# Preparation

- Set your InnoDB Buffer Pool size properly (512+ M ?)
- Make your InnoDB Log File size bigger / big enough (32 – 64 M)
- Enable innodb_files_per_table (= 1)
- Load FoodMart-2.tar.gz (http://www.fromdual.com/mysql-consulting-tools)

# Exercises

## Vertical Partitioning

1. Copy table `customer` to schema `test` and load some random files into the `picture` field (blow the table really up!).
2. Then do some queries on the database an measure time (abolish first value and consider Query Cache).
3. Then split apart the picture field into a separate table and do the queries again on the first table.
4. How big is the size of both tables?
5. How many blocks does MySQL touch for the queries?

## Horizontal partitioning

1. Which ones of the `foodmart` tables would you consider for partitioning?
2. Partition the `sales_fact` table with per month range partitions (on `time_id`).
3. Partition the `inventory_fact` table by a warehouse list (`warehouse_id`).
4. Look in the Information Schema how skewed or even the data distribution is.
5. Drop the oldest partition of the `sales_fact` table.
6. Delete the data from the 2nd oldest partition with `DELETE` and from the 3rd oldest with `TRUNCATE PARTITON`. What is the performance difference?
7. Add a new partition to the `sales_fact` table.
8. Insert some rows into this new partition and above!
9. Add a new partition with `MAXVALUE` to the table `sales_fact`.
10. Insert some rows into the `MAXVALUE` partition and above!
11. You have realized that you forgot to add the partition for the next month! Split the `MAXVALUE` partition accordingly.
12. How much work for MySQL do you expect when you have already much data into the partition?
13. You want to delete the oldest month of the `inventory_fact` table. What do you think? Is it a good idea, what went wrong?
14. Compare the execution plans when you want to delete the last 7 days of the `inventory_fact` and the `sales_fact` table. Did the optimizer decide correctly?

## InnoDB

1. Preparation: Create the test table:

```
CREATE TABLE test.test (
  id int unsigned NOT NULL AUTO_INCREMENT
, data varchar(64) DEFAULT NULL
, ts timestamp NOT NULL
  DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
, PRIMARY KEY (id)
) ENGINE=InnoDB
;
```

2. Simple transactions:
   - Start a transaction, insert some rows then do a `COMMIT`.
   - Start a transaction, insert some rows then do a `ROLLBACK`.
   What is the difference?

3. Have 2 connections open:

```
conn1> SET session tx_isolation = 'read-uncommitted';
conn1> START TRANSACTION;
conn2> START TRANSACTION;
conn1> INSERT INTO test VALUES (NULL, 'Read uncommitted', NULL);
```

   What do you expect from the following queries?

```
conn1> SELECT * FROM test;
conn2> SELECT * FROM test;
conn1> ROLLBACK;
conn1> SELECT * FROM test;
conn2> SELECT * FROM test;
conn2> ROLLBACK;
conn1> SELECT * FROM test;
conn2> SELECT * FROM test;
```

4. Now do the same thing with the default isolation level and play around with the other isolation levels...

5. Create the tables `orders` and `order_items` in the schema `test`:

```
CREATE TABLE test.orders (
  id INT UNSIGNED NOT NULL AUTO_INCREMENT
, order_date DATE NOT NULL
, order_title VARCHAR(45)
, PRIMARY KEY (id)
) ENGINE = InnoDB;

CREATE TABLE test.order_items (
  id INT UNSIGNED NOT NULL AUTO_INCREMENT
, position SMALLINT UNSIGNED NOT NULL
, item VARCHAR(45) NOT NULL
, amount SMALLINT SIGNED NOT NULL
, order_id INT UNSIGNED NOT NULL
, PRIMARY KEY (id)
```

```
, INDEX order_id (order_id)
, CONSTRAINT order_id_fk FOREIGN KEY (order_id)
  REFERENCES test.orders(id) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE = InnoDB;
```

Then insert some rows into the `orders` and `order_items` table:

```
INSERT INTO orders
VALUES (1, '2011-07-28', 'Order 1')
     , (2, '2011-07-29', 'Order 2')
     , (3, '2011-07-30', 'Order 3');
INSERT INTO order_items
VALUES (1, 1, 'Item 1', 2, 1), (2, 2, 'Item 2', 4, 1)
     , (3, 1, 'Item 1', 3, 2), (4, 2, 'Item 2', 5, 2)
     , (5, 3, 'Item 3', 1, 2), (6, 1, 'Item 1', 6, 3);
INSERT INTO order_items
VALUES (7, 1, 'Item 1', 9, 4);
```

Why is the 3rd insert not working? Look at SHOW ENGINE INNODB STATUS\G

6. Change the `id` of table `orders` from 1 to 4. What happens with the `order_id` of the `order_items` table?

7. Delete the order with `id` = 3. What happens with the order items of this order?

8. InnoDB locking, you have 2 connections:

```
conn1> START TRANSACTION;
conn1> SELECT * FROM order_items FOR UPDATE;
conn2> START TRANSACTION;
conn2> DELETE FROM order_items WHERE id = 5;
```

What happens? Be a bit patient! After how many seconds does it happen? What InnoDB variable does this fit to?

9. InnoDB Deadlock, you have 2 connections:

```
conn1> START TRANSACTION;
conn2> START TRANSACTION;
conn1> SELECT * FROM order_items WHERE order_id = 2 FOR UPDATE;
conn2> DELETE FROM orders WHERE id = 2;
conn1> UPDATE order_items SET order_id = 3 WHERE order_id = 2;
```

What happens? What happens with the other transaction? Look at the output of `SHOW ENGINE INNODB STATUS\G`

10. `innodb_file_per_table`: I hope you have followed the advice in the preparations in the very beginning... Otherwise you have to go back to step 1 and do all the steps until here again! :-P
    Search and find the InnoDB files of your partitions: How does it look like??

11. Examine the Information of the InnoDB Buffer Pool with:
    - MySQL Status variables,
    - InnoDB Status and
    - the INFORMATION_SCHEMA.

12. Start the InnoDB Table Monitor and examine the `foodmart` tables in the MySQL error log. Especially the big tables (with no PK) are interesting. What is the difference between a table with a PK and one with no PK?
    It would be interesting as well to compare the pages of partitioned tables with non-partitioned tables.

13. InnoDB Primary Key and Locality of data: Create 2 copies of `sales_fact`. Then add a column with an id `INT UNSIGNED AUTO_INCREMENT` to this copy and modify the PK as follows:
    - With a PK on (id, customer_id) and
    - With a PK on (customer_id, id).
    Then look at `SHOW TABLE STATUS` and compare both tables.
    Look at the Query Execution Plans for those 2 statements. What do you see?

```
EXPLAIN
SELECT * FROM T1 WHERE customer_id BETWEEN 1000 AND 1499;

EXPLAIN
SELECT * FROM T2 WHERE customer_id BETWEEN 1000 AND 1499;
```

    What is performance difference and why? What says the Optimizer / QEP? How many blocks do we touch?

14. Do an `OPTIMIZE TABLE` and compare both table with `SHOW TABLE STATUS`. What do you see?

15. Drop and re-create all secondary indexes and compare with `SHOW TABLE STATUS` from above. How long does it take per index? Look into the process list during this step.
    What do you see, let us discuss in class?
    How does it work with partitions?

16. Find the information about:
    - Adaptive Hash Index
    - Double Write Buffer
    - Purge threads
    - Change Buffer etc.
    in in the MySQL status or InnoDB status information.

## Performance Tuning and Monitoring

1. Do the steps above (optimizing a table and optimizing indexes with fast index drop/create) again but this time monitor with:
   - `top`
   - `vmstat 1`
   - `iostat -xk 1`
   - `dstat 1`
   - `watch -d -n 1 'ifconfig'`

   What do you see. Where/who is the bottleneck?

2. Why is `vmstat`, `iostat` and `top` (with SMP view disabled) bad for seeing CPU bottlenecks? Use `mpstat -P ALL 1` instead!

3. Do a MySQL Profile of an `OPTIMIZE TABLE` and a fast index `DROP/CREATE`. Compare those two Profiles. Where is most of the time lost?

4. Study the sample program (profiling.php), let it run and play a bit around with it. Could you add something to your application as well? Let us discuss in class, why or why not?

5. Try to find out how much "non accounted" time do you have in the program (profiling.php)? What methods do you have to find those? What could be the advantages and disadvantages of the different approaches?

## Benchmarking and Benchmarking Tools

1. Define your personal benchmark goal for the foodmart application. Define a result to achieve with tuning / benchmarking. What is your question you want to answer with this benchmark?

2. Run some non-database benchmarks with `sysbench`.
   ```
   shell> sysbench --help
   ```
   Do those benchmarks help you answering your question or find the goal of Exercise 1? What types of questions do those benchmarks answer?

3. Run the OLTP benchmark from `sysbench` against your MySQL database.

4. Change some of the MySQL parameters which you expect to have an influence on the OLTP benchmark and benchmark again.

5. Is this benchmark useful to give you an answer to your original question defined in Exercise 1? Discuss in class!

6. Try the same with `slamd` and `grinder`.

7. Let us do a simple webShop Benchmark with `JMeter` (you can also try to do it on your own with `slamd` and `grinder` if you prefer but then you are on your own!).
   I have defined the webShop Benchmark as follows according to by needs and the question I want to have answered from Exercise 1:
   - login
   - add 1-n articles to the basket
   - buy these articles
   - logout

   You can have a look in the script webshop_benchmark.php

8. Where are the problems of our Benchmark, where are the weaknesses. What is missing?

9. I am pretty sure in the web-shop application is still some optimization potential. Does the Benchmarking Tool help us finding this potential? What about a profile of our web-shop benchmark?

## Memcached

1. Everybody should set-up a Memcached.

2. Use telnet first for some simple set key and get key tests.

3. Let us move the session part of our Webshop Benchmark into the Memcached. See example webshop_benchmark_memcached.php

4. Run the webshop_benchmark_memcached.php example.

5. Change the JMeter Benchmark so it can use the Memcached (if this is possible at all?)

6. Let us do a big cluster of Memcached's and adapt our our webshop_benchmark_memcached.php script accordingly. Verify that you really get some data from the others in your memcached?

7. What happens if somebody suddenly stops his memcached?

## HA and Replication Solutions

1. Who has experience with Master/Slave replication?

2. Two and two together, set-up a Master/Slave or Master/Master replication.

3. Set-up a Semi-Sync Replication. What happens when you make it “sharp” and Slaves goes down?

## BLOB Streaming Engine

1. Patch and Compile MySQL 5.5.12 with PBMS. Install PHP modules. Try to write and read a BLOB from MySQL.

## HandlerSocket

1. Download MariaDB or Percona Server and do some simple fetches over telnet.

## Memcached API

1. Download MySQL 5.6.3 and try to do some accesses over the Memcached API via telnet.

## Solutions

Will be provided on request...