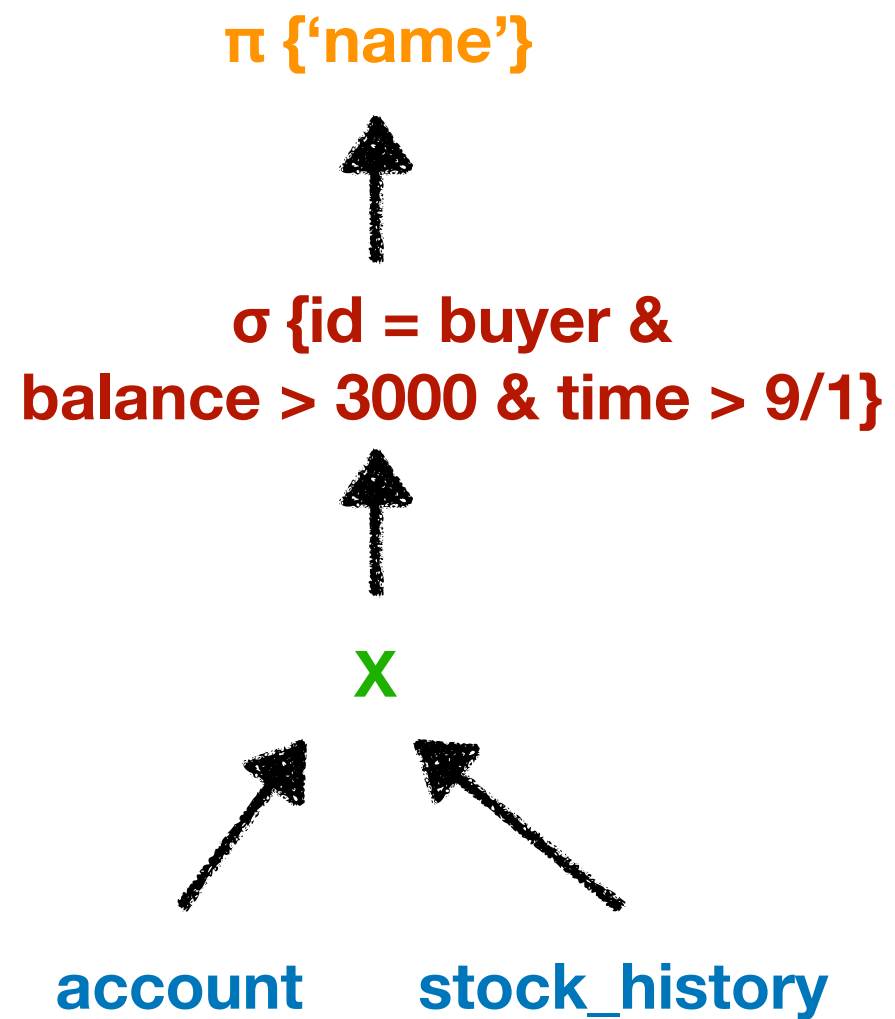
Table {account}



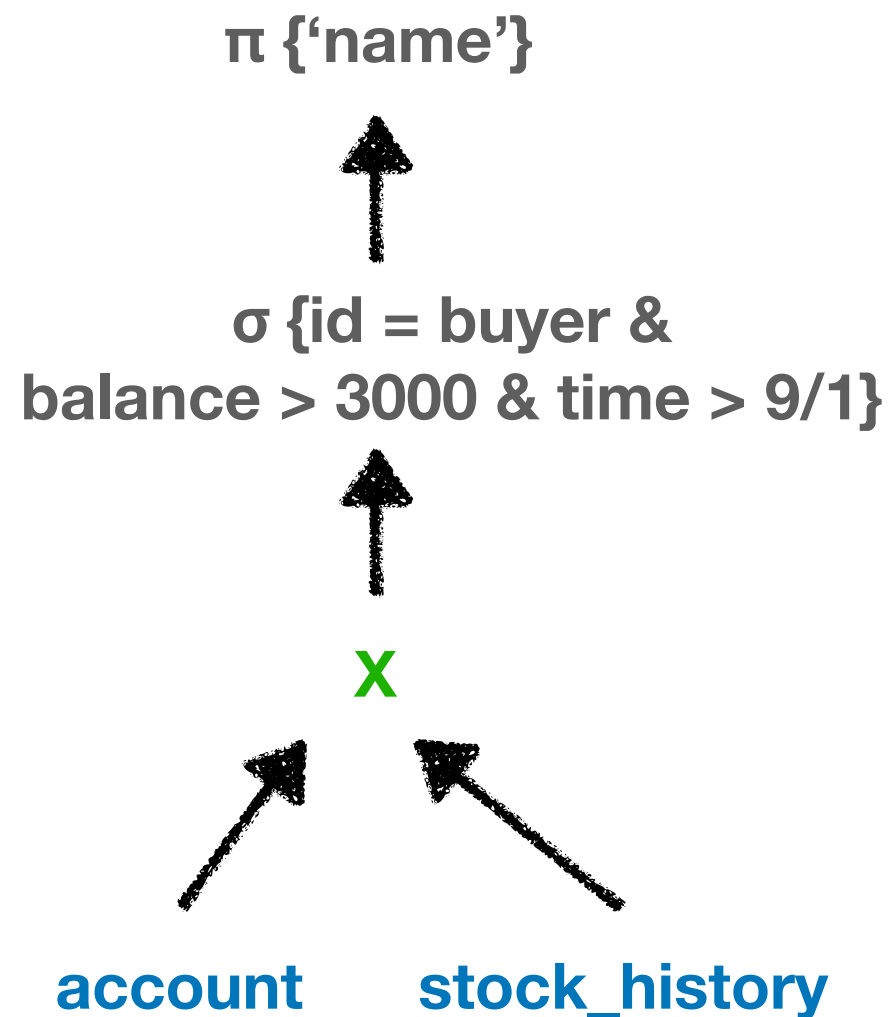
Table {stock_history}

These are called Relational Algebra

Executing A Plan



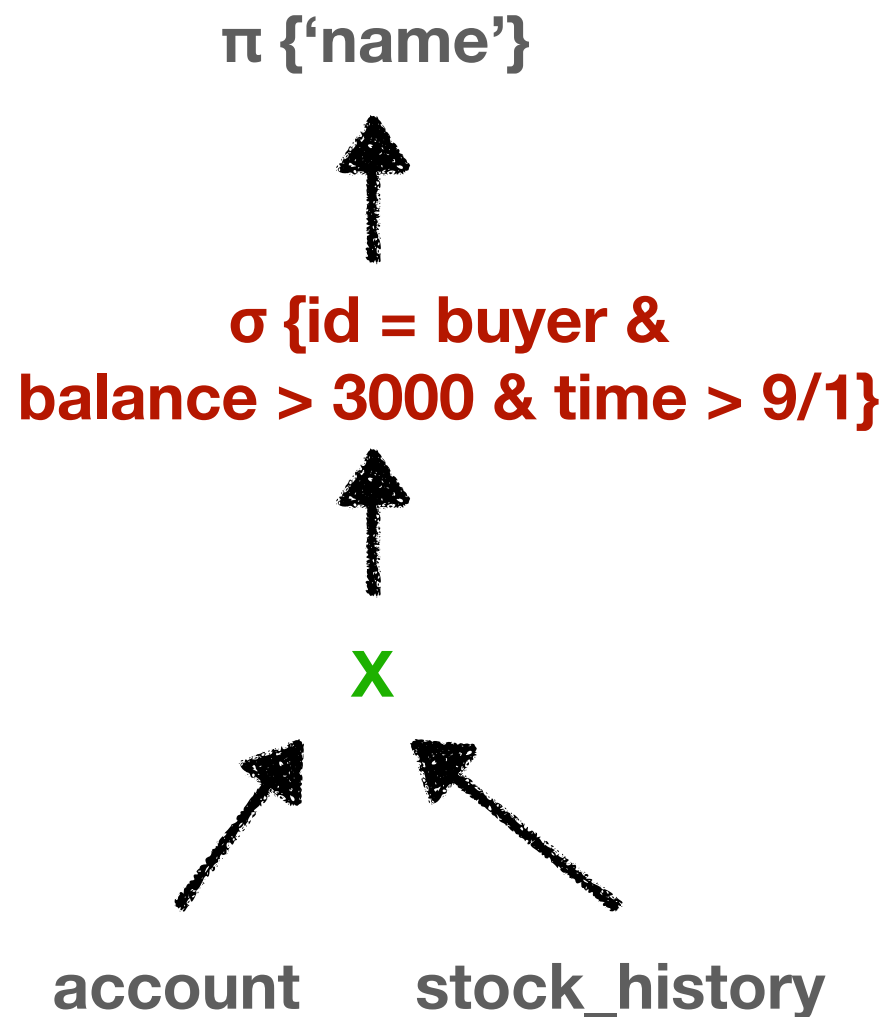
Executing A Plan



id	name	balance	buyer	stock_id	amount	time
1	Red	3300	1	103	50	7/19
1	Red	3300	1	297	300	8/1
1	Red	3300	1	31	230	8/5
1	Red	3300	2	45	40	8/7
...

id	name	balance	buyer	stock_id	amount	time
1	Red	3300	1	103	50	7/19
2	Blue	2200	1	297	300	8/1
3	Green	4500	1	31	230	8/5
			2	45	40	8/7
			3	24	100	9/2

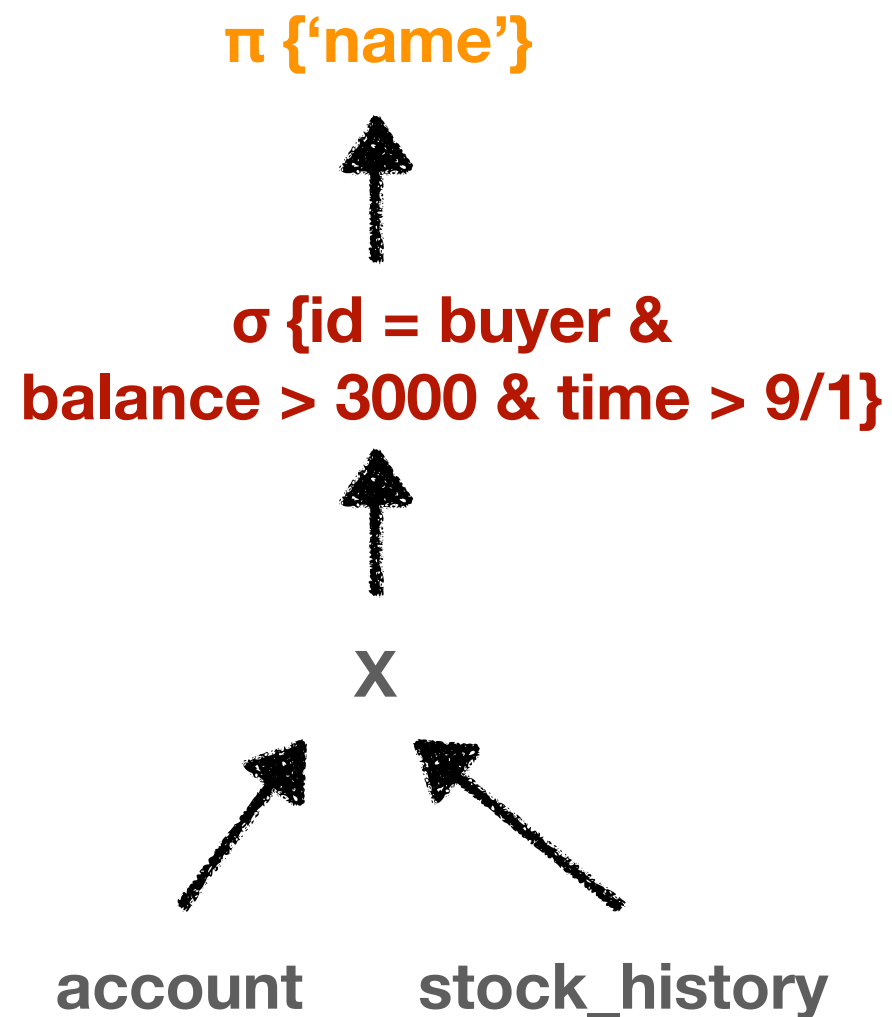
Executing A Plan



id	name	balance	buyer	stock_id	amount	time
3	Green	4000	3	24	100	9/2

id	name	balance	buyer	stock_id	amount	time
1	Red	3300	1	103	50	7/19
1	Red	3300	1	297	300	8/1
1	Red	3300	1	31	230	8/5
1	Red	3300	2	45	40	8/7
...

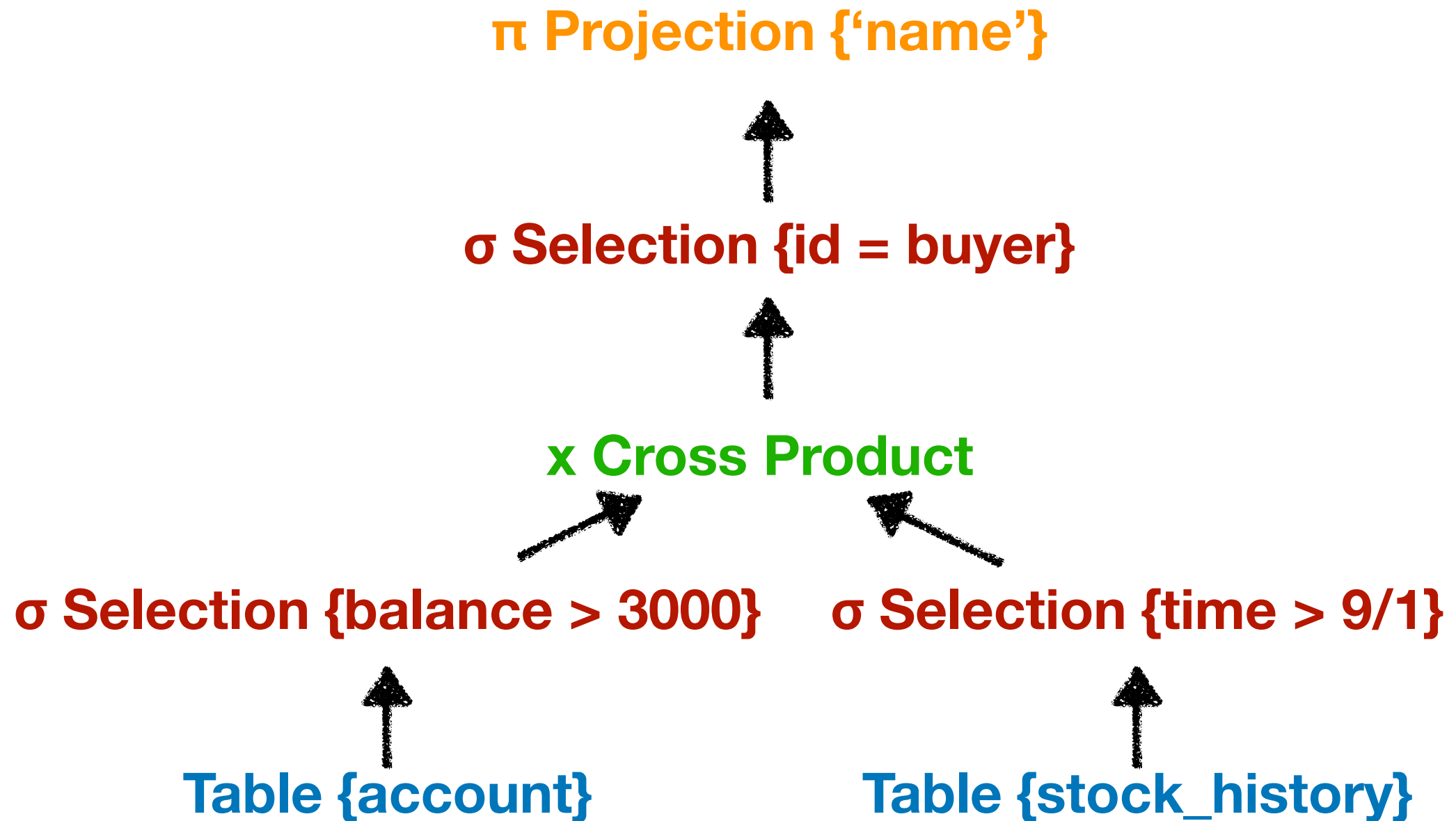
Executing A Plan



name
Green

id	name	balance	buyer	stock_id	amount	time
3	Green	4500	3	24	100	9/2

A Query May Have Multiple Plan Trees



Planners

- Also known as “**Query Optimizer**”.
- A DBMS records the statistics while executing updates.
- Then, a planner tries to find the best plan tree for a query using the statistics.

How to Know The Plan Tree My DBMS Used ?

Ask your DBMS to **EXPLAIN** your query !!

```
SLMT=# EXPLAIN SELECT name FROM account, stock_history WHERE id = buyer  
AND balance > 3000 AND time >= 901;
```

QUERY PLAN

Hash Join (cost=24.16..86.76 rows=835 width=58)

Hash Cond: (stock_history.buyer = account.id)

-> Seq Scan on stock_history (cost=0.00..32.12 rows=590 width=4)
Filter: ("time" >= 901)

-> Hash (cost=20.62..20.62 rows=283 width=62)

-> Seq Scan on account (cost=0.00..20.62 rows=283 width=62)
Filter: (balance > 3000)

(7 rows)

A real example executed on PostgreSQL

**Ok, A Good Query Engine
Is Hard to Write**

Maybe... other parts are easier ?

Outline

- Motivations
- Introduction to RDBMS
- A Day of a Query in VanillaDB
- **Some Challenges of Developing a RDBMS**
- VanillaDB Project
- Our Next Step ?

Don't forget we need
to support **ACID**  **!!**

How To Ensure Atomicity ?



We need logs.

Logging

- A DBMS logs each updates, maybe along with the old value and the new value.

```
BEGIN TRANSACTION;  
UPDATE account SET balance = balance - 100 WHERE name = "Red";  
UPDATE account SET balance = balance + 100 WHERE name = "Blue";  
COMMIT TRANSACTION;
```

SQLs

```
<Tx 1, Begin>  
<Tx 1, Set Value, Record 1, Offset 30, Old 3300, New 3200>  
<Tx 1, Set Value, Record 2, Offset 30, Old 2200, New 2300>  
<Tx 1, Commit>
```

Logs

Undoing

- When a transaction rolls back, it undoes the actions it has performed.

```
BEGIN TRANSACTION;  
UPDATE account SET balance = balance - 100 WHERE name = "Red";  
UPDATE account SET balance = balance + 100 WHERE name = "Blue";  
COMMIT TRANSACTION;
```

<Tx 1, Begin>

<Tx 1, Set Value, Record 1, Offset 30, Old 3300, New 3200>

Undo: Set the value back to 3300.

A Question

**Update The Record First ?
or
Write The Log First ?**

Quick Answer: Write-Ahead Logging

Detail left for your homework :)

How To Ensure Consistency ?



Locks can help you !


Locks The Records You Are Accessing

```
UPDATE account  
SET balance = balance - 100  
WHERE name = "Red";
```

Transaction 1

```
SELECT balance FROM account  
WHERE name = "Red";
```

Transaction 2



id	name	balance
1	Red	3300
2	Blue	2200
3	Green	4000

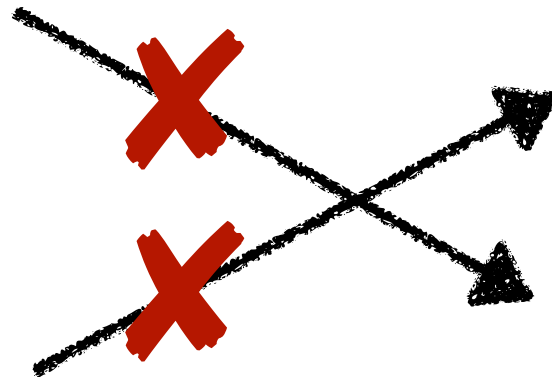
There Can Be Multiple Types of Locks

- Shared Locks (S)
- Exclusive Locks (X)
- Multiple granularity locking (MGL)
 - Intention Shared Locks (IS)
 - Intention Exclusive Locks (IX)
 - Shared with Intention Exclusive Locks (SIX)

Don't Forget Deadlocks !

Transaction 1

Transaction 2



	id	name	balance
Tx1	1	Red	3300
Tx2	2	Blue	2200
	3	Green	4000

How To Solve Deadlocks ?

- Let's see how your Operating Systems learned.
- Algorithms
 - Deadlock-detection
 - Deadlock-avoidance
 - Deadlock-free locking
- Trade-off ?

How To Ensure Isolation ?



Locks again !

Isolation Levels

- The point is
 - When to acquire locks ?
 - When to release locks ?
 - Which locks does it need to acquire ?
- Details left for your homework :)

Durability ?

All You Need Is ...

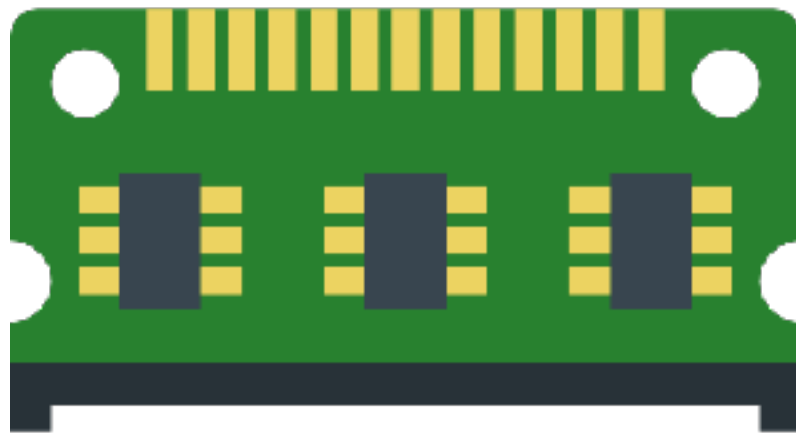


A Disk.

Just save all of them to the disk !

Wait... Disks Are Slow

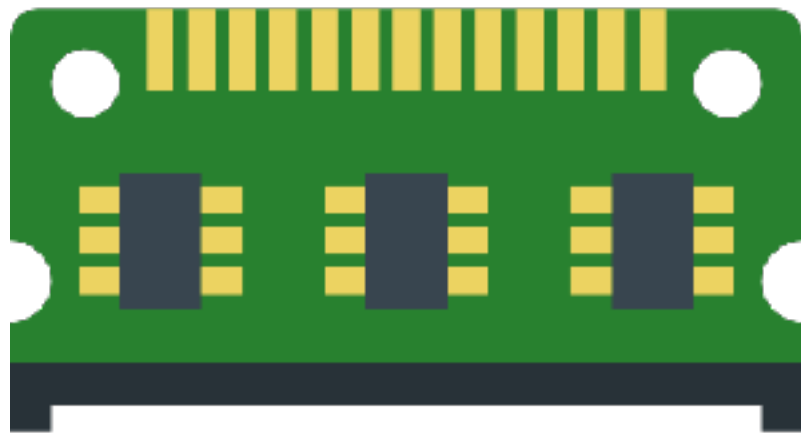
Ok... You may need ...



some memory space to cache data.

Mmm.. What If The DBMS Crashes During Execution ?

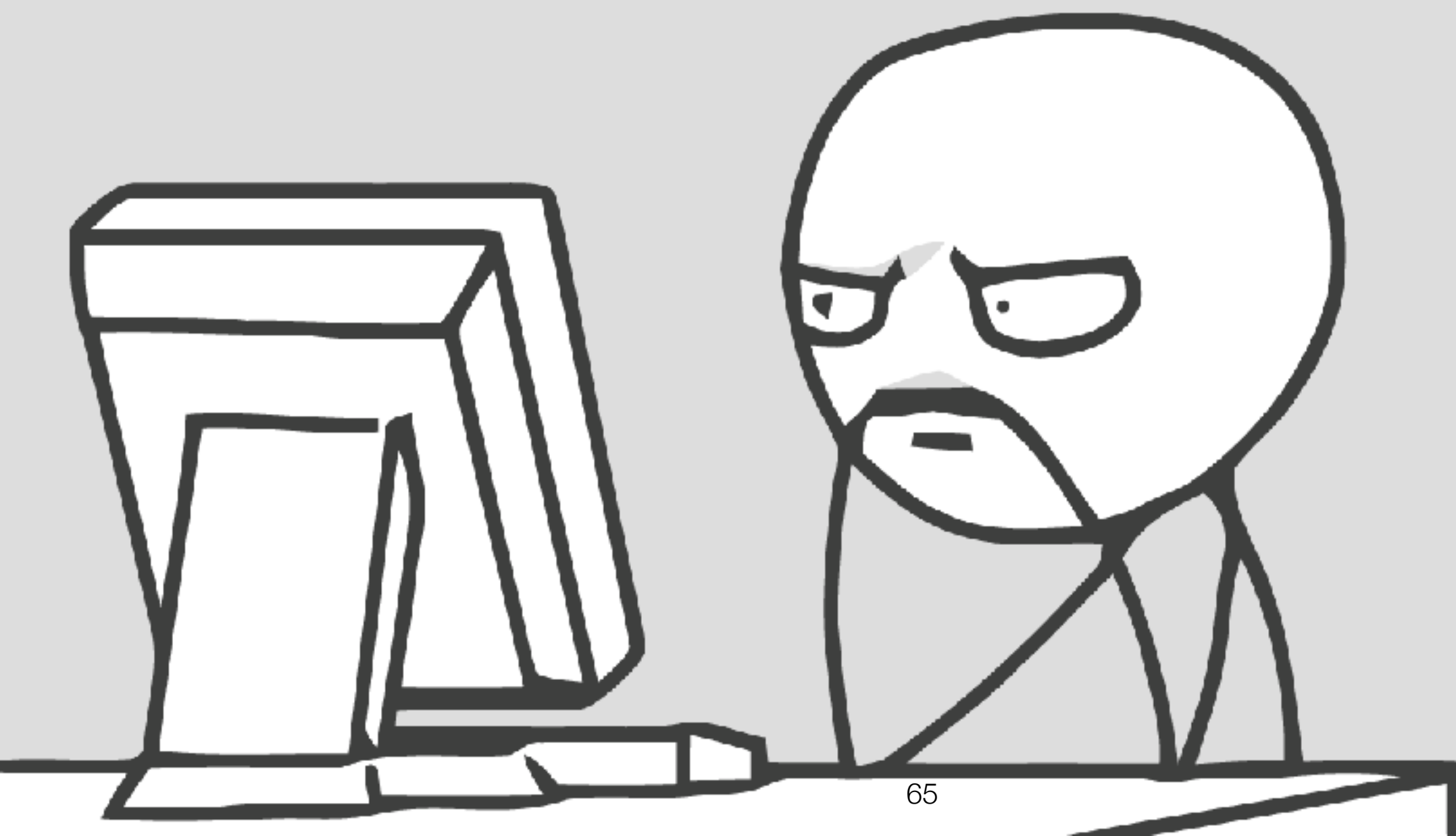
Well... You need to ensure the data are flushed to disk during committing

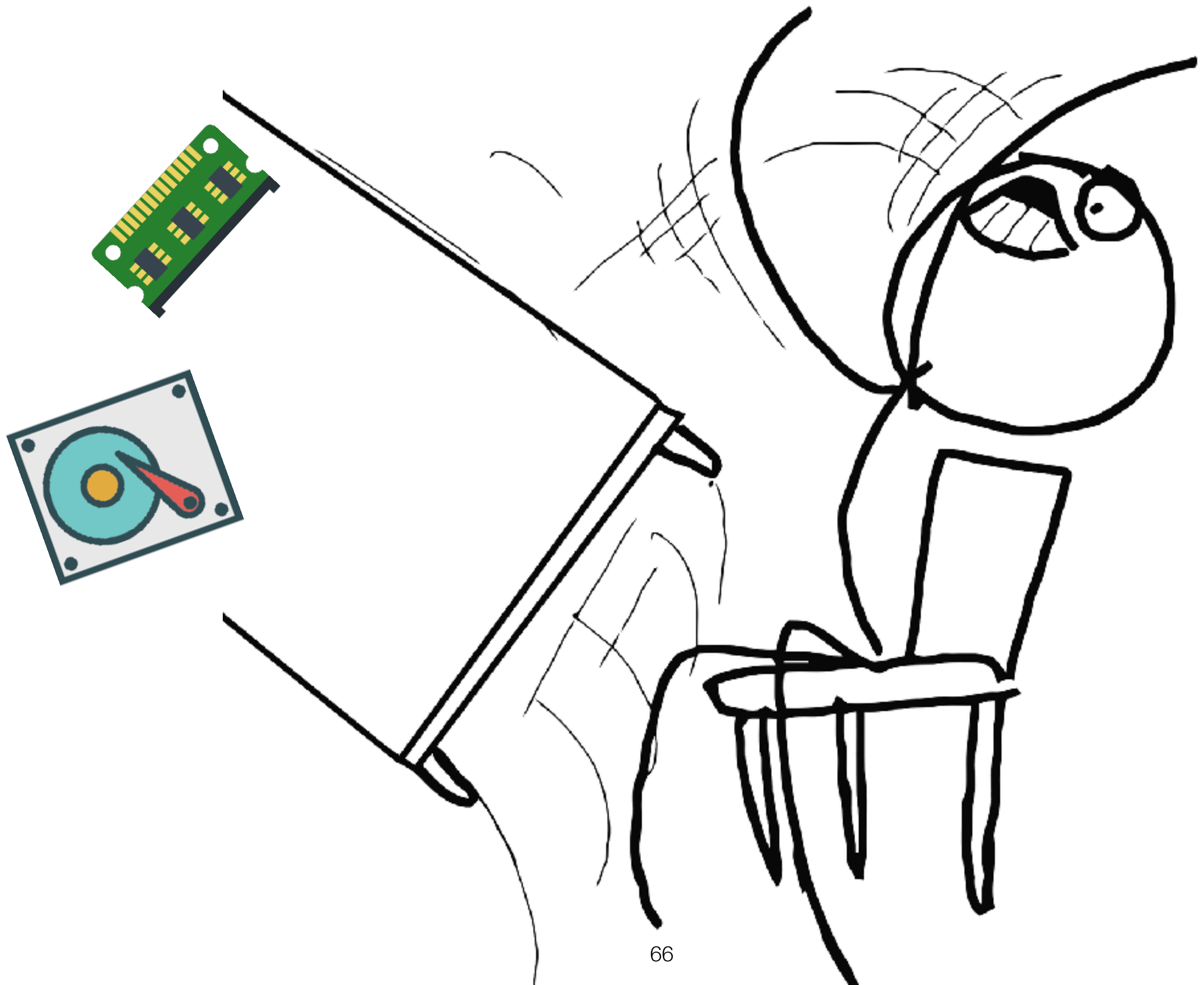


Flushing



**But... That Might Make
Committing Slow !!**





**Ok ! Not So Difficult.
Just flush logs instead.**

A DBMS can recover the data by redoing the actions !

There Are More !!

- How to manage the records on the files ?
 - Block management
- Which cached data need to be swapped ?
 - Buffer replacement strategies
- Indexes
 - What are they ? Can we eat them ?

**“If you are good enough to write code
for a DBMS, then you can write code on
almost anything else.”**

- Andy Pavlo @ CMU 15-721

Outline

- Motivations
- Introduction to RDBMS
- A Day of a Query in VanillaDB
- Some Challenges of Developing a RDBMS
- **VanillaDB Project**
- Our Next Step ?



VanillaDB

<http://www.vanilladb.org/>

Why Do We Write Our Own Database ?

- A modern DBMS uses lots of **optimization** technique and has **complicated structures**.
 - E.g. PostgreSQL, MySQL
- It is hard for beginners to read the source code of such systems.
- Only a few DBMSs designed for tutorial purposes.
 - have almost not been maintaining for a long time.

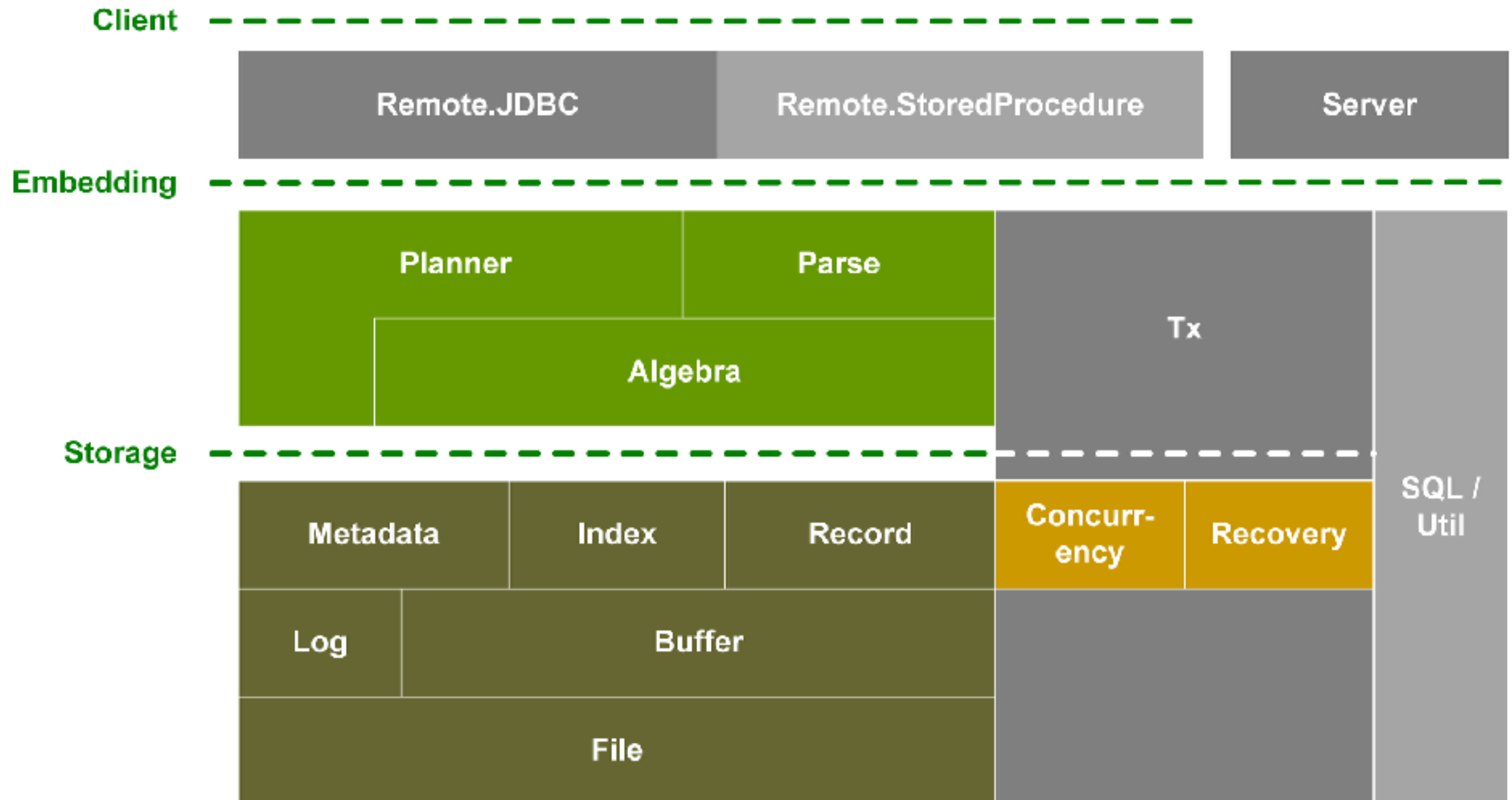
Sub-Projects of VanillaDB

- VanillaCore
 - Single node, multi-threaded RDBMS.
- VanillaBench
 - Benchmarks for testing VanillaCore.
- VanillaComm
 - A collection of reliable group communication primitives.

VanillaCore

- Features
 - **Made in Taiwan.**
 - Highly Modularized.
 - Implemented all necessary components for a RDBMS.
 - Most of them are state-of-the-art.
 - Using Serializable Isolation as default.
 - Written in Java.
 - No need to worry about segmentation fault.

Architecture of VanillaCore



VanillaBench

- A benchmark project focusing on testing VanillaCore.
- Implemented standard benchmarks
 - (most of) TPC-C
 - (part of) TPC-E
 - (planning) YCSB
- There is also a **micro-benchmark** with multiple adjustable parameters for fine-grained experiments.

VanillaComm

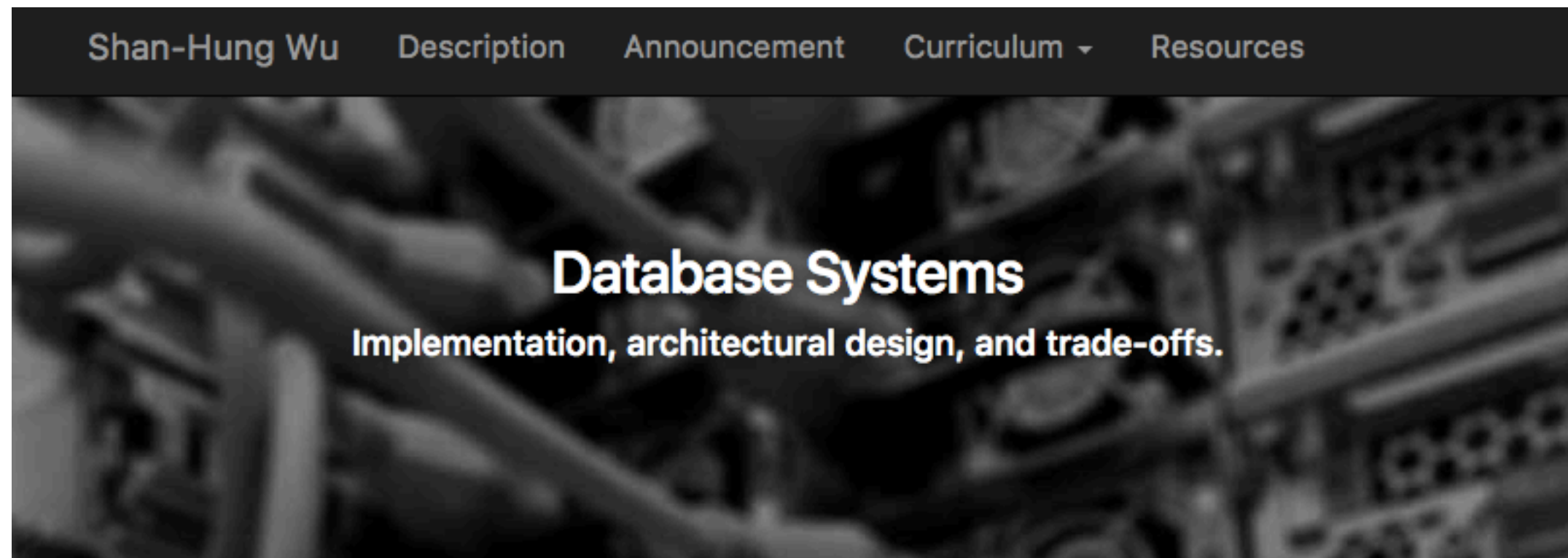
- A collection of communication primitives focusing on provide **reliable communication** for distributed systems.
- This will be a part of our distributed DBMS project in the next step.

How to Contribute ?

- Test & Firing Issues
- Documentation
- Let more people know !

Database Course

- A course that introduces database implementation from scratch.
- <https://nthu-datalab.github.io/clouddb/>



Lecture 03

Architecture and Interfaces

Architecture Overview | SQL | JDBC | Native Interface | RecordFile | MetaData

 Video

 Slides

Introducing Benchmark Project

This lab guides you through the Benchmark Project and how to configure the VanillaDB

 Slides

Assignment 1

Implement JDBC version and Stored procedure Procedures version of Read/Write transaction in VanillaDB.

 Solution

 GitLab

Outline

- Motivations
- Introduction to RDBMS
- A Day of a Query in VanillaDB
- Some Challenges of Developing a RDBMS
- VanillaDB Project
- Our Next Step ?

ElaSQL

A relational database system made scalable, available, and elastic.

<http://www.elasql.org/>

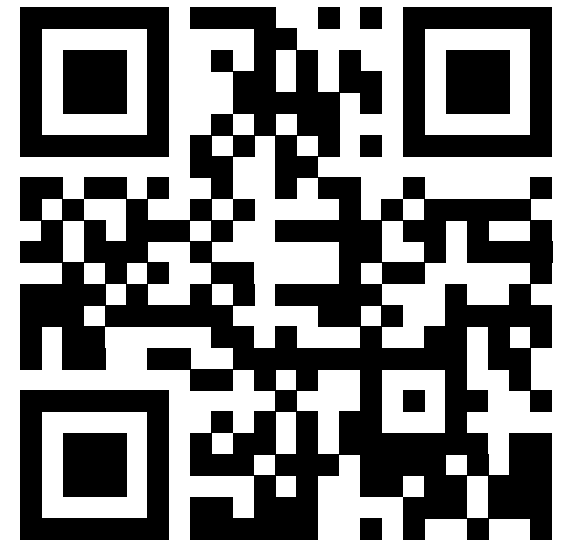
Credit

- {Freepik, Madebyoliver, Dave Gandy, Alfredo Hernandez, Madebyoliver, Swifticons} @ FLATICON for icons
- My adviser: Shan-Hung Wu @ CS, NTHU
- The predecessors of this projects:
 - Tsai-Yu Feng @ Appier
 - {Meng-Kai Liao, Shao-Kan Pi} @ SeekrTech
- My team members:
 - Ching Tsai, Tz-Yu Lin @ CS, NTHU
- Contributors: kye chou, johnnylu305, Dingleet, cyhsutw, jserv



VanillaDB

Q & A



ElaSQL