Boosting performance with MySQL partitions

Giuseppe Maxia
MySQL Community Team Lead at Oracle
about me - Giuseppe Maxia

- a.k.a. The Data Charmer
- MySQL Community Team Lead
- Long time hacking with MySQL features
- Formerly, database consultant, designer, coder.
- A passion for QA
- An even greater passion for open source
- ... and community
- Passionate blogger
- http://datacharmer.blogspot.com
Slides for this presentation

http://tinyurl.com/mysql-partitions

This work is licensed under the Creative Commons Attribution-Share Alike 3.0 Unported License.
Table of contents

- Your needs
- What
- Partition pruning
- Benchmarking
- Syntax
- Partitioning by date
- hands on
- tools
- tips
- leveraging replication
The problem(s)

• Too much data
• Not enough RAM
• Historical data
• Growing data
• Rotating data
Too much data
Not enough RAM
Not enough RAM
Table of contents

Your needs
- What
  - Partition pruning
  - Syntax
  - Benchmarking
  - Partitioning by date
  - hands on
  - tools
  - tips
  - leveraging replication
What exactly is this "partitions" thing?

- Logical splitting of tables
- Transparent to user
Remember the MERGE tables?

- separate tables
- risk of duplicates
- insert in each table
- no constraints
It isn't a merge table!

- One table
- No risk of duplicates
- insert in one table
- constraints enforced
Wait a minute ...

• WHAT THE HELL DOES "LOGICAL SPLIT" REALLY MEANS?
• LET ME EXPLAIN ...
Physical partitioning (1)

- Take a map
Physical partitioning (2)

• cut it into pieces
Physical partitioning (3)

- What you have, is several different pieces
Physical partitioning (4)

• If you want the map back, you need some application (adhesive tape) and you may get it wrong
Logical partitioning (1)

- Take a map
Logical partitioning (2)

• fold it to show the piece you need
Logical partitioning (3)

• what you have is still a map, even if you see only one part.
Logical partitioning (4)

• if you unfold the map, you still have the whole thing
Table of contents

Your needs

What

Partition pruning

Syntax

Benchmarking

Partitioning by date

hands on

tools

tips

leveraging replication
Partition pruning
1a - unpartitioned table - SINGLE RECORD

select *
from
table_name
where colx = 120
Partition pruning
1a - unpartitioned table - SINGLE RECORD

```
select *
from
table_name
where colx = 120
```
Partition pruning
1b - unpartitioned table - SINGLE RECORD

```sql
select *
from
table_name
where colx = 350
```
Partition pruning
1c - unpartitioned table - RANGE

select *  
from table_name  
where colx between 120 and 230
Partition pruning
2a - table partitioned by colx - SINGLE REC

```sql
select *
from table_name
where colx = 120
```
Partition pruning
2a - table partitioned by colx - SINGLE REC

```sql
select *
from table_name
where colx = 120
```
Partition pruning
2b - table partitioned by colx - SINGLE REC

select *
from
table_name
where colx = 350
Partition pruning
2c - table partitioned by colx - RANGE

```
select *
from
table_name
where colx
between 120
and 230
```
Partition pruning

**before**

```sql
EXPLAIN
select *
from table_name
where colx = 120
```

**in 5.1**

```sql
EXPLAIN PARTITIONS
select *
from table_name
where colx = 120
```
Partition pruning - unpartitioned table

```
explain partitions select count(*)
from table_name where colx=120\G

***** 1. row ****
  id: 1
  select_type: SIMPLE
  table: table_name
  partitions: NULL
  type: index

...
Partition pruning - unpartitioned table

```
explain partitions select count(*)
from table_name where colx between 120 and 230
G

***** 1. row *****
id: 1
  select_type: SIMPLE
  table: table_name
  partitions: NULL
  type: index

...
**Partition pruning - table partitioned by colx**

```
explain partitions select count(*)
from table_name where colx between 120 and 230\G

***** 1. row ****

  id: 1
  select_type: SIMPLE
  table: table_name
  partitions: p02,p03
  type: index

...
Partitioning Types

- Partition a table using CREATE TABLE or ALTER TABLE

- CREATE TABLE `<table_name>` ( `<columns>` )
  
  ENGINE=`<engine_name>`

  PARTITION BY `<type>` ( `<partition_expression>` );

- `<type>` can be
  - RANGE
  - LIST
  - HASH
  - KEY
CREATE TABLE Employee (
    emp_id INT AUTO_INCREMENT,
    name VARCHAR(50),
    store_id TINYINT,
    PRIMARY KEY (emp_id)
) ENGINE=MyISAM

PARTITION BY RANGE (emp_id) (
    PARTITION p0 VALUES LESS THAN (10000),
    PARTITION p1 VALUES LESS THAN (20000),
    PARTITION p2 VALUES LESS THAN (30000),
    PARTITION p3 VALUES LESS THAN (40000),
    PARTITION p4 VALUES LESS THAN MAXVALUE)

- MAXVALUE is optional
- Partition ranges must be listed smallest to greatest and must be integers
Benchmarking partitions

- Compare results
- Unpartitioned vs partitioned
- ISOLATION
- Repeatability
- Check your resources!
Benchmarking partitions - Compare results

- Execute query
- Record execution time
- CHECK RETRIEVED RECORDS!
Benchmarking partitions - Unpartitioned vs partitioned

- Make it simple.
- Do not change structure
- If needed, remove PK from partitioned table
Benchmarking partitions - ISOLATION

- Try to reproduce working conditions
- No other servers running while benchmarking
- Restart the server before each test
- Do NOT mix partitioned and unpartitioned tables in the same server
- Use MySQL Sandbox
Benchmarking partitions - Repeatability

- Measure more than once
- Make sure you have the same conditions
- Make sure your results are consistent between runs
Benchmarking partitions - Check resources

- InnoDB
  - check disk space (uses more than MyISAM)
  - check CPU usage
- Partitioned MyISAM tables
  - use 2 file handles per partition
  - If you use more than one partitioned table, count total file handles
- If you use Archive partitioned tables
  - check CPU usage
  - check memory usage
Table of contents

- Your needs
- What
- Partition pruning
- Benchmarking
- Partitioning by date
  - hands on
  - tools
  - tips
  - leveraging replication
Partitioning by date - limits of partitioning

- Can partition only by INTEGER columns
- OR you can partition by an expression, which must return an integer
Partitioning by date - your (GOOD) options

- GOOD ideas
  - YEAR(date_column)
  - TO_DAYS(date_column)

- WHY?
  - Both functions optimized for partitions
  - Partition pruning does not kick with other functions
CREATE TABLE t1 ( 
    d date
);
PARTITION BY RANGE (YEAR(d))
(
    PARTITION p01 VALUES LESS THAN (1999),
    PARTITION p02 VALUES LESS THAN (2000),
    PARTITION p03 VALUES LESS THAN (2001),
    PARTITION p04 VALUES LESS THAN (MAXVALUE)
);
CREATE TABLE t1 ( 
    d date 
) 
PARTITION BY RANGE (TO_DAYS(d)) 
( 
    PARTITION p01 VALUES LESS THAN 
        (TO_DAYS('2008-01-01')) , 
    PARTITION p02 VALUES LESS THAN 
        (TO_DAYS('2008-02-01')) , 
    PARTITION p03 VALUES LESS THAN 
        (TO_DAYS('2008-03-01')) , 
    PARTITION p04 VALUES LESS THAN (MAXVALUE) 
);
Partitioning by date - How TO

- How it works
  - partition BY FUNCTION
  - query BY COLUMN
Partitioning by date - WRONG!!!

```
PARTITION BY RANGE (year(from_date))

select count(*) from salaries where year(from_date) = 1998;

+----------+
| count(*) |
+----------+
| 247489   |
+----------+

1 row in set (2.25 sec)
```
Partitioning by date - RIGHT

PARTITION BY RANGE(\text{year(from\_date)})

\text{select count(*) from salaries where from\_date between '1998-01-01' and '1998-12-31';}

\begin{tabular}{l}
\text{+----------+} \\
| count(*) | \\
\text{+----------+} \\
| 247489 | \\
\text{+----------+} \\
\end{tabular}

1 row in set (0.46 sec)
Partitioning by date - EXPLAIN

explain partitions select count(*) from salaries where year(from_date) = 1998\G

***** 1. row ****

    id: 1

    select_type: SIMPLE

    table: salaries

    partitions:
    p01,p02,p03,p04,p05,p06,p07,p08,p09,p10,p11,p12,p13,p14,p15,p16,p17,p18,p19

    type: index

...
Partitioning by date - EXPLAIN

```
explain partitions select count(*) from salaries where from_date between '1998-01-01' and '1998-12-31'

***** 1. row ****

id: 1

select_type: SIMPLE

table: salaries

partitions: p14,p15

...
```
Partitioning by date with different sizes

- Mixing partitions by year, month, day in the same table

HOW TO:
- Use the "TO_DAYS" function
- Set appropriate intervals
Partitioning by date with different sizes

ALTER TABLE salaries

partition by range (to_days(from_date))

(
  # 5 years
  partition p01 values less than (to_days('1985-01-01')),
  partition p06 values less than (to_days('1990-01-01')),
  # 1 year
  partition p11 values less than (to_days('1995-01-01')),
  partition p12 values less than (to_days('1996-01-01')),
  partition p13 values less than (to_days('1997-01-01')),
  partition p14 values less than (to_days('1998-01-01')),
  partition p15 values less than (to_days('1999-01-01')),
  partition p16 values less than (to_days('2000-01-01')),
Partitioning by date with different sizes

ALTER TABLE salaries
partition by range (to_days(from_date))
(
  # 1 month
  partition p17 values less than (to_days('2001-01-01')),
  partition p18 values less than (to_days('2001-02-01')),
  partition p19 values less than (to_days('2001-03-01')),
  partition p20 values less than (to_days('2001-04-01')),
  partition p21 values less than (to_days('2001-05-01')),
  partition p22 values less than (to_days('2001-06-01')),
  partition p23 values less than (to_days('2001-07-01')),
  partition p24 values less than (to_days('2001-08-01')),
  partition p25 values less than (to_days('2001-09-01')),
  partition p26 values less than (to_days('2001-10-01')),
  partition p27 values less than (to_days('2001-11-01')),
  partition p28 values less than (to_days('2001-12-01')))
Hands on - Partitioning with MyISAM

- Primary key matters - not always needed
- Size of indexes matters
components used for testing

- MySQL Sandbox
  - created MySQL instances in seconds
  - http://launchpad.net/mysql-sandbox

- MySQL Employees Test Database
  - has ~ 4 mil records in 6 tables
  - http://launchpad.net/test-db

- Command line ability
- fingers
- ingenuity
employees test database
How many partitions

dataframe from `information_schema.partitions`

<table>
<thead>
<tr>
<th>pname</th>
<th>expr</th>
<th>descr</th>
</tr>
</thead>
<tbody>
<tr>
<td>p01</td>
<td><code>year(from_date)</code></td>
<td>1985</td>
</tr>
<tr>
<td>p02</td>
<td><code>year(from_date)</code></td>
<td>1986</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>p13</td>
<td><code>year(from_date)</code></td>
<td>1997</td>
</tr>
<tr>
<td>p14</td>
<td><code>year(from_date)</code></td>
<td>1998</td>
</tr>
<tr>
<td>p15</td>
<td><code>year(from_date)</code></td>
<td>1999</td>
</tr>
<tr>
<td>p16</td>
<td><code>year(from_date)</code></td>
<td>2000</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>p19</td>
<td><code>year(from_date)</code></td>
<td>MAXVALUE</td>
</tr>
</tbody>
</table>
How many records

```
select count(*) from salaries;
```

```
+----------+
| count(*) |
+----------+
| 2844047  |
+----------+
```
How many records in 1998 not partitioned

```sql
select count(*) from salaries where from_date between '1998-01-01' and '1998-12-31';
```

```
+----------+
| count(*) |
+----------+
|   247489 |
+----------+
1 row in set (1.52 sec)
```

# NOT PARTITIONED
How many records in 1998 partitioned

```sql
select count(*) from salaries where from_date between '1998-01-01' and '1998-12-31';
```

```
+----------+
| count(*) |
+----------+
|   247489 |
+----------+
1 row in set (0.41 sec)
```

# partition p15
Deleting records in 1998

NOT partitioned

delete from salaries where from_date between '1998-01-01' and '1998-12-31';
Query OK, 247489 rows affected (19.13 sec)

# NOT PARTITIONED
Deleting records in 1998 partitioned

```
alter table salaries drop partition p15;
Query OK, 0 rows affected (1.35 sec)
```
Partitions with InnoDB

- Slower than MyISAM
- But more robust
- Requires more storage
CREATE TABLE table_name (  
  ...  
  ...  
) ENGINE = INNODB  
ROW_FORMAT=COMPRESSED KEY_BLOCK_SIZE=8  
PARTITION BY XXXX
Partitions with InnoDB (laptop)

- Key points:
- Takes much more storage than other engines

<table>
<thead>
<tr>
<th>engine</th>
<th>storage (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>innodb</td>
<td>221</td>
</tr>
<tr>
<td>myisam</td>
<td>181</td>
</tr>
<tr>
<td>archive</td>
<td>74</td>
</tr>
<tr>
<td>innodb partitioned (whole)</td>
<td>289</td>
</tr>
<tr>
<td>innodb partitioned (file per table)</td>
<td>676</td>
</tr>
<tr>
<td>myisam partitioned</td>
<td>182</td>
</tr>
<tr>
<td>archive partitioned</td>
<td>72</td>
</tr>
</tbody>
</table>

Monday, 13 September 2010
## Benchmarking results (laptop)

<table>
<thead>
<tr>
<th>engine</th>
<th>query year 2000</th>
<th>query year 2002</th>
</tr>
</thead>
<tbody>
<tr>
<td>InnoDB</td>
<td>1.25</td>
<td>1.25</td>
</tr>
<tr>
<td>MyISAM</td>
<td>1.72</td>
<td>1.73</td>
</tr>
<tr>
<td>Archive</td>
<td>2.47</td>
<td>2.45</td>
</tr>
<tr>
<td>InnoDB partitioned whole</td>
<td>0.24</td>
<td>0.10</td>
</tr>
<tr>
<td>InnoDB Partitioned (file per table)</td>
<td>0.45</td>
<td>0.10</td>
</tr>
<tr>
<td>MyISAM partitioned</td>
<td>0.18</td>
<td>0.12</td>
</tr>
<tr>
<td>Archive partitioned</td>
<td>0.22</td>
<td>0.12</td>
</tr>
</tbody>
</table>
Partitions with InnoDB (huge server)

- Key points:
  - Takes much more storage than other engines

<table>
<thead>
<tr>
<th>engine</th>
<th>storage (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>innodb (with PK)</td>
<td>330</td>
</tr>
<tr>
<td>myisam (with PK)</td>
<td>141</td>
</tr>
<tr>
<td>archive</td>
<td>13</td>
</tr>
<tr>
<td>innodb partitioned (no PK)</td>
<td>237</td>
</tr>
<tr>
<td>myisam partitioned (no PK)</td>
<td>107</td>
</tr>
<tr>
<td>archive partitioned</td>
<td>13</td>
</tr>
</tbody>
</table>
### Benchmarking results (huge server)

<table>
<thead>
<tr>
<th>engine</th>
<th>6 month range</th>
</tr>
</thead>
<tbody>
<tr>
<td>InnoDB</td>
<td>4 min 30s</td>
</tr>
<tr>
<td>MyISAM</td>
<td>25.03s</td>
</tr>
<tr>
<td>Archive</td>
<td>22 min 25s</td>
</tr>
<tr>
<td>InnoDB partitioned by month</td>
<td>13.19</td>
</tr>
<tr>
<td>MyISAM partitioned by year</td>
<td>6.31</td>
</tr>
<tr>
<td>MyISAM partitioned by month</td>
<td>4.45</td>
</tr>
<tr>
<td>Archive partitioned by year</td>
<td>16.67</td>
</tr>
<tr>
<td>Archive partitioned by month</td>
<td>8.97</td>
</tr>
</tbody>
</table>

Monday, 13 September 2010
Partitions with Archive

- Key points:
- REMOVE INDEXES (not supported by archive)
- For pure statistics applications, ARCHIVE can be ALMOST AS FAST AS MyISAM!
Partitions with ARCHIVE - in practice

```
CREATE TABLE titles (  
    emp_no INT NOT NULL,
    title VARCHAR(50) NOT NULL,
    from_date DATE NOT NULL,
    to_date DATE,
    KEY (emp_no),
    FOREIGN KEY (emp_no) REFERENCES employees
        (emp_no) ON DELETE CASCADE,
    PRIMARY KEY (emp_no,title,
                  from_date)
) ENGINE = InnoDB;
```
CREATE TABLE titles (  
emp_no INT NOT NULL,  
title VARCHAR(50) NOT NULL,  
from_date DATE NOT NULL,  
to_date DATE #,  
# KEY (emp_no),  
# FOREIGN KEY (emp_no)  
# REFERENCES employees  
# (emp_no) ON DELETE CASCADE,  
# PRIMARY KEY (emp_no,title,  
# from_date)  
) ENGINE = ARCHIVE;
Table of contents

- Your needs
- What
- Partition pruning
- Benchmarking
- Partitioning by date
- hands on tools
- tips
- leveraging replication
TOOLS

- The INFORMATION_SCHEMA.PARTITIONS table
- The partition helper
  - A Perl script that creates partitioning statements
- The mysqldump_partition_backup
- ... anything you are creating right now :)
| TABLE_NAME                    |          |
| PARTITION_NAME               |          |
| SUBPARTITION_NAME            |          |
| PARTITION_ORDINAL_POSITION   |          |
| SUBPARTITION_ORDINAL_POSITION|          |
| PARTITION_METHOD             |          |
| SUBPARTITION_METHOD          |          |
| PARTITION_EXPRESSION         |          |
| SUBPARTITION_EXPRESSION      |          |
| PARTITION_DESCRIPTION        |          |
| TABLE_ROWS                   |          |
| PARTITION_COMMENT            |          |
INFORMATION SCHEMA PARTITIONS table

```
select * from partitions where table_name='salaries'
and table_schema='employees' limit 1

TABLE_SCHEMA: employees
TABLE_NAME: salaries
PARTITION_NAME: p01
SUBPARTITION_NAME: NULL
PARTITION_ORDINAL_POSITION: 1
SUBPARTITION_ORDINAL_POSITION: NULL
PARTITION_METHOD: RANGE COLUMNS
SUBPARTITION_METHOD: NULL
PARTITION_EXPRESSION: from_date
SUBPARTITION_EXPRESSION: NULL
PARTITION_DESCRIPTION: '1985-12-31'
TABLE_ROWS: 18238
```
```sql
select partition_name, partition_description, table_rows
from partitions where table_name='salaries' and table_schema='employees';
```

<table>
<thead>
<tr>
<th>partition_name</th>
<th>partition_description</th>
<th>table_rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>p01</td>
<td>'1985-12-31'</td>
<td>18238</td>
</tr>
<tr>
<td>p02</td>
<td>'1986-12-31'</td>
<td>37915</td>
</tr>
<tr>
<td>p03</td>
<td>'1987-12-31'</td>
<td>57395</td>
</tr>
<tr>
<td>p04</td>
<td>'1988-12-31'</td>
<td>76840</td>
</tr>
<tr>
<td>p05</td>
<td>'1989-12-31'</td>
<td>95890</td>
</tr>
<tr>
<td>p06</td>
<td>'1990-12-31'</td>
<td>114520</td>
</tr>
<tr>
<td>p07</td>
<td>'1991-12-31'</td>
<td>132578</td>
</tr>
<tr>
<td>p08</td>
<td>'1992-12-31'</td>
<td>151019</td>
</tr>
<tr>
<td>p09</td>
<td>'1993-12-31'</td>
<td>168103</td>
</tr>
<tr>
<td>p10</td>
<td>'1994-12-31'</td>
<td>185121</td>
</tr>
<tr>
<td>p11</td>
<td>'1995-12-31'</td>
<td>201576</td>
</tr>
<tr>
<td>p12</td>
<td>'1996-12-31'</td>
<td>218244</td>
</tr>
<tr>
<td>p13</td>
<td>'1997-12-31'</td>
<td>233144</td>
</tr>
<tr>
<td>p14</td>
<td>'1998-12-31'</td>
<td>247458</td>
</tr>
</tbody>
</table>

Monday, 13 September 2010
The Partition helper

- Introduction, syntax, and examples
The Partition Helper, version 1.0.1
This program creates a ALTER TABLE statement to add or reorganize partitions for MySQL 5.1 or later
(C) 2008 Giuseppe Maxia

Syntax: partitions_helper [options]
-t --table = name
-c --column = name
-i --interval = name
   "year", "month", or a number
-p --partitions = number

...
The partition helper

... 

`--first_partition = number`
`--reorganize = name`
`s --start = name`
`e --end = name`
`f --function = name`
`l --list`
`--prefix = name`
`--explain`
The partition helper example

partitions_helper --table=t1 \n  --column=mydate \n  --interval=year \n  --start=1998-01-01 --end=2002-12-01

# partitions: 5
ALTER TABLE t1
PARTITION by range (to_days(mydate))
(
  partition p001 VALUES LESS THAN (to_days('1998-01-01'))
  , partition p002 VALUES LESS THAN (to_days('1999-01-01'))
  , partition p003 VALUES LESS THAN (to_days('2000-01-01'))
  , partition p004 VALUES LESS THAN (to_days('2001-01-01'))
  , partition p005 VALUES LESS THAN (to_days('2002-01-01'))
);
The partition helper example

```
partitions_helper --table=t1 \  
   --column=mydate \  
   --interval=month \  
   --start=1998-01-01 --end=2002-12-01
#

# partitions: 60

ALTER TABLE t1
PARTITION by range (to_days(mydate))
(
   partition p001 VALUES LESS THAN (to_days('1998-01-01'))
, partition p002 VALUES LESS THAN (to_days('1998-02-01'))
, partition p003 VALUES LESS THAN (to_days('1998-03-01'))
# [...]
, partition p058 VALUES LESS THAN (to_days('2002-10-01'))
, partition p059 VALUES LESS THAN (to_days('2002-11-01'))
, partition p060 VALUES LESS THAN (to_days('2002-12-01'))
);
```
The partition helper example

```
partitions_helper --table=t1
    --column=mydate
    --interval=week
    --start=1998-01-01 --end=2002-12-01

# partitions: 255
ALTER TABLE t1
PARTITION by range (to_days(mydate))
(
    partition p001 VALUES LESS THAN (to_days('1998-01-01'))
    , partition p002 VALUES LESS THAN (to_days('1998-01-08'))
    , partition p003 VALUES LESS THAN (to_days('1998-01-15'))
    #[...]
    , partition p253 VALUES LESS THAN (to_days('2002-10-31'))
    , partition p254 VALUES LESS THAN (to_days('2002-11-07'))
    , partition p255 VALUES LESS THAN (to_days('2002-11-14'))
);
```
The partition helper example

```
partitions_helper --table=t1 \
  --column=mynum \ 
  --interval=250 \ 
  --start=1000 --end=3000

# partitions: 8
ALTER TABLE t1
PARTITION by range (mydate)
(
  partition p001 VALUES LESS THAN (1250)
, partition p002 VALUES LESS THAN (1500)
, partition p003 VALUES LESS THAN (1750)
, partition p004 VALUES LESS THAN (2000)
, partition p005 VALUES LESS THAN (2250)
, partition p006 VALUES LESS THAN (2500)
, partition p007 VALUES LESS THAN (2750)
, partition p008 VALUES LESS THAN (3000)
);
```
Partitions maintenance through events

- An article by Greg Haase about this topic
- [http://dev.mysql.com/tech-resources/articles/partitioning-event_scheduler.html](http://dev.mysql.com/tech-resources/articles/partitioning-event_scheduler.html)
mysql_dump_partition_backup

- a script created by Roland Bouman
- detects the partitions for a table, and generates mysqldump statements for each one
Partitions maintenance through events
HOW TO

1. Gather information about partitions (from INFORMATION_SCHEMA)
2. dynamically create the partitions info
3. fire the event
the NULL partition

- when partitioning by date
- using TO_DAYS
- the first partition matters

<table>
<thead>
<tr>
<th>partition</th>
<th>rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>100,000</td>
</tr>
<tr>
<td>p2</td>
<td>150,000</td>
</tr>
<tr>
<td>p3</td>
<td>120,000</td>
</tr>
<tr>
<td>p4</td>
<td>100,000</td>
</tr>
<tr>
<td>partition</td>
<td>condition</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>p1</td>
<td>to_days('1990-01-01')</td>
</tr>
<tr>
<td>p2</td>
<td>to_days('1990-02-01')</td>
</tr>
<tr>
<td>p3</td>
<td>to_days('1990-03-01')</td>
</tr>
<tr>
<td>p4</td>
<td>to_days('1990-04-01')</td>
</tr>
<tr>
<td>p5</td>
<td>to_days('1990-05-01')</td>
</tr>
<tr>
<td>p6</td>
<td>to_days('1990-06-01')</td>
</tr>
</tbody>
</table>
# date_field between '1990-04-10' and '1990-04-15'

<table>
<thead>
<tr>
<th>partition</th>
<th>condition</th>
<th>rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>to_days('1990-01-01')</td>
<td>1,000,000</td>
</tr>
<tr>
<td>p2</td>
<td>to_days('1990-02-01')</td>
<td>1,000,000</td>
</tr>
<tr>
<td>p3</td>
<td>to_days('1990-03-01')</td>
<td>1,000,000</td>
</tr>
<tr>
<td>p4</td>
<td>to_days('1990-04-01')</td>
<td>1,000,000</td>
</tr>
<tr>
<td>p5</td>
<td>to_days('1990-05-01')</td>
<td>1,000,000</td>
</tr>
<tr>
<td>p6</td>
<td>to_days('1990-06-01')</td>
<td>1,000,000</td>
</tr>
</tbody>
</table>

= 2,000,000 rows
alter table t1
reorganize partition p1 into (  
  partition p0 values less than (0),  
  partition p1  
    values less than (to_days('1990-01-01'))) ;

<table>
<thead>
<tr>
<th>partition</th>
<th>condition</th>
<th>rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>p0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>p1</td>
<td>to_days('1990-01-01')</td>
<td>1,000,000</td>
</tr>
<tr>
<td>p2</td>
<td>to_days('1990-02-01')</td>
<td>1,000,000</td>
</tr>
<tr>
<td>p3</td>
<td>to_days('1990-03-01')</td>
<td>1,000,000</td>
</tr>
<tr>
<td>p4</td>
<td>to_days('1990-04-01')</td>
<td>1,000,000</td>
</tr>
<tr>
<td>p5</td>
<td>to_days('1990-05-01')</td>
<td>1,000,000</td>
</tr>
<tr>
<td>p6</td>
<td>to_days('1990-06-01')</td>
<td>1,000,000</td>
</tr>
</tbody>
</table>
date_field between '1990-04-10' and '1990-04-15'

<table>
<thead>
<tr>
<th>partition</th>
<th>condition</th>
<th>rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>p0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>p1</td>
<td>to_days('1990-01-01')</td>
<td>1,000,000</td>
</tr>
<tr>
<td>p2</td>
<td>to_days('1990-02-01')</td>
<td>1,000,000</td>
</tr>
<tr>
<td>p3</td>
<td>to_days('1990-03-01')</td>
<td>1,000,000</td>
</tr>
<tr>
<td>p4</td>
<td>to_days('1990-04-01')</td>
<td>1,000,000</td>
</tr>
<tr>
<td>p5</td>
<td>to_days('1990-05-01')</td>
<td>1,000,000</td>
</tr>
<tr>
<td>p6</td>
<td>to_days('1990-06-01')</td>
<td>1,000,000</td>
</tr>
</tbody>
</table>

= 1,000,000 rows
Lock before inserting single records

- If you send MANY **SINGLE** INSERT at once
- LOCK before inserting

- No need to do it for bulk inserts
Table of contents

Your needs
What
Partition pruning
Benchmarking
Partitioning by date
hands on
tools
tips
leveraging replication
Replication schemes

MASTER

INNODB
NOT
PARTITIONED

SLAVE

INNODB
NOT
PARTITIONED

concurrent insert

SLAVE

INNODB
PARTITIONED
BY RANGE

MyISAM
PARTITIONED
BY RANGE

SLAVE

concurrent read

concurrent batch processing

large batch processing

Monday, 13 September 2010
Replication schemes

MASTER

INNODB
PARTITIONED
BY HASH

concurrent insert

MyISAM
PARTITIONED
BY RANGE

batch processing

SLAVE

INNODB
NON
PARTITIONED

concurrent reads

SLAVE
Replication schemes - dimensions

- **MASTER**
  - INNODB
  - PARTITIONED BY HASH
  - concurrent insert

- **SLAVE**
  - ARCHIVE
  - PARTITIONED BY RANGE (date)
  - dimensional processing

- **SLAVE**
  - ARCHIVE
  - PARTITIONED BY RANGE (product)
  - dimensional processing

- **SLAVE**
  - ARCHIVE
  - PARTITIONED BY RANGE (locations)
  - dimensional processing
The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.
Slides for this presentation

http://tinyurl.com/mysql-partitions

This work is licensed under the Creative Commons Attribution-Share Alike 3.0 Unported License.
THANKS!

Q&A

Comment on Twitter: @datacharmer
MySQL 5.5 enhancements

• PARTITION BY RANGE COLUMNS
• PARTITION BY LIST COLUMNS
• TO_SECONDS
MySQL 5.5 enhancements

CREATE TABLE t (  
  dt date  
)  
PARTITION BY RANGE (TO_DAYS(dt))  
(
  PARTITION p01 VALUES LESS THAN (TO_DAYS('2007-01-01')),
  PARTITION p02 VALUES LESS THAN (TO_DAYS('2008-01-01')),
  PARTITION p03 VALUES LESS THAN (TO_DAYS('2009-01-01')),
  PARTITION p04 VALUES LESS THAN (MAXVALUE));
SHOW CREATE TABLE `t` 

Table: `t`

Create Table: CREATE TABLE `t` ( 
  `dt` date DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1

/*!50100 PARTITION BY RANGE (TO_DAYS (dt))
(PARTITION p01 VALUES LESS THAN (733042) ENGINE = MyISAM,
[...]
MySQL 5.5 enhancements

CREATE TABLE t (  
  dt date  
)  
PARTITION BY RANGE COLUMNS (dt)  
(  
  PARTITION p01 VALUES LESS THAN ('2007-01-01'),  
  PARTITION p02 VALUES LESS THAN ('2008-01-01'),  
  PARTITION p03 VALUES LESS THAN ('2009-01-01'),  
  PARTITION p04 VALUES LESS THAN (MAXVALUE))

Monday, 13 September 2010
SHOW CREATE TABLE t

Table: t

Create Table: CREATE TABLE `t` (  
`dt` date DEFAULT NULL  
)

ENGINE=MyISAM DEFAULT CHARSET=latin1

/*!50500 PARTITION BY RANGE COLUMNS
(dt)
(PARTITION p01 VALUES LESS THAN
(  '2007-01-01'  ) ENGINE = MyISAM,
[...]

MySQL 5.5 enhancements

Monday, 13 September 2010
MySQL 5.5 - Multiple columns

CREATE TABLE t (  
a int,
b int
) PARTITION BY RANGE COLUMNS (a,b) (
    PARTITION p01 VALUES LESS THAN (10,1),
    PARTITION p02 VALUES LESS THAN (10,10),
    PARTITION p03 VALUES LESS THAN (10,20),
    PARTITION p04 VALUES LESS THAN (MAXVALUE, MAXVALUE)) ;
### Partition Definition

#### By Range \((a, b)\)

<table>
<thead>
<tr>
<th>Records</th>
<th>(a)</th>
<th>(b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>p01</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>p02</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>p03</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>p04</td>
<td>10</td>
<td>MAXVALUE</td>
</tr>
<tr>
<td>p05</td>
<td>MAXVALUE</td>
<td>MAXVALUE</td>
</tr>
</tbody>
</table>

For each partition, \(a\) and \(b\) define the range of values for which records are partitioned.

- **p01**: 1 to 10
- **p02**: 10 to 20
- **p03**: 10 to 30
- **p04**: 10 to MAXVALUE
- **p05**: MAXVALUE to MAXVALUE
### Partition Definition

**By Range (a, b)**

<table>
<thead>
<tr>
<th>partition</th>
<th>LESS THAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>p01</td>
<td>10</td>
</tr>
<tr>
<td>p02</td>
<td>10</td>
</tr>
<tr>
<td>p03</td>
<td>10</td>
</tr>
<tr>
<td>p04</td>
<td>10</td>
</tr>
<tr>
<td>p05</td>
<td>MAXVALUE</td>
</tr>
</tbody>
</table>

#### Records

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
</tr>
</tbody>
</table>

#### Query

\[(1, 10) < (10, 10) \?
\]

\[
(a < 10) \\
\text{OR} \\
((a = 10) \text{ AND } (b < 10))
\]

\[
(1 < 10) \\
\text{OR} \\
((1 = 10) \text{ AND } (10 < 10))
\]

**TRUE**
### Partition Definition
by range \((a, b)\)

<table>
<thead>
<tr>
<th>partition</th>
<th>LESS THAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>p01</td>
<td>10</td>
</tr>
<tr>
<td>p02</td>
<td>10</td>
</tr>
<tr>
<td>p03</td>
<td>10</td>
</tr>
<tr>
<td>p04</td>
<td>10</td>
</tr>
<tr>
<td>p05</td>
<td>MAXVALUE</td>
</tr>
</tbody>
</table>

#### Records

<table>
<thead>
<tr>
<th>(a)</th>
<th>(b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
</tr>
</tbody>
</table>

\((10, 9) < (10, 10)\) ?

\[(a < 10)\]
\[\text{OR}\]
\[\((a = 10) \text{ AND } (b < 10)\)\]

\((10 < 10)\]
\[\text{OR}\]
\[\((10 = 10) \text{ AND } (9 < 10)\)\]

\[\text{TRUE}\]
(10,10) < (10,10) ?

(a < 10)  
OR  
((a = 10) AND (b < 10))

(10 < 10)  
OR  
((10 = 10) AND (10 < 10))

FALSE
partition definition

by range \((a, b)\)

<table>
<thead>
<tr>
<th>partition</th>
<th>LESS THAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>p01</td>
<td>10</td>
</tr>
<tr>
<td>p02</td>
<td>10</td>
</tr>
<tr>
<td>p03</td>
<td>10</td>
</tr>
<tr>
<td>p04</td>
<td>10</td>
</tr>
<tr>
<td>p05</td>
<td>MAXVALUE</td>
</tr>
</tbody>
</table>

\((10, 10) < (10, 20)\) ?

\((a < 10)\)

OR

\(((a = 10) \text{ AND } (b < 20))\)

\((10 < 10)\)

OR

\(((10 = 10) \text{ AND } (10 < 20))\)

TRUE

\begin{table}[h]
\centering
\begin{tabular}{|c|c|}
\hline
\textbf{records} & \textbf{a} & \textbf{b} \\
\hline
1 & 10 & \\
10 & 9 & \\
10 & 10 & \\
10 & 11 & \\
\hline
\end{tabular}
\end{table}
CREATE TABLE employees (  
    emp_no int(11) NOT NULL,  
    birth_date date NOT NULL,  
    first_name varchar(14) NOT NULL,  
    last_name varchar(16) NOT NULL,  
    gender char(1) DEFAULT NULL,  
    hire_date date NOT NULL  
 ) ENGINE=MyISAM  
PARTITION BY RANGE  
    COLUMNS(gender,hire_date)  
(PARTITION p01 VALUES LESS THAN ('F','1990-01-01'),  
 PARTITION p02 VALUES LESS THAN ('F','2000-01-01'),  
 PARTITION p03 VALUES LESS THAN ('F',MAXVALUE),  
 PARTITION p04 VALUES LESS THAN ('M','1990-01-01'),  
 PARTITION p05 VALUES LESS THAN ('M','2000-01-01'),  
 PARTITION p06 VALUES LESS THAN ('M',MAXVALUE),  
 PARTITION p07 VALUES LESS THAN (MAXVALUE,MAXVALUE)
MySQL 5.5 enhancements

- TRUNCATE PARTITION
- TO_SECONDS()
MySQL 5.6 enhancements

• ALTER TABLE … EXCHANGE PARTITION
ALTER TABLE t1
EXCHANGE PARTITION p2
WITH TABLE t2
<table>
<thead>
<tr>
<th></th>
<th>t1</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>f1</td>
<td>f2</td>
<td>f3</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>a</td>
<td>xxx</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>b</td>
<td>xxx</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>c</td>
<td>yyy</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>d</td>
<td>xxx</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>e</td>
<td>zzz</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>f</td>
<td>www</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>g</td>
<td>ccc</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>h</td>
<td>sss</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>i</td>
<td>www</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>j</td>
<td>xxx</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>k</td>
<td>yyy</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>l</td>
<td>zzz</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>t2</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>f1</td>
<td>f2</td>
<td>f3</td>
<td></td>
</tr>
<tr>
<td>(empty)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>t1</td>
<td></td>
<td>t2</td>
</tr>
<tr>
<td>---</td>
<td>----</td>
<td>---</td>
<td>----</td>
</tr>
<tr>
<td>f1</td>
<td>f2</td>
<td>f3</td>
<td>f1</td>
</tr>
<tr>
<td>1</td>
<td>a</td>
<td>xxx</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>b</td>
<td>xxx</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>c</td>
<td>yyy</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>d</td>
<td>xxx</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>e</td>
<td>zzz</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>f</td>
<td>www</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>g</td>
<td>ccc</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>h</td>
<td>sss</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>i</td>
<td>www</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>j</td>
<td>xxx</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>k</td>
<td>yyy</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>l</td>
<td>zzz</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(empty)</td>
</tr>
<tr>
<td></td>
<td>t1</td>
<td>t2</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>--------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>f1</td>
<td>f2</td>
<td>f3</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>a</td>
<td>xxx</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>b</td>
<td>xxx</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>c</td>
<td>yyy</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>d</td>
<td>xxx</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>e</td>
<td>zzz</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>f</td>
<td>www</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>g</td>
<td>ccc</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>h</td>
<td>sss</td>
<td></td>
</tr>
<tr>
<td>p1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>