David Rolfe
Director of Solution Architecture, EMEA

Doug Jauregui
Sr. Solutions Engineer
Legacy database technology is obsolete

- Legacy RDBMS designs date from about 1985.
- Vendors are finding legacy databases increasingly uneconomic.
- Legacy databases struggle to scale beyond 2 nodes.
- But demand for transactions is increasing all the time.
- Meanwhile Moore’s law means hardware and RAM keeps getting cheaper
21st Century Requirements for transaction processing

• Virtualization friendly.
• ACID transactions.
• Millisecond response times.
• No "Long Tail"
• Supports complicated logic
• Easily scalable beyond 2 nodes.
• HA “Just Happens”
• Geo replication
• “Translytics”/”HTAP”
RDBMS - How We Thought an RDBMS Worked

```
SELECT * FROM PRODUCTS WHERE ID = 1 FOR UPDATE OF qty;
UPDATE users SET BAL = 190 WHERE ID = 1;
INSERT INTO sales (userid, productId, cost) VALUES (42, 1, 10);
UPDATE products SET qty = 199 WHERE ID = 1;
COMMIT;
SELECT * FROM PRODUCTS WHERE ID = 1 FOR UPDATE OF qty;
UPDATE users SET BAL = 190 WHERE ID = 1;
INSERT INTO sales (userid, productId, cost) VALUES (43, 1, 10);
UPDATE products SET qty = 199 WHERE ID = 1;
COMMIT;
```
RDBMS - What Actually Happens – Part 1...

SELECT * FROM PRODUCTS WHERE ID = 1 FOR UPDATE OF qty;
UPDATE users SET BAL = 190 WHERE ID = 1;
INSERT INTO sales (userid, productId, cost) VALUES (42, 1, 10);
UPDATE products SET qty = 199 WHERE ID = 1;
COMMIT;

SELECT * FROM PRODUCTS WHERE ID = 1 FOR UPDATE OF qty;
UPDATE users SET BAL = 190 WHERE ID = 1;
INSERT INTO sales (userid, productId, cost) VALUES (43, 1, 10);
UPDATE products SET qty = 199 WHERE ID = 1;
COMMIT;

Inflight Transactions

CPU

RAM
DATA

WAITING

WAITING

Inflight Transactions

DISK
DATA

CPU
RDBMS - What Actually Happens – Part 2

Inflight Transactions

RAM DATA

WAITING

WAITING

Inflight Transactions

SAN

CPU

CORE

CPU

CORE

VOLTDB
If we tried this in a supermarket...
Dr. Michael Stonebraker found a solution..
How VoltDB works
How a supermarket works...
VoltDB’s Role

- Transactions
- Scale
- Milliseconds
- Legacy
- OLTP
- ACID Transactions
- Batch Processing/HDFS
- NoSQL
## The only 3 ways to interact with any database

<table>
<thead>
<tr>
<th>Approach</th>
<th>Examples</th>
<th>Strengths</th>
<th>Weaknesses</th>
</tr>
</thead>
</table>
| Many SQL Statements + Commit or Rollback | JDBC, ODBC, PL/SQL | Liked by developers, initial development is rapid | • Doesn’t handle scaling OLTP loads well – DB spends its time figuring out who can see what instead of working  
• Constant locking problems for shared, finite resources  
• Failure of a client to Commit or Rollback causes a temporary resource leak |
| Move all the data to the client and back again | NoSQL, KV Stores | Very developer friendly                    | • Multiple updated copies of the data can arrive at the same time for scaling OLTP loads  
• All of the data gets moved across the network, every time. |
| Stored Procedures                | VoltDB, PL/SQL    | Predictable speed and best possible scaling characteristics | • Not in fashion with developers.  
• PL/SQL created perception of complexity.  
• Other implementations of Java Stored Procedures really slow. |
A Proven and Reliable Partner

**Telco**
Billing/rights management, subscriber data, etc.

**Financial Services**
Risk, market data management, customer mgt.

**Personalize, Customize, Target**
Ad optimization, audience segmenting, customer service

**IoT Platforms, Energy, Sensor**
Smart grid/meters, asset tracing & management

**Infrastructure, Dashboards, KPIs**
Data pipeline, system performance, streaming ETL.
VoltDB & Machine Learning
Near Real Time Data for Models and Rules

VoltDB

Spark + Hadoop

New Data

Rules

Application/Use Case
- Fraud Prevention
- Single sign-in of all Huawei phones
- Consumer banking risk management

Why VoltDB?
- > 50% reduction in fraud cases
- > $15M/year saved from fraud loss
- 10k complex Transactions Per Second
- 99.99% transactions finish < 50ms
- 10x better performance than traditional fraud detection
VoltDB & Machine Learning

• VoltDB has a C++ core with a Java layer on top for running stored procedures

• VoltDB implements High Availability by running the same code in two places at once.

• Any Java class can be used in a stored procedure call provided:
  • It’s deterministic (all copies of the code have to act the same way...)
  • It doesn’t access network resources (which would make it non-deterministic)

• Examples: H20.AI and (J)PMML
ML Example – User Defined Function in H20

```java
public class AirlineDemoUDF {
    private static String modelClassName = "gbm_pojo_test";

    public String ademo(String cRSDepTime, String year, String month, String dayOfMonth, String dayOfWeek,
        String uniqueCarrier, String origin, String dest) {
        try {
            hex.genmodel.GenModel rawModel;
            rawModel = (hex.genmodel.GenModel) Class.forName(modelClassName).newInstance();
            EasyPredictModelWrapper model = new EasyPredictModelWrapper(rawModel);
            RowData row = new RowData();
            row.put("Year", year);
            row.put("Month", month);
            row.put("DayOfMonth", dayOfMonth);
            row.put("DayOfWeek", dayOfWeek);
            row.put("cRSDepTime", cRSDepTime);
            row.put("UniqueCarrier", uniqueCarrier);
            row.put("Origin", origin);
            row.put("Dest", dest);
            BinomialModelPrediction p = model.predictBinomial(row);
            return (p.label);
        } catch (Exception e) {
            System.err.println(e.getMessage());
            return null;
        }
    }
}
```

CREATE FUNCTION ademo FROM METHOD h20.AirlineDemoUDF.ademo;

CREATE PROCEDURE flight_hist
PARTITION ON TABLE flights COLUMN f_FlightNum AS
SELECT f_cRSDepTime, f_year, f_month, f_dayOfMonth, f_dayOfWeek, f_uniqueCarrier, f_origin, f_dest,
ademo(f_cRSDepTime, f_year, f_month, f_dayOfMonth, f_dayOfWeek, f_uniqueCarrier, f_origin, f_dest) ademo
FROM flights
WHERE f_FlightNum = ?
ORDER BY f_year, f_month, f_dayOfMonth, f_cRSDepTime;
ML Example – Calling JPMML from a Procedure

```java
public VoltTable[] runModel(String pmmlFileName, VoltTable inputParams) throws Exception {
    VoltEvaluator evaluator = pmmlEvaluators.get(pmmlFileName);
    if (evaluator == null) {
        throw new Exception("Model " + pmmlFileName + " not found");
    }

    List<InputField> inputFields = evaluator.getInputFields();
    Map<String, Object> arguments = new HashMap<String, Object>();
    // Sanity check input params
    if (inputFields == null) {
        throw new Exception("VoltTable inputParams can't be null");
    }
    if (inputParams.getRowCount() == 0) {
        throw new Exception("VoltTable inputParams must have one row");
    }
    if (inputParams.getColumnCount() != inputFields.size()) {
        throw new Exception("VoltTable inputParams must match length of inputFields, inputParams + columns, expect " + inputFields.size());
    }

    inputParams.advanceRow();
    for (InputField inputField : inputFields) {
        mapVoltParamToPmmlParam(inputParams, arguments, inputField);
    }
    Map<String, Object> result = evaluator.evaluate(arguments);
    // Processing results
    List<TargetField> targetFields = evaluator.getTargetFields();
    VoltTable resultTable = mapPmmlTargetFieldsToVoltTable(result, targetFields);
    VoltTable[] outputFields = evaluator.getOutputFields();
    VoltTable otherTable = mapPmmlOutputFieldsToVoltTable(result, outputFields);
    VoltTable[] outputParams = { resultTable, otherTable };
    return outputParams;
}
```

```java
public class GolfDemo extends VoltProcedure {

    public VoltTable[] run(double temperature, double humidity,
        String windy, String outlook) throws VoltAbortException {

        VoltTable[] pmmlOut;

        try {
            JPMMLImpl i = JPMMLImpl.getInstance();
            VoltDBPMMMLWrangler w = i.getPool().borrowObject();
            final String modelName = "tree.model";
            VoltTable paramtable = w.getEmptyTable(modelName);
            paramtable.addRow(temperature, humidity, windy, outlook);
            pmmlOut = w.runModel(modelName, paramtable);
        }
        catch (Exception e) {
            System.err.println(e.getMessage());
            throw new VoltAbortException(e);
        }

        voltExecuteSQL(true);
        return pmmlOut;
    }
}
```
For more information:
www.voltdb.com
drolfe@voltdb.com