# SQL versus NoSQL

Guido Schmutz

Philipp Salvisberg

BASEL BERN BRUGG LAUSANNE ZÜRICH DÜSSELDORF FRANKFURT A.M. FREIBURG I.BR. HAMBURG MUNICH STUTTGART VIENNA

trivadis
makes IT easier.

# Agenda

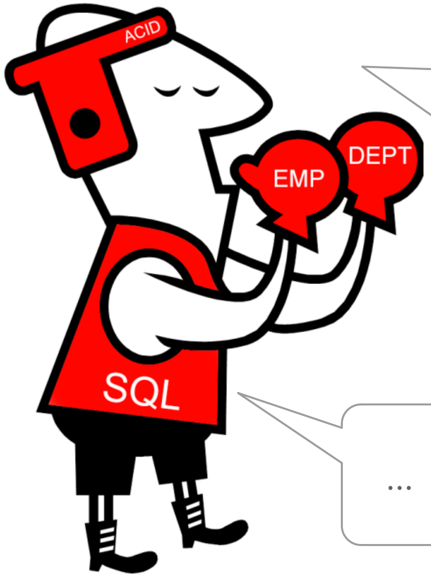
1. Motivation
2. Overview of SQL and NoSQL Data Stores
3. Use Cases – Let's Get Ready to Rumble
4. Recommended Reads
5. Core Messages

# Understanding the Merits

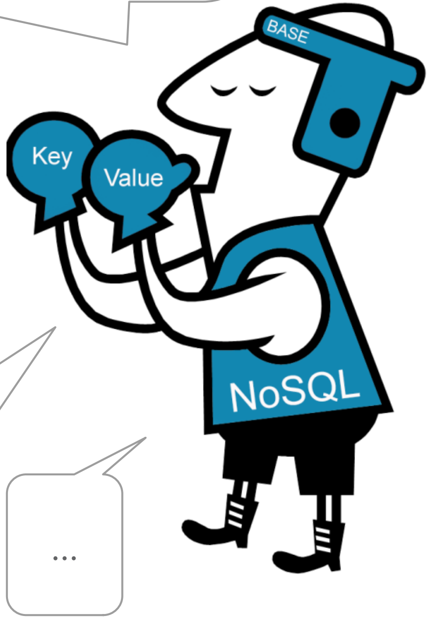
If new relational systems can do everything that a NoSQL system can, with analogous performance and scalability, and with the convenience of transactions and SQL, why would you choose a NoSQL system?

We haven't yet seen good benchmarks showing that RDBMSs can achieve scaling comparable with NoSQL systems like Google's BigTable.

If you only require a lookup of objects based on a single key, then a key-value store is adequate and probably easier to understand than a relational DBMS. Likewise for a document store on a simple application: you only pay the learning curve for the level of complexity you require.



While we don't see "one size fits all" in the SQL products themselves, we do see a common interface with SQL, transactions, and relational schema that give advantages in training, continuity, and data interchange.



Some applications require a flexible schema, allowing each object in a collection to have different attributes. While some RDBMSs allow efficient "packing" of tuples with missing attributes, and some allow adding new attributes at runtime, this is uncommon.

…

…

Source: Scalable SQL and NoSQL Data Stores, Rick Catell, 2010

# Agenda

1. Motivation
2. Overview of SQL and NoSQL Data Stores
3. Use Cases – Let's Get Ready to Rumble
4. Recommended Reads
5. Conclusion

# SQL Data Stores

- Relational Model
- Standardized, SQL:2011 is the 7th major revision since SQL-86
  - 9 parts, more than 4000 pages
  - But no single database implements all standards/features
- Rich set of features
  - Incl. SQL/PSM, SQL/MED, SQL/XML, SQL/RPR, Temporal Features
  - Incl. User-defined Types and Collection Types (since SQL:1999)
- ACID Transactions
  - **A**tomicity: all or nothing
  - **C**onsistency: from valid state to valid state considering constraints, triggers, …
  - **I**solation: result is not affected through concurrent execution
  - **D**urability: committed data stays available after crash, power loss or errors
- Good support by different languages, frameworks and tools
- Good understanding of basic concepts by IT professionals

# NoSQL Definition

- Next Generation Databases mostly addressing some of the points:
  - being non-relational,
  - distributed,
  - open-source and
  - horizontally scalable.
- Often more characteristics apply such as:
  - schema-free,
  - easy replication support,
  - simple API,
  - eventually consistent / BASE (not ACID),
  - a huge amount of data
  - and more.

BASE

- **B**asically **A**vailable: Availability is more important than consistency
- **S**oft State: Higher availability results in an eventual consistent state
- **E**ventually Consistent: If no new updates are made to a given data item, eventually all accesses to that item will return the last updated value

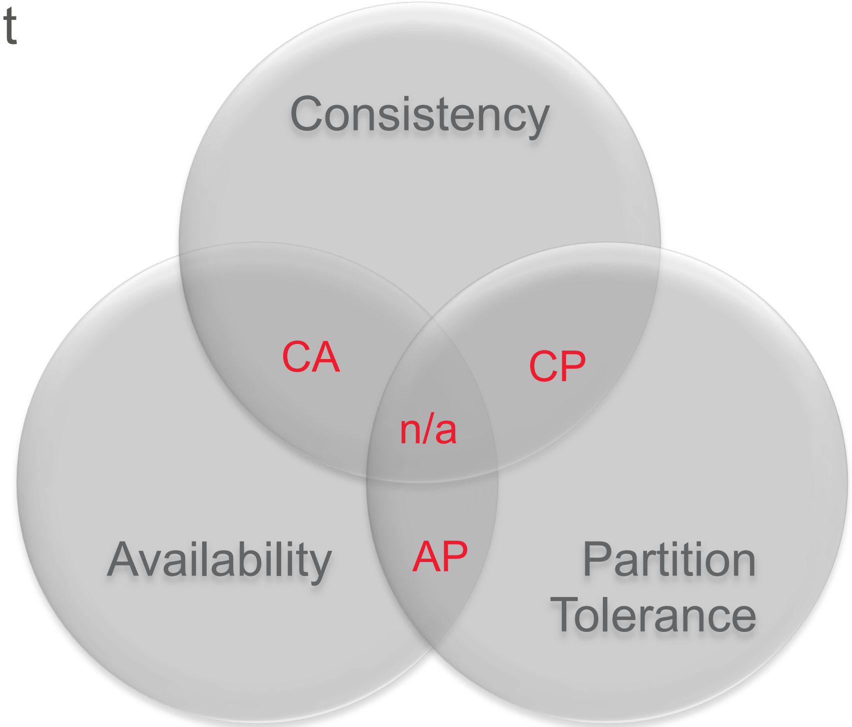
- The misleading term "nosql" (the community now translates it mostly with "not only sql") should be seen as an alias to something like the definition above

Source: http://nosql-database.org

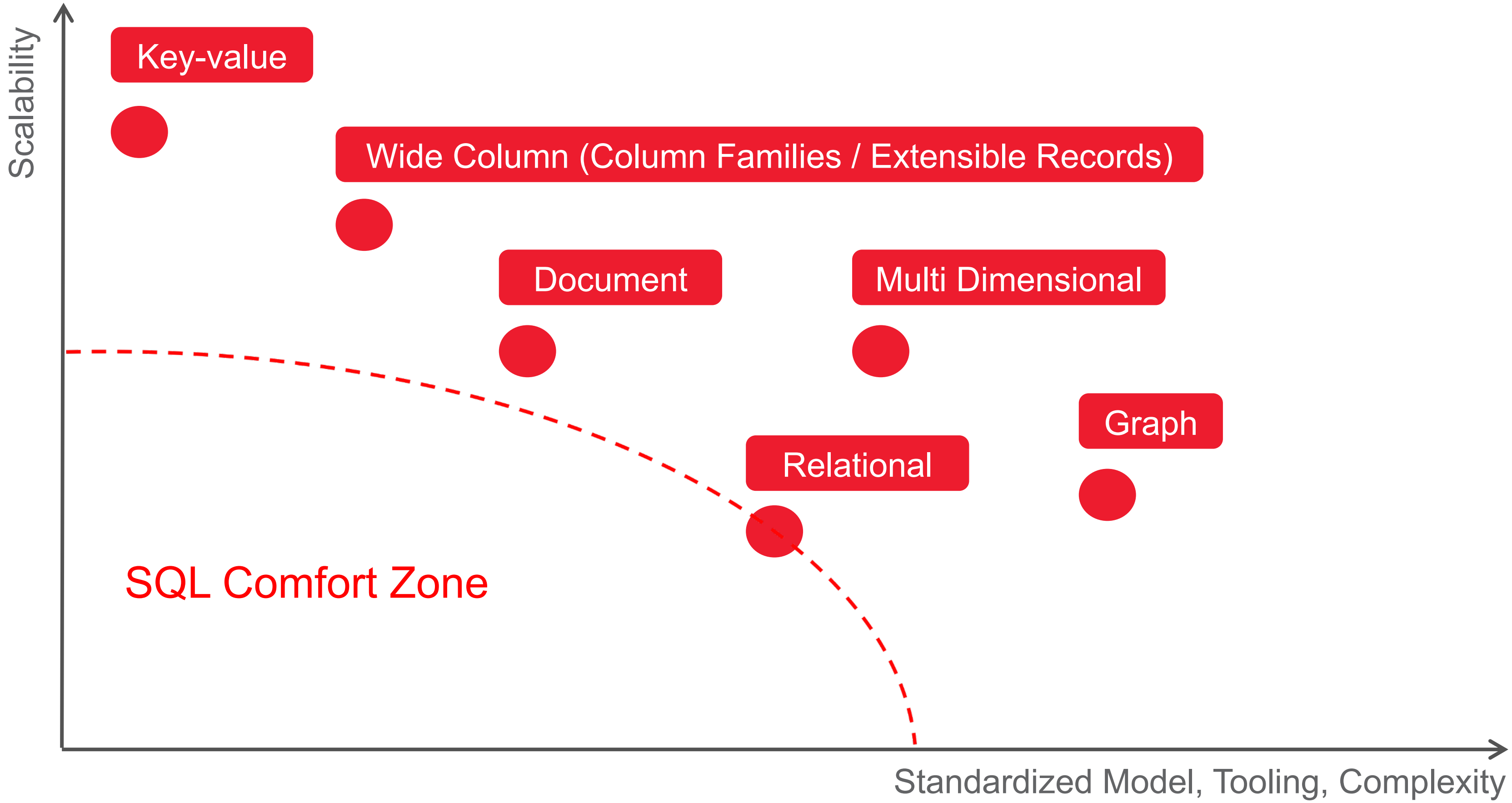
# Brewer's CAP Theorem

Any networked shared-data system can have at most two of the three desirable properties:

- Consistency
  All of the nodes see the same data at the same time, regardless of where the data is stored
- Availability
  Node failures do not prevent survivors from continuing to operate
- Partition tolerance
  The system continues to operate despite arbitrary message loss



# Data Store Positioning



# NoSQL Data Stores for Scalability

- Key-value Stores
  - These systems store values and an index to find them, based on a programmer defined key
  - Products: Voldemort, Redis, Riak, Amazon DynamoDB, Oracle NoSQL, Oracle Berkeley DB, Memcached
- Document Stores
  - These systems store documents. A document allows values to be nested documents or lists as well as scalar values, and the attribute names are dynamically defined for each document at runtime. The documents are indexed and a simple query mechanism is provided
  - Products: MongoDB, CouchDB, RavenDB, OrientDB, Couchbase
- Wide Column Stores (Column Families / Extensible Record Store)
  - These systems store extensible records that can be partitioned vertically and horizontally across nodes
  - Products: Hbase, Cassandra, Accumulo, Amazon SimpleDB, HyperTable

# More NoSQL Data Stores

- Graph Databases
  - These systems use graph structures with nodes, edges, and properties to represent and store data. Specialized graph databases such as triplestores and network databases exists beside general graph databases. SPARQL is used to query graphs in RDF format.
  - Products: Neo4J, Titan, Jena, Sesame, Allegrograph, Virtuoso, BigData, Oracle Spatial and Graph, Oracle NoSQL with Graph Options
- Multidimensional Databases
  - These systems support multi-dimensional online analytical processing (MOLAP) by storing data in an optimized multi-dimensional array storage (data cubes), rather than in a relational database. MDX is typically used to query data in multidimensional databases.
  - Products: Microsoft Analysis Services, Oracle Essbase, Palo (Jedox), Mondrian (Pentaho), SAS OLAP Server, IBM Cognos TM1
- and many more

# Agenda

1. Motivation
2. Overview of SQL and NoSQL Data Stores
3. Use Cases – Let's Get Ready to Rumble
4. Recommended Reads
5. Core Messages

# Round 1

# Smart Meter

# Temperature – 1 Value per Second and Sensor

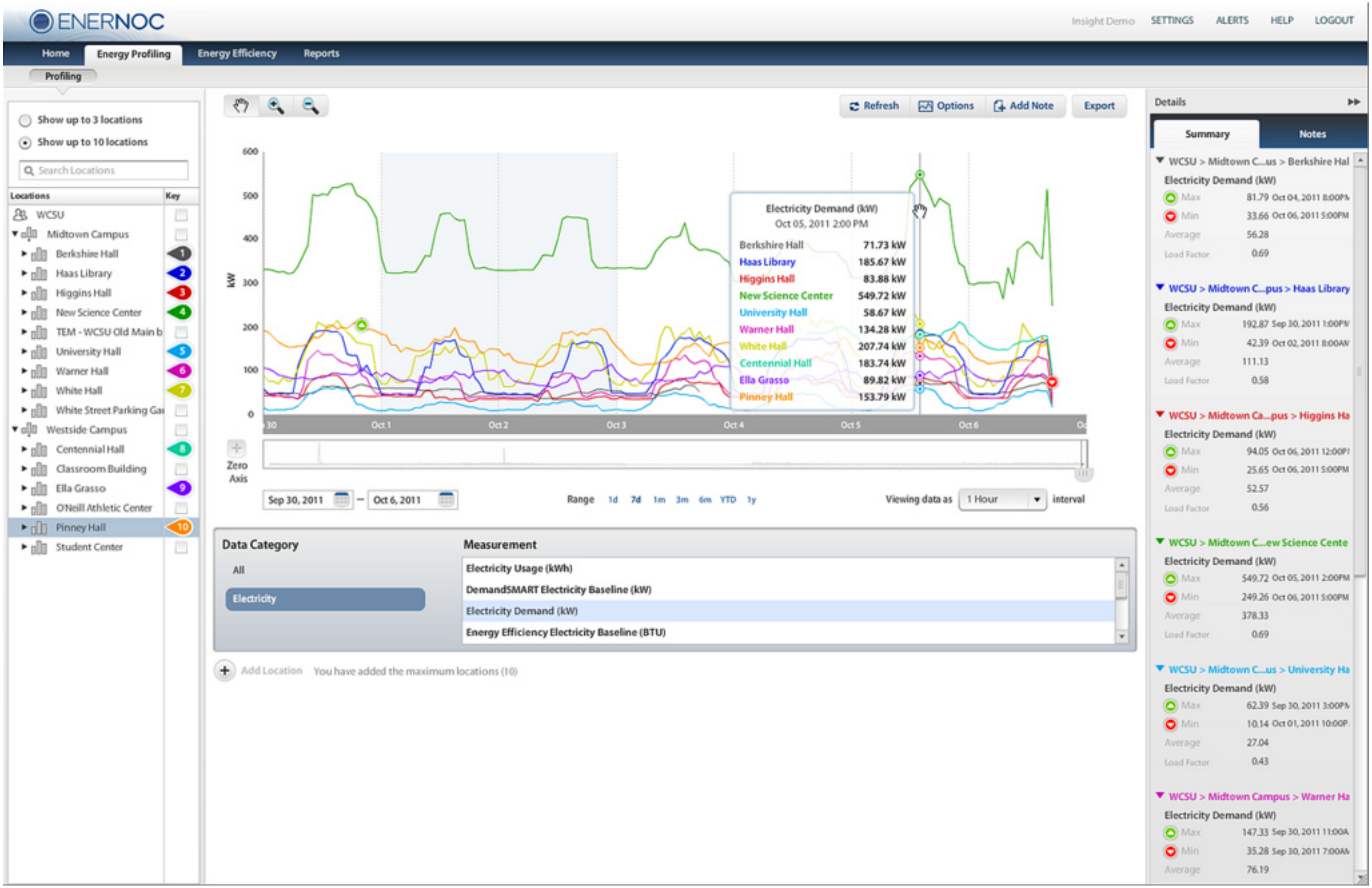
## Starting Position

- Data is captured from 2 Mio sensors (10 sub-sensors, e.g. fridge, stove, dish-washer, wash-machine, TV, computer, …)
  - Smart Meters delivering the current usage
  - One meter per household
  - Delivery interval between 1 second and 5 minutes
- Every sensor delivers the current usage per second (kWh)
- AP Characteristics

## Use Case Description

- Insert sensor and its sub-sensor values
- Query usage per sensor and its sub-sensors to visualize a time series on a customer dashboard
  - Available in different granularities, values are aggregated in
    - Minute
    - Quarter of hour (15-minutes)
    - Hour
    - Day
  - Responsive UI

# Query 1: Customer Dashboard



○ Show up to 3 locations
● Show up to 10 locations

Search Locations

| Locations | Key |
|---|---|
| WCSU | □ |
| Midtown Campus | □ |
| Berkshire Hall | 1 |
| Haas Library | 2 |
| Higgins Hall | 3 |
| New Science Center | 4 |
| TEM - WCSU Old Main b | □ |
| University Hall | 5 |
| Warner Hall | 6 |
| White Hall | 7 |
| White Street Parking Ga | □ |
| Westside Campus | □ |
| Centennial Hall | 8 |
| Classroom Building | □ |
| Ella Grasso | 9 |
| O'Neill Athletic Center | □ |
| Pinney Hall | 10 |
| Student Center | □ |

Electricity Demand (kW)
Oct 05, 2011 2:00 PM

| | |
|---|---|
| Berkshire Hall | 71.73 kW |
| Haas Library | 185.67 kW |
| Higgins Hall | 83.88 kW |
| New Science Center | 549.72 kW |
| University Hall | 58.67 kW |
| Warner Hall | 134.28 kW |
| White Hall | 207.74 kW |
| Centennial Hall | 183.74 kW |
| Ella Grasso | 89.82 kW |
| Pinney Hall | 153.79 kW |

Sep 30, 2011 – Oct 6, 2011

Viewing data as 1 Hour interval

| Data Category | Measurement |
|---|---|
| All | Electricity Usage (kWh) |
| Electricity | DemandSMART Electricity Baseline (kW) |
| | Electricity Demand (kW) |
| | Energy Efficiency Electricity Baseline (BTU) |

You have added the maximum locations (10)

WCSU > Midtown C...us > Berkshire Hal
Electricity Demand (kW)

| | | |
|---|---|---|
| Max | 81.79 | Oct 04, 2011 8:00PM |
| Min | 33.66 | Oct 06, 2011 5:00PM |
| Average | 56.28 | |
| Load Factor | 0.69 | |

WCSU > Midtown C...pus > Haas Library
Electricity Demand (kW)

| | | |
|---|---|---|
| Max | 192.87 | Sep 30, 2011 1:00PM |
| Min | 42.39 | Oct 02, 2011 8:00AM |
| Average | 111.13 | |
| Load Factor | 0.58 | |

WCSU > Midtown Ca...pus > Higgins Ha
Electricity Demand (kW)

| | | |
|---|---|---|
| Max | 94.05 | Oct 06, 2011 12:00PI |
| Min | 25.65 | Oct 06, 2011 5:00PM |
| Average | 52.57 | |
| Load Factor | 0.56 | |

WCSU > Midtown C...ew Science Cente
Electricity Demand (kW)

| | | |
|---|---|---|
| Max | 549.72 | Oct 05, 2011 2:00PM |
| Min | 249.26 | Oct 06, 2011 5:00PM |
| Average | 378.33 | |
| Load Factor | 0.69 | |

WCSU > Midtown C...us > University Ha
Electricity Demand (kW)

| | | |
|---|---|---|
| Max | 62.39 | Sep 30, 2011 3:00PM |
| Min | 10.14 | Oct 01, 2011 10:00P |
| Average | 27.04 | |
| Load Factor | 0.43 | |

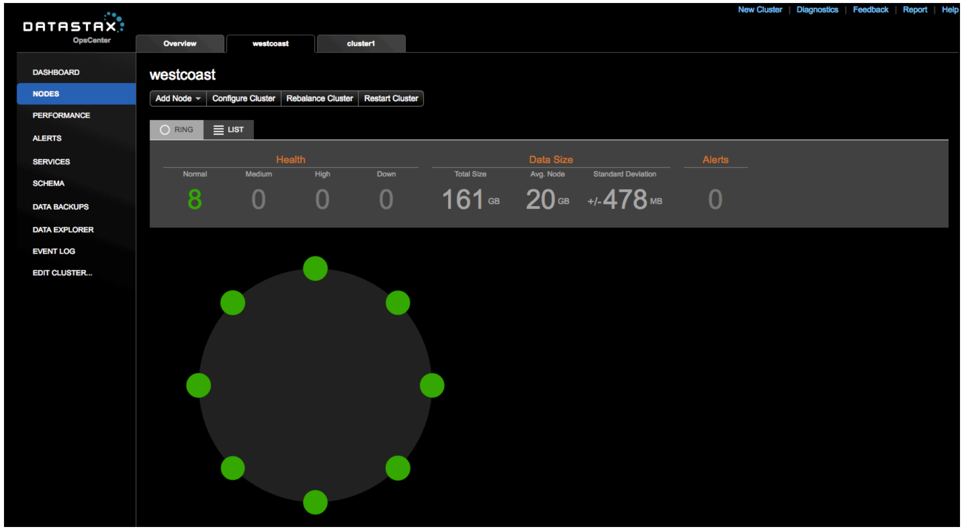
WCSU > Midtown Campus > Warner Ha
Electricity Demand (kW)

| | | |
|---|---|---|
| Max | 147.33 | Sep 30, 2011 11:00A |
| Min | 35.28 | Sep 30, 2011 7:00AM |
| Average | 76.19 | |

trivadis
makes IT easier.

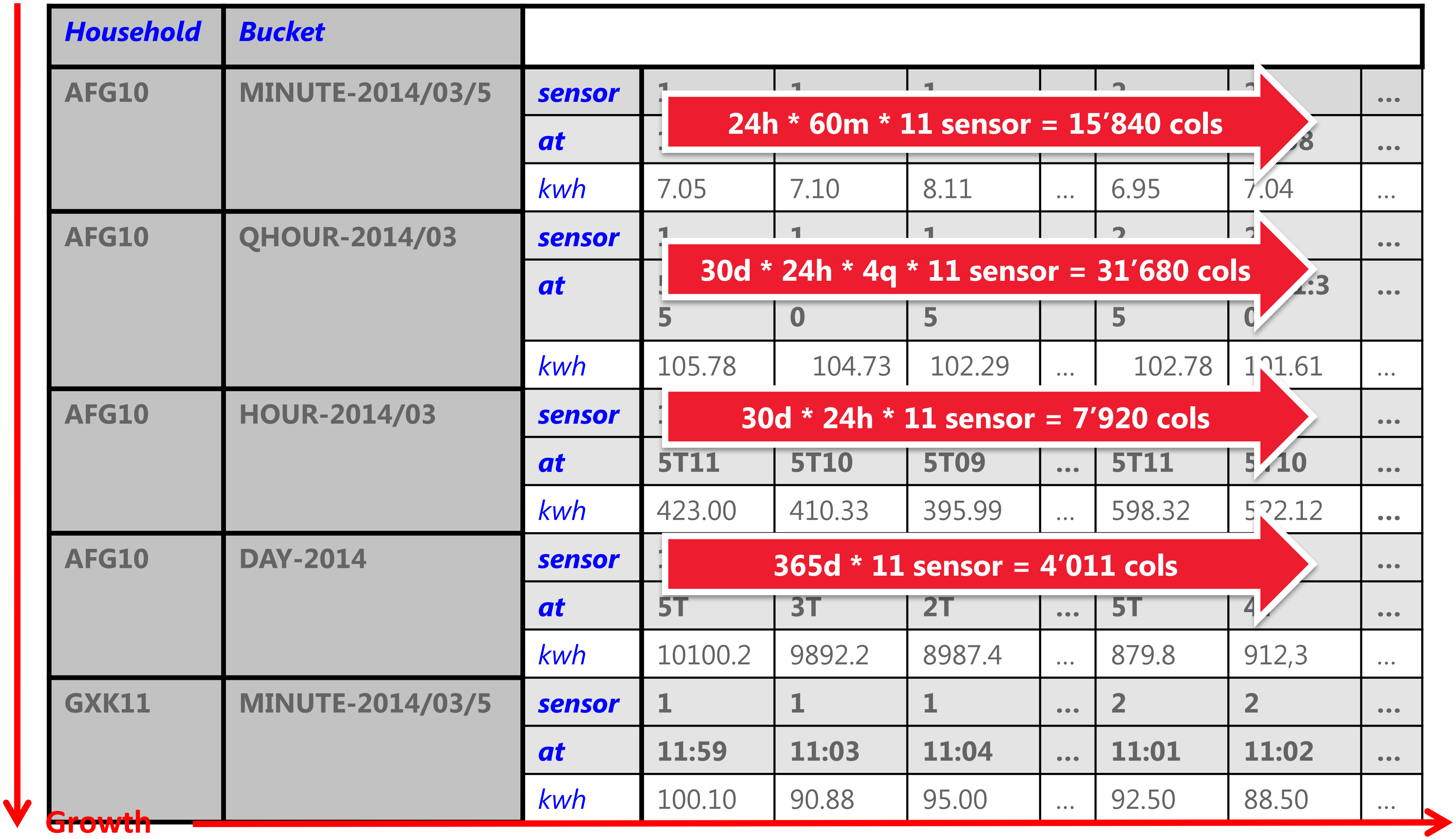
# Cassandra NoSQL Datastore

- Wide-Column Store
- Developed at Facebook
- Professional grade support from DataStax
- Main Features
  - Real-Time
  - Highly Distributed
  - Support for Multiple Data Center
  - Highly Scalable
  - No Single Point of Failure
  - Fault Tolerant
  - Tunable Consistency
  - CQL – Cassandra Query Language



| Health | | | | Data Size | | | Alerts |
|---|---|---|---|---|---|---|---|
| Normal | Medium | High | Down | Total Size | Avg. Node | Standard Deviation | |
| 8 | 0 | 0 | 0 | 161 GB | 20 GB | +/-478 MB | 0 |

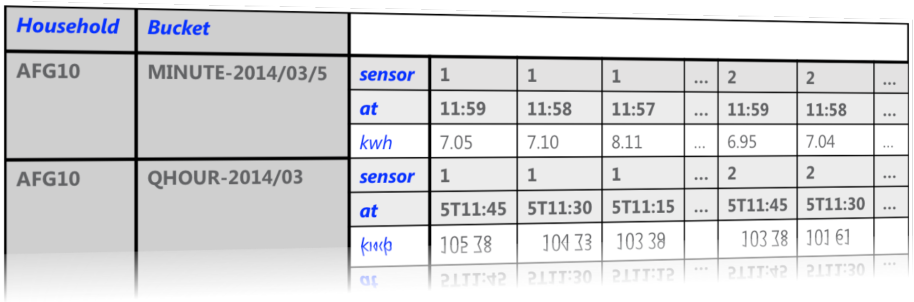
# The Cassandra Way



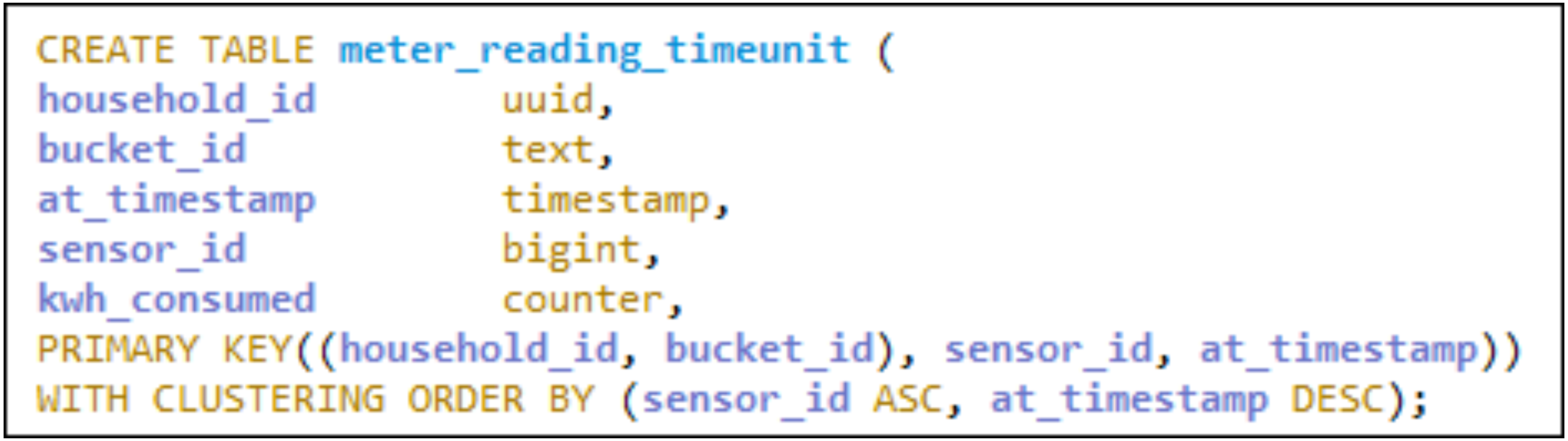
| Household | Bucket | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| AFG10 | MINUTE-2014/03/5 | sensor | 1 | 1 | 1 | [illegible] | 2 | 2 | ... |
| | | at | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible]8 | ... |
| | | kwh | 7.05 | 7.10 | 8.11 | ... | 6.95 | 7.04 | ... |
| AFG10 | QHOUR-2014/03 | sensor | 1 | 1 | 1 | [illegible] | 2 | 2 | ... |
| | | at | [illegible]<br>5 | [illegible]<br>0 | [illegible]<br>5 | [illegible] | [illegible]<br>5 | [illegible]:3<br>0 | ... |
| | | kwh | 105.78 | 104.73 | 102.29 | ... | 102.78 | 101.61 | ... |
| AFG10 | HOUR-2014/03 | sensor | 1 | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | ... |
| | | at | 5T11 | 5T10 | 5T09 | ... | 5T11 | 5T10 | ... |
| | | kwh | 423.00 | 410.33 | 395.99 | ... | 598.32 | 522.12 | ... |
| AFG10 | DAY-2014 | sensor | 1 | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | ... |
| | | at | 5T | 3T | 2T | ... | 5T | 4[illegible] | ... |
| | | kwh | 10100.2 | 9892.2 | 8987.4 | ... | 879.8 | 912,3 | ... |
| GXK11 | MINUTE-2014/03/5 | sensor | 1 | 1 | 1 | ... | 2 | 2 | ... |
| | | at | 11:59 | 11:03 | 11:04 | ... | 11:01 | 11:02 | ... |
| | | kwh | 100.10 | 90.88 | 95.00 | ... | 92.50 | 88.50 | ... |

24h * 60m * 11 sensor = 15’840 cols
30d * 24h * 4q * 11 sensor = 31’680 cols
30d * 24h * 11 sensor = 7’920 cols
365d * 11 sensor = 4’011 cols
Growth

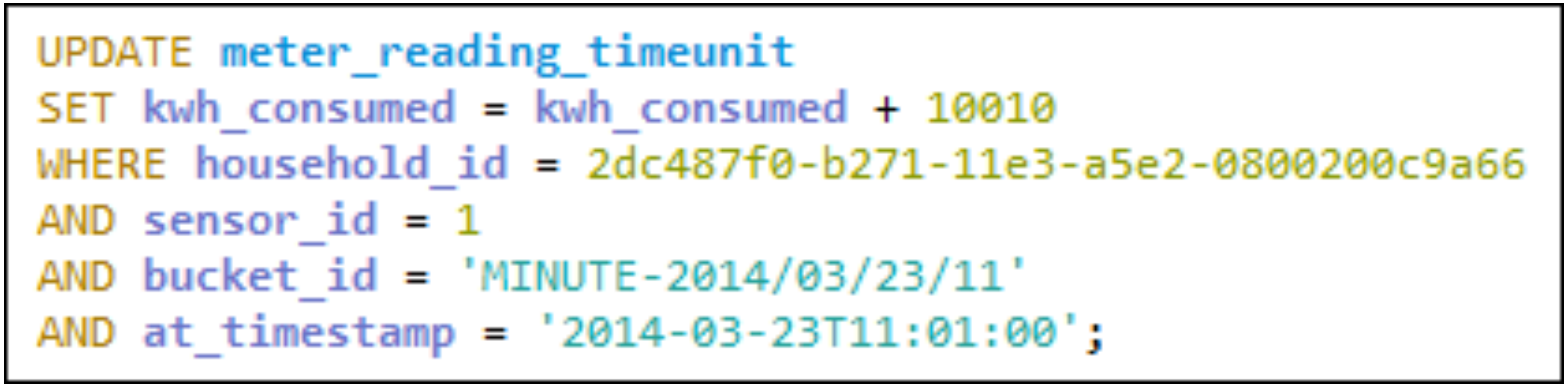
# The Cassandra Way



| Household | Bucket | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| AFG10 | MINUTE-2014/03/5 | sensor | 1 | 1 | 1 | ... | 2 | 2 | ... |
| | | at | 11:59 | 11:58 | 11:57 | ... | 11:59 | 11:58 | ... |
| | | kwh | 7.05 | 7.10 | 8.11 | ... | 6.95 | 7.04 | ... |
| AFG10 | QHOUR-2014/03 | sensor | 1 | 1 | 1 | ... | 2 | 2 | ... |
| | | at | 5T11:45 | 5T11:30 | 5T11:15 | ... | 5T11:45 | 5T11:30 | ... |
| | | kwh | 105.38 | 104.33 | 103.39 | | 103.38 | 101.61 | |

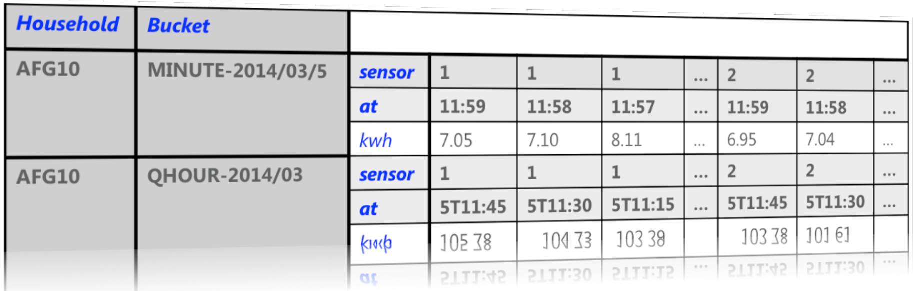


```
CREATE TABLE meter_reading_timeunit (
household_id         uuid,
bucket_id            text,
at_timestamp         timestamp,
sensor_id            bigint,
kwh_consumed         counter,
PRIMARY KEY((household_id, bucket_id), sensor_id, at_timestamp))
WITH CLUSTERING ORDER BY (sensor_id ASC, at_timestamp DESC);
```



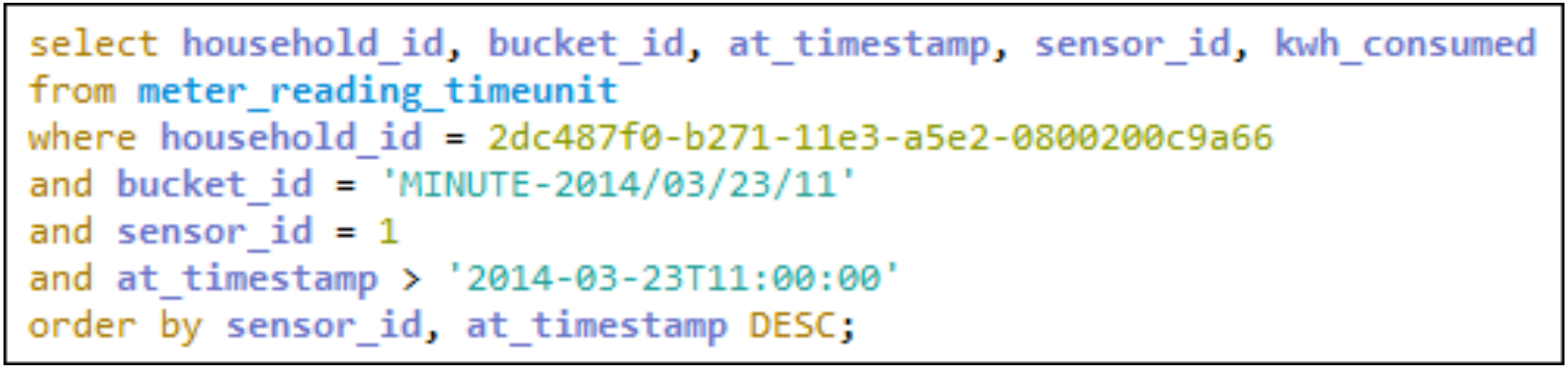
```
UPDATE meter_reading_timeunit
SET kwh_consumed = kwh_consumed + 10010
WHERE household_id = 2dc487f0-b271-11e3-a5e2-0800200c9a66
AND sensor_id = 1
AND bucket_id = 'MINUTE-2014/03/23/11'
AND at_timestamp = '2014-03-23T11:01:00';
```

# The Cassandra Way



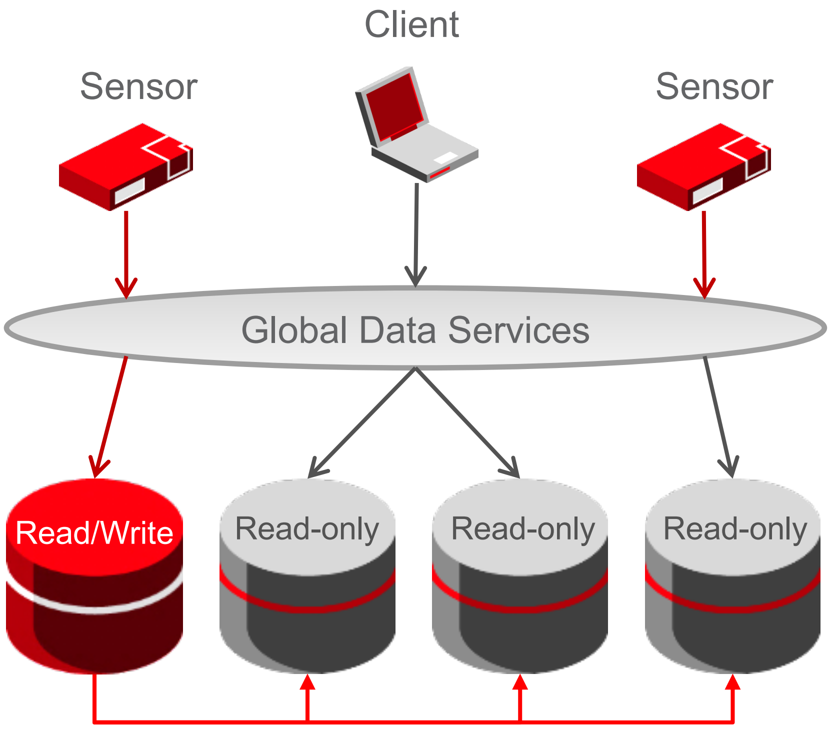
| Household | Bucket | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| AFG10 | MINUTE-2014/03/5 | sensor | 1 | 1 | 1 | ... | 2 | 2 | ... |
| | | at | 11:59 | 11:58 | 11:57 | ... | 11:59 | 11:58 | ... |
| | | kwh | 7.05 | 7.10 | 8.11 | ... | 6.95 | 7.04 | ... |
| AFG10 | QHOUR-2014/03 | sensor | 1 | 1 | 1 | ... | 2 | 2 | ... |
| | | at | 5T11:45 | 5T11:30 | 5T11:15 | ... | 5T11:45 | 5T11:30 | ... |
| | | kwh | 105.38 | 104.33 | 103.39 | | 103.38 | 101.61 | |

- 288 nodes on EC2
- Over 1 Mio writes/sec => 60 Mio writes/min
- Rolling counters, always up to date



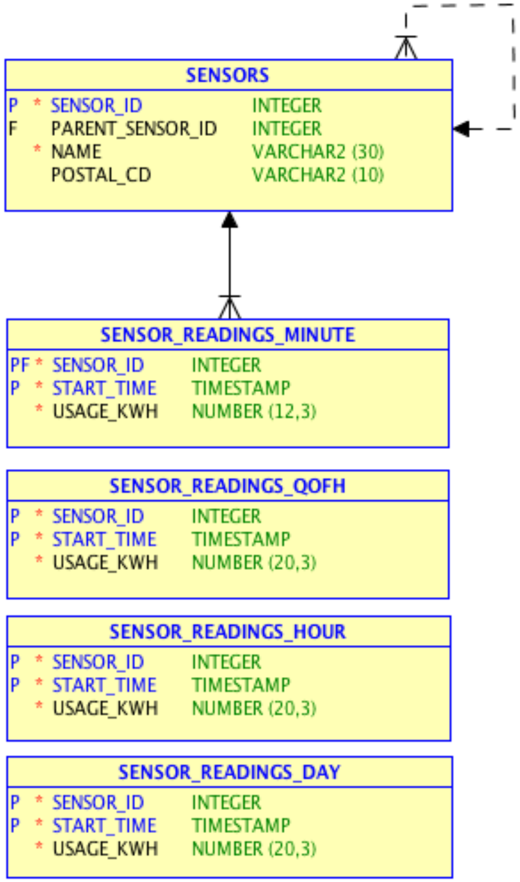
```
select household_id, bucket_id, at_timestamp, sensor_id, kwh_consumed
from meter_reading_timeunit
where household_id = 2dc487f0-b271-11e3-a5e2-0800200c9a66
and bucket_id = 'MINUTE-2014/03/23/11'
and sensor_id = 1
and at_timestamp > '2014-03-23T11:00:00'
order by sensor_id, at_timestamp DESC;
```

# Relational Architecture



- Active Data Guard Configuration
- Global Data Services redirects requests based on
  - Server loads
  - Request type (read/write)
- Reader farm is geographically spread
- Failover/switchover to any node in the reader farm is possible
  - Read services are not affected
  - Write services are unavailable for a short period of time
- Scalability of the write services is the bottleneck of the system

# Relational Data Model



**SENSORS**

| | | | |
|---|---|---|---|
| P | * | SENSOR_ID | INTEGER |
| F | | PARENT_SENSOR_ID | INTEGER |
| | * | NAME | VARCHAR2 (30) |
| | | POSTAL_CD | VARCHAR2 (10) |

**SENSOR_READINGS_MINUTE**

| | | | |
|---|---|---|---|
| PF | * | SENSOR_ID | INTEGER |
| P | * | START_TIME | TIMESTAMP |
| | * | USAGE_KWH | NUMBER (12,3) |

**SENSOR_READINGS_QOFH**

| | | | |
|---|---|---|---|
| P | * | SENSOR_ID | INTEGER |
| P | * | START_TIME | TIMESTAMP |
| | * | USAGE_KWH | NUMBER (20,3) |

**SENSOR_READINGS_HOUR**

| | | | |
|---|---|---|---|
| P | * | SENSOR_ID | INTEGER |
| P | * | START_TIME | TIMESTAMP |
| | * | USAGE_KWH | NUMBER (20,3) |

**SENSOR_READINGS_DAY**

| | | | |
|---|---|---|---|
| P | * | SENSOR_ID | INTEGER |
| P | * | START_TIME | TIMESTAMP |
| | * | USAGE_KWH | NUMBER (20,3) |

- SENSOR_READINGS_...
  - Index-organized tables
  - Daily partitions
- JDBC Batch Merges
  - A transaction per sensor delivery
  - A single network roundtrip to merge 55 readings of a sensor delivery
  - Average between
    - 0.4 Mio tpm (delivery per 5 minutes)
    - 120 Mio tpm (delivery per second)
  - Top TPC-C Benchmark: 8.5 Mio tpm
- Batch job to aggregate readings every 15 minutes, avoiding intermediate results (updates)
  - Quarter of hour (5760 times a day)
  - Hour (24 times a day)
  - Day (once a day)

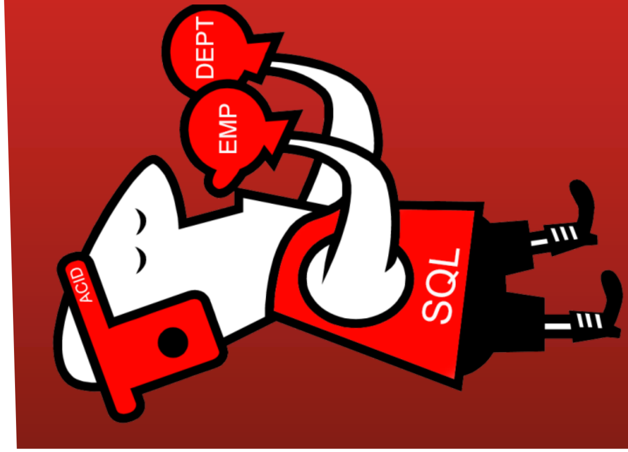
# Query Sensor Data – The SQL Way

Use aggregate tables to change granularity (quarter of hours, hours, days)

```
SELECT sensor_id, start_time, usage_kwh
  FROM sensor_readings_minute
 WHERE sensor_id IN (SELECT sensor_id
                       FROM sensors
                      WHERE sensor_id = :p_parent_sensor
                            OR parent_sensor_id = :p_parent_sensor)
   AND start_time BETWEEN :p_from AND :p_to
 ORDER BY sensor_id, start_time;
```

## Smart Meter

0 – 1



# Round 2

# Order Entry

# Order Entry - Change Quantity in Stock

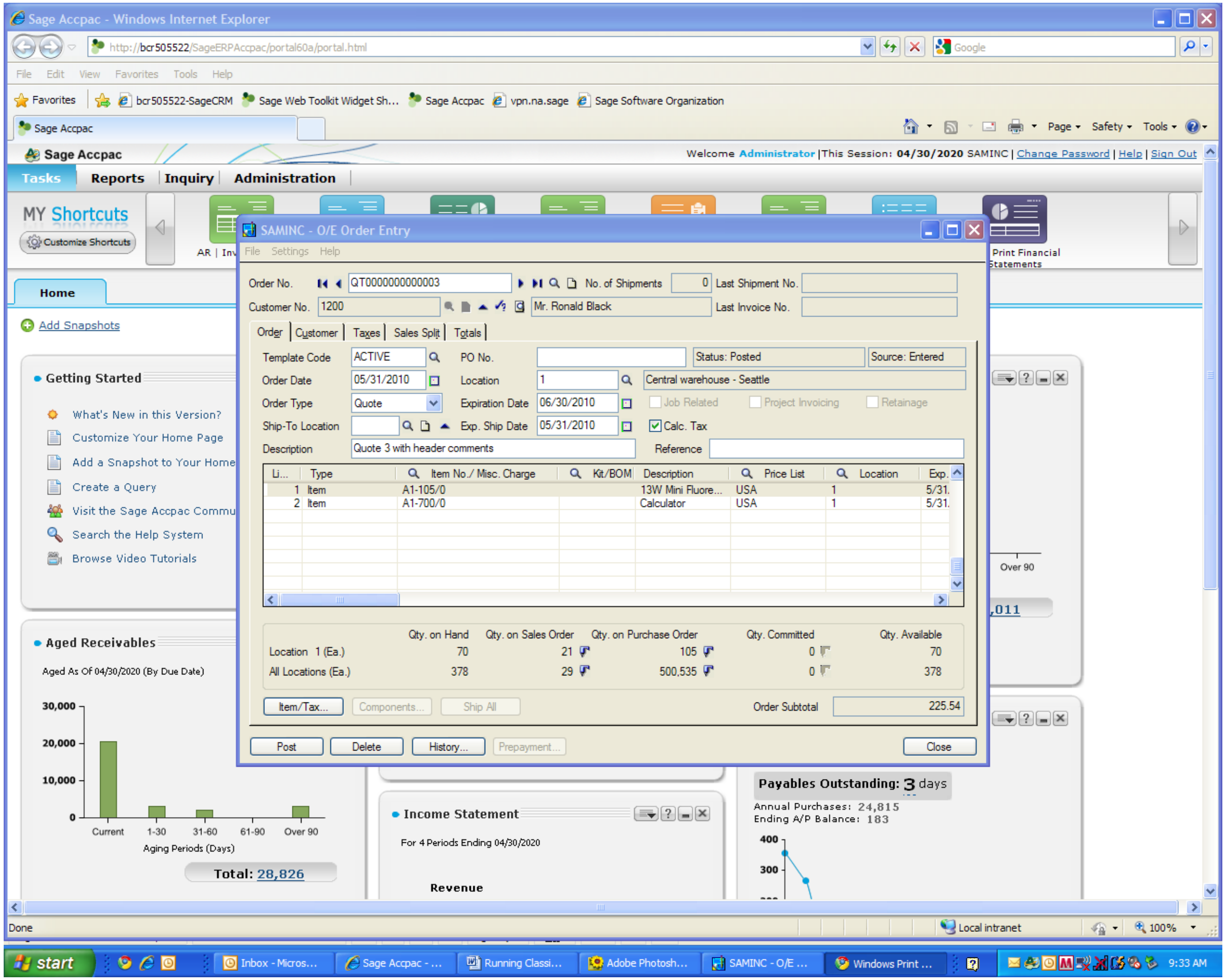
## Starting Position

- An incomplete order with multiple order items is stored in the database
  - Order status is "incomplete"
  - Data is complete, just the final approval is missing

## Use Case Description

- Change the quantity in stock of all ordered products
  - When order status changes from "incomplete" to "complete"
  - When order status changes from "complete" to "cancelled"
- Ensure that the quantity in stock is correctly amended
  - No lost updates or similar

# Order Entry



Welcome Administrator | This Session: 04/30/2020 SAMINC | Change Password | Help | Sign Out

SAMINC - O/E Order Entry

Order No. QT0000000000003 No. of Shipments 0 Last Shipment No.

Customer No. 1200 Mr. Ronald Black Last Invoice No.

Template Code ACTIVE PO No. Status: Posted Source: Entered

Order Date 05/31/2010 Location 1 Central warehouse - Seattle

Order Type Quote Expiration Date 06/30/2010 □ Job Related □ Project Invoicing □ Retainage

Ship-To Location Exp. Ship Date 05/31/2010 ☑ Calc. Tax

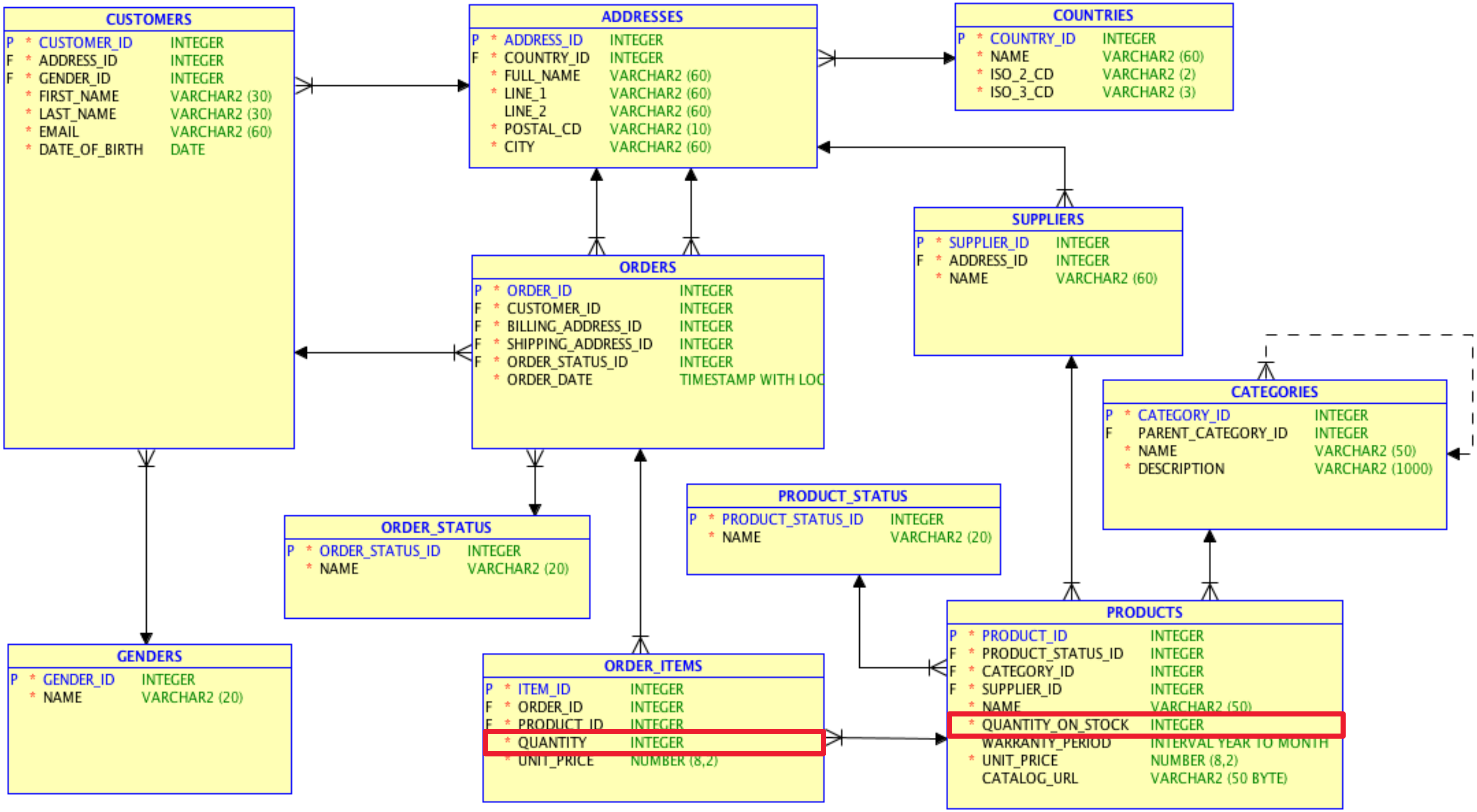
Description Quote 3 with header comments Reference

| Li... | Type | Item No./ Misc. Charge | Kit/BOM | Description | Price List | Location | Exp. |
|---|---|---|---|---|---|---|---|
| 1 | Item | A1-105/0 | | 13W Mini Fluore... | USA | 1 | 5/31. |
| 2 | Item | A1-700/0 | | Calculator | USA | 1 | 5/31. |

| | Qty. on Hand | Qty. on Sales Order | Qty. on Purchase Order | Qty. Committed | Qty. Available |
|---|---|---|---|---|---|
| Location 1 (Ea.) | 70 | 21 | 105 | 0 | 70 |
| All Locations (Ea.) | 378 | 29 | 500,535 | 0 | 378 |

Order Subtotal 225.54

# Relational Model



CUSTOMERS

| | | | |
|---|---|---|---|
| P | * | CUSTOMER_ID | INTEGER |
| F | * | ADDRESS_ID | INTEGER |
| F | * | GENDER_ID | INTEGER |
| | * | FIRST_NAME | VARCHAR2 (30) |
| | * | LAST_NAME | VARCHAR2 (30) |
| | * | EMAIL | VARCHAR2 (60) |
| | * | DATE_OF_BIRTH | DATE |

ADDRESSES

| | | | |
|---|---|---|---|
| P | * | ADDRESS_ID | INTEGER |
| F | * | COUNTRY_ID | INTEGER |
| | * | FULL_NAME | VARCHAR2 (60) |
| | * | LINE_1 | VARCHAR2 (60) |
| | | LINE_2 | VARCHAR2 (60) |
| | * | POSTAL_CD | VARCHAR2 (10) |
| | * | CITY | VARCHAR2 (60) |

COUNTRIES

| | | | |
|---|---|---|---|
| P | * | COUNTRY_ID | INTEGER |
| | * | NAME | VARCHAR2 (60) |
| | * | ISO_2_CD | VARCHAR2 (2) |
| | * | ISO_3_CD | VARCHAR2 (3) |

SUPPLIERS

| | | | |
|---|---|---|---|
| P | * | SUPPLIER_ID | INTEGER |
| F | * | ADDRESS_ID | INTEGER |
| | * | NAME | VARCHAR2 (60) |

ORDERS

| | | | |
|---|---|---|---|
| P | * | ORDER_ID | INTEGER |
| F | * | CUSTOMER_ID | INTEGER |
| F | * | BILLING_ADDRESS_ID | INTEGER |
| F | * | SHIPPING_ADDRESS_ID | INTEGER |
| F | * | ORDER_STATUS_ID | INTEGER |
| | * | ORDER_DATE | TIMESTAMP WITH LOC |

CATEGORIES

| | | | |
|---|---|---|---|
| P | * | CATEGORY_ID | INTEGER |
| F | | PARENT_CATEGORY_ID | INTEGER |
| | * | NAME | VARCHAR2 (50) |
| | * | DESCRIPTION | VARCHAR2 (1000) |

PRODUCT_STATUS

| | | | |
|---|---|---|---|
| P | * | PRODUCT_STATUS_ID | INTEGER |
| | * | NAME | VARCHAR2 (20) |

ORDER_STATUS

| | | | |
|---|---|---|---|
| P | * | ORDER_STATUS_ID | INTEGER |
| | * | NAME | VARCHAR2 (20) |

GENDERS

| | | | |
|---|---|---|---|
| P | * | GENDER_ID | INTEGER |
| | * | NAME | VARCHAR2 (20) |

ORDER_ITEMS

| | | | |
|---|---|---|---|
| P | * | ITEM_ID | INTEGER |
| F | * | ORDER_ID | INTEGER |
| F | * | PRODUCT_ID | INTEGER |
| | * | QUANTITY | INTEGER |
| | * | UNIT_PRICE | NUMBER (8,2) |

PRODUCTS

| | | | |
|---|---|---|---|
| P | * | PRODUCT_ID | INTEGER |
| F | * | PRODUCT_STATUS_ID | INTEGER |
| F | * | CATEGORY_ID | INTEGER |
| F | * | SUPPLIER_ID | INTEGER |
| | * | NAME | VARCHAR2 (50) |
| | * | QUANTITY_ON_STOCK | INTEGER |
| | | WARRANTY_PERIOD | INTERVAL YEAR TO MONTH |
| | * | UNIT_PRICE | NUMBER (8,2) |
| | | CATALOG_URL | VARCHAR2 (50 BYTE) |

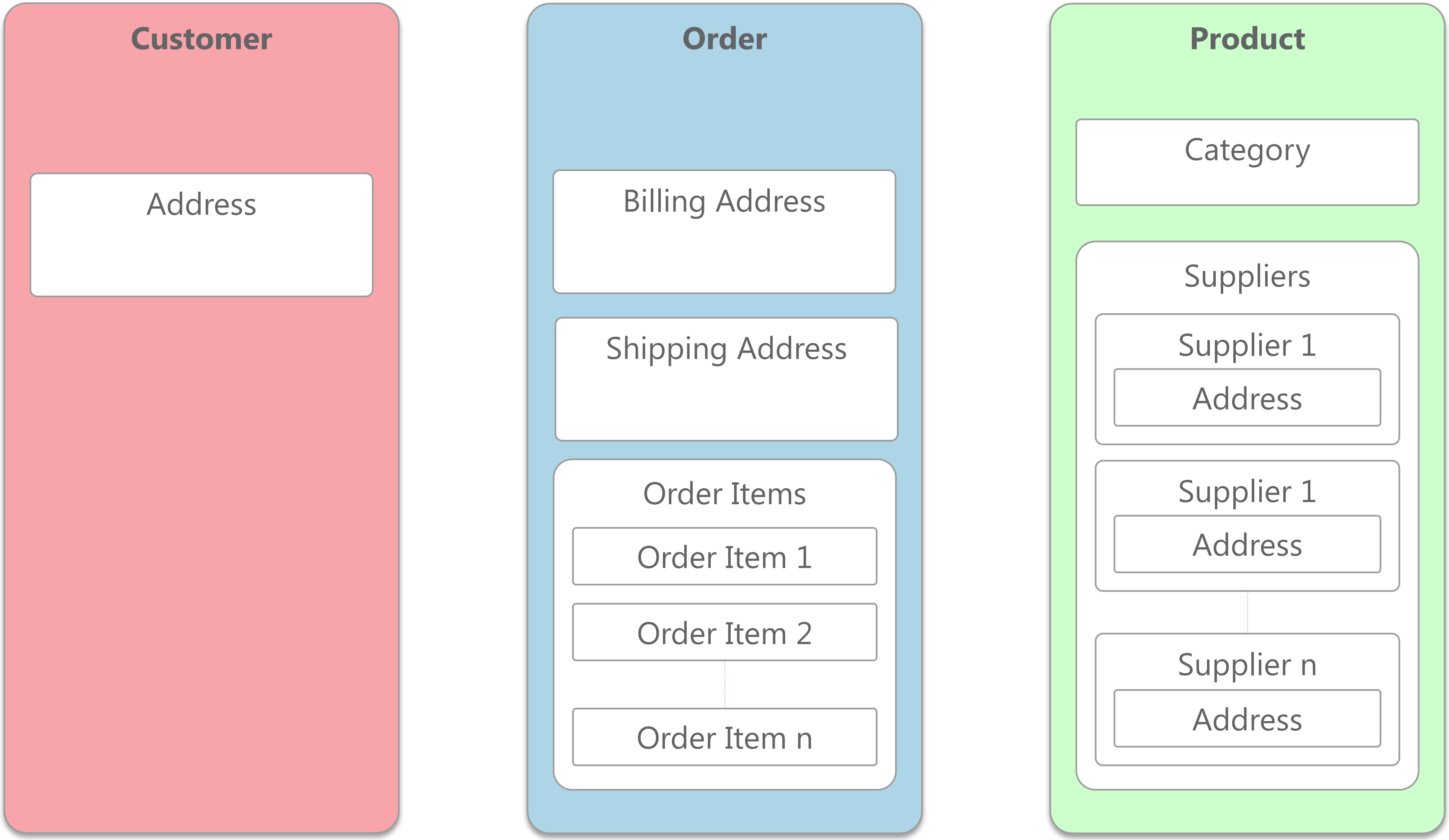
# Change Quantity in Stock – The SQL Way

```
UPDATE ORDERS
   SET order_status = :p_value_for_complete
 WHERE order_id = :p_order_id;

MERGE INTO PRODUCTS t
USING (SELECT product_id,
              SUM(quantity) AS quantity
         FROM order_items
        WHERE order_id = :p_order_id
        GROUP BY product_id) s
   ON (t.product_id = s.product_id)
 WHEN MATCHED THEN
    UPDATE SET t.quantity_on_stock =
               t.quantity_on_stock - s.quantity;

COMMIT;
```

# MongoDB NoSQL Store

- Document Store
- Developed by 10gen, now MongoDB Inc.
- Professional grade support by MongoDB Inc.
- Main Features
  - JSON Data Model with Dynamic Schemas
  - Auto-Sharding for Horizontal Scalability
  - Built-In Replication for High Availability
  - Rich Secondary Indexes, including geospatial and TTL indexes
  - Text Search
  - Aggregation Framework & Native MapReduce

# MongoDB Document Datamodel (Aggregate Pattern)



# Change Quantity in Stock – The MongoDB Way

```
db.orders.update( { orderId: 1},
                  { $set : { orderStatus: "COMPLETE" } },
                  { multi: false } );
```

```
db.orders.find ( { orderId: 1}
);
```

```
ForEach orderItems.item {

      db.products.update( { productId : 101 },
                          { $inc : { quantity: -10 } },
                          { multi: false }
                        );

}
```

# Order Entry - Ad Hoc Analysis

## Starting Position

- A lot of data is available in the system
- Some sales volume analysis are wanted

## Use Case Description

- Create a report to shows all sales for a year per country
- Create a report for the 5 top-selling products for a year

# Sales Volume per Country – The SQL Way

```
SELECT c.name as country_name,
       SUM(i.quantity * i.unit_price) AS sales_volume
  FROM order_items i
 INNER JOIN orders o
    ON o.order_id = i.order_id
 INNER JOIN addresses a
    ON a.address_id = o.shipping_address_id
 INNER JOIN countries c
    ON c.country_id = a.country_id
 WHERE o.order_date >= DATE '2013-01-01'
       AND o.order_date < DATE '2014-01-01'
 GROUP BY c.name
 ORDER BY 2 DESC;
```

# 5 Top-Selling Products – The SQL Way

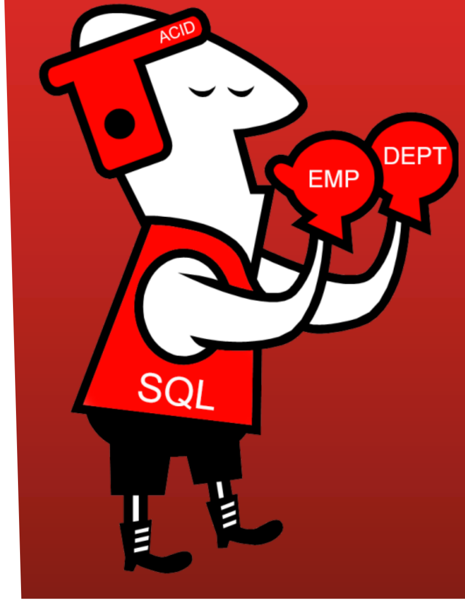
```
SELECT p.name AS product_name,
       SUM(i.quantity * i.unit_price) AS sales_volume
  FROM order_items i
 INNER JOIN orders o
    ON o.order_id = i.order_id
 INNER JOIN products p
    ON p.product_id = i.product_id
 WHERE o.order_date > DATE '2013-01-01'
       AND o.order_date <= DATE '2014-01-01'
       AND o.order_status = :p_value_for_complete
 GROUP BY p. name
 ORDER BY 2 DESC
 FETCH FIRST 5 ROWS WITH TIES;
```

# Sales Volume per Country – The MongoDB Way

```
var mapFunction = function() {
    for (var idx = 0; idx < this.orderItems.length; idx++) {
        var value = this.orderItems[idx].unitPrice *
                         this.orderItems[idx].quantity;
        emit(this.shippingAddress.country, value);
    } };
var reduceFunction = function(name, valuesPrices) {
    return Array.sum(valuesPrices);
};
db.orders.mapReduce(mapFunction,
    reduceFunction,
    { out : {inline:1},
      query: { orderStatus: "COMPLETE",
               orderDate: { $gt: ISODate("2014-01-01"),
                            $lt: ISODate("2014-04-01") }
             }
    });
```

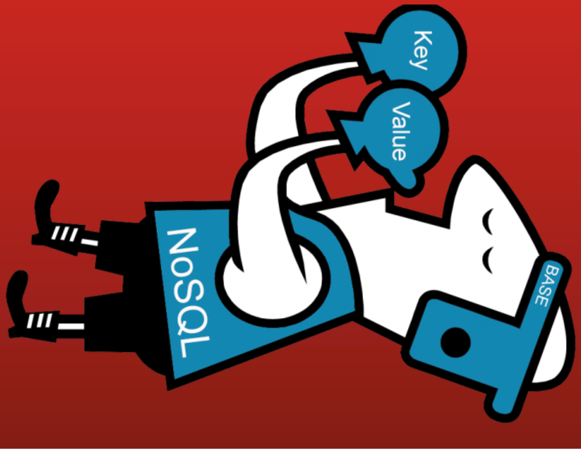
# 5 Top-Selling Products – The MongoDB Way

```
db.orders.aggregate([
    { $match : {
           orderStatus: "COMPLETE",
           orderDate: { $gt: ISODate("2014-01-01"),
                        $lt: ISODate("2014-04-01") }
                      } },
    { $unwind : "$orderItems" },
    { $project : { _id: 0,
           productId: "$orderItems.productId",
           total : { $multiply : ["$orderItems.quantity",
                                "$orderItems.unitPrice”] }
    } },
    { $group : { _id: "$productId",
                total : { $sum : "$total"} } },
    { $sort : { total: -1 }},
    { $limit : 5 }
])
```



# Order Entry

## 1 – 1



# Round 3

# Spotify

# Spotify

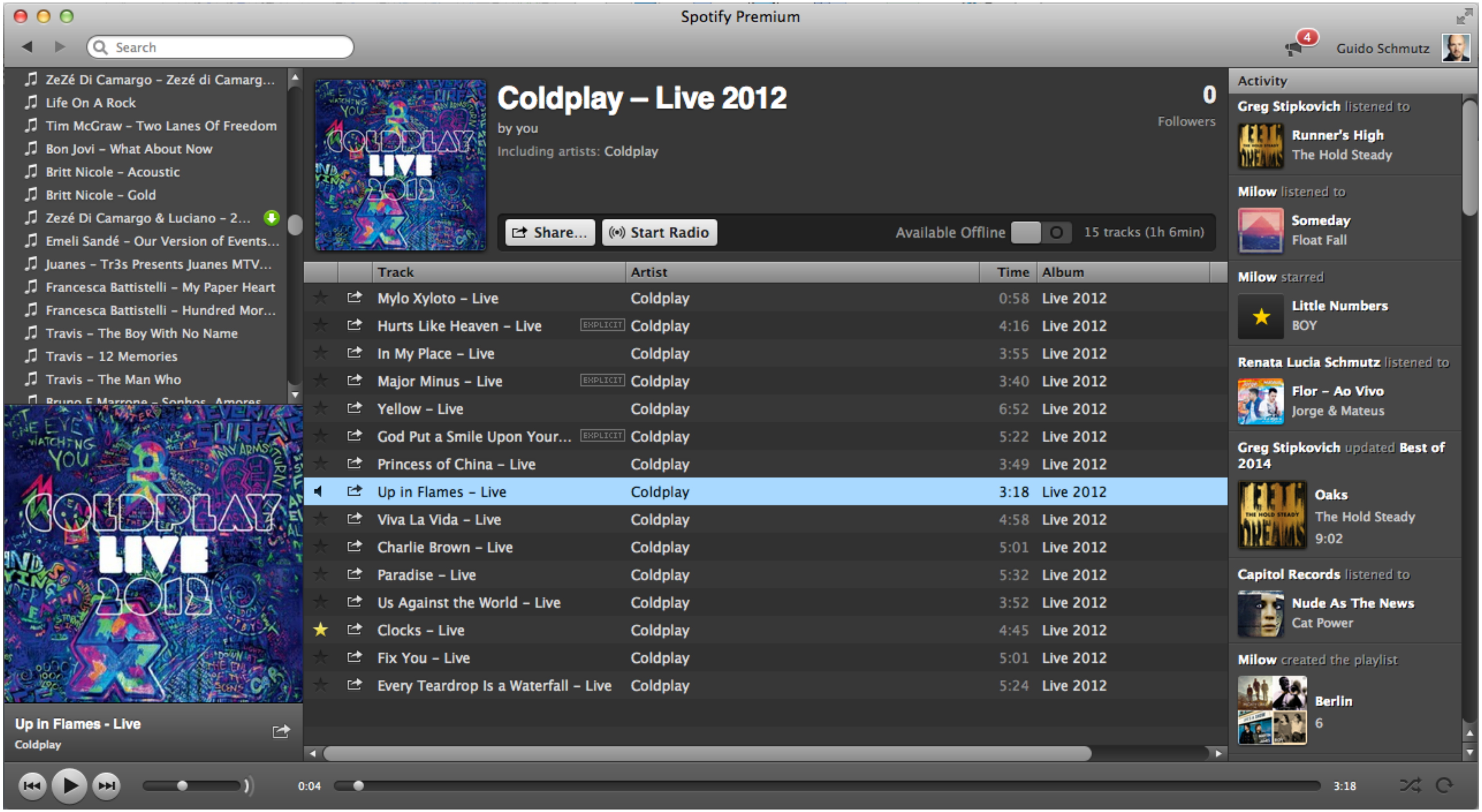
## Starting Position

- Music service which is available worldwide
  - Over 20’000’000 music tracks available
  - Millions of users
  - Each user has dozens of playlists
- Many AP but also some CA use cases

## Use Case Description

- Playlist, Showing Ads, Following Artists … are all uses cases which have to be highly available, and accessible worldwide
  - Needs to be distributed to be fast
  - Service should be available even if a partition happen (due to network failure/machine failure)
- First time subscription and subscription renewal must be absolutely consistent
  - Customer should only pay once!

# Spotify



Spotify Premium

Search

4 Guido Schmutz

- ZeZé Di Camargo – Zezé di Camarg…
- Life On A Rock
- Tim McGraw – Two Lanes Of Freedom
- Bon Jovi – What About Now
- Britt Nicole – Acoustic
- Britt Nicole – Gold
- Zezé Di Camargo & Luciano – 2…
- Emeli Sandé – Our Version of Events…
- Juanes – Tr3s Presents Juanes MTV…
- Francesca Battistelli – My Paper Heart
- Francesca Battistelli – Hundred Mor…
- Travis – The Boy With No Name
- Travis – 12 Memories
- Travis – The Man Who
- Bruno E Marrone – Sonhos, Amores

## Coldplay – Live 2012

by you

Including artists: Coldplay

0 Followers

Share… | Start Radio | Available Offline | 15 tracks (1h 6min)

| | | Track | | Artist | Time | Album |
|---|---|---|---|---|---|---|
| ★ | | Mylo Xyloto – Live | | Coldplay | 0:58 | Live 2012 |
| ★ | | Hurts Like Heaven – Live | EXPLICIT | Coldplay | 4:16 | Live 2012 |
| ★ | | In My Place – Live | | Coldplay | 3:55 | Live 2012 |
| ★ | | Major Minus – Live | EXPLICIT | Coldplay | 3:40 | Live 2012 |
| ★ | | Yellow – Live | | Coldplay | 6:52 | Live 2012 |
| ★ | | God Put a Smile Upon Your… | EXPLICIT | Coldplay | 5:22 | Live 2012 |
| ★ | | Princess of China – Live | | Coldplay | 3:49 | Live 2012 |
| ◀ | | Up in Flames – Live | | Coldplay | 3:18 | Live 2012 |
| ★ | | Viva La Vida – Live | | Coldplay | 4:58 | Live 2012 |
| ★ | | Charlie Brown – Live | | Coldplay | 5:01 | Live 2012 |
| ★ | | Paradise – Live | | Coldplay | 5:32 | Live 2012 |
| ★ | | Us Against the World – Live | | Coldplay | 3:52 | Live 2012 |
| ★ | | Clocks – Live | | Coldplay | 4:45 | Live 2012 |
| ★ | | Fix You – Live | | Coldplay | 5:01 | Live 2012 |
| ★ | | Every Teardrop Is a Waterfall – Live | | Coldplay | 5:24 | Live 2012 |

Activity

**Greg Stipkovich** listened to
Runner's High
The Hold Steady

**Milow** listened to
Someday
Float Fall

**Milow** starred
Little Numbers
BOY

**Renata Lucia Schmutz** listened to
Flor – Ao Vivo
Jorge & Mateus

**Greg Stipkovich** updated **Best of 2014**
Oaks
The Hold Steady
9:02

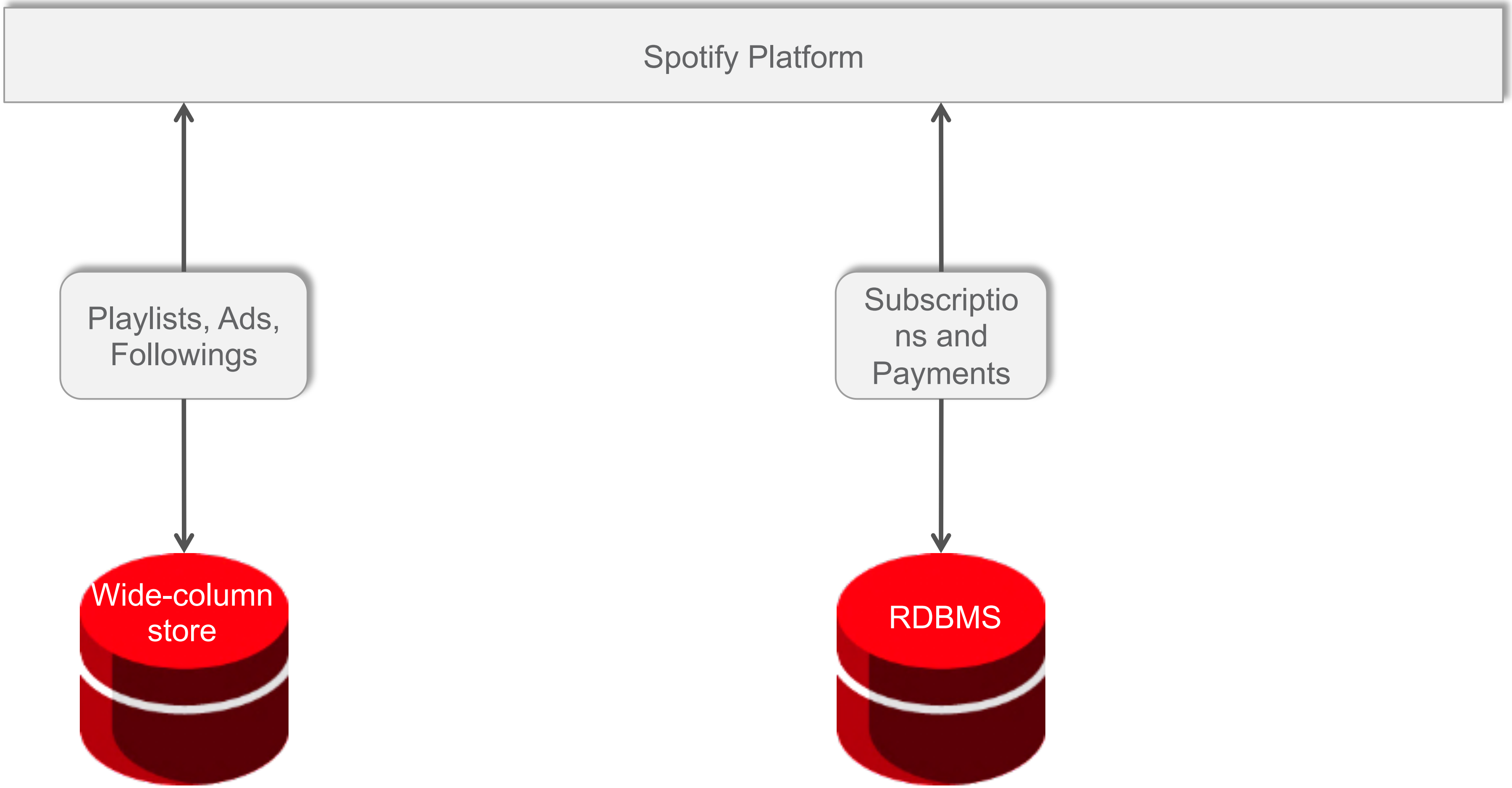
**Capitol Records** listened to
Nude As The News
Cat Power

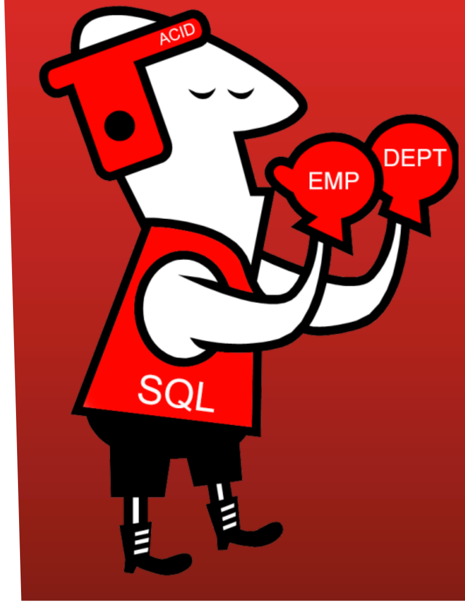
**Milow** created the playlist
Berlin
6

Up in Flames - Live
Coldplay

0:04 3:18

# Polyglot Persistence – SQL And NoSQL





# Spotify

2 – 2

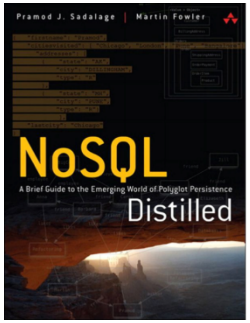
Draw!

# Agenda

1. Motivation
2. Overview of SQL and NoSQL Data Stores
3. Use Cases – Let's Get Ready to Rumble
4. Recommended Reads
5. Core Messages

# Further information ...

- http://martinfowler.com/books/nosql.html
- http://www.manning.com/mccreary/
- http://highlyscalable.wordpress.com
- http://nosql-database.org
- http://db-engines.com/

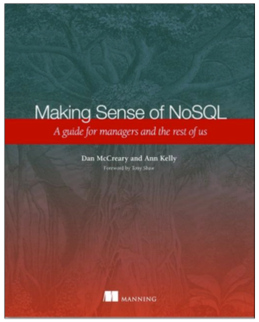


Pramod J. Sadalage | Martin Fowler

NoSQL

A Brief Guide to the Emerging World of Polyglot Persistence

Distilled



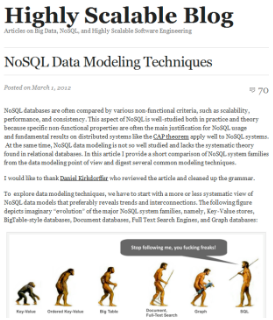
Making Sense of NoSQL

*A guide for managers and the rest of us*

Dan McCreary and Ann Kelly

Foreword by Tony Shaw

MANNING



**Highly Scalable Blog**

Articles on Big Data, NoSQL, and Highly Scalable Software Engineering

NoSQL Data Modeling Techniques

*Posted on March 1, 2012*

70

NoSQL databases are often compared by various non-functional criteria, such as scalability, performance, and consistency. This aspect of NoSQL is well-studied both in practice and theory because specific non-functional properties are often the main justification for NoSQL usage and fundamental results on distributed systems like the CAP theorem apply well to NoSQL systems. At the same time, NoSQL data modeling is not so well studied and lacks the systematic theory found in relational databases. In this article I provide a short comparison of NoSQL system families from the data modeling point of view and digest several common modeling techniques.

I would like to thank Daniel Kirkdorffer who reviewed the article and cleaned up the grammar.

To explore data modeling techniques, we have to start with a more or less systematic view of NoSQL data models that preferably reveals trends and interconnections. The following figure depicts imaginary "evolution" of the major NoSQL system families, namely, Key-Value stores, BigTable-style databases, Document databases, Full Text Search Engines, and Graph databases:

Stop following me, you fucking freaks!
Key-Value
Ordered Key-Value
Big Table
Document, Full-Text Search
Graph
SQL

# Agenda

1. Motivation
2. Overview of SQL and NoSQL Data Stores
3. Use Cases – Let's Get Ready to Rumble
4. Recommended Reads
5. Core Messages

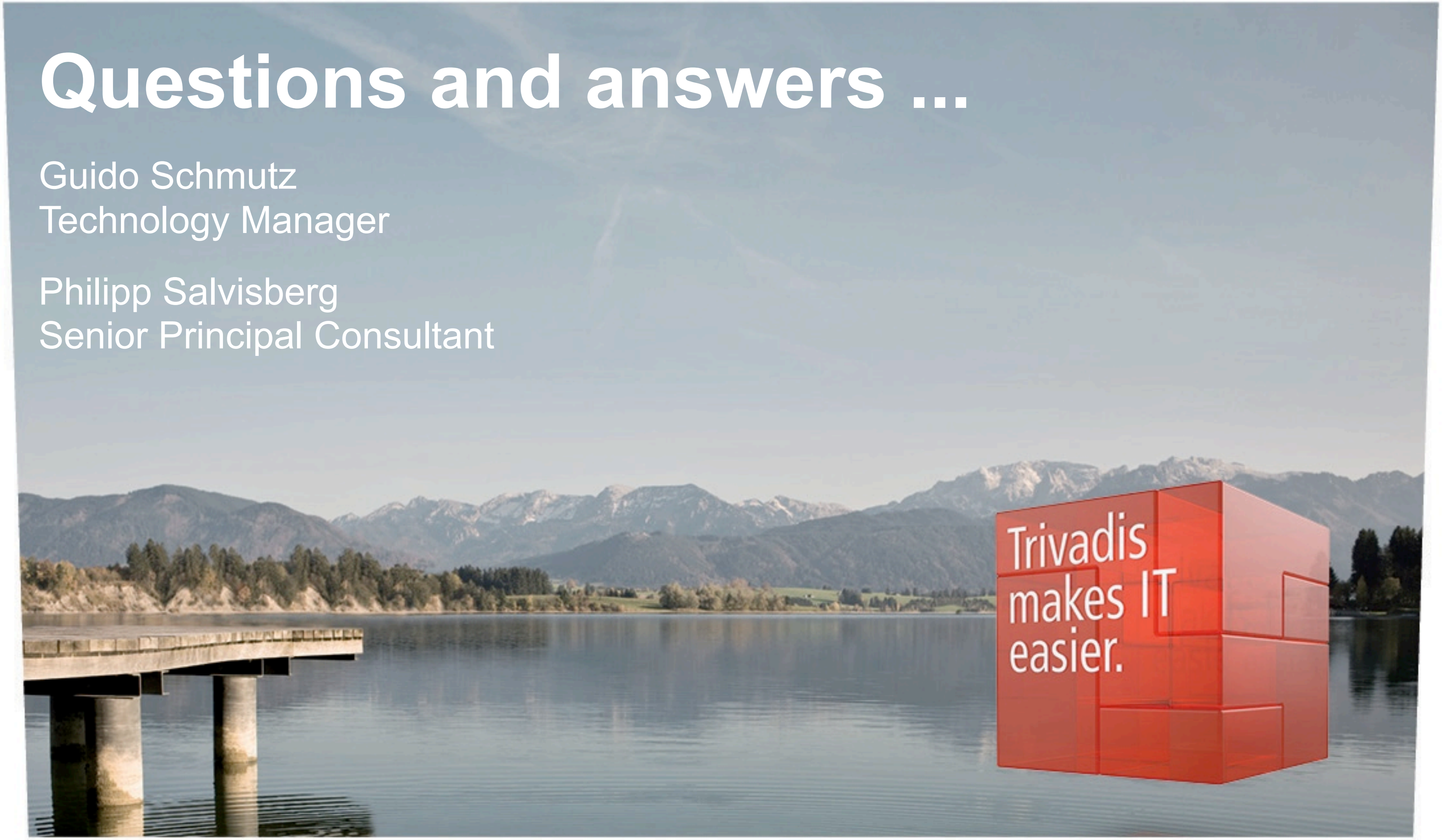
# Core Messages

- We will see a major consolation in the NoSQL area
- SQL is and stays important
- Polyglot persistence will be part of every solution design in the near future
- Enterprise capabilities are required
  - Tooling (monitoring, backup & recovery, data security, …)
  - Organization, skills
  - Opportunity for cloud based solutions

# Questions and answers ...

Guido Schmutz
Technology Manager

Philipp Salvisberg
Senior Principal Consultant



BASEL BERN BRUGES LAUSANNE ZÜRICH DÜSSELDORF FRANKFURT A.M. FREIBURG I.BR. HAMBURG MUNICH STUTTGART VIENNA