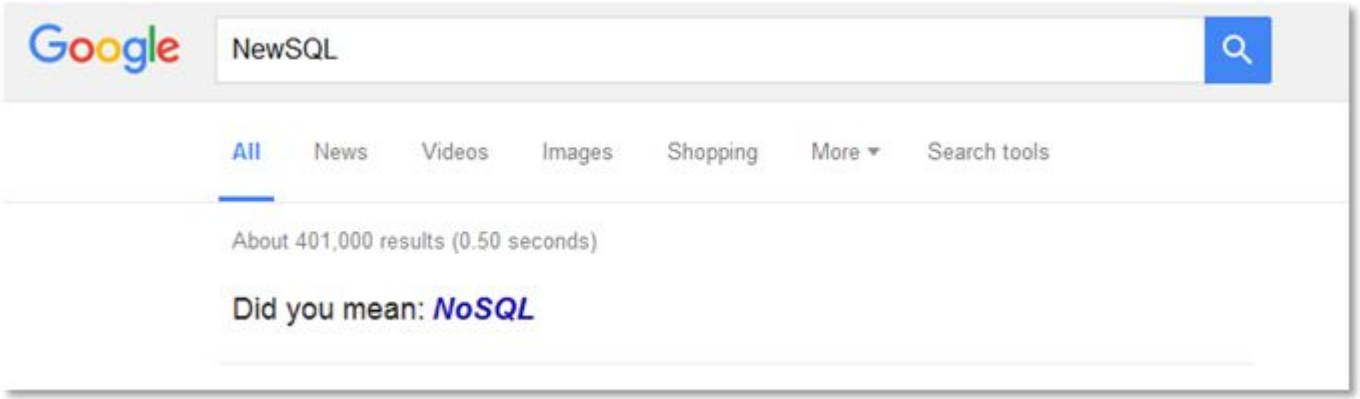
Lukáš Císar, Erik Horváth, Radoslav Karlík, Mikuláš Mráz

NewSQL

- New relational database systems:
  - Scalability and performance of NoSQL systems
  - ACID guarantees of a traditional database systems
- First use of term by Matthew Aslett from „the 451 Group“ in 2011



Comparison

	Traditional RDBMS	NoSQL	NewSQL
SQL	Supported	Not supported	Supported
Machine dependency	Single machine	Distributed	Distributed
DBMS type	Relational	Non-relational	Relational
Schema	Table	Key-value, column-store, etc.	Both
Properties support	ACID	CAP through BASE	ACID
OLTP	Not fully supported	Supported	Fully supported
Cloud support	Not fully supported	Supported	Fully supported

- Fast
- Really fast!
- Horizontally scales with ease
- ACID compliant
- Easy to setup and deploy with Docker
- Perfect for high-demand use case

VoltDB architecture in detail - Key ideas

- Research based
- Memory easier available
- Page Bugger management
- Concurrency management
- Horizontal scaling

VoltDB architecture in detail - Disk persistence

- Append-Only stream writes
- Snapshots
- Asynchronous logging

VoltDB architecture in detail - No-Wait design

- Single threaded
- No waiting on users
- Stored procedures
- Not shared memory
- Concurrency through scheduling

VoltDB architecture in detail - Partitions 1

One of the most important features. Similar to **sharding**.

- Tables partitioned automatically (based on specified column)
- Multiple partitions on a single server
- Partition both **data** and **processing**

Tables with defined partition column - **Partitioned** tables.

Tables without any partition column - **Replicated** tables.

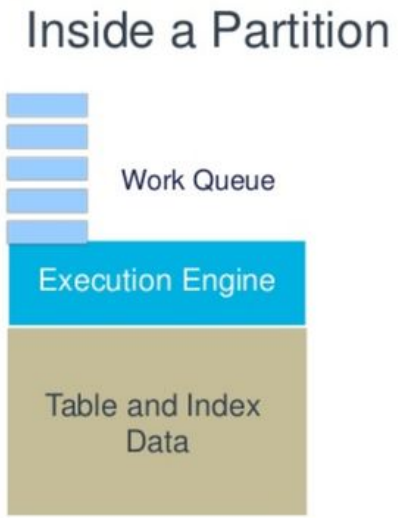
VoltDB architecture in detail - Partitions 2

VoltDB partition contains:

- Data
- Execution engine

The execution engine contains queue for transaction requests.

Requests are executed sequentially (**single threaded**).



VoltDB architecture in detail - Partitions 3

```
CREATE TABLE towns (
  town VARCHAR(64),
  state VARCHAR(2),
  state_num TINYINT NOT NULL,
  county VARCHAR(64),
  county_num SMALLINT NOT NULL,
  elevation INTEGER
);

PARTITION TABLE towns ON COLUMN state_num;
```

VoltDB architecture in detail - Partitions 4

```
./csvloader --file towns.csv towns
```

No partition:

CSVLoader elapsed: **7.788** seconds
Number of input lines skipped: 0
Number of lines read from input: 195082
Number of rows discovered: 195083
Number of rows successfully inserted: 195082
Number of rows that could not be inserted: 0
CSVLoader rate: **25 049.049** row/s

With partition on state_num:

CSVLoader elapsed: **1.067** seconds
Number of input lines skipped: 0
Number of lines read from input: 195082
Number of rows discovered: 195083
Number of rows successfully inserted: 195082
Number of rows that could not be inserted: 0
CSVLoader rate: **182 832.23** row/s

VoltDB architecture in detail - Partitions 5

Partition column rules:

- **Integer** or **string**
- Cannot be **null**
- **One** per table
- Must be included in primary key (if primary key exists)

Recommendations:

- Reasonable distribution of values
- Maximizing use of single-partitioned stored procedures
- Important for INSERT, UPDATE & DELETE
- Use statistics

VoltDB architecture in detail - Partitions 6

Replicated tables - replicated to all partitions

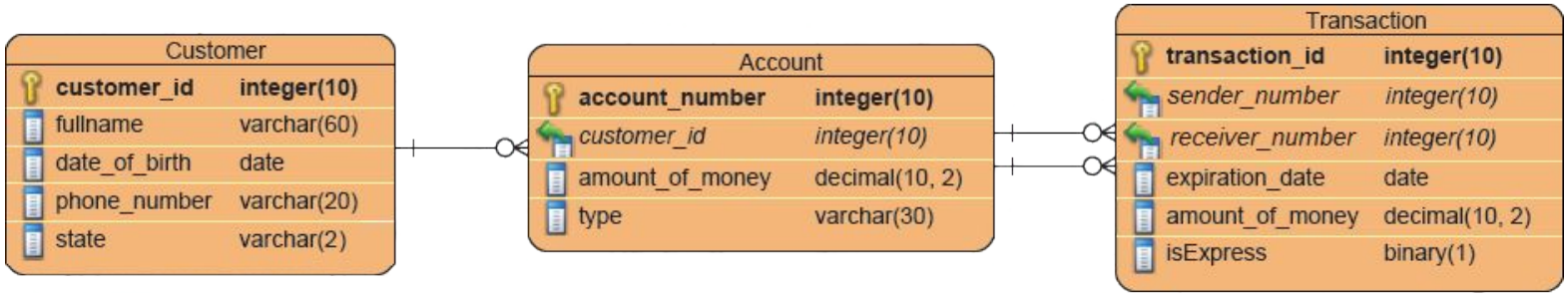
```
CREATE TABLE states (
   abbreviation VARCHAR(20),
   state_num TINYINT,
   name VARCHAR(20),
   PRIMARY KEY (state_num)
);
```

- Small tables
- Data does not change often
- Good for stored procedure joins
- Just use **CREATE TABLE** without **PARTITION** in sql

More in documentation https://docs.voltdb.com/tutorial/Part3.php

Demo / Practical usage

- Simple bank system model:
  - Customer - 2 million rows
  - Account - 3 million rows
  - Transaction - 10 million rows



Questions?

More resources about VoltDB:

Official website: https://www.voltdb.com/