



# Performance

# Today's Topics

- Different types of requirements
- Performance
  - Stored procedures
  - Indexes
- Exercises



# Requirements

# FURPS+

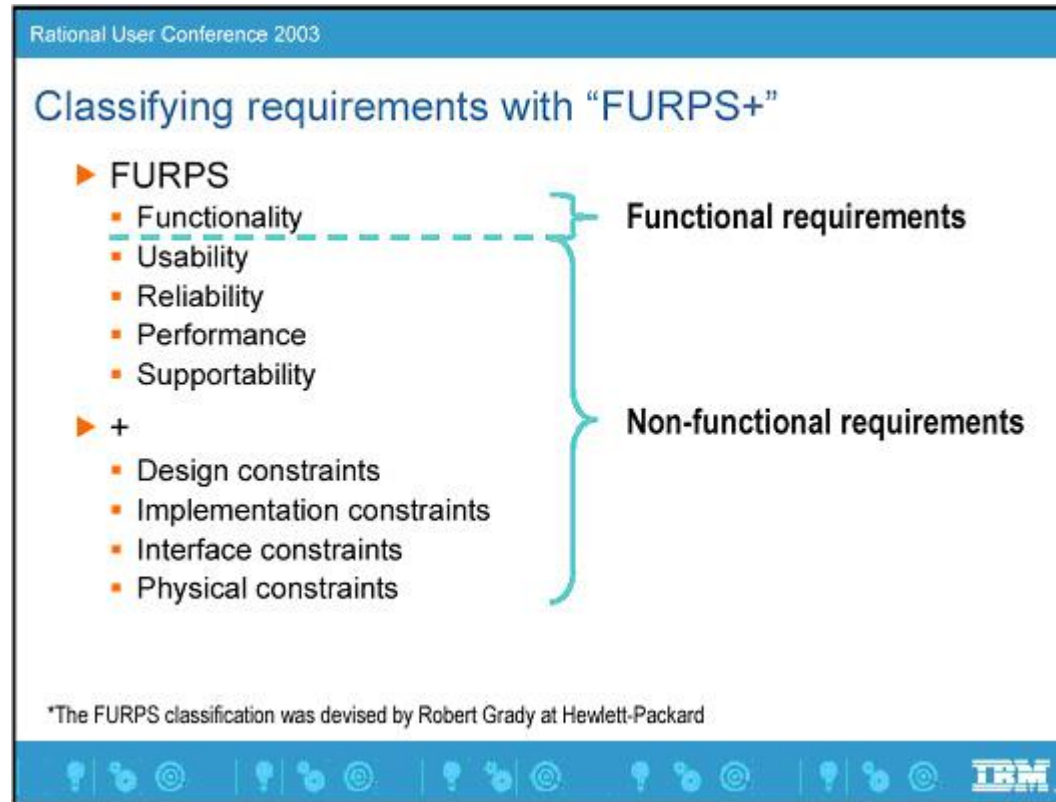
- What does FURPS acronym mean?

FURPS+	
Functionality	Plus:
Usability	Design constraints
Reliability	Implementation req'ts
Performance	Interface req'ts
Supportability	Physical req'ts

Source: <https://en.wikipedia.org/wiki/FURPS> &  
<http://agileinaflash.blogspot.dk/2009/04/furps.html>

# FURPS+

- What does FURPS acronym mean?



Source: <http://www.ibm.com/developerworks/rational/library/3975.html>

# Capturing constraints

- Design and implementation constraint example
  - What database type to choose?
  - What database product to choose?

Rational User Conference 2003

Architectural mechanisms

Analysis Mechanism	Design Mechanism	Implementation Mechanism
Persistence	RDBMS	DB2
		Oracle
	OODBMS	ObjectStore

IBM

Source: <http://www.ibm.com/developerworks/rational/library/3975.html>

# How to improve performance

- Techniques to improve response time
  - Indexes
  - Stored procedures
- Both are database techniques and both come at a price 😊
- In module 2, you will learn about performance improvement of your Java code



# Indexes



# Indexing

- If you often search by certain non-key column(s), you can speed up response time by putting an index on the column(s).



# Zipcodes example 1

- Get zipcodes from Danish postal service:
  - <http://www.postnord.dk/da/Privat/Kundeservice/postnummerkort/Sider/postnummerkort.aspx>
- We only need postnr + and bynavn in excel file. Remove rest of the fields.
- Remove duplicate zipcodes, e.g. 1055, 1165.
- Convert to UTF-8 format. Windows users might do this via Notepad.

# Zipcodes example 2

- Create table in SQL Workbench:

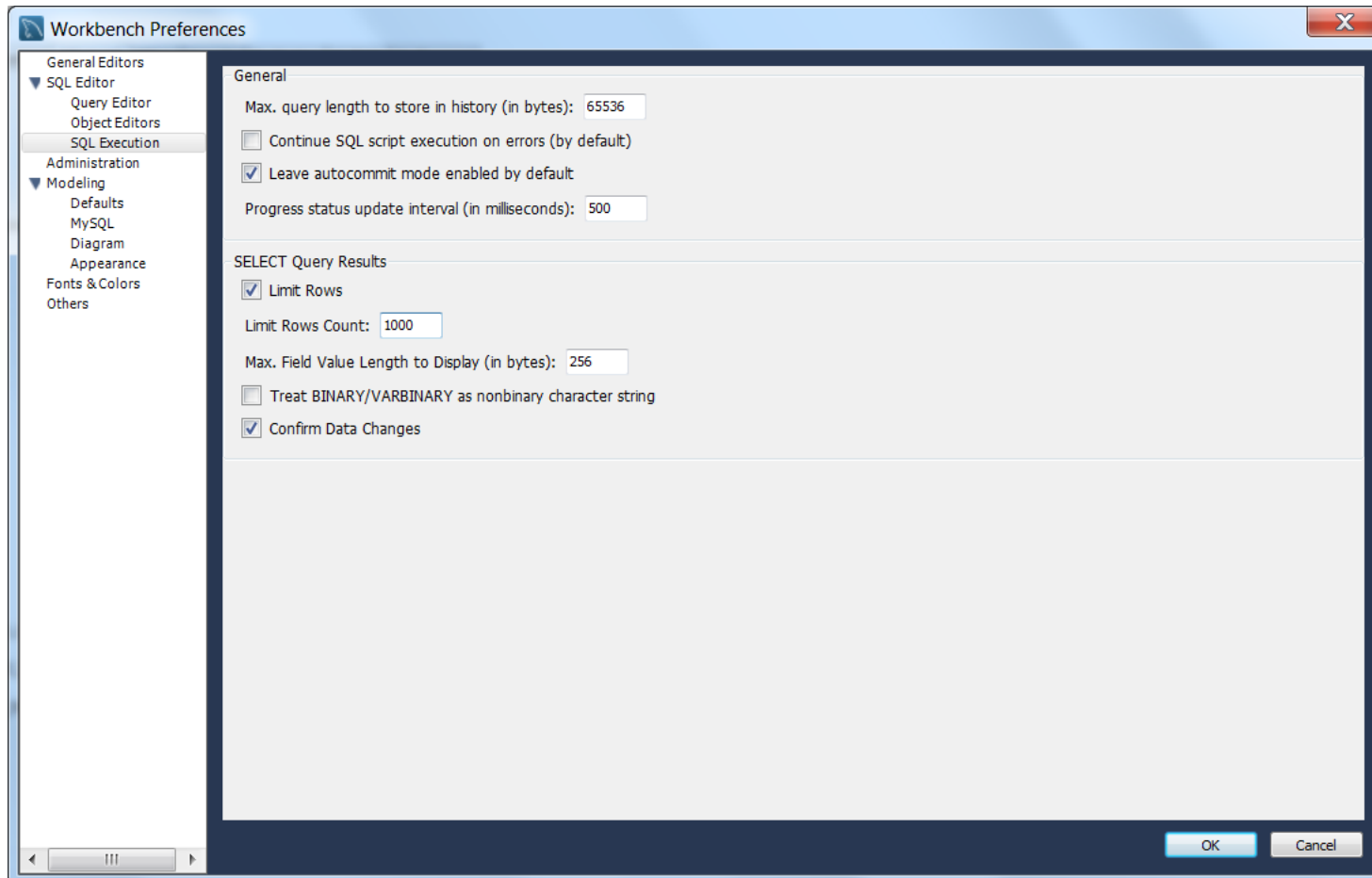
```
create table zipcodes(  
    zip int(4) not null,  
    city varchar(60) not null,  
    primary key (zip)  
)
```

- Import from file with LOAD DATA statement:

```
LOAD DATA INFILE 'C:/2sem2017/postnummerutf8.txt'  
INTO TABLE zipcodes  
FIELDS TERMINATED BY ';'   
LINES TERMINATED BY '\r\n'  
IGNORE 2 ROWS;
```

# Zipcodes example 3

- Workbench → Edit → Preferences – change **Limit Rows Count** to 2000.



# Zipcodes example 4

- Create copy of zip codes table (notice, you don't get any primary key!!!)

```
create table zipcodesindex as  
select * FROM zipcodes;
```

- Alternatively, create a table as normally in order to get a primary key and insert copy of data afterwards with this statement:

```
insert into zipcodesindex  
select * from zipcodes
```

# Zipcodes example 5

- Create index on new table

```
CREATE INDEX idx_zipcodes_city  
ON zipcodesindex (city)
```

Info

Columns

Indexes



Triggers

Foreign keys

Partitions

Grants

Indexes in Table

Key	Type	Uni...	Columns
 PRIMARY	BTREE	YES	zip
 idx_zipcodes_city	BTREE	NO	city

<

III

>

Index Details

Key Name:

idx\_zipcodes\_city

Index Type:

BTREE

Allows NULL:



Cardinality:

1340

Comment:

User Comment:

Columns in table

Column	Type	Nulla...	Indexes
 zip	int(11)	NO	PRIMARY
 city	varchar(60)	NO	idx_zipcodes_city

# Zipcodes example 6

## How to measure search with/without index

### Without index

```
SELECT * FROM zipcodes  
where city like 'M%';
```

#### Timing (as measured at client side):

Execution time: 0:00:0.00000000

#### Timing (as measured by the server):

Execution time: 0:00:0.00062948

Table lock wait time: 0:00:0.00000000

#### Errors:

Had Errors: NO

Warnings: 0

#### Rows Processed:

Rows affected: 0

Rows sent to client: 28

Rows examined: 1340

#### Joins per Type:

Full table scans (Select\_scan): 1

Joins using table scans (Select\_full\_join): 0

Joins using range search (Select\_full\_range\_join): 0

Joins with range checks (Select\_range\_check): 0

Joins using range (Select\_range): 0

#### Sorting:

Sorted rows (Sort\_rows): 0

Sort merge passes (Sort\_merge\_passes): 0

Sorts with ranges (Sort\_range): 0

Sorts with table scans (Sort\_scan): 0

#### Index Usage:

No Index used

### With index

```
SELECT * FROM zipcodesindex  
where city like 'M%';
```

#### Timing (as measured at client side):

Execution time: 0:00:0.00000000

#### Timing (as measured by the server):

Execution time: 0:00:0.00024974

Table lock wait time: 0:00:0.00000000

#### Errors:

Had Errors: NO

Warnings: 0

#### Rows Processed:

Rows affected: 0

Rows sent to client: 28

Rows examined: 28

#### Joins per Type:

Full table scans (Select\_scan): 0

Joins using table scans (Select\_full\_join): 0

Joins using range search (Select\_full\_range\_join): 0

Joins with range checks (Select\_range\_check): 0

Joins using range (Select\_range): 1

#### Sorting:

Sorted rows (Sort\_rows): 0

Sort merge passes (Sort\_merge\_passes): 0

Sorts with ranges (Sort\_range): 0

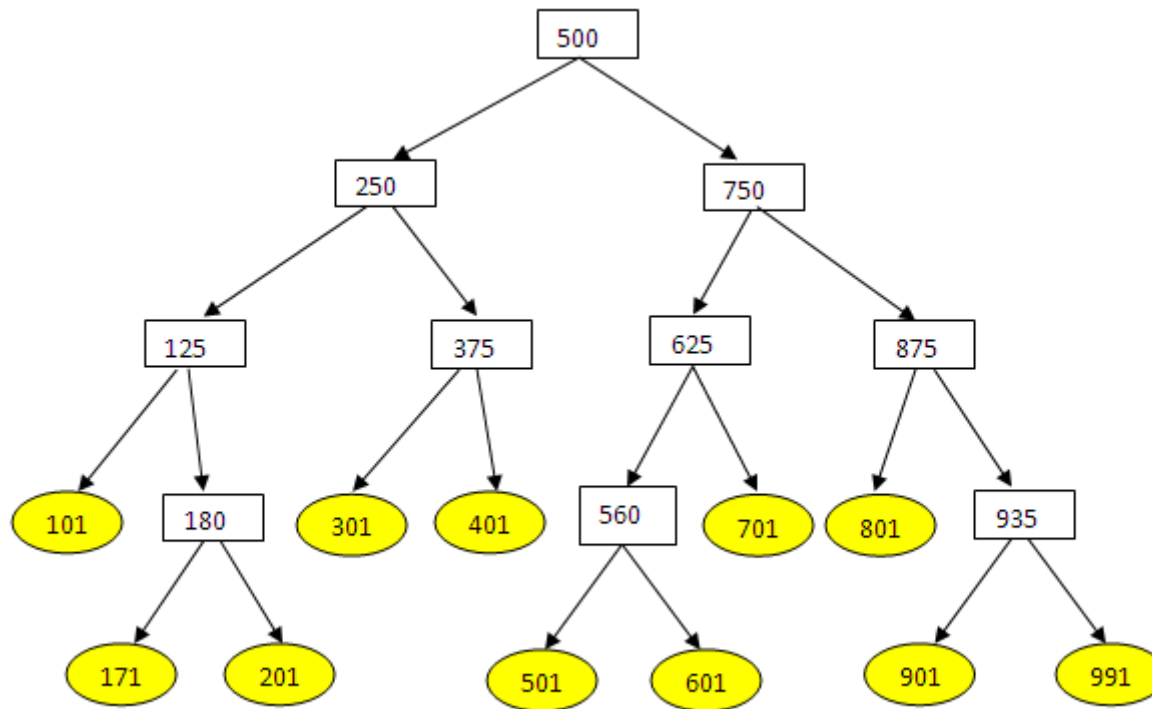
Sorts with table scans (Sort\_scan): 0

#### Index Usage:

At least one Index was used

# Zipcodes example 7

- Indexing comes at a price: every time a data record is changed or inserted, the B tree must be updated







# Stored procedures

# Stored procedure example 1

- Create stored procedure that retrieves all zip codes

```
DELIMITER //  
CREATE PROCEDURE GetAllZipcodes()  
BEGIN  
SELECT * FROM zipcodes;  
END //
```

Resource: <http://www.mysqltutorial.org/getting-started-with-mysql-stored-procedures.aspx>

# Stored procedure example 2

- Call stored procedure (in MySQL Workbench):

```
call GetAllZipcodes()
```

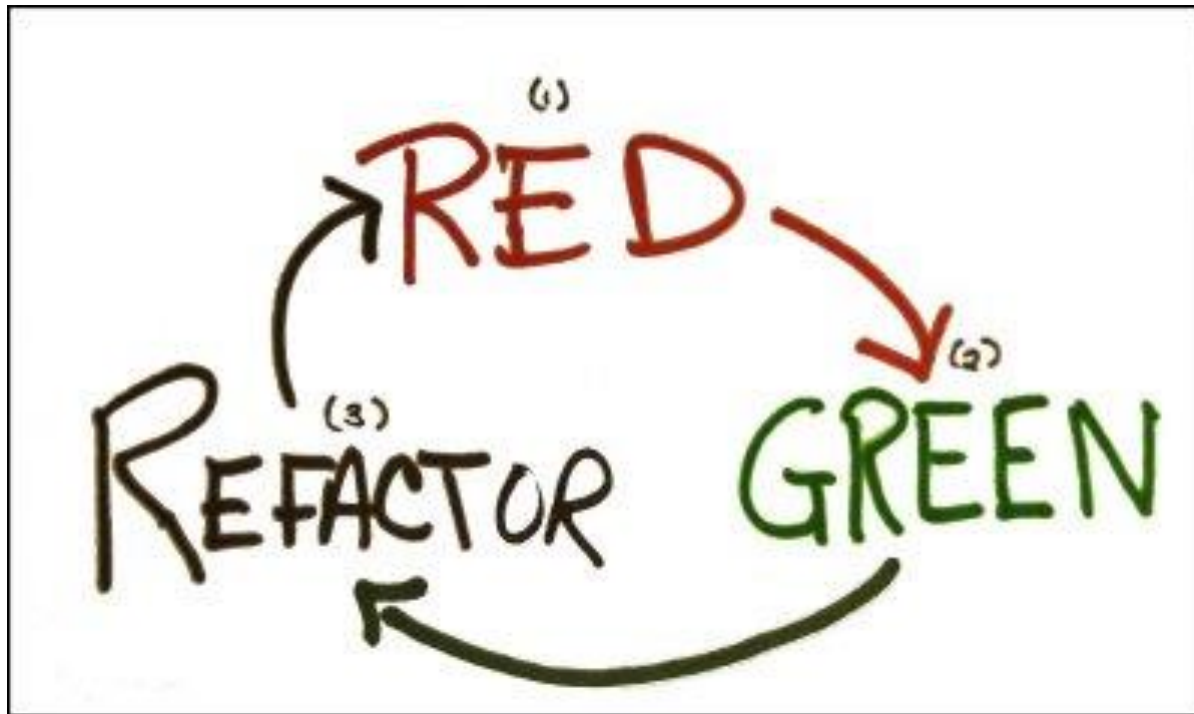
# Stored procedure example 3

Call stored procedure from jdbc code:

```
...
CallableStatement call = null;
try {
    call = con.prepareCall("{call GetAllZipcodes()}");
    boolean hadResults = call.execute();
    if (hadResults) {
        rs = call.getResultSet();
        while (rs.next()) {
            city = rs.getString("city");
            zip = rs.getInt("zip");
            codes.add(new Zipcode(zip, city));
        }
    }
}
...
```

# Test-Driven Development

- Let's see the stored procedure run for real!
- Let's do it in a JUnit test (tests give us confidence that our code is on the right track)



# Stored procedure parameters

```
DELIMITER //  
CREATE PROCEDURE GetZipcodes(IN cityName varchar(50))  
BEGIN  
SELECT * FROM zipcodes  
  where city like cityName;  
END //
```

## Call in JDBC code:

```
call = con.prepareCall("{call GetZipcodes(?)}");  
call.setString(1, city);
```

# Exercises

- See document in exercise folder