Get Back in Control of your SQL

SQL and Java could work together so much better if we only let them.

Me – @lukaseder

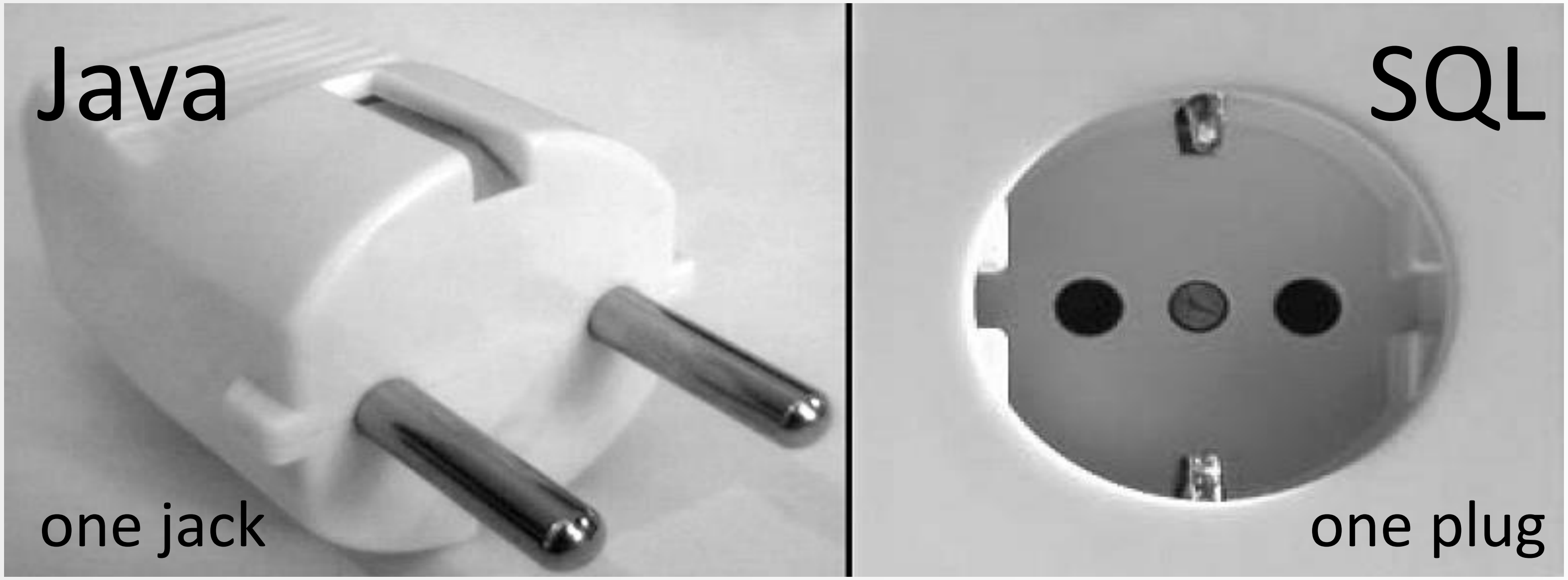
- Founder and CEO at Data Geekery
- SQL Aficionado
- Java Aficionado

“SQL is a device whose mystery is only exceeded by its power!”

Legal Disclaimer

THE FOLLOWING IS COMMUNICATED TO YOU SOLELY FOR ENTERTAINMENT PURPOSES. NO ONE SANE WOULD BELIEVE A GUY WHO CLAIMS HE IS A SQL AFICIONADO OR WORSE WHO CLAIMS THAT SQL IS ANYTHING NEAR BEAUTIFUL. IF YOU STILL FIND THE FOLLOWING INTERESTING AND IF YOU BASE YOUR PURCHASING DECISIONS UPON THAT, YOU DEFINITELY NEED PROFESSIONAL HELP. WE ACTUALLY PROVIDE SUCH HELP.

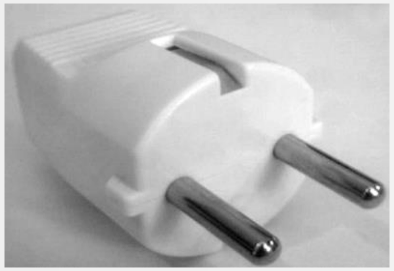
SQL and Java – in theory



In this metaphor, electricity is the data (SQL) that flows into your appliance / application (Java)

SQL and Java – in practice

Java



one jack

SQL

lots of plugs

Images from: http://en.wikipedia.org/wiki/AC_power_plugs_and_sockets. License: public domain

JDBC

```
PreparedStatement stmt = connection.prepareStatement(
  "SELECT text FROM products WHERE cust_id = ? AND value < ?");
stmt.setInt(1, custID);
stmt.setBigDecimal(2, BigDecimal.ZERO);
ResultSet rs = stmt.executeQuery();

while (rs.next()) {
  System.out.println(rs.getString("TEXT"));
}
```

JDBC – the naked truth

```
PreparedStatement stmt = connection.prepareStatement(
  "SELECT p.text txt" +
  (isAccount ? ", NVL(a.type, ?) " : "") +
  "FROM products p " +
  (isAccount ? " INNER JOIN accounts a USING (prod_id) " : "") +
  " WHERE p.cust_id = ? AND p.value < ?" +
  (isAccount ? " AND a.type LIKE '%" + type + "%'" : "");
stmt.setInt(1, defaultType);
stmt.setInt(2, custID);
stmt.setBigDecimal(3, BigDecimal.ZERO);
ResultSet rs = stmt.executeQuery();

while (rs.next()) {
  Clob clob = rs.getClob("TEXT");
  System.out.println(clob.getSubString(1, (int) clob.length());
}

rs.close();
stmt.close();
```

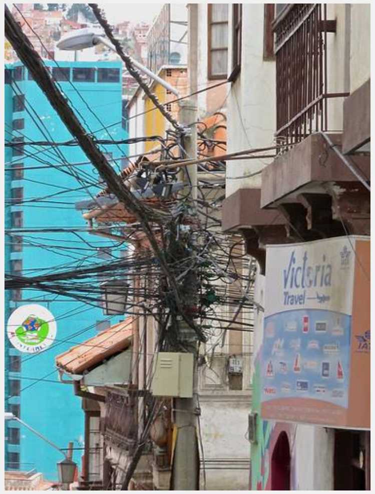
JDBC – the naked truth

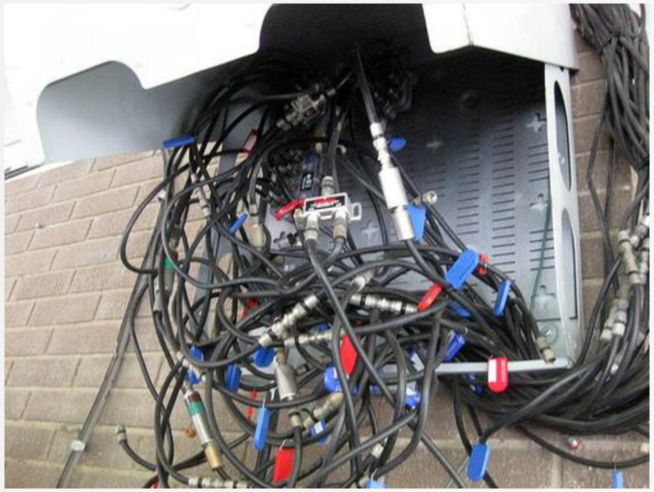
```
PreparedStatement stmt = connection.prepareStatement(           //
  "SELECT p.text txt" +                                          //
  (isAccount ? ", NVL(a.type, ?) " : "") +                       //
  "FROM products p " +                                           // Syntax error when isAccount == false
  (isAccount ? " INNER JOIN accounts a USING (prod_id) " : "") + //
  " WHERE p.cust_id = ? AND p.value < ?" +                       //
  (isAccount ? " AND a.type LIKE '%" + type + "%'" : "");        // Syntax error and SQL injection possible
stmt.setInt(1, defaultType);                                     // Wrong bind index
stmt.setInt(2, custID);                                          //
stmt.setBigDecimal(3, BigDecimal.ZERO);                          //
ResultSet rs = stmt.executeQuery();                              //

while (rs.next()) {                                              //
  Clob clob = rs.getClob("TEXT");                                // Wrong column name
  System.out.println(clob.getSubString(1, (int) clob.length());  // ojdbc6: clob.free() should be called
}                                                                //

rs.close();                                                      // close() not really in finally block
stmt.close();                                                    //
```

What JDBC means for developers





With JDBC, your developers have to do a lot of manual, error-prone (dangerous) and inefficient work

Images from Flickr. To the left by: Matthew Straubmuller, Greg Grossmeier. License: CC BY SA 2.0. Electric Engineers to the right copyright by Marco Sarli, all rights reserved.

EJB 2.0 EntityBeans

```
public interface CustomerRequest extends EJBObject {
  BigInteger getId();
  String getText();
  void setText(String text);
  @Override
  void remove();
}

public interface CustomerRequestHome extends EJBHome {
  CustomerRequest create(BigInteger id);
  CustomerRequest find(BigInteger id);
}
```

EJB 2.0 – the naked truth

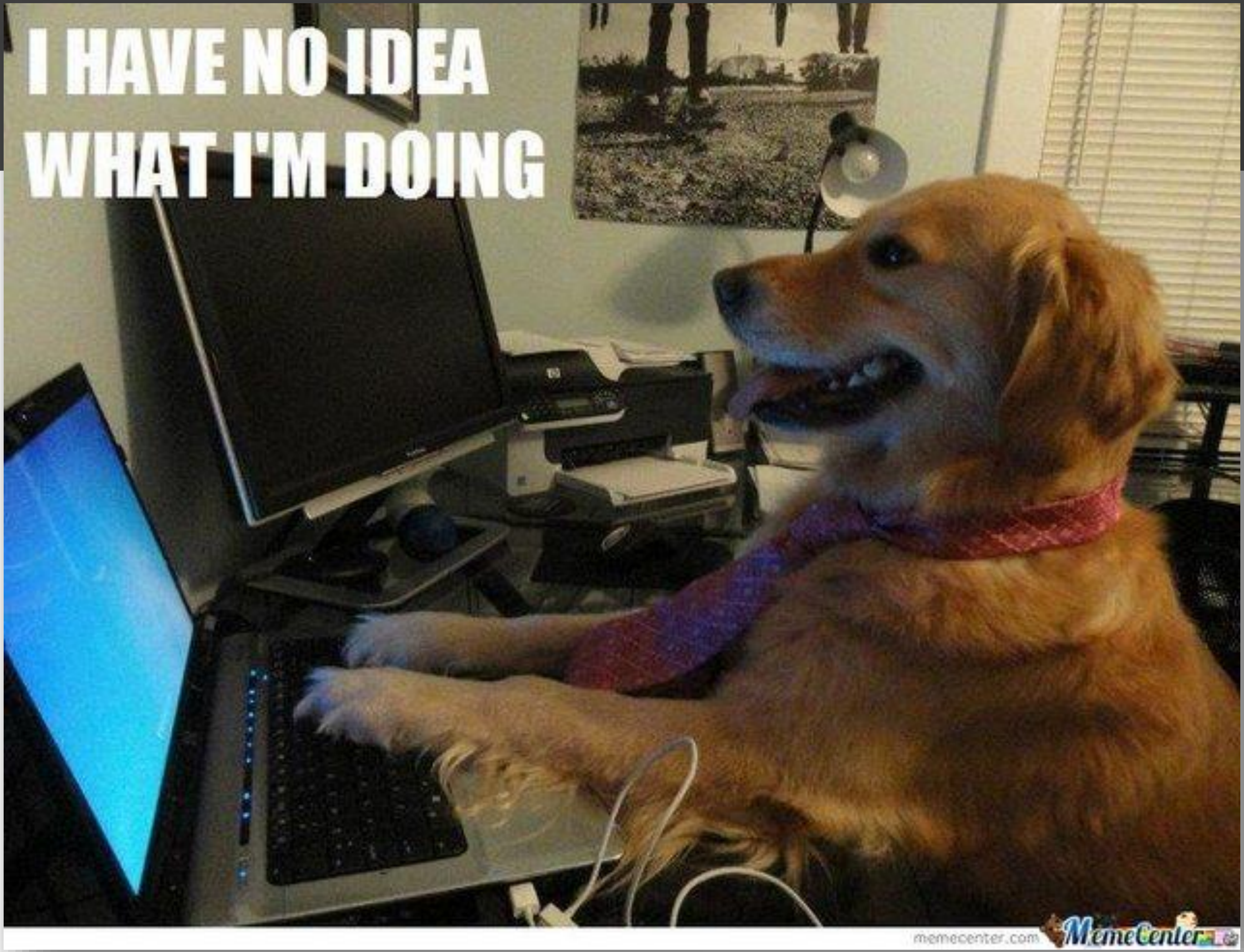
```
<weblogic-enterprise-bean>
  <ejb-name>com.example.CustomerRequestHome</ejb-name>
  <entity-descriptor>
    <pool>
      <max-beans-in-free-pool>100</max-beans-in-free-pool>
    </pool>
    <entity-cache>
      <max-beans-in-cache>500</max-beans-in-cache>
      <idle-timeout-seconds>10</idle-timeout-seconds>
      <concurrency-strategy>Database</concurrency-strategy>
    </entity-cache>
    <persistence>
      <delay-updates-until-end-of-tx>True</delay-updates-until-end-of-tx>
    </persistence>
    <entity-clustering>
      <home-is-clusterable>False</home-is-clusterable>
      <home-load-algorithm>round-robin</home-load-algorithm>
    </entity-clustering>
  </entity-descriptor>
  <transaction-descriptor/>
  <enable-call-by-reference>True</enable-call-by-reference>
  <jndi-name>com.example.CustomerRequestHome</jndi-name>
</weblogic-enterprise-bean>
```

EJB 2.0 – the naked truth

o_O

```
<pool>
  <max-beans-in-free-pool>100</max-beans-in-free-pool>
</pool>
<entity-cache>
  <max-beans-in-cache>500</max-beans-in-cache>
  <idle-timeout-seconds>10</idle-timeout-seconds>
  <concurrency-strategy>Database</concurrency-strategy>
</entity-cache>
<persistence>
  <delay-updates-until-end-of-tx>True</delay-updates-…>
</persistence>
```

EJB 2.0



JPA and EJB 3.0

```java
EntityManager em = factory.createEntityManager();
em.getTransaction().begin();

em.persist(new Event("Conference", new Date());
em.persist(new Event("After Party", new Date());

List result = em.createQuery("from Event").getResultList();
for (Event event : (List<Event>) result) {
  System.out.println("Event : " + event.getTitle());
}

em.getTransaction().commit();
em.close();
```

EJB 3.0 – the naked truth

```
@Entity @Table(name = "EVENTS")
public class Event {
  private Long id;
  private String title;
  private Date date;

  @Id @GeneratedValue(generator = "increment")
  @GenericGenerator(name = "increment", strategy = "increment")
  public Long getId() { /* … */ }

  @Temporal(TemporalType.TIMESTAMP)
  @Column(name = "EVENT_DATE")
  public Date getDate() { /* … */ }
```

EJB 3.0 – Yep, annotations!

```
@OneToMany(mappedBy = "destCustomerId")
@ManyToMany
@Fetch(FetchMode.SUBSELECT)
@JoinTable(
    name = "customer_dealer_map",
    joinColumns = {
        @JoinColumn(name = "customer_id", referencedColumnName = "id")
    },
    inverseJoinColumns = {
        @JoinColumn(name = "dealer_id", referencedColumnName = "id")
    }
)
private Collection dealers;
```

Found at http://stackoverflow.com/q/17491912/521799

JPA 3.0 Preview – Annotatiomania™

```
@SeveralAndThenNothing @MaybeThisDoesSomething
@TweakThisWithThat(
    tweak = {
        @TweakID(name = "id", preferredValue = 1839),
        @TweakID(name = "test", preferredValue = 839),
        @TweakID(name = "test.old", preferredValue = 34),
    },
    inCaseOf = {
        @ConditionalXMLFiltering(run = 5),
    }
)
@OneToMany @OneToManyMore @AnyOne @AnyBody @DoesThisEvenMeanAnything @DoesAnyoneEvenReadThis
@ManyToMany @Many @AnnotationsTotallyRock @DeclarativeProgrammingRules @NoMoreExplicitAlgorithms
@Fetch @FetchMany @FetchWithDiscriminator(name = "no_name")
@JoinTable(joinColumns = {
    @JoinColumn(name = "customer_id", referencedColumnName = "id")
})
@PrefetchJoinWithDiscriminator @JustTrollingYouKnow @LOL
@IfJoiningAvoidHashJoins @ButUseHashJoinsWhenMoreThan(records = 1000)
@XmlDataTransformable @SpringPrefechAdapter
private Collection employees;
```

Might not be true

What's next?

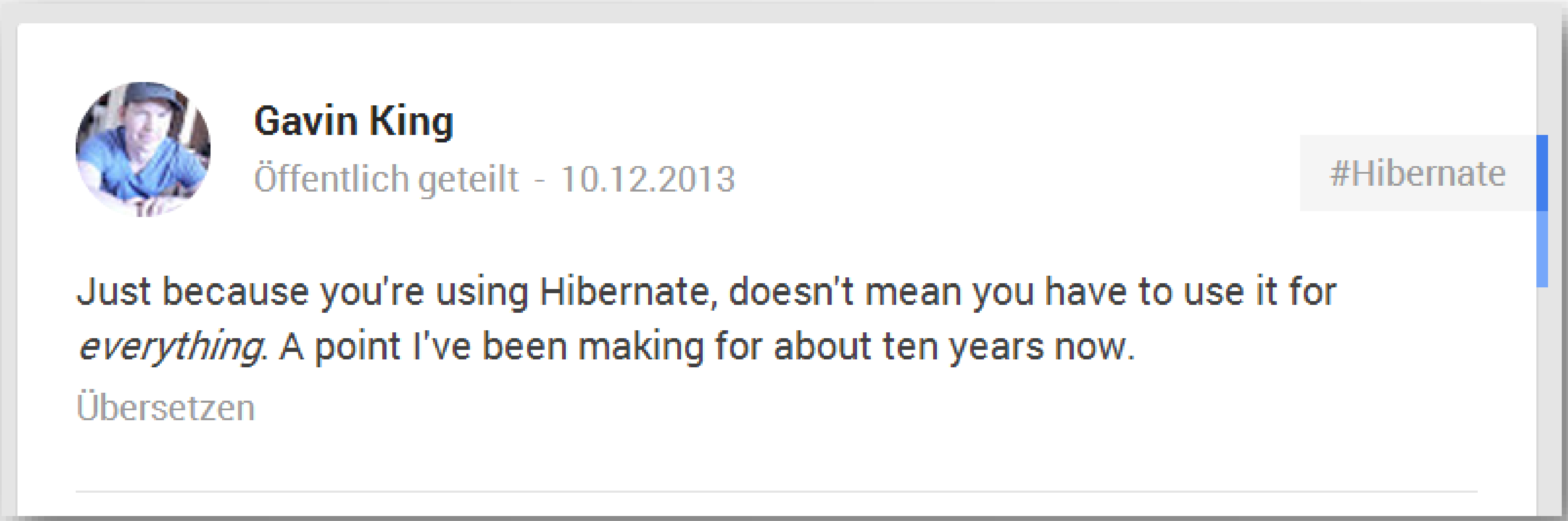
What JPA means for developers…

With JPA, your developers use a huge framework with lots of complexity that can get hard to manage

Images from Wikimedia. License: public domain. High voltage power lines by Simon Koopmann. License: CC-BY SA 3.0

… when developers actually wanted this

Note, we're talking about SQL. Not Persistence…



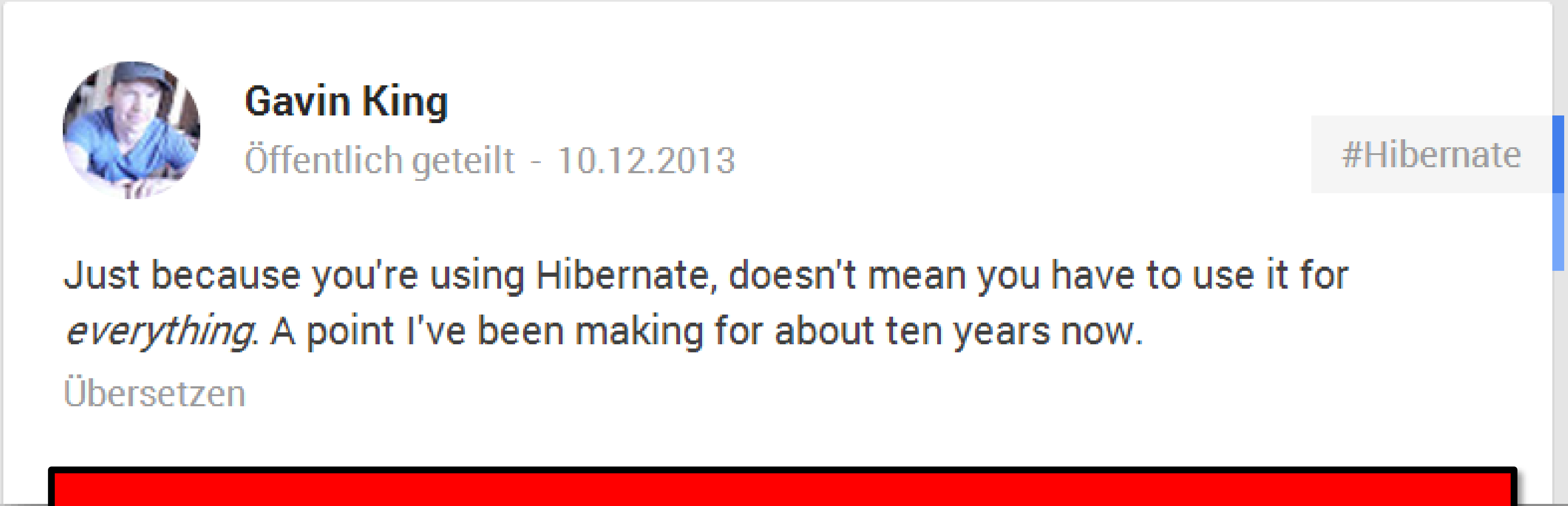
Gavin King
Öffentlich geteilt - 10.12.2013

#Hibernate

Just because you're using Hibernate, doesn't mean you have to use it for *everything*. A point I've been making for about ten years now.

Übersetzen

Note, we're talking about SQL. Not Persistence…



Gavin King

Öffentlich geteilt - 10.12.2013

#Hibernate

Just because you're using Hibernate, doesn't mean you have to use it for *everything*. A point I've been making for about ten years now.

Übersetzen

FYI: Gavin King: Creator of Hibernate!

NoSQL?

...

... so, should we maybe abandon SQL?

NoSQL? Who said it?

“Our service ran at 99.99 percent uptime in the first quarter of 2009, runs more than 200 million transactions a day, and has subsecond response time; and we are constantly making advances to deliver it even faster.”

NoSQL? Marc Benioff – salesforce.com

“ Our service ran at 99.99 percent uptime in the first quarter of 2009, runs more than 200 mil
transactions a day, and
response time; and we
making advances to del
faster.

NoSQL? Marc Benioff – salesforce.com

He's talking about salesforce.com's Oracle database.

He "invented" the cloud

NoSQL? Who said it?

“

- 300 TB of data files for production DBs in total
- LHC logging database ~140TB, expected growth up to ~70 TB / year
- 13 Production experiments' database ~120 TB in total

”

NoSQL? CERN, collecting LHC data

“

- 300 TB of data files for production DBs in total
- LHC logging database ~ 140TB expected growth up to year
- 13 Production experim database ~120 TB in to

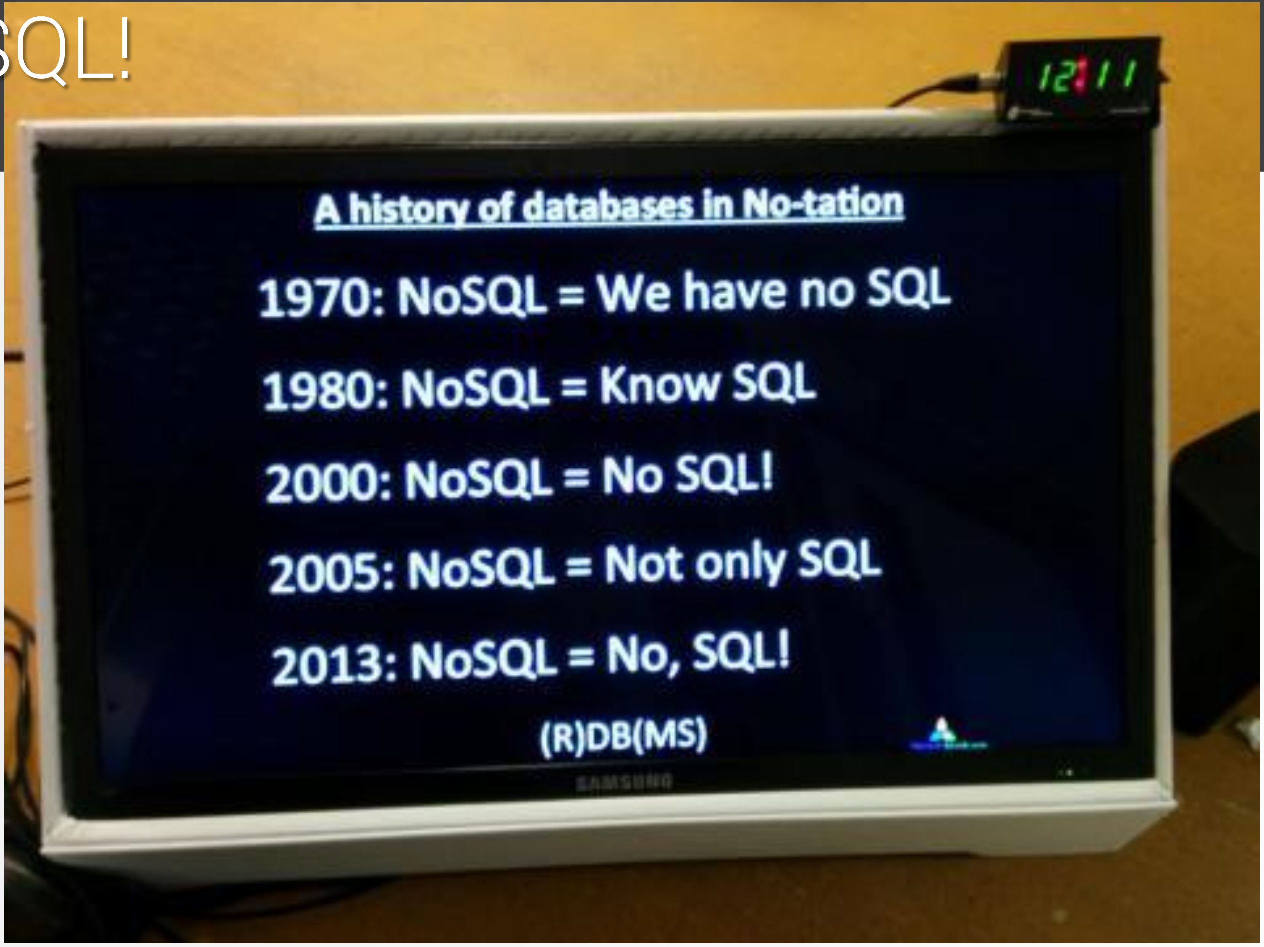
NoSQL for Big Data?

- You're giving up on **ACID**
- You're giving up on **type safety**
- You're giving up on **standards**
- You're giving up on **tooling**
- You're giving up on **relational algebra**
- You haven't asked operations
- You don't actually have «Big Data»

NoSQL for Big Data?

- You're giving up on **ACID**
- You're giving up on **type safety**
- You're giving up on **standards**
- You're giving up on **tooling**
- You're giving up on **relational algebra**
- You haven't asked operations
- You don't actually have «Big Data»

NoSQL? No, SQL!


A history of databases in No-tation

1970: NoSQL = We have no SQL

1980: NoSQL = Know SQL

2000: NoSQL = No SQL!

2005: NoSQL = Not only SQL

2013: NoSQL = No, SQL!

(R)DB(MS)

Seen at the O'Reilly Strata Conf:
History of NoSQL by Mark Madsen. Picture published by Edd Dumbill

Let’s calculate a running total

```
| ID   | VALUE_DATE | AMOUNT |
|------|------------|--------|
| 9997 | 2014-03-18 |  99.17 |
| 9981 | 2014-03-16 |  71.44 |
| 9979 | 2014-03-16 | -94.60 |
| 9977 | 2014-03-16 |  -6.96 |
| 9971 | 2014-03-15 | -65.95 |
```

Let's calculate a running total

```
| ID   | VALUE_DATE | AMOUNT |  BALANCE |
|------|------------|--------|----------|
| 9997 | 2014-03-18 |  99.17 | 19985.81 |
| 9981 | 2014-03-16 |  71.44 | 19886.64 |
| 9979 | 2014-03-16 | -94.60 | 19815.20 |
| 9977 | 2014-03-16 |  -6.96 | 19909.80 |
| 9971 | 2014-03-15 | -65.95 | 19916.76 |
```

Let's calculate a running total

```
| ID   | VALUE_DATE | AMOUNT |    BALANCE |
|------|------------|--------|------------|
| 9997 | 2014-03-18 | +99.17     =19985.81 |
| 9981 | 2014-03-16 |  71.44 |  +19886.64 |
| 9979 | 2014-03-16 | -94.60 |   19815.20 |
| 9977 | 2014-03-16 |  -6.96 |   19909.80 |
| 9971 | 2014-03-15 | -65.95 |   19916.76 |
```

Let's calculate a running total

```
| ID   | VALUE_DATE | AMOUNT |    BALANCE |
|------|------------|--------|------------|
| 9997 | 2014-03-18 |  99.17 |   19985.81 |
| 9981 | 2014-03-16 | +71.44 |  =19886.64 |
| 9979 | 2014-03-16 | -94.60 |  +19815.20 |
| 9977 | 2014-03-16 |  -6.96 |   19909.80 |
| 9971 | 2014-03-15 | -65.95 |   19916.76 |
```

Let's calculate a running total

```
| ID   | VALUE_DATE | AMOUNT |    BALANCE |
|------|------------|--------|------------|
| 9997 | 2014-03-18 |  99.17 |   19985.81 |
| 9981 | 2014-03-16 | +71.44   =19886.64 | n
| 9979 | 2014-03-16 | -94.60 | +19815.20 | n+1
| 9977 | 2014-03-16 |  -6.96 |   19909.80 |
```

$\mathrm{BALANCE}(\mathrm{ROW}_n) = \mathrm{BALANCE}(\mathrm{ROW}_{n+1}) + \mathrm{AMOUNT}(\mathrm{ROW}_n)$

$\mathrm{BALANCE}(\mathrm{ROW}_{n+1}) = \mathrm{BALANCE}(\mathrm{ROW}_n) - \mathrm{AMOUNT}(\mathrm{ROW}_n)$

```
SELECT
  t.*,
  t.current_balance - NVL(
    SUM(t.amount) OVER (
      PARTITION BY t.account_id
      ORDER BY     t.value_date DESC,
                   t.id         DESC
      ROWS BETWEEN UNBOUNDED PRECEDING
           AND     1         PRECEDING
    ),
  0) AS balance
FROM     v_transactions t
WHERE    t.account_id = 1
ORDER BY t.value_date DESC,
         t.id         DESC
```

```
SUM(t.amount) OVER (
  PARTITION BY t.account_id
  ORDER BY     t.value_date DESC,
               t.id         DESC
  ROWS BETWEEN UNBOUNDED PRECEDING
       AND     1         PRECEDING
)
```

```
SUM(t.amount) OVER (
  PARTITION BY t.ac
  ORDER BY                        SC,
                                 DESC
  RC                NDED PRECEDING
               1              PRECEDING
)
```

KEYWORDS!

```
SUM(t.amount) OVER (
  PARTITION BY t.account_id
  ORDER BY     t.value_date DESC,
               t.id         DESC
  ROWS BETWEEN UNBOUNDED PRECEDING
       AND     1         PRECEDING
)
```

```
SUM(t.amount) OVER (
  PARTITION BY t.account_id
  ORDER BY     t.value_date DESC,
               t.id         DESC
  ROWS BETWEEN UNBOUNDED PRECEDING
       AND     1         PRECEDING
)
```

```sql
SUM(t.amount) OVER (
  PARTITION BY t.account_id
  ORDER BY     t.value_date DESC,
               t.id         DESC
  ROWS BETWEEN UNBOUNDED PRECEDING
       AND     1         PRECEDING
)
```

```
| ID     | VALUE_DATE   | AMOUNT    |      BALANCE |
|------|------------|---------|------------|
| 9997 | 2014-03-18  -(99.17)|    +19985.81 |
| 9981 | 2014-03-16  -(71.44)|     19886.64 |
| 9979 | 2014-03-16 -(-94.60)|     19815.20 |
| 9977 | 2014-03-16 |   -6.96 |    =19909.80 |
| 9971 | 2014-03-15 |  -65.95 |     19916.76 |
```

Don’t you think that’s beautiful?

Stockholm Syndrome:

“We love ~~COBOL~~ SQL”

Winston Churchill

“SQL is the worst form of database querying, except for all the other forms.”

More SQL Calculations

```
|        TEXT | VOTES |         RANK | PERCENT |
|-------------|-------|--------------|---------|
|   Hibernate |  1383 |            1 |    32 % |
|        jOOQ |  1029 |            2 |    23 % |
|  EclipseLink |   881 |            3 |    20 % |
|        JDBC |   533 |            4 |    12 % |
| Spring JDBC |   451 |            5 |    10 % |
```

Data may not be accurate...

More SQL Calculations

```
SELECT    p.text,
          p.votes,
          DENSE_RANK() OVER (ORDER BY p.votes DESC) AS "rank",
          LPAD(
            (p.votes * 100 / SUM(p.votes) OVER ()) || ' %',
            4, ' '
          ) AS "percent"
FROM      poll_options p
WHERE     p.poll_id = 12
ORDER BY  p.votes DESC
```

The same with jOOQ

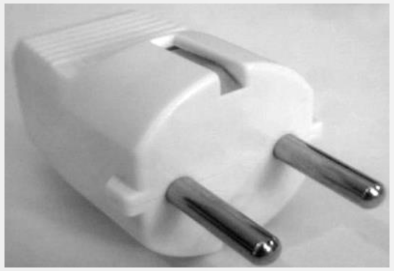
```
select (p.TEXT,
        p.VOTES,
        denseRank().over().orderBy(p.VOTES.desc()).as("rank"),
        lpad(
          p.VOTES.mul(100).div(sum(p.VOTES).over()).concat(" %"),
          4, " "
        ).as("percent"))
.from   (POLL_OPTIONS.as("p"))
.where  (p.POLL_ID.eq(12))
.orderBy(p.VOTES.desc());
```

The same with jOOQ in Scala (!)

```
select (p.TEXT,
        p.VOTES,
        denseRank() over() orderBy(p.VOTES desc) as "rank",
        lpad(
          (p.VOTES * 100) / (sum(p.VOTES) over()) || " %",
          4, " "
        ) as "percent")
from    (POLL_OPTIONS as "p")
where   (p.POLL_ID === 12)
orderBy (p.VOTES desc)
```

What jOOQ means for developers

Java

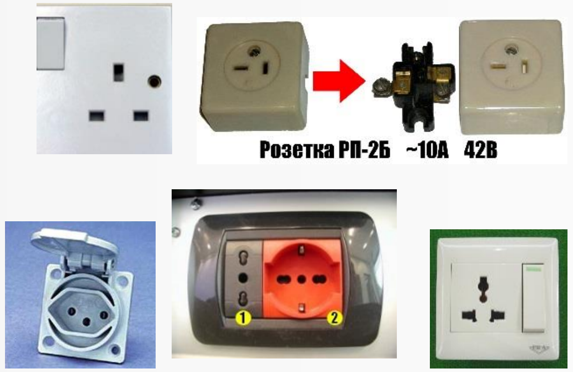


one jack

jOOQ

one adaptor

SQL



all plugs

With jOOQ, Java plugs into SQL intuitively, letting your developers focus on business-logic again.

Just to be sure you get the message

“ jOOQ is the best way to write SQL in Java ”

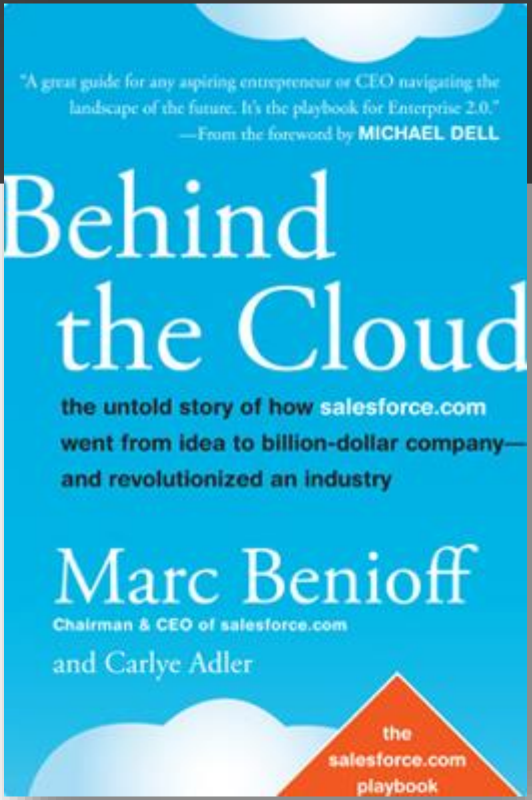
Examples

Who said it?

“All companies benefit when they can afford to focus on innovation rather than infrastructure”

Marc Benioff:

“All companies benefit when they can afford to focus on innovation rather than infrastructure”


"A great guide for any aspiring entrepreneur or CEO navigating the landscape of the future. It's the playbook for Enterprise 2.0." —From the foreword by MICHAEL DELL
Behind the Cloud
the untold story of how salesforce.com went from idea to billion-dollar company— and revolutionized an industry
Marc Benioff
Chairman & CEO of salesforce.com
and Carlye Adler
the salesforce.com playbook

And a shameless tool recommendation

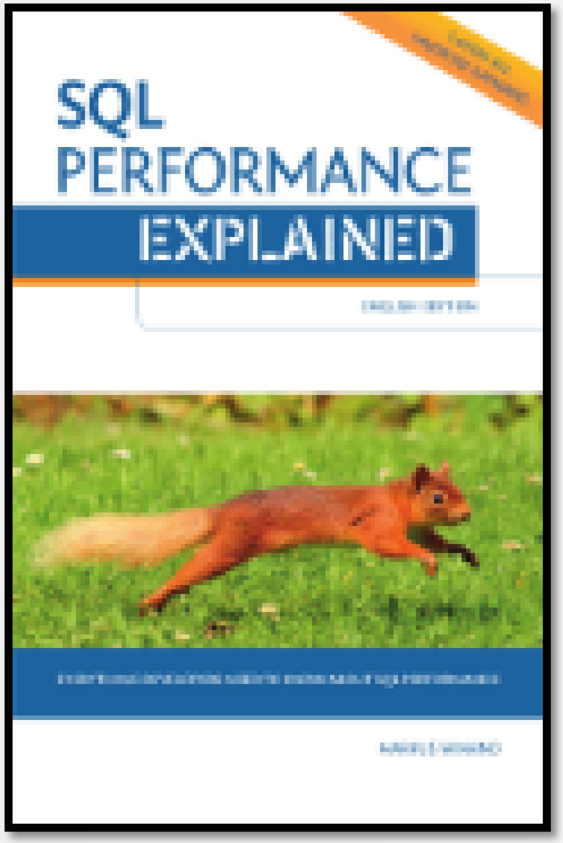
Open source databases:

- Free / Apache license

Commercial databases:

- Commercial license

And a shameless book recommendation



Markus Winand from

[Use-The-Index-Luke.com]

ROI north of 83'174%

Achieve proper indexing and performance in popular RDBMS

«jOOQ» 10% discount code

That's it folks

More free Java / SQL knowledge on:

- Blog: http://blog.jooq.org
- Twitter: @JavaOOQ / @lukaseder
- Newsletter: http://www.jooq.org