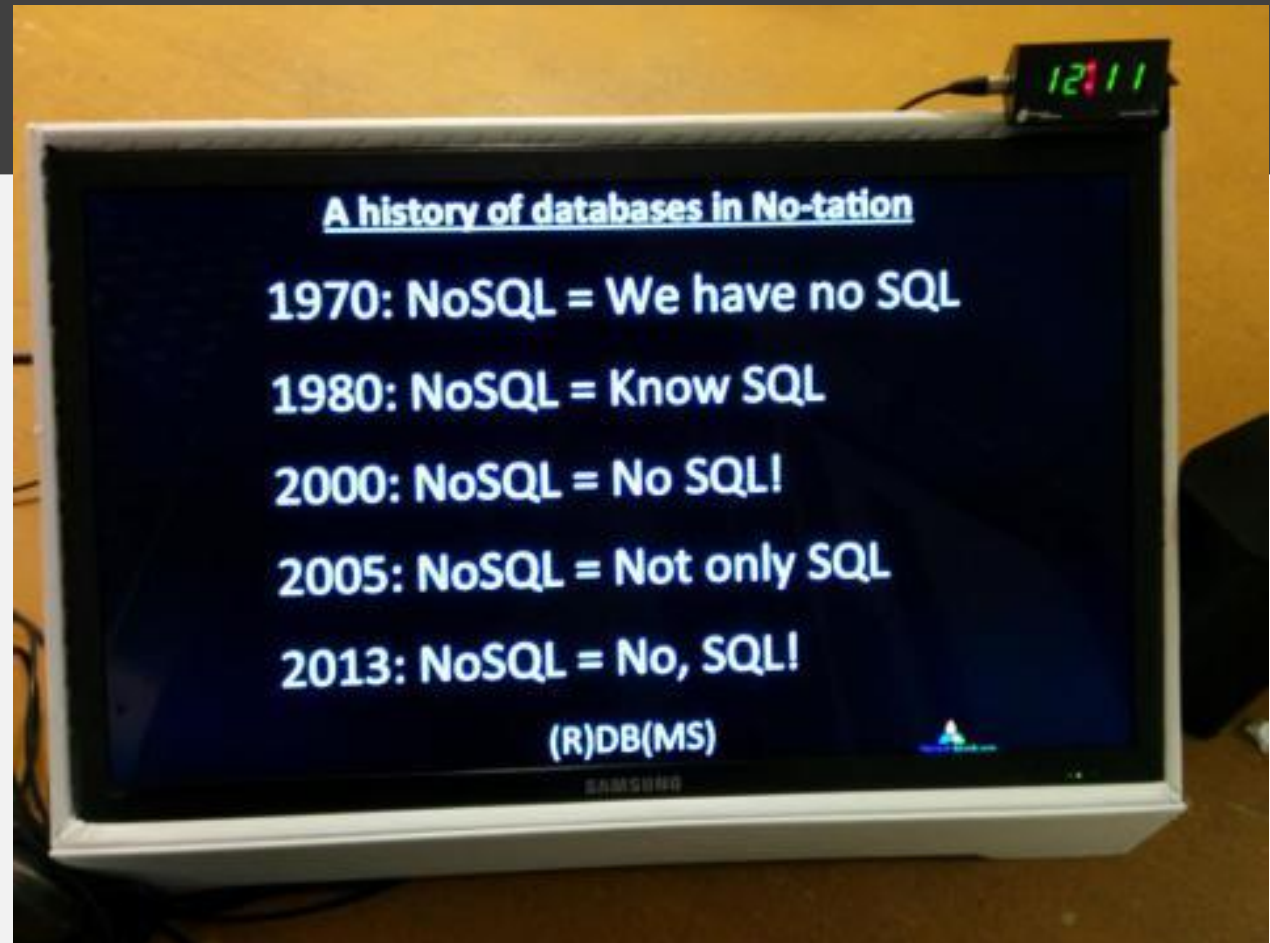


SQL, the
underestimated
“Big Data” technology

No – tation



Seen at the 2013 O'Reilly Strata Conf:

History of NoSQL by [Mark Madsen](#). Picture published by [Edd Dumbill](#)

NoSQL?

NoSQL?

No, SQL!

Our vision at Data Geekery

- SQL dominates database systems
- SQL is very expressive
- SQL is very type safe

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

Me – @lukaseder



- Head of R&D at Data Geekery GmbH
- SQL Aficionado
- Java Aficionado

“ Java developers can get back in control of SQL with jOOQ ”

Big Data? NoSQL?

- 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»

Big Data? NoSQL?

- 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»

Also Not SQL

```
@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() { /* ... */ }
```


Also Not SQL – Annotatiomania™

```
@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;
```

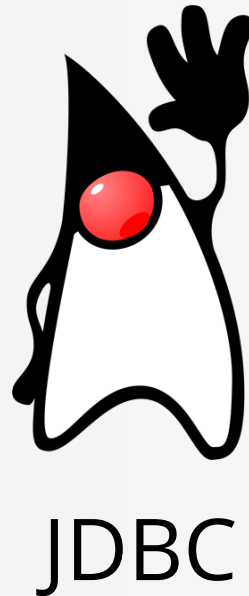
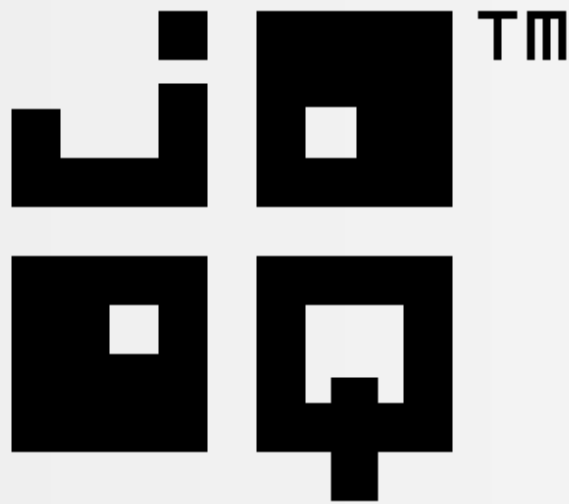
Found at <http://stackoverflow.com/q/17491912/521799>

Also Not SQL – JPA 3.0 Preview

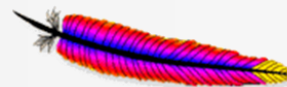
```
@OneToMany @OneToManyMore @AnyOne @AnyBody
@ManyToMany @Many
@Fetch @FetchMany @FetchWithDiscriminator(name = "no_name")
@JoinTable(joinColumns = {
    @JoinColumn(name = "customer_id", referencedColumnName = "id")
})
@PrefetchJoinWithDiscriminator
@ifJoiningAvoidHashJoins @ButUseHashJoinsWhenMoreThan(records = 1000)
@XmlDataTransformable @SpringPrefetchAdapter
private Collection employees;
```

Might not be true

Shocker! You can now write SQL in Java.



MyBatis



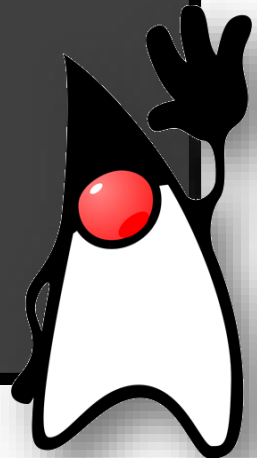
Apache CommonsTM
<http://commons.apache.org/>

commons
*dbutils*TM

SQL in Java 7 – JDBC

```
try (PreparedStatement stmt = c.prepareStatement(sql);
     ResultSet rs          = stmt.executeQuery()) {

    while (rs.next()) {
        System.out.println(
            new Schema(rs.getString("SCHEMA_NAME"),
                       rs.getBoolean("IS_DEFAULT"))
        );
    }
}
```



SQL in Java 8 – jOOQ

```
DSL.using(c)
    .fetch(sql)
    .map(rs -> new Schema(
        rs.getValue("SCHEMA_NAME", String.class),
        rs.getValue("IS_DEFAULT", boolean.class)
    ))
    .forEach(System.out::println);
```



Typesafe SQL in Java – jOOQ

```
DSL.using(c)
  .select(s.SCHEMA_NAME, s.IS_DEFAULT)
  .from(INFORMATION_SCHEMA.SCHEMATA.as("s"))
  .orderBy(s.SCHEMA_NAME)
  .map(rs -> new Schema(
    rs.getValue(s.SCHEMA_NAME),
    rs.getValue(s.IS_DEFAULT)
  ))
  .forEach(System.out::println);
```



SQL in Java 8 – Spring JDBC

```
new JdbcTemplate(  
    new SingleConnectionDataSource(c, true))  
.query(sql, (rs, rowNum) ->  
    new Schema(  
        rs.getString("SCHEMA_NAME"),  
        rs.getBoolean("IS_DEFAULT")  
    ))  
.forEach(System.out::println);
```



SQL in Java 8 – Apache DbUtils

```
new QueryRunner()  
    .query(c, sql, new ArrayListHandler())  
    .stream()  
    .map(array -> new Schema(  
        (String) array[0],  
        (Boolean) array[1]  
    ))  
    .forEach(System.out::println);
```



commons
dbutils

SQL in Groovy

```
sql.eachRow( 'select * from tableName' ) {  
    println "$it.id -- ${it.firstName} --"  
}
```

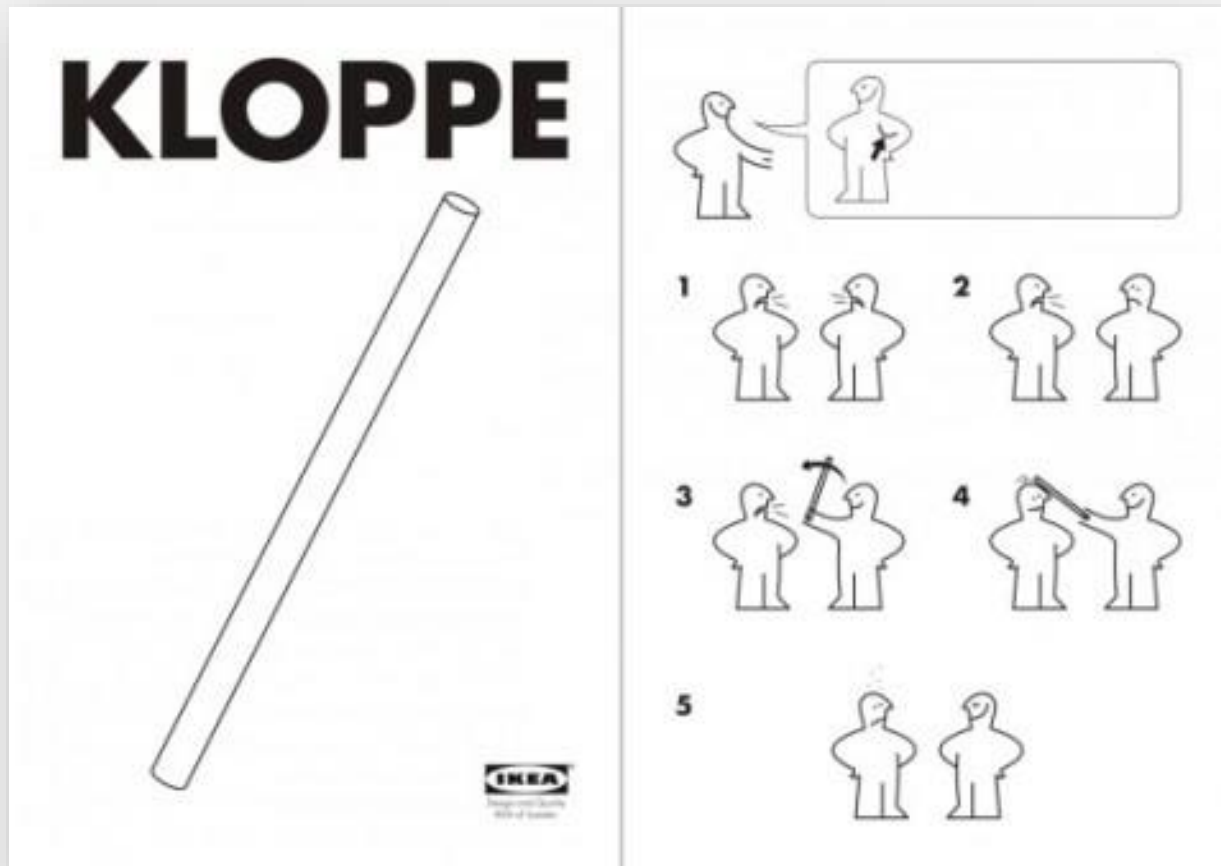


When you should use SQL – indicators

- You need JOINS, UNIONS
- You need functions, aggregations
- You need bulk reads / writes

“ Calculations should be done close to the data ”

Please, run that calculation in your DB



SQL Trivia – NULL

```
-- What does this query return?  
SELECT 1 AS a FROM dual  
WHERE 1 IN (NULL)  
UNION ALL  
SELECT 2 AS a FROM dual  
WHERE NOT(1 IN (NULL))
```

SQL Trivia – NULL

```
-- What does this query return?  
SELECT 1 AS a FROM dual  
WHERE 1 IN (NULL)  
UNION ALL  
SELECT 2 AS a FROM dual  
WHERE NOT(1 IN (NULL))
```

SQL Trivia – NULL

```
-- Nothing! It's the same as this
SELECT 1 AS a FROM dual
WHERE 1 = NULL
UNION ALL
SELECT 2 AS a FROM dual
WHERE 1 != NULL
```

SQL Trivia – NULL

-- Nothing! It's the same as this

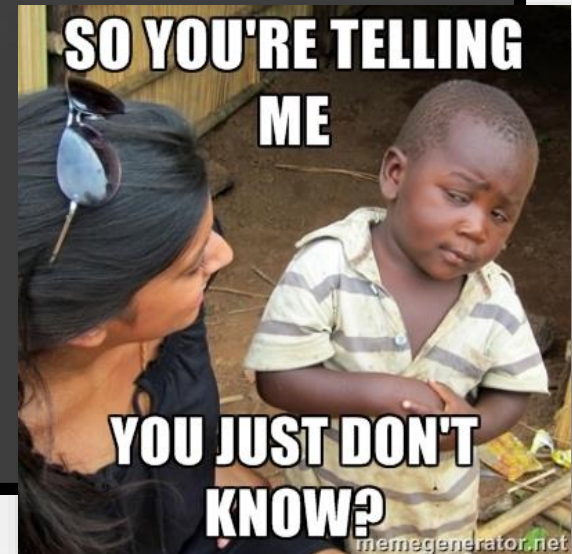
```
SELECT 1 AS a FROM dual
```

```
WHERE "UNKNOWN"
```

```
UNION ALL
```

```
SELECT 2 AS a FROM dual
```

```
WHERE "UNKNOWN"
```



SQL Trivia – Oracle VARCHAR2

```
-- What does this query return?  
SELECT 1 AS a FROM dual  
WHERE '' = ''  
UNION ALL  
SELECT 2 AS a FROM dual  
WHERE 'a' != ''
```


SQL Trivia – Oracle VARCHAR2

```
-- Nope! Nothing again (only in Oracle).  
SELECT 1 AS a FROM dual  
WHERE NULL = NULL  
UNION ALL  
SELECT 2 AS a FROM dual  
WHERE 'a' != NULL
```

SQL Trivia – Oracle VARCHAR2

-- Nope! Nothing again (only in Oracle).

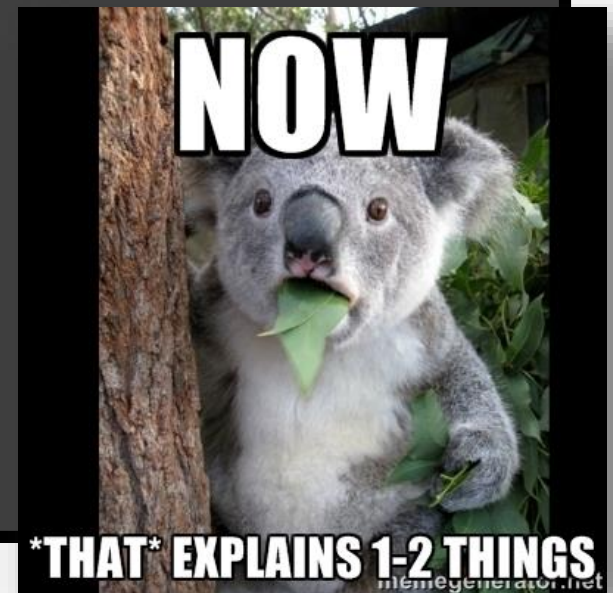
```
SELECT 1 AS a FROM dual
```

```
WHERE NULL = NULL
```

```
UNION ALL
```

```
SELECT 2 AS a FROM dual
```

```
WHERE 'a' != NULL
```



Stockholm Syndrome:

“ We love ~~JavaScript~~ SQL ”

Winston Churchill:

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

Let's calculate a running total

```
SELECT      *
FROM        v_transactions
WHERE       account_id = 1
ORDER BY    value_date DESC,
            id          DESC
```

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	

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

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

“ How can we do it?

”

How can we do it?

- In Java
- Calculate on UPDATE
- Nested SELECT
- Recursive SQL
- Window functions
- MODEL clause (Oracle)
- Stored procedures

How can we do it? – With SQL!

- ~~- In Java~~
- ~~- Calculate on UPDATE~~
- Nested SELECT
- Recursive SQL
- Window functions
- MODEL clause (Oracle)
- ~~- Stored procedures~~

“ Using nested SELECTs ”

Ex
1

```
SELECT
  t1.*,
  t1.current_balance - (
    SELECT NVL(SUM(amount), 0)
    FROM v_transactions t2
    WHERE t2.account_id = t1.account_id
    AND (t2.value_date, t2.id) >
        (t1.value_date, t1.id)
  ) AS balance
FROM v_transactions t1
WHERE t1.account_id = 1
ORDER BY t1.value_date DESC, t1.id DESC
```

```
SELECT
  t1.*,
  t1.current_balance - (
    SELECT NVL(SUM(amount), 0)
    FROM v_transactions t2
    WHERE t2.account_id = t1.account_id
    AND (t2.value_date, t2.id) >
        (t1.value_date, t1.id)
  ) AS balance
FROM v_transactions t1
WHERE t1.account_id = 1
ORDER BY t1.value_date DESC, t1.id DESC
```



```
SELECT
  t1.*,
  t1.current_balance - (
    SELECT NVL(SUM(amount), 0)
    FROM v_transactions t2
    WHERE t2.account_id = t1.account_id
    AND ((t2.value_date > t1.value_date) OR
        (t2.value_date = t1.value_date AND
         t2.id > t1.id))
  ) AS balance
FROM v_transactions t1
WHERE t1.account_id = 1 ORDER BY ...
```

Using nested SELECTs

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

Id	Operation	Name	A-Rows	A-Time
0	SELECT STATEMENT		50	00:00:00.77
1	SORT AGGREGATE		1101	00:00:00.76
* 2	TABLE ACCESS BY INDEX ROWID	T_TRANSACTIONS	605K	00:00:00.69
* 3	INDEX RANGE SCAN	I_TRX_ACCO_ID	1212K	00:00:00.21
4	SORT ORDER BY		50	00:00:00.77
5	NESTED LOOPS		1101	00:00:00.01
6	TABLE ACCESS BY INDEX ROWID	T_ACCOUNTS	1	00:00:00.01
* 7	INDEX UNIQUE SCAN	SYS_C006991	1	00:00:00.01
8	TABLE ACCESS BY INDEX ROWID	T_TRANSACTIONS	1101	00:00:00.01
* 9	INDEX RANGE SCAN	I_TRX_ACCO_ID	1101	00:00:00.01



I HAVE MUCH TO LEARN
memegenerator.net

“Using recursive SQL”

We need to number transactions

ID	VALUE_DATE	AMOUNT	TRANSACTION_NR
9997	2014-03-18	99.17	1
9981	2014-03-16	71.44	2
9979	2014-03-16	-94.60	3
9977	2014-03-16	-6.96	4
9971	2014-03-15	-65.95	5

```
CREATE OR REPLACE VIEW v_transactions_by_time
AS
SELECT
  t.*,
  ROW_NUMBER() OVER (
    PARTITION BY account_id
    ORDER BY      t.value_date DESC,
                 t.id DESC
  ) AS transaction_number
FROM
  v_transactions t;
```

Ex
2

```
WITH ordered_with_balance (  
    account_id, value_date, amount, balance, transaction_number  
)  
AS (  
    SELECT t1.account_id, t1.value_date, t1.amount, t1.current_balance,  
           t1.transaction_number  
    FROM   v_transactions_by_time t1  
    WHERE  t1.transaction_number = 1  
  
    UNION ALL  
  
    SELECT t1.account_id, t1.value_date, t1.amount, t2.balance - t2.amount,  
           t1.transaction_number  
    FROM   ordered_with_balance t2  
    JOIN   v_transactions_by_time t1  
    ON     t1.transaction_number = t2.transaction_number + 1  
    AND    t1.account_id = t2.account_id  
)  
SELECT *  
FROM   ordered_with_balance  
WHERE  account_id = 1  
ORDER BY transaction number ASC
```

```
WITH ordered_with_balance (  
  account_id, value_date, amount, balance, transaction_number  
)  
AS (  
  SELECT t1.account_id, t1.value_date, t1.amount, t1.current_balance,  
         t1.transaction_number  
  FROM   v_transactions_by_time t1  
 WHERE  t1.transaction_number = 1  
  
  UNION ALL  
  
  SELECT t1.account_id, t1.value_date, t1.amount, t2.balance - t2.amount,  
         t1.transaction_number  
  FROM   ordered_with_balance t2  
  JOIN   v_transactions_by_time t1  
  ON     t1.transaction_number = t2.transaction_number + 1  
  AND    t1.account_id = t2.account_id  
)  
SELECT *  
FROM   ordered_with_balance  
WHERE  account_id = 1  
ORDER BY transaction number ASC
```



```
WITH ordered_with_balance (  
    account_id, value_date, amount, balance, transaction_number  
)  
AS (  
    SELECT t1.account_id, t1.value_date, t1.amount, t1.current_balance,  
           t1.transaction_number  
    FROM   v_transactions_by_time t1  
    WHERE  t1.transaction_number = 1  
  
    UNION ALL  
  
    SELECT t1.account_id, t1.value_date, t1.amount, t2.balance - t2.amount,  
           t1.transaction_number  
    FROM   ordered_with_balance t2  
    JOIN   v_transactions_by_time t1  
    ON     t1.transaction_number = t2.transaction_number + 1  
    AND    t1.account_id = t2.account_id  
)  
SELECT *  
FROM   ordered_with_balance  
WHERE  account_id = 1  
ORDER BY transaction number ASC
```

Id	Operation	Name	A-Rows	A-Time
0	SELECT STATEMENT		50	00:00:35.29
1	SORT ORDER BY		50	00:00:35.29
* 2	VIEW		1101	00:00:35.29
3	UNION ALL (RECURSIVE WITH) BREADTH FIRST		9999	00:00:35.28
* 4	VIEW	V_TRANSACTIONS_BY_TIME	9	00:00:00.03
* 5	WINDOW SORT PUSHED RANK		18	00:00:00.03
6	NESTED LOOPS		9999	00:00:00.01
7	NESTED LOOPS		9999	00:00:00.01
8	TABLE ACCESS FULL	T_ACCOUNTS	10	00:00:00.01
* 9	INDEX RANGE SCAN	I_TRX_ACCO_ID	9999	00:00:00.01
10	TABLE ACCESS BY INDEX ROWID	T_TRANSACTIONS	9999	00:00:00.01
* 11	HASH JOIN		9990	00:00:35.08
12	VIEW	V_TRANSACTIONS_BY_TIME	11M	00:00:29.13
13	WINDOW SORT		11M	00:00:27.19
14	NESTED LOOPS		11M	00:00:13.62
15	NESTED LOOPS		11M	00:00:03.89
16	INDEX FAST FULL SCAN	SYS_C006991	11450	00:00:00.06
* 17	INDEX RANGE SCAN	I_TRX_ACCO_ID	11M	00:00:02.18
18	TABLE ACCESS BY INDEX ROWID	T_TRANSACTIONS	11M	00:00:06.15
	PUMP		9999	00:00:00.01



THIS FAR, NO FARTHER!
memegenerator.net

“ Using window
functions

”

Ex
3

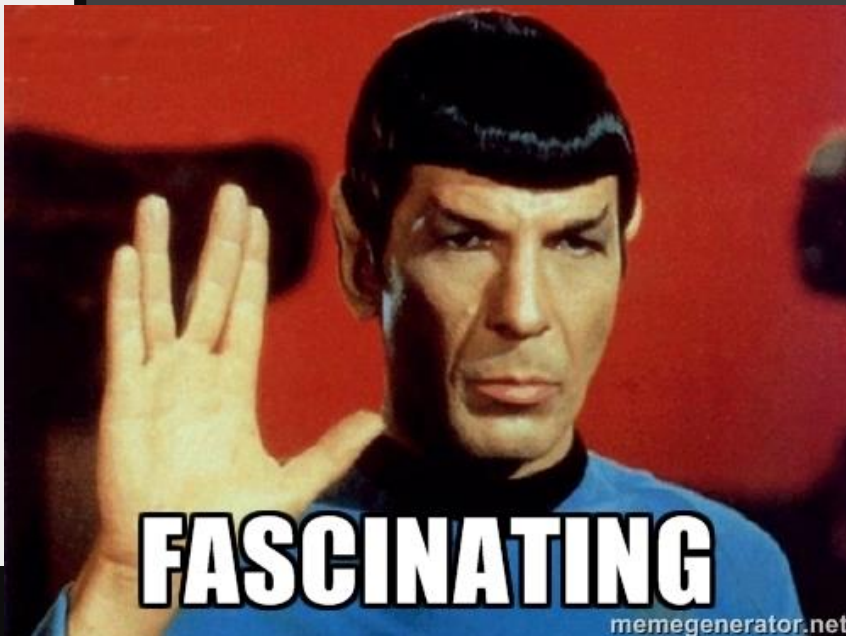
```
SELECT
  t.*,
  t.current_balance - NVL(
    SUM(t.amount) OVER (
      PARTITION BY t.account_id
      ORDER BY    t.value_date DESC,
                 t.id          DESC
      ROWS BETWEEN UNBOUNDED PRECEDING
                 AND 1          PRECEDING
    ),
    0) AS balance
FROM   v_transactions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

```
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
```

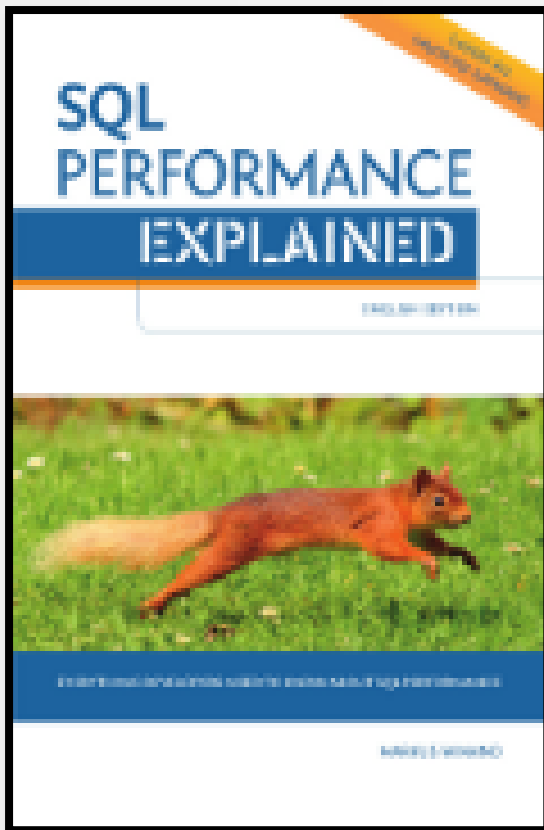
Using window functions

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

Id	Operation	Name	A-Rows	A-Time
0	SELECT STATEMENT		50	00:00:00.01
1	WINDOW SORT		50	00:00:00.01
2	NESTED LOOPS		1101	00:00:00.01
3	TABLE ACCESS BY INDEX ROWID	T_ACCOUNTS	1	00:00:00.01
* 4	INDEX UNIQUE SCAN	SYS_C006991	1	00:00:00.01
5	TABLE ACCESS BY INDEX ROWID	T_TRANSACTIONS	1101	00:00:00.01
* 6	INDEX RANGE SCAN	I_TRX_ACCO_ID	1101	00:00:00.01



Performance – Please read this book



Markus Winand from
Use-The-Index-Luke.com

ROI of 83'800% (time AND money)
Achieve proper indexing and
performance in popular RDBMS

«jOOQ» 10% discount code

“ Using the Oracle
MODEL clause

”

Ex
4

```
SELECT account_id, value_date, amount, balance
FROM (
  SELECT id, account_id, value_date, amount,
         current_balance AS balance
  FROM v_transactions
) t
WHERE account_id = 1
MODEL
  PARTITION BY (account_id)
  DIMENSION BY (
    ROW_NUMBER() OVER (ORDER BY value_date DESC, id DESC) AS rn
  )
  MEASURES (value_date, amount, balance)
  RULES (
    balance[rn > 1] = balance[cv(rn) - 1] - amount[cv(rn) - 1]
  )
ORDER BY rn ASC
```

```
SELECT account_id, value_date, amount, balance
FROM (
  SELECT id, account_id, value_date, amount,
         current_balance AS balance
  FROM v_transactions
) t
WHERE account_id = 1
MODEL
  PARTITION BY (account_id)
  DIMENSION BY (
    ROW_NUMBER() OVER (ORDER BY value_date DESC, id DESC) AS rn
  )
  MEASURES (value_date, amount, balance)
  RULES (
    balance[rn > 1] = balance[cv(rn) - 1] - amount[cv(rn) - 1]
  )
ORDER BY rn ASC
```

RULES (

$$\text{balance}[\text{rn} > 1] = \text{balance}[\text{cv}(\text{rn}) - 1] - \text{amount}[\text{cv}(\text{rn}) - 1]$$

)

-- does it look familiar?

	A	B	C	D
1	value_date	amount	balance	
2	17.03.2014	15.87	13222.45	
3	16.03.2014	-33.14	13206.58	
4	16.03.2014	-93.77	=C3-B3	
5	13.03.2014	10.65	13333.49	
6	11.03.2014	19.16	13322.84	
7	11.03.2014	-59.25	13303.68	
8	11.03.2014	94.86	13362.93	
9	10.03.2014	80.42	13268.07	
10	10.03.2014	38.43		
11	09.03.2014	-4.41		
12	08.03.2014	80.45		
13	07.03.2014	-56.45		



Id	Operation	Name	A-Rows	A-Time
0	SELECT STATEMENT		50	00:00:00.02
1	SORT ORDER BY		50	00:00:00.02
2	SQL MODEL ORDERED		1101	00:00:00.02
3	WINDOW SORT		1101	00:00:00.01
4	NESTED LOOPS		1101	00:00:00.01
5	TABLE ACCESS BY INDEX ROWID	T_ACCOUNTS	1	00:00:00.01
* 6	INDEX UNIQUE SCAN	SYS_C006991	1	00:00:00.01
* 7	TABLE ACCESS FULL	T_TRANSACTIONS	1101	00:00:00.01

I LOVE IT



WHEN A PLAN CONTAINS A MODEL
CLAUSE

“ The MODEL clause is
Oracle's most
powerful and
underused feature ”

Our vision at Data Geekery - Revisited

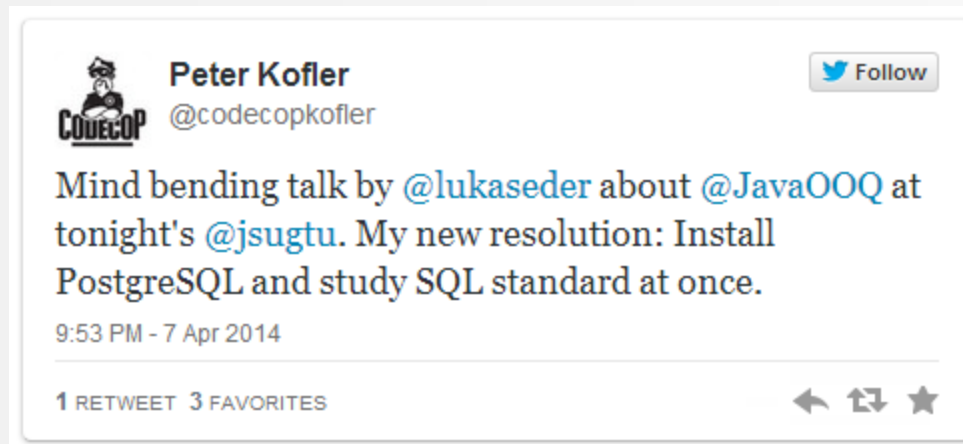
- SQL dominates database systems
- SQL is expressive
- SQL is type safe

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

Our vision at Data Geekery - Revisited

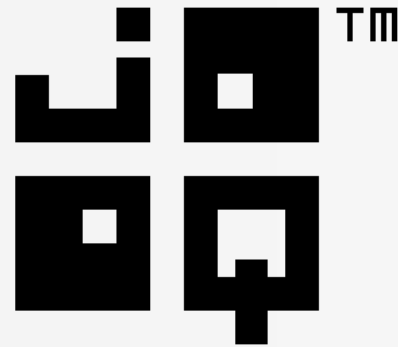
- SQL dominates database systems
- SQL is expressive
- SQL is type safe

“



”

Our vision at Data Geekery - Revisited



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

```
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
```

```
DSL.using(connection)
  .select(t.VALUE_DATE,
         t.AMOUNT,
         t.CURRENT_BALANCE.sub(
           sum(t.AMOUNT).over(
             partitionBy(t.ACCOUNT_ID)
             .orderBy    (t.VALUE_DATE.desc(),
                        t.ID      .desc())
             .rowsBetweenUnboundedPreceding()
             .andPreceding(1)
           )
         ).nvl(0).as("balance"))
  .from  (V_TRANSACTIONS.as("t"))
  .where (t.ACCOUNT_ID.eq(1))
  .orderBy(t.VALUE_DATE.desc(),
          t.ID      .desc())
```

Thank you

3-month jOOQ Enterprise trial:

- Send «JUGS-BE-SQL-2014» to sales@datageekery.com



More free Java / SQL knowledge on:

- Blog: <http://blog.jooq.org>
- Twitter: [@JavaOOQ](https://twitter.com/JavaOOQ)

This just in...

“ There is SQL before and
after window functions ”

Use-case: Choreo export

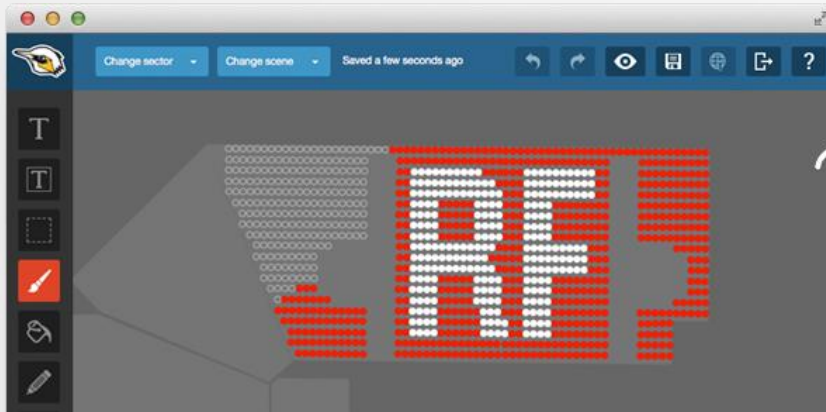


Image Copyright © fanpictor.com

Use-case: Choreo export as Excel

	Sector	Row	Seat	Scene1	Scene2
7935					
7936	S2	52	33	#176FC1	#FFFFFF
7937	S2	52	34	#176FC1	#FFFFFF
7938	S2	52	35	#176FC1	#FFFFFF
7939	S2	52	36	#176FC1	#FFFFFF
7940	T1	11	1	#176FC1	#176FC1
7941	T1	11	2	#176FC1	#176FC1
7942	T1	11	3	#176FC1	#176FC1
7943	T1	11	4	#176FC1	#176FC1
7944	T1	11	5	#176FC1	#176FC1
7945	T1	11	6	#176FC1	#176FC1
7946	T1	11	7	#176FC1	#176FC1
7947	T1	11	8	#176FC1	#176FC1
7948	T1	11	9	#176FC1	#FFFFFF
7949	T1	12	1	#176FC1	#176FC1
7950	T1	12	2	#176FC1	#176FC1
7951	T1	12	3	#176FC1	#176FC1
7952	T1	12	4	#176FC1	#176FC1
7953	T1	12	5	#176FC1	#176FC1
7954	T1	12	6	#176FC1	#176FC1
7955	T1	12	7	#176FC1	#176FC1
7956	T1	12	8	#176FC1	#FFFFFF
7957	T1	12	9	#176FC1	#FFFFFF
7958	T1	13	1	#176FC1	#176FC1
7959	T1	13	2	#176FC1	#176FC1

Use-case: Choreo export as Excel

	Sector	Row	Seat	Scene1	Scene2	Start / Stop	Count
7935							
7936	S2	52	33	#176FC1	#FFFFFF		36
7937	S2	52	34	#176FC1	#FFFFFF		36
7938	S2	52	35	#176FC1	#FFFFFF		36
7939	S2	52	36	#176FC1	#FFFFFF	stop	36
7940	T1	11	1	#176FC1	#176FC1	start	8
7941	T1	11	2	#176FC1	#176FC1		8
7942	T1	11	3	#176FC1	#176FC1		8
7943	T1	11	4	#176FC1	#176FC1		8
7944	T1	11	5	#176FC1	#176FC1		8
7945	T1	11	6	#176FC1	#176FC1		8
7946	T1	11	7	#176FC1	#176FC1		8
7947	T1	11	8	#176FC1	#176FC1	stop	8
7948	T1	11	9	#176FC1	#FFFFFF	start / stop	1
7949	T1	12	1	#176FC1	#176FC1	start	7
7950	T1	12	2	#176FC1	#176FC1		7
7951	T1	12	3	#176FC1	#176FC1		7
7952	T1	12	4	#176FC1	#176FC1		7
7953	T1	12	5	#176FC1	#176FC1		7
7954	T1	12	6	#176FC1	#176FC1		7
7955	T1	12	7	#176FC1	#176FC1	stop	7
7956	T1	12	8	#176FC1	#FFFFFF	start	2
7957	T1	12	9	#176FC1	#FFFFFF	stop	2
7958	T1	13	1	#176FC1	#176FC1	start	6
7959	T1	13	2	#176FC1	#176FC1		6

Use-case: Choreo export as Excel

	Sector	Row	Seat	Scene1	Scene2	Start / Stop	Count
7935	S2	52	33	#176FC1	#FFFFFF		36
7937	S2	52	34	#176FC1	#FFFFFF		36
7938	S2	52	35	#176FC1	#FFFFFF		36
7939	S2	52	36	#176FC1	#FFFFFF	stop	36
7940	T1	11	1	#176FC1	#176FC1	start	8
7941	T1	11	2	#176FC1	#176FC1		8
7942	T1	11	3	#176FC1	#176FC1		8
7943	T1	11	4	#176FC1	#176FC1		8
7944	T1	11	5	#176FC1	#176FC1		8
7945	T1	11	6	#176FC1	#176FC1		8
7946	T1	11	7	#176FC1	#176FC1		8
7947	T1	11	8	#176FC1	#176FC1	stop	8
7948	T1	11	9	#176FC1	#FFFFFF	start / stop	1
7949	T1	12	1	#176FC1	#176FC1	start	7
7950	T1	12	2	#176FC1	#176FC1		7
7951	T1	12	3	#176FC1	#176FC1		7
7952	T1	12	4	#176FC1	#176FC1		7
7953	T1	12	5	#176FC1	#176FC1		7
7954	T1	12	6	#176FC1	#176FC1		7
7955	T1	12	7	#176FC1	#176FC1	stop	7
7956	T1	12	8	#176FC1	#FFFFFF	start	2
7957	T1	12	9	#176FC1	#FFFFFF	stop	2
7958	T1	13	1	#176FC1	#176FC1	start	6
7959	T1	13	2	#176FC1	#176FC1		6

```

WITH data AS (SELECT d.*,
                    row(sector, row, scene1, scene2) block
                FROM d)
SELECT data.*,
       CASE WHEN LAG (block) OVER (o) IS DISTINCT FROM block
            AND LEAD(block) OVER (o) IS DISTINCT FROM block
            THEN 'start / stop'
            WHEN LAG (block) OVER (o) IS DISTINCT FROM block
            THEN 'start'
            WHEN LEAD(block) OVER (o) IS DISTINCT FROM block
            THEN 'stop'
            ELSE '' END start_stop,
       COUNT(*) OVER (PARTITION BY sector, row, scene1, scene2)
FROM data
WINDOW o AS (ORDER BY sector, row, seat)
ORDER BY sector, row, seat

```

Full example: <http://blog.jooq.org/2014/04/15/how-to-do-this-with-sql-of-course>

```
WITH data AS (SELECT d.*,  
                    row(sector, row, scene1, scene2) block  
                FROM d)  
SELECT data.*,  
       CASE WHEN LAG (block) OVER (o) IS DISTINCT FROM block  
             AND LEAD(block) OVER (o) IS DISTINCT FROM block  
             THEN 'start / stop'  
             WHEN LAG (block) OVER (o) IS DISTINCT FROM block  
             THEN 'start'  
             WHEN LEAD(block) OVER (o) IS DISTINCT FROM block  
             THEN 'stop'  
             ELSE '' END start stop,
```

We can compare rows with each other, not only columns!

Full example: <http://blog.jooq.org/2014/04/15/how-to-do-this-with-sql-of-course>

We can reuse window specifications!

```

        row(sector, row, scene1, scene2) block
    FROM d)
SELECT data.*,
    CASE WHEN LAG (block) OVER (o) IS DISTINCT FROM block
        AND LEAD(block) OVER (o) IS DISTINCT FROM block
    THEN 'start / stop'
    WHEN LAG (block) OVER (o) IS DISTINCT FROM block
    THEN 'start'
    WHEN LEAD(block) OVER (o) IS DISTINCT FROM block
    THEN 'stop'
    ELSE '' END start_stop,
    COUNT(*) OVER (PARTITION BY sector, row, scene1, scene2)
FROM data
WINDOW o AS (ORDER BY sector, row, seat)
ORDER BY sector, row, seat

```

Full example: <http://blog.jooq.org/2014/04/15/how-to-do-this-with-sql-of-course>

