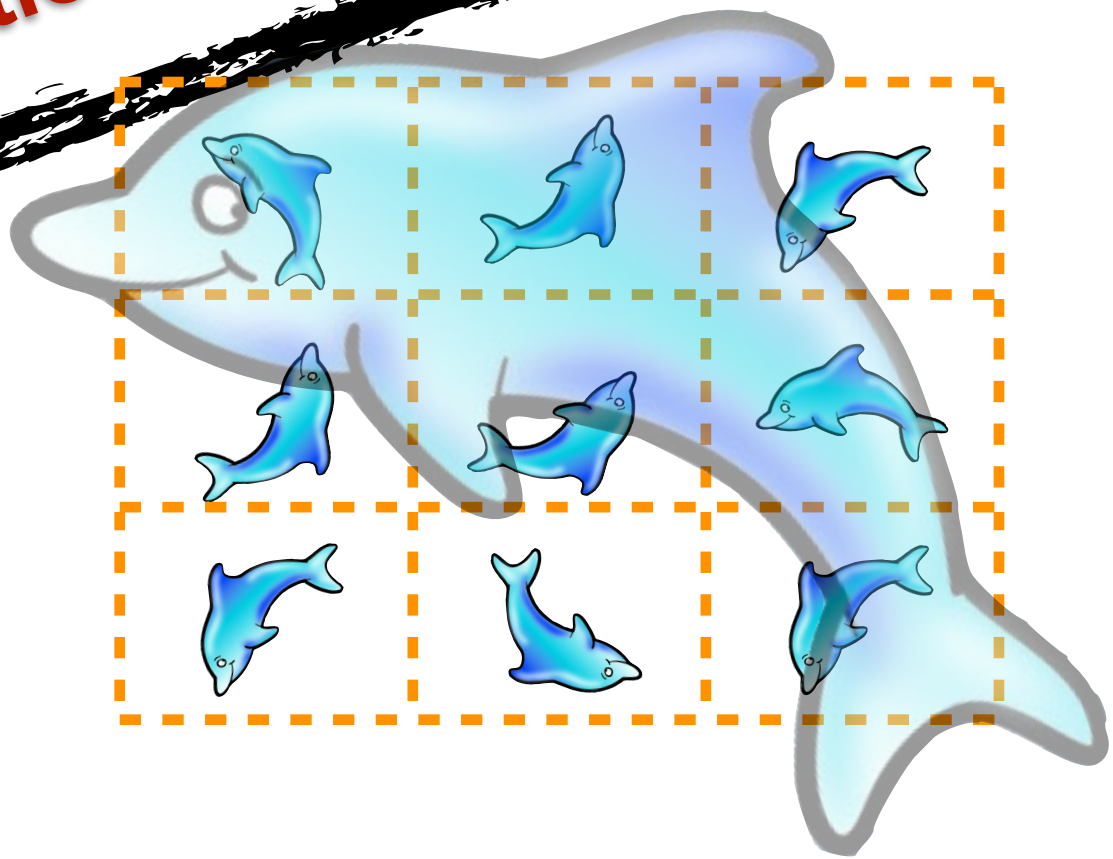
Boost performance with MySQL 5.1 and 5.5 partitioning



Giuseppe Maxia
MySQL Community Team Lead
Sun Microsystems

Who's this guy?

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

MySQL 5.1 GA

MySQL 5.1 GA


MySQL
Sun
Brian Aker
Marc Alff
Davi Arnaut
Kaj Arno
David Axmark
Igor Babaev
Patrik Backman
Alexander Barkov
Omer BarNir
Anthony Bedford
Chuck Bell
Shane Bester
Guilhem Bichot
Kent Boortz
Alexey Botchkov
Tomash Brechko
Martin Brown
Jörg Brühe
Reggie Burnett
Sanja Byelkin
Michelle Caisse
Petr Chardin
Colin Charles
Patrick Crews
Antony Curtis
James Day
Adam Donnison
Liz Drachnik
Paul DuBois
Susanne Ebrecht
Andrei Elkin
Dean Ellis
Daniel Fischer
Ignacio Galarza
Patrick Galbraith
Sergey Glukhov
Sergei Golubchik
Ian Greenhoe
Tonci Grgin
Lenz Grimmer
Peter Gulutzan
Christoffer Hall
Martin Hansson
Justin He
Lars Heill
Mike Hillyer
Stefan Hinz
Rayson Ho
Andrey Hristov
Horst Hunger
Alexi Ivanov
Mattias Jonsson
Mads Martin Jørgensen
Ramil Kalimullin
Timour Katchaounov
Damien Katz
Kolbe Kegel
Alexander Kermidarski
Mats Kindahl
Georgi Kodinov
Alexey Kopytov
Serge Kozlov
Valerly Kravchuk
Hakan Küçükylmaz
Peter Lavin
Matthias Leich
Mark Leith
Dmitri Lenev
Giuseppe Maxia
Mårten Mickos
Brian Miezejewski
Sinisa Milivojevic
Jeb Miller
Chad Miller
Domas Mituzas
Bjørn Munch
Elliot Murphy
V Narayanan
Guangbao Ni
Kristian Nielsen
Pekka Nousiainen
Alexander Nozdrin
Tatjana Nurnberg
Konstantin Osipov
Vemund Østgaard
Allan Packer
Trudy Pelzer
Jonathan Perkin
Trimbakeshwar Pershad
Sergey Petrunia
Kristofer Pettersson
Jay Pipes
Evgeny Potemkin
Jeffrey Pugh
Georg Richter
Mikael Ronström
Ulf Sanberg
Sven Sandberg
Robin Schumacher
Gleb Shchepa
Nidhi Shrotriya
Martin Sköld
Sveta Smirnova
Stewart Smith
Tim Smith
Miguel Solorzano
Rafal Somla
Hema Sridharan
Elena Stepanova
Jon Stephens
Philip Stoev
Alexey Stroganov
Ingo Strüwing
Calvin Sun
Ygnve Svendsen
Magnus Svensson
Lars Thalmann
Jani Tolonen
Tomas Ulin
Zack Urlocker
Vladislav Vaintroub
Sergey Vojtovich
Monty Widenius
Jim Winstead
He Zhenxing
LI Zhou
Michael Zinner
Plus many more from Support, Community, Marketing
and the wider MySQL community

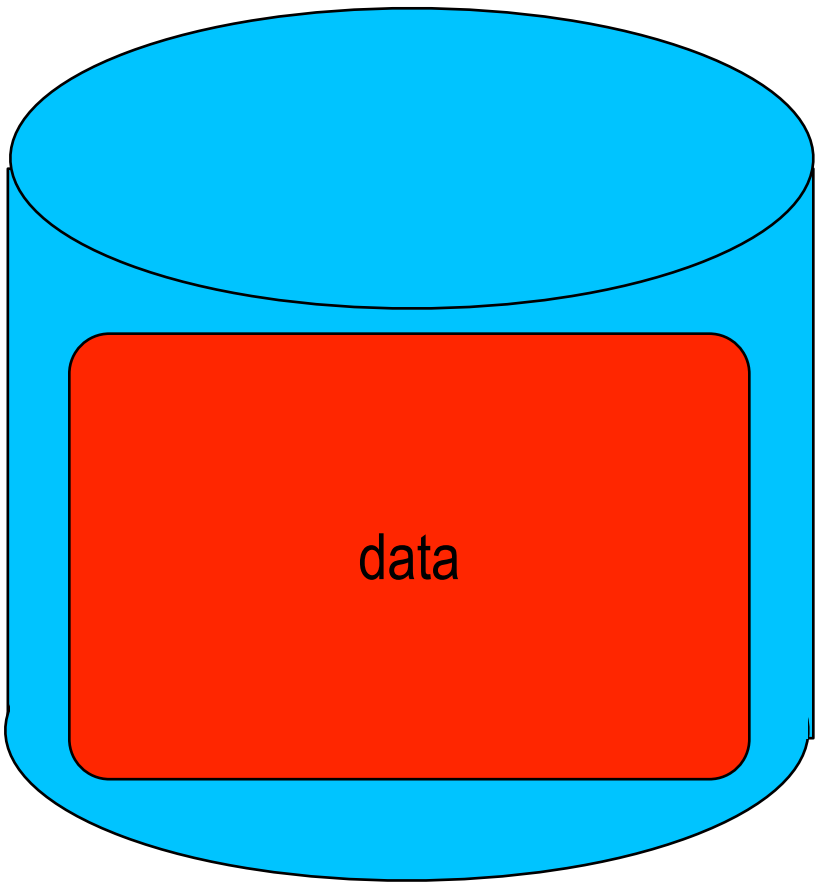
Defining the problem

YOUR NEEDS

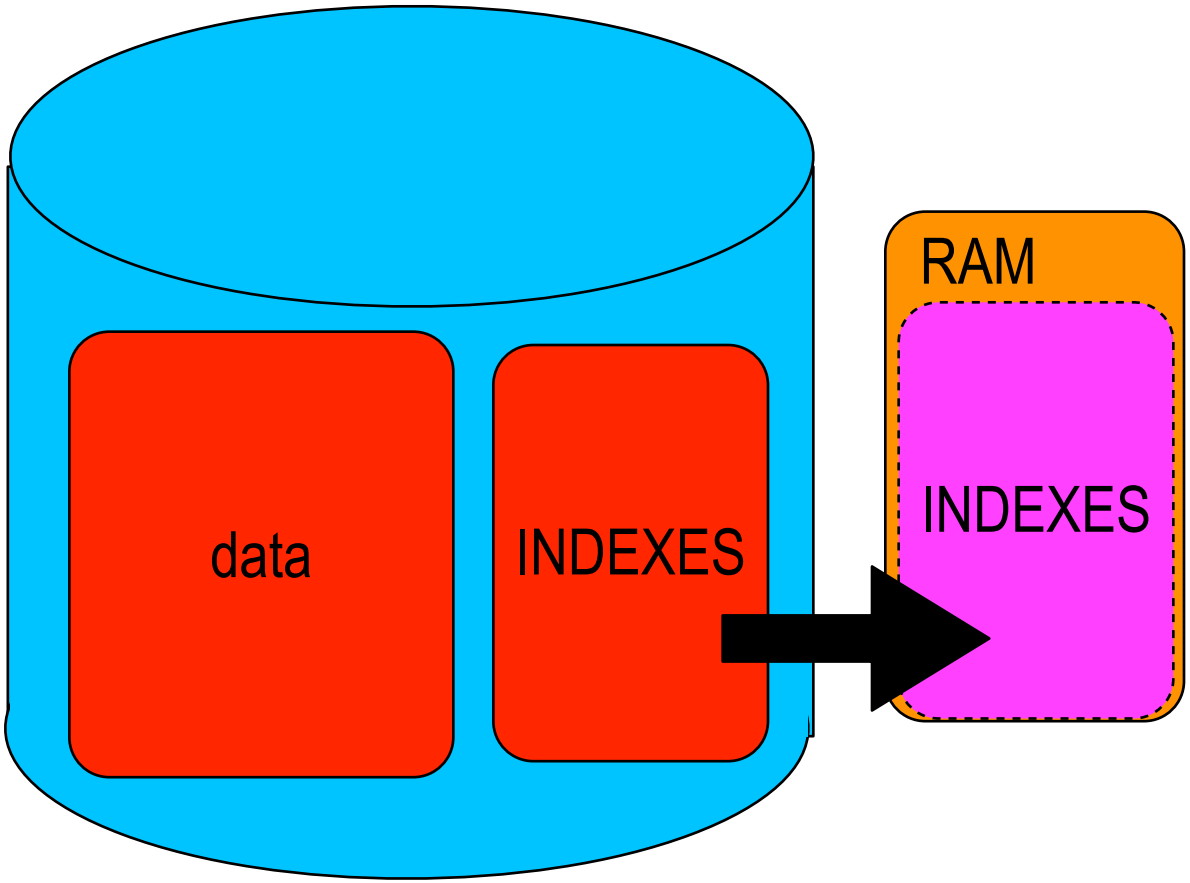
The problem(s)

- Too much data
- Not enough RAM
- Historical data
- Growing data
- Rotating data

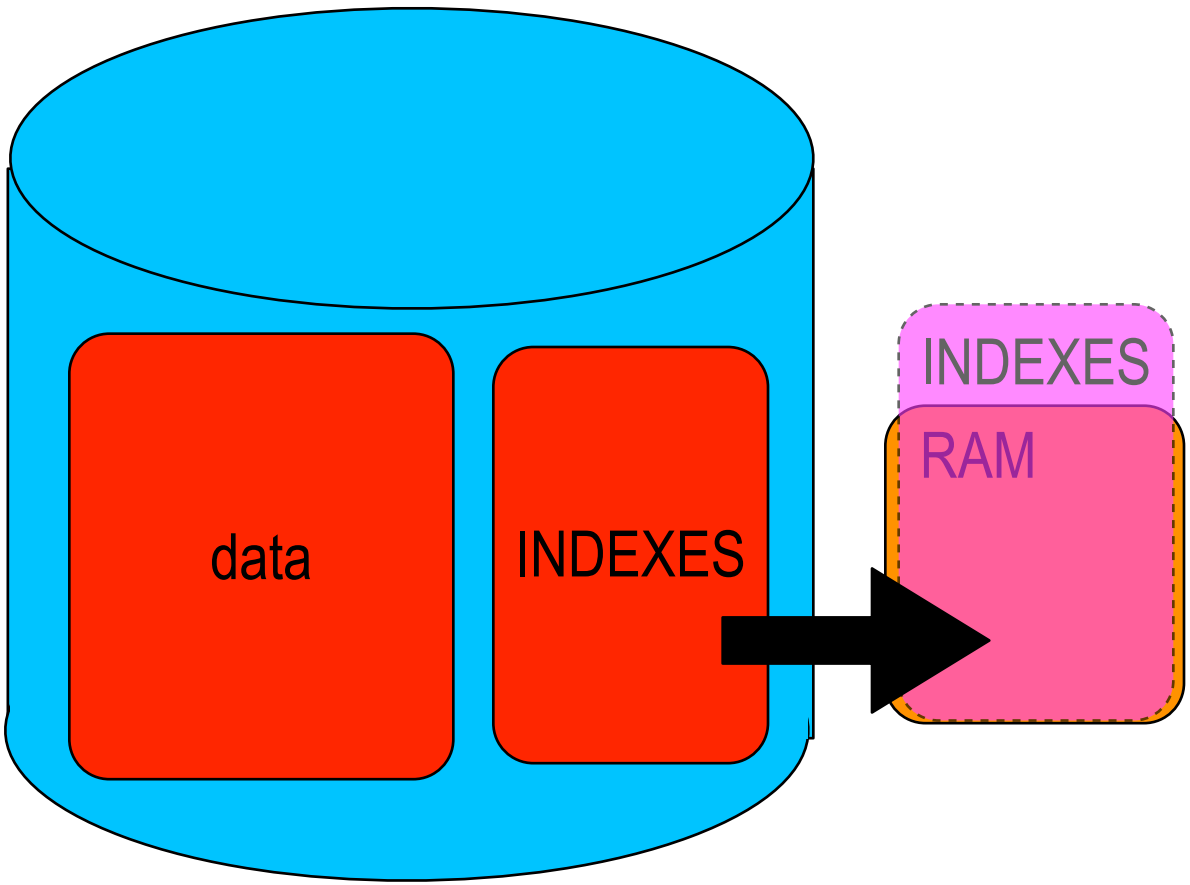
Too much data



Not enough RAM



Not enough RAM



MySQL 5.1 partitions

WHAT

What is it?

- Logical splitting of tables
- Transparent to user

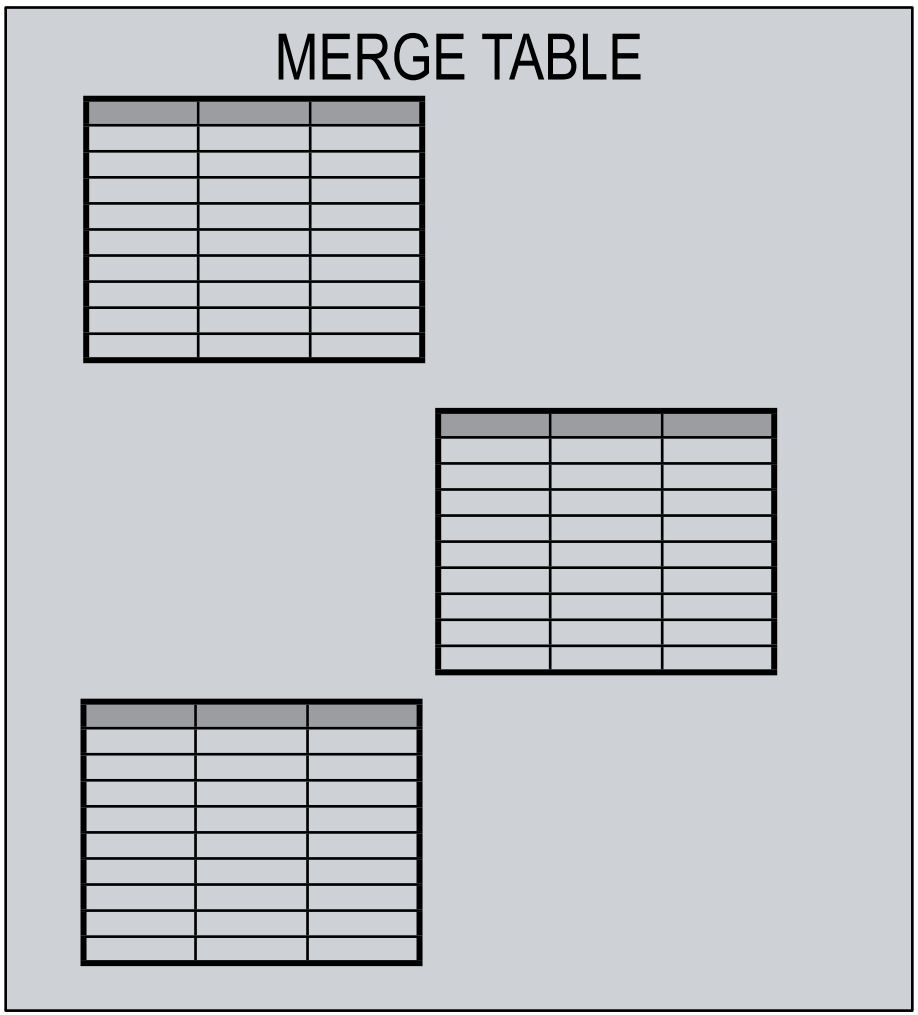
Logical splitting

- No need to create separate tables
- No need to move chunks of data across files

MySQL 5.1 partitions

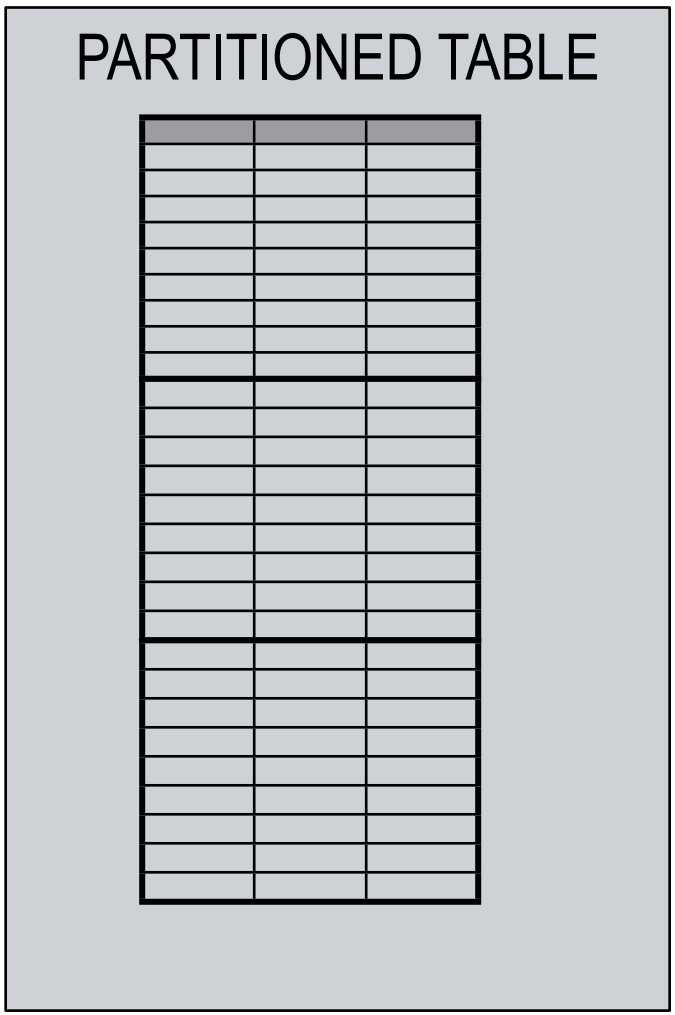
TRANSPARENT SQL

COMPARED TO MERGE TABLES



- separate tables
- risk of duplicates
- insert in each table
- no constraints

COMPARED TO MERGE TABLES



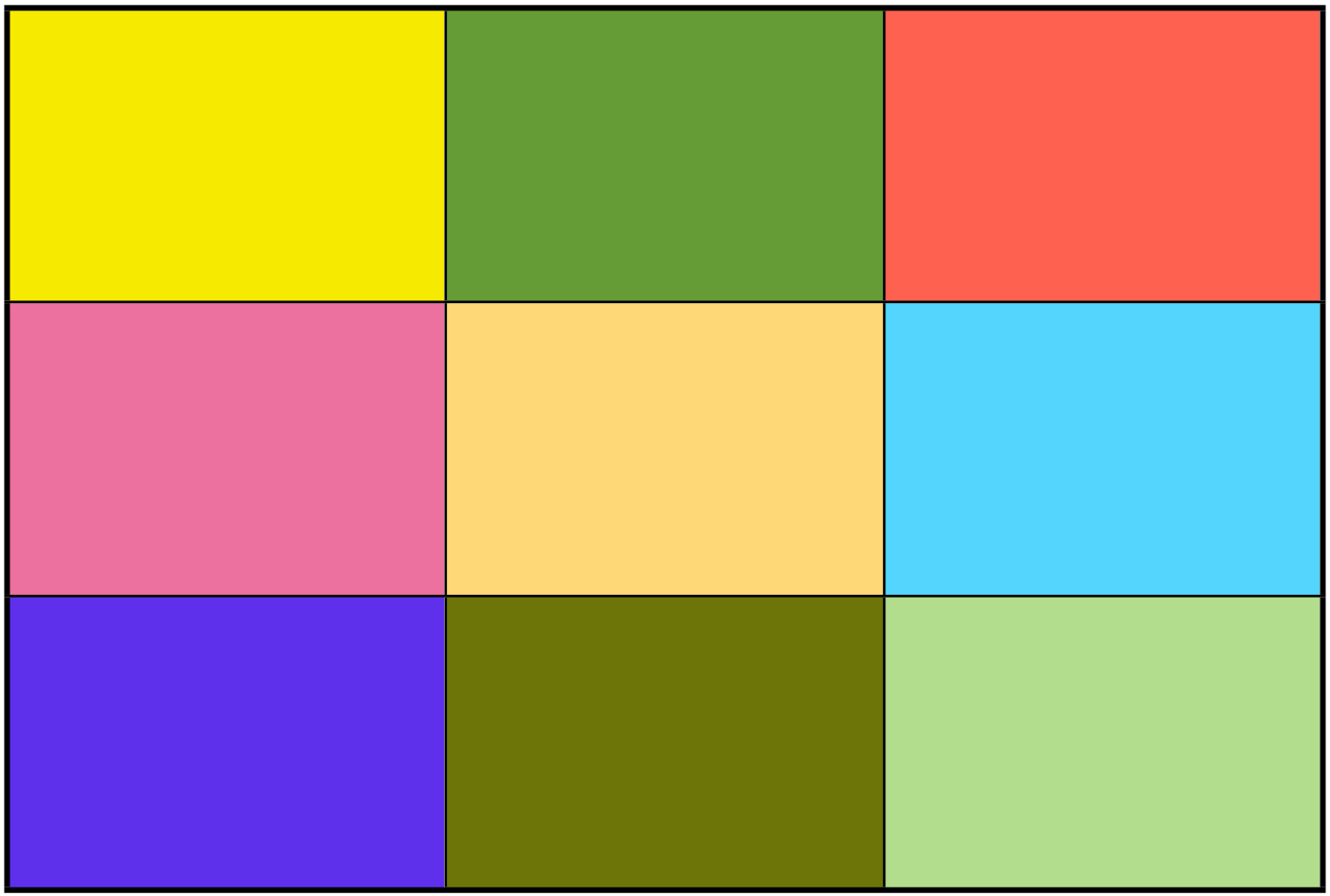
- One table
- No risk of duplicates
- insert in one table
- constraints enforced

Wait a minute ...

- WHAT THE HELL DOES "**LOGICAL SPLIT**" REALLY MEANS?
- LET ME EXPLAIN ...

Physical partitioning (1)

- Take a map



Physical partitioning (2)

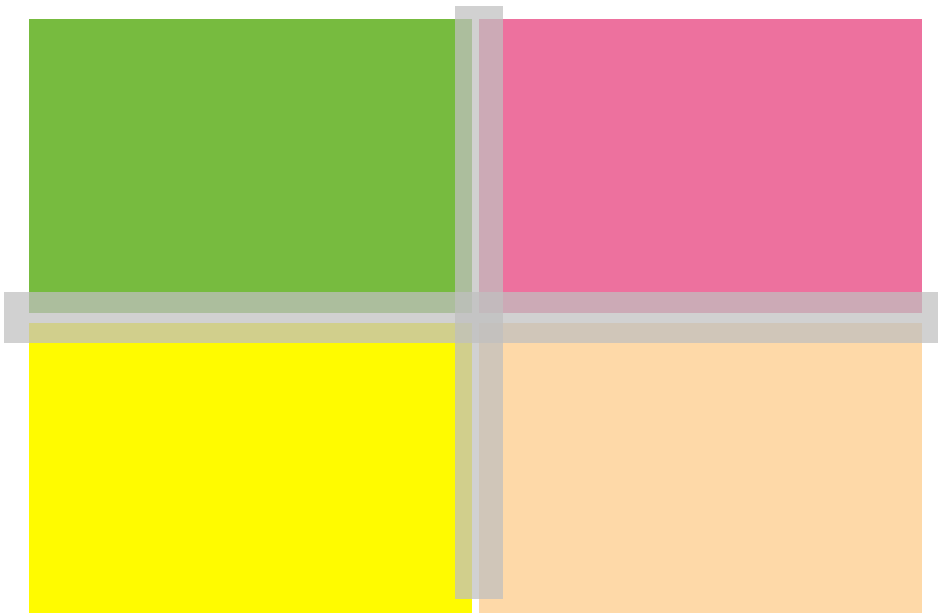
- cut it into pieces

Physical partitioning (3)

- What you have, is several different pieces

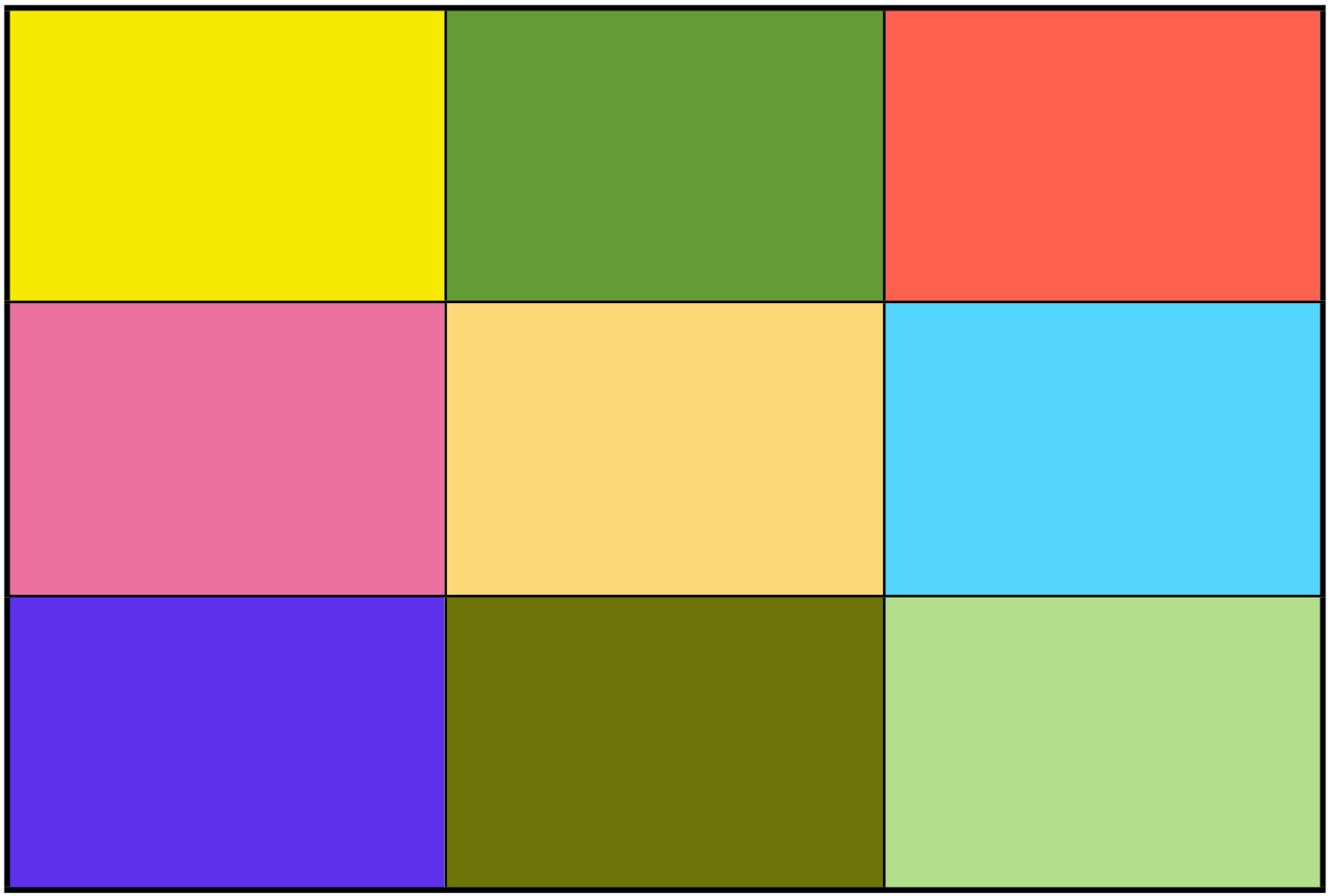
Physical partitioning (4)

- If you want the map back, you need some application (adhesive tape) and you may get it wrong



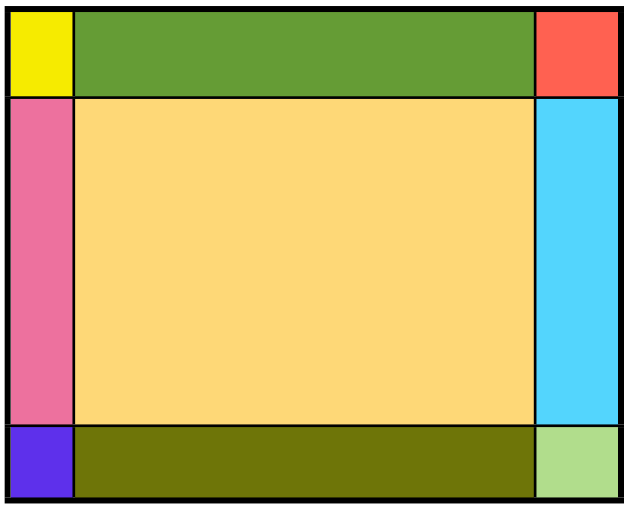
Logical partitioning (1)

- Take a map



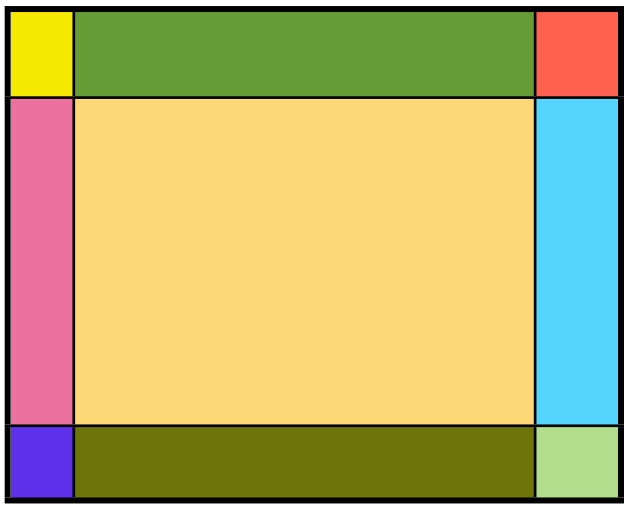
Logical partitioning (2)

- fold it to show the piece you need



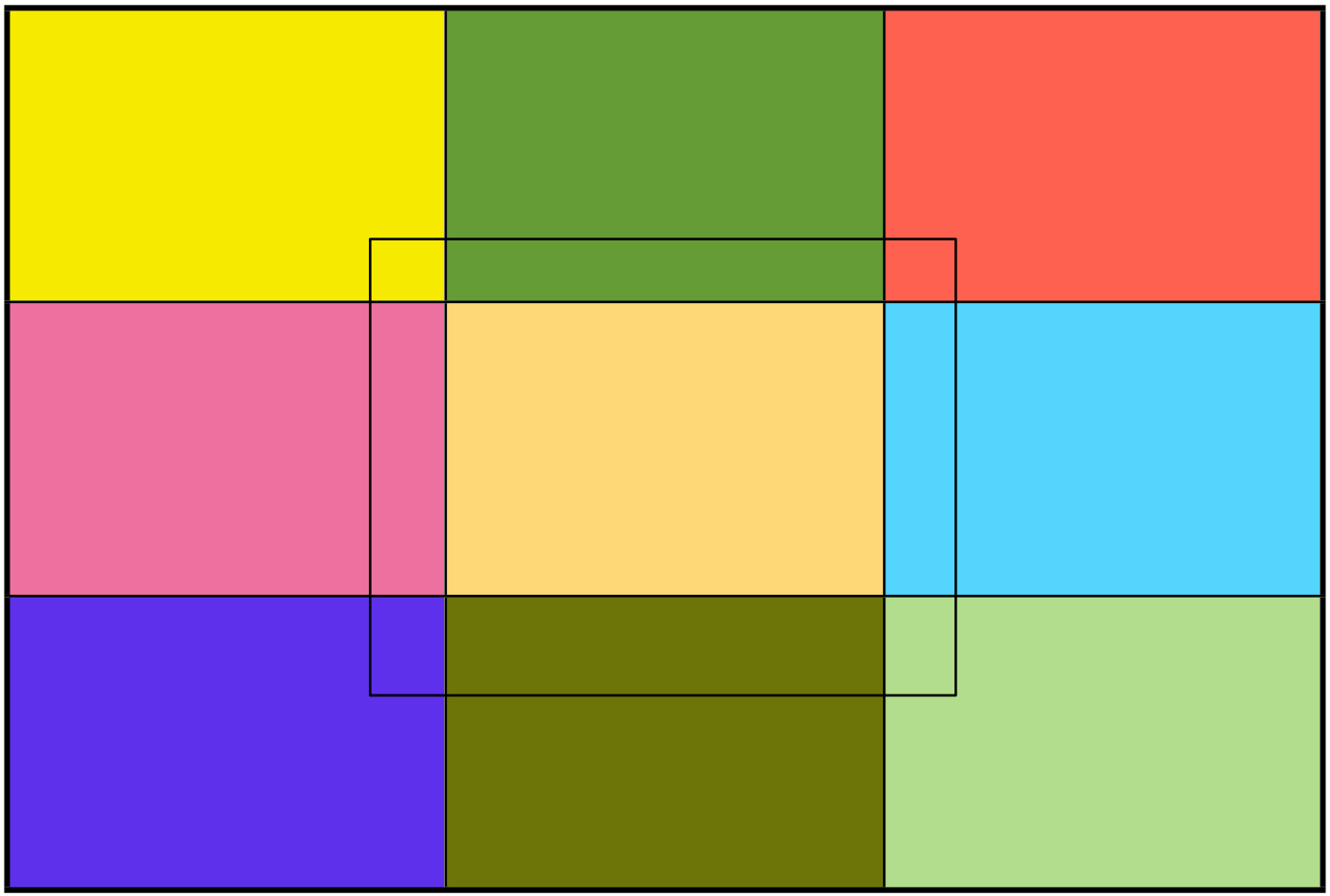
Logical partitioning (3)

- what you have is still a map, even if you see only one part.



Logical partitioning (4)

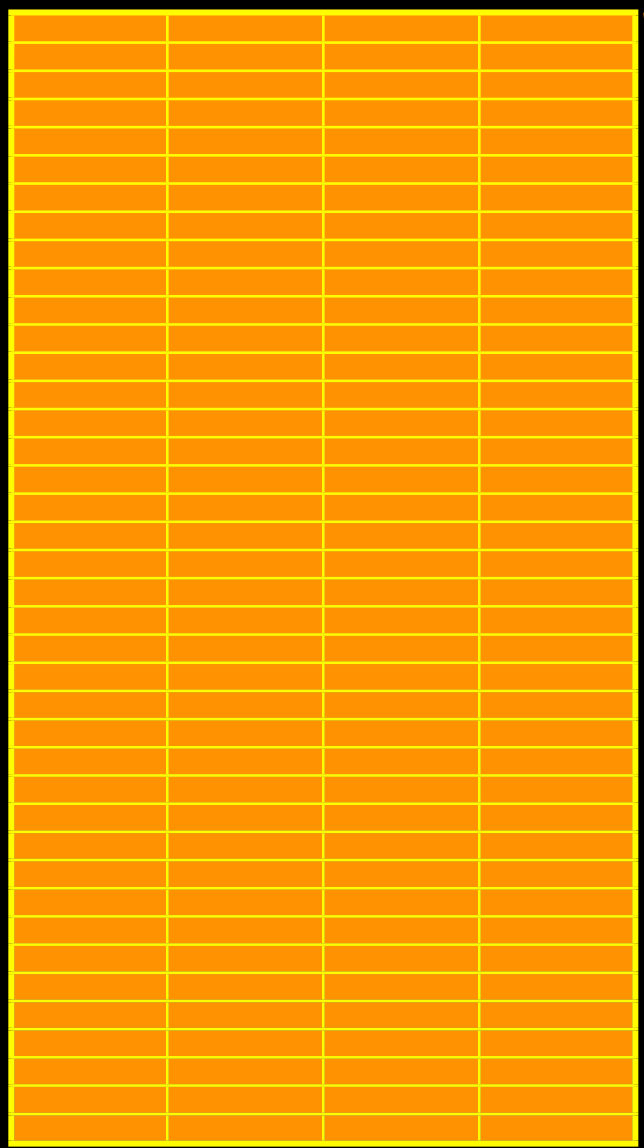
- if you unfold the map, you still have the whole thing



What partitions can do

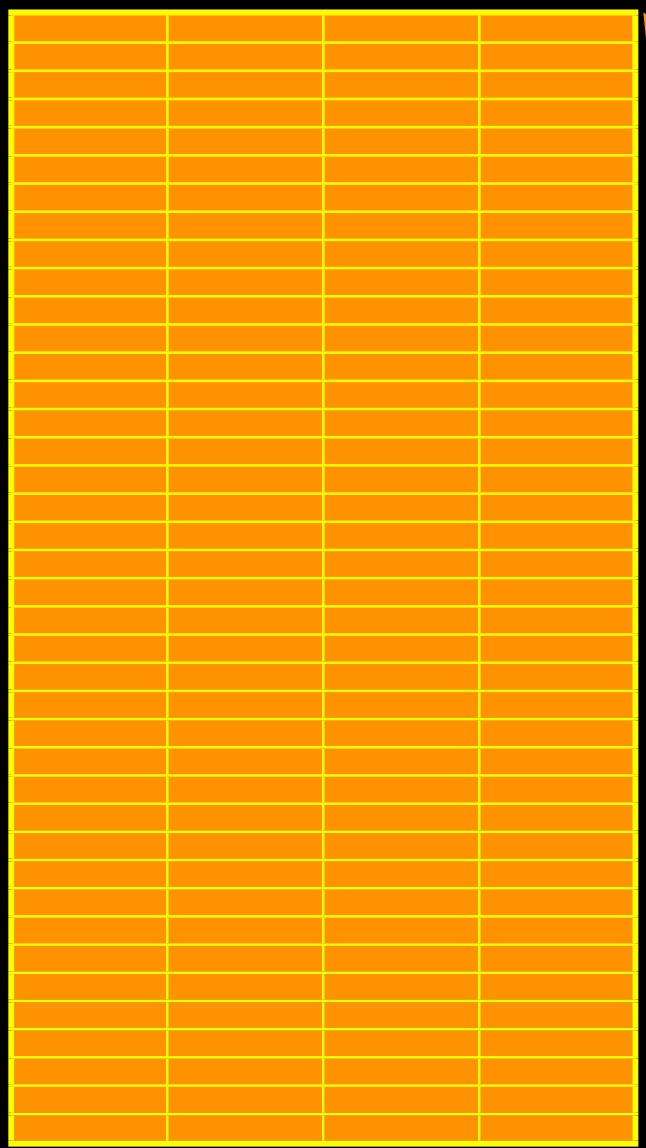
- logical split
- but you can also split the data physically
- granular partition (subpartitioning)
- different methods (range, list, hash, key)

Partition pruning
1a - unpartitioned table - SINGLE RECORD



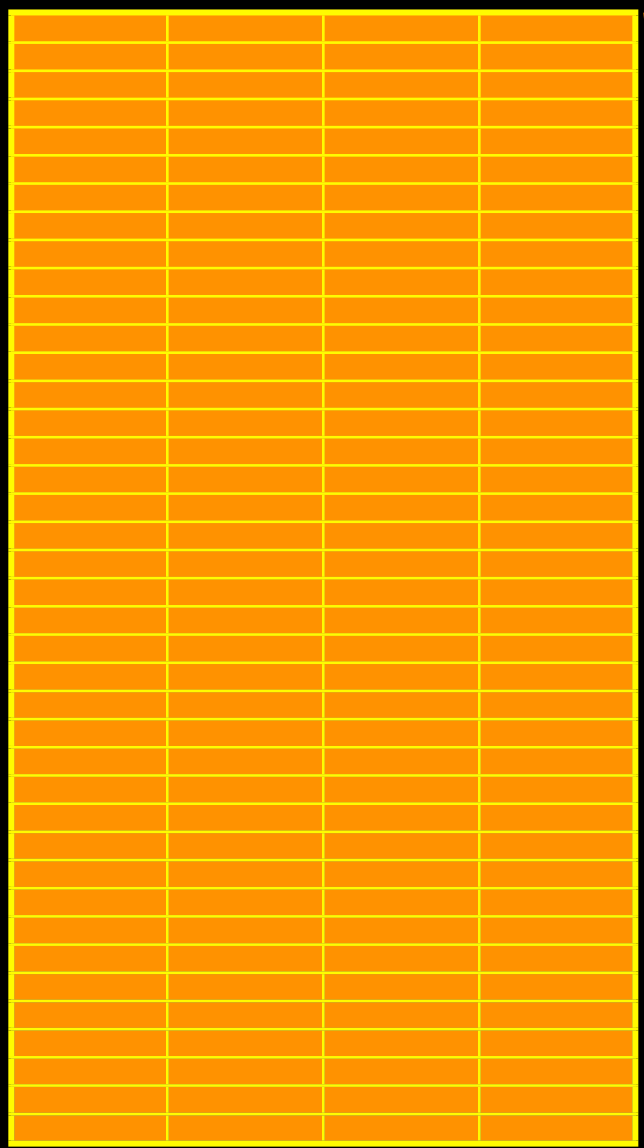
```
select *
from
table_name
where colx =
120
```

Partition pruning
1b - unpartitioned table - SINGLE RECORD



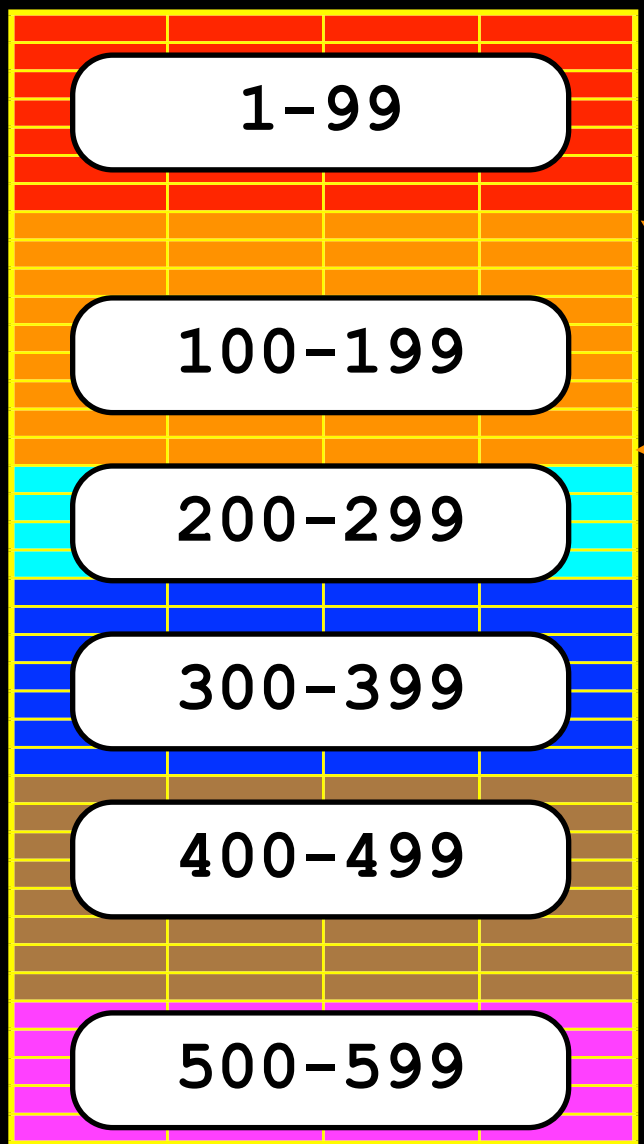
```
select *
from
table_name
where colx =
350
```

Partition pruning
1c - unpartitioned table - RANGE



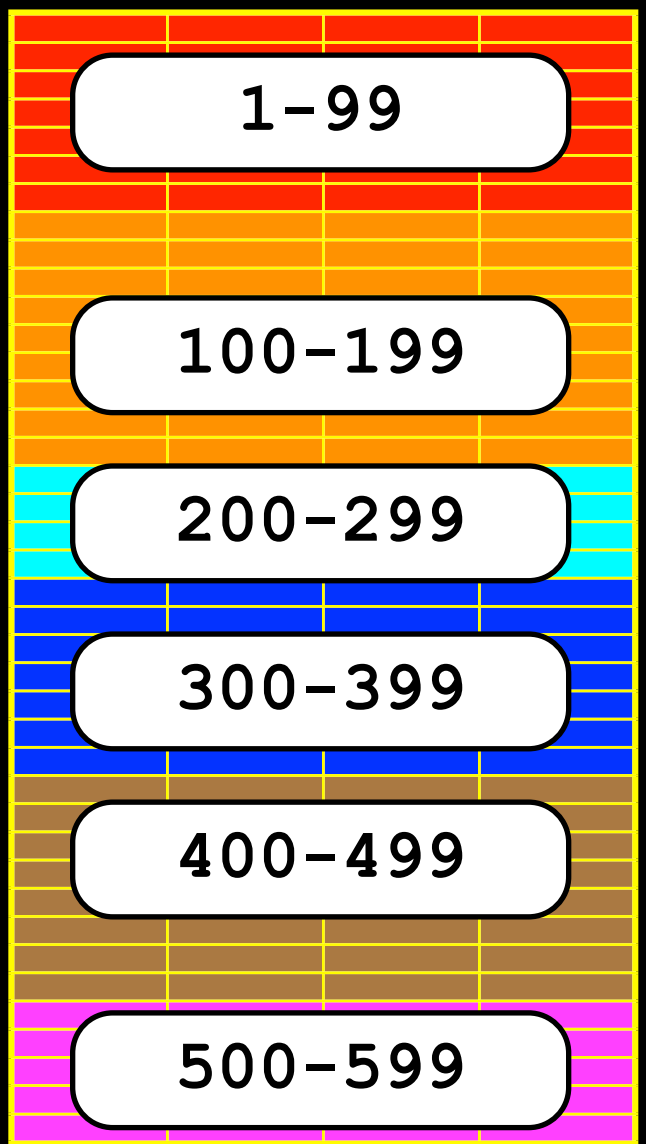
```
select *
from
table_name
where colx
between 120
and 230
```

Partition pruning
2a - table partitioned by colx - SINGLE REC



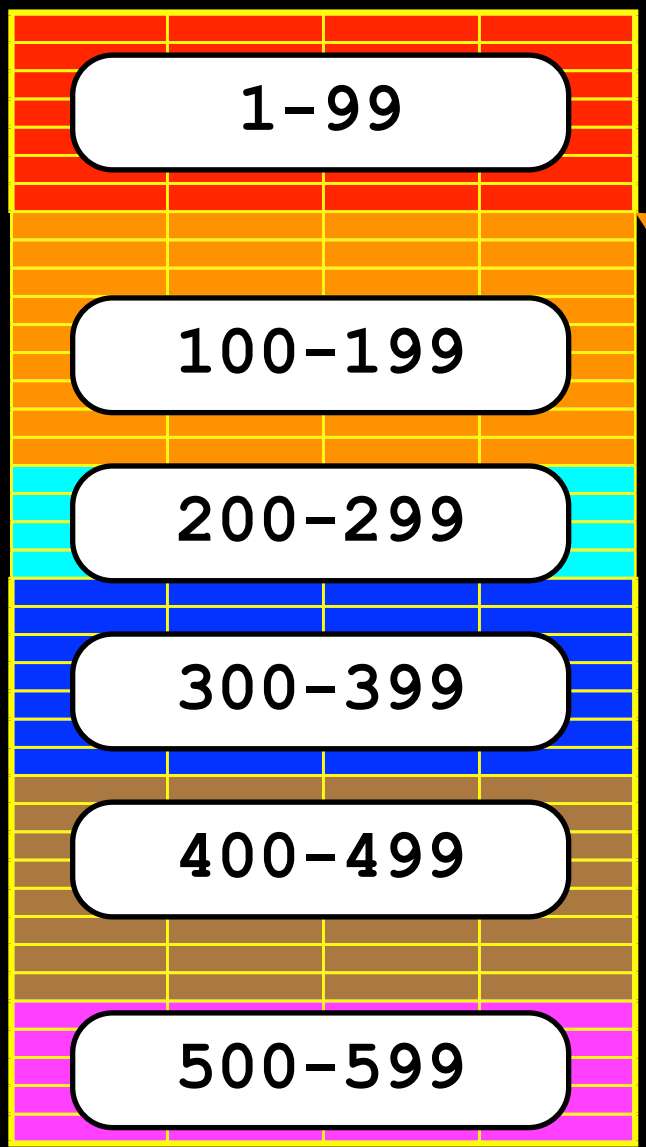

```
select *
from
table_name
where colx =
120
```

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
```

MySQL 5.1 partitions

WHY

4 main reasons to use partitions

- To make single inserts and selects faster
- To make range selects faster
- To help split the data across different paths
- To store historical data efficiently
- If you need to delete large chunks of data INSTANTLY

MySQL 5.1 partitions

HOW

HOW TO MAKE PARTITIONS

- **RTFM ...**
  - No, seriously, the manual has everything
  - But if you absolutely insist ...

HOW TO MAKE PARTITIONS

```
CREATE TABLE t1 (
   id int
) ENGINE=InnoDB
   # or MyISAM, ARCHIVE
PARTITION BY RANGE (id)
(
 PARTITION P1 VALUES LESS THAN (10),
 PARTITION P2 VALUES LESS THAN (20)
)
```

HOW TO MAKE PARTITIONS

```
CREATE TABLE t1 (
   id int
) ENGINE=InnoDB
PARTITION BY LIST (id)
(
 PARTITION P1 VALUES IN (1,2,4),
 PARTITION P2 VALUES IN (3,5,9)
)
```

HOW TO MAKE PARTITIONS

```
CREATE TABLE t1 (
   id int not null primary key
) ENGINE=InnoDB
PARTITION BY HASH (id)
PARTITIONS 10;
```

HOW TO MAKE PARTITIONS

```
CREATE TABLE t1 (
   id int not null primary key
) ENGINE=InnoDB
PARTITION BY KEY ()
PARTITIONS 10;
```

Limitations

- Can partition only by INTEGER columns
- OR you can partition by an expression, which must return an integer
- Maximum 1024 partitions

- If you have a Unique Key or PK, the partition column **must** be part of that key
- No Foreign Key support
- No Fulltext or GIS support

PARTITIONING BY DATE

```
CREATE TABLE t1 (
 d date
) ENGINE=InnoDB
PARTITION BY RANGE (year(d))
(
 PARTITION P1 VALUES
    LESS THAN (1999),
 PARTITION P2 VALUES
   LESS THAN (2000)
)
```

PARTITIONING BY DATE

```
CREATE TABLE t1 (
 d date
) ENGINE=InnoDB
PARTITION BY RANGE (to_days(d))
(
 PARTITION P1 VALUES
  LESS THAN (to_days('1999-01-01')),
 PARTITION P2 VALUES
 LESS THAN (to_days('2000-01-01'))
)
```

MySQL 5.1 partitions

WHEN

When to use partitions?

- if you have large tables
- if you know that you will always query for the partitioning column
- if you have historical tables that you want to purge quickly
- if your indexes are larger than the available RAM

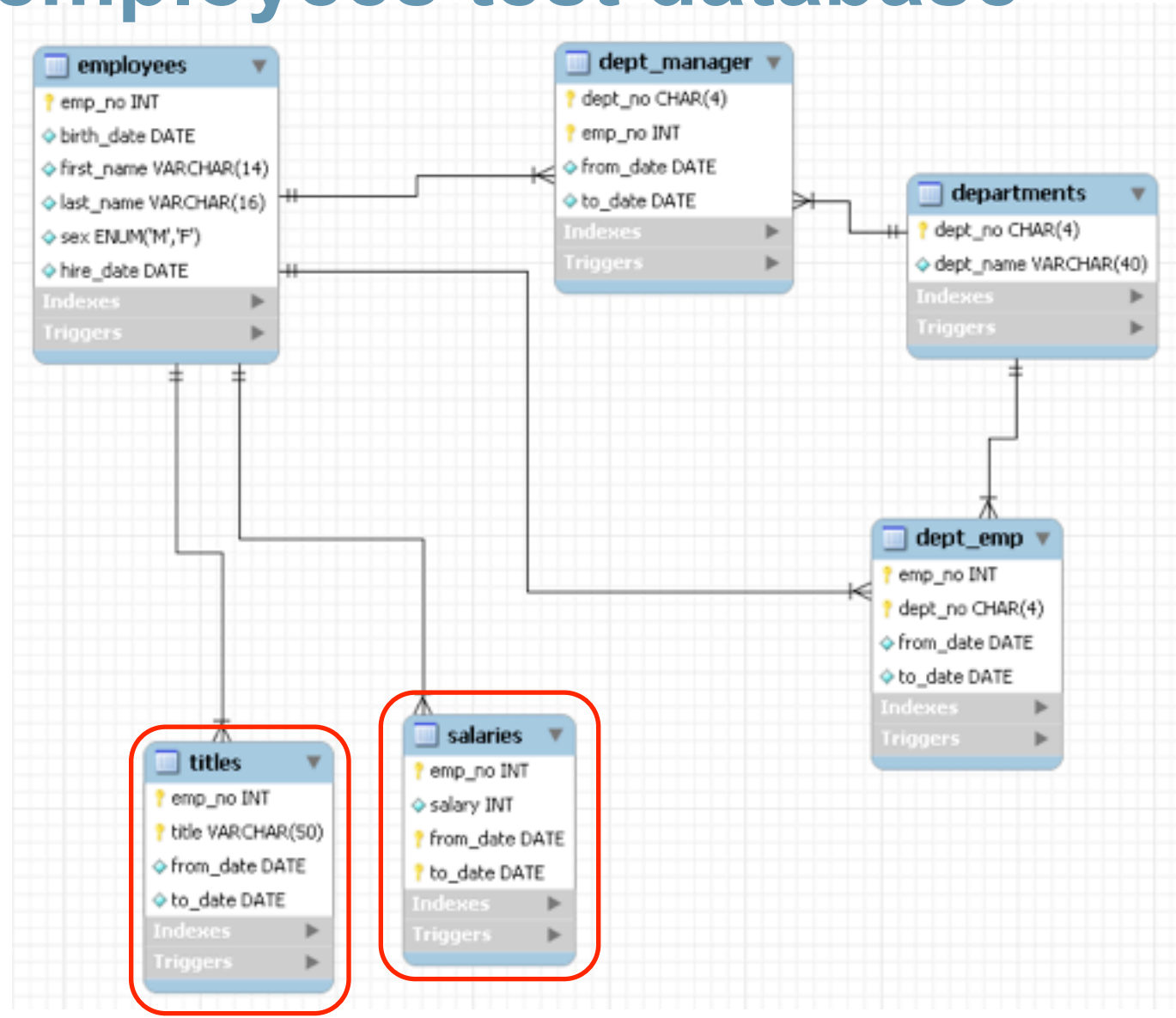
MySQL 5.1 partitions

HANDS ON

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

```
from information_schema.partitions
+-------+-------------------+----------+
| pname | expr              | descr    |
+-------+-------------------+----------+
| p01   | year(from_date)   | 1985     |
| p02   | year(from_date)   | 1986     |

...
| p13   | year(from_date)   | 1997     |
| p14   | year(from_date)   | 1998     |
| p15   | year(from_date)   | 1999     |
| p16   | year(from_date)   | 2000     |
  ...
| p19   | year(from_date)   | MAXVALUE |
+-------+-------------------+----------+
```

How many records

```
select count(*) from salaries;
 +----------+
 | count(*) |
 +----------+
 |  2844047 |
 +----------+
1 row in set (0.01 sec)
```

How many records in 1998 not partitioned

```
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

```
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
```

How many records in 1999 not partitioned

```
select count(*) from salaries
where from date between
'1999-01-01' and '1999-12-31';
+-----------+
| count(*)  |
+-----------+
|    260957 |
+-----------+
1 row in set (1.54 sec)

# NOT PARTITIONED
```

How many records in 1999 PARTITIONED

```
select count(*) from salaries
where from date between
'1999-01-01' and '1999-12-31';
+-----------+
| count(*)  |
+-----------+
|    260957 |
+-----------+
1 row in set (0.17 sec)

# partition p16
```

Deleting 1998 records 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 1998 records partitioned

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

MySQL 5.1 partitions

LEVERAGING REPLICATION

Replication schemes

MASTER

concurrent insert

INNODB
NOT
PARTITIONED

INNODB
NOT
PARTITIONED

SLAVE

concurrent read

INNODB
PARTITIONED
BY RANGE

SLAVE

concurrent batch processing

MyISAM
PARTITIONED
BY RANGE

SLAVE

large batch processing

Replication schemes

MASTER

INNODB
PARTITIONED

concurrent insert

SLAVE

MyISAM
PARTITIONED

batch processing

INNODB
NON
PARTITIONED

SLAVE

concurrent reads

MySQL 5.1 partitions

PITFALLS

Expressions

- Partition by function
- Search by column

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)
```

RIGHT

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

select count(*) from salaries where
from_date between '1998-01-01' and
'1998-12-31';
+-----------+
| count(*)  |
+-----------+
|    247489 |
+-----------+
1 row in set (0.46 sec)
```

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,p1
8,p19
         type: index
...
```

EXPLAIN

```
explain partitions select count(*)
from salaries where from_date
between '1998-01-01' and
'1998-12-31'\G
***** 1. row ****
           id: 1
  select_type: SIMPLE
        table: salaries
   partitions: p14,p15
...
```

MySQL 5.1 partitions

BENCHMARKS

Partitions benchmarks (laptop)

engine	storage (MB)
innodb	221
myisam	181
archive	74
innodb partitioned (whole)	289
innodb partitioned (file per table)	676
myisam partitioned	182
archive partitioned	72

Benchmarking results (laptop)

engine	query year 2000	query year 2002
InnoDB	1.25	1.25
MyISAM	1.72	1.73
Archive	2.47	2.45
InnoDB partitioned whole	0.24	0.10
InnoDB Partitioned (file per table)	0.45	0.10
MyISAM partitioned	0.18	0.12
Archive partitioned	0.22	0.12

Partitioning storage (huge server)

engine	storage (GB)
innodb (with PK)	330
myisam (with PK)	141
archive	13
innodb partitioned (no PK)	237
myisam partitioned (no PK)	107
archive partitioned	13

Benchmarking results (huge server)

engine	6 month range
InnoDB	4 min 30s
MyISAM	25.03s
Archive	22 min 25s
InnoDB partitioned by month	13.19
MyISAM partitioned by year	6.31
MyISAM partitioned by month	4.45
Archive partitioned by year	16.67
Archive partitioned by month	8.97

MySQL 5.1 partitions

TOOLS

The partition helper

- The INFORMATION_SCHEMA.PARTITIONS table
- The partition helper
  - http://datacharmer.blogspot.com/2008/12/partition-helper-improving-usability.html
  - A Perl script that creates partitioning statements
- ... anything **you** are creating right now :)

MySQL 5.1 partitions

TIPS

TIPS

- For large table (indexes > available RAM)
  - DO NOT USE INDEXES
  - PARTITIONS ARE MORE EFFICIENT

TIPS

- Before adopting partitions in production, benchmark!
- you can create partitions of different sizes
  - in historical tables, you can partition
    - current data by day
    - recent data by month
    - distant past data by year
  - ALL IN THE SAME TABLE!
- If you want to enforce a constraint on a integer column, you may set a partition by list, with only the values you want to admit.

TIPS

- For large historical tables, consider using ARCHIVE + partitions
  - You see benefits for very large amounts of data (> 500 MB)
  - Very effective when you run statistics
  - Almost the same performance of MyISAM
  - but 1/10 of storage

MySQL 5.5 partitions

NEW FEATURES

MySQL 5.5 enhancements

- PARTITION BY RANGE COLUMNS
- PARTITION BY LIST COLUMNS
- TO_SECONDS

MySQL 5.5 enhancements

BEFORE

5.1

```
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));
```

MySQL 5.5 enhancements

BEFORE

5.1

```
SHOW CREATE TABLE t \G
       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));
```

MySQL 5.5 enhancements

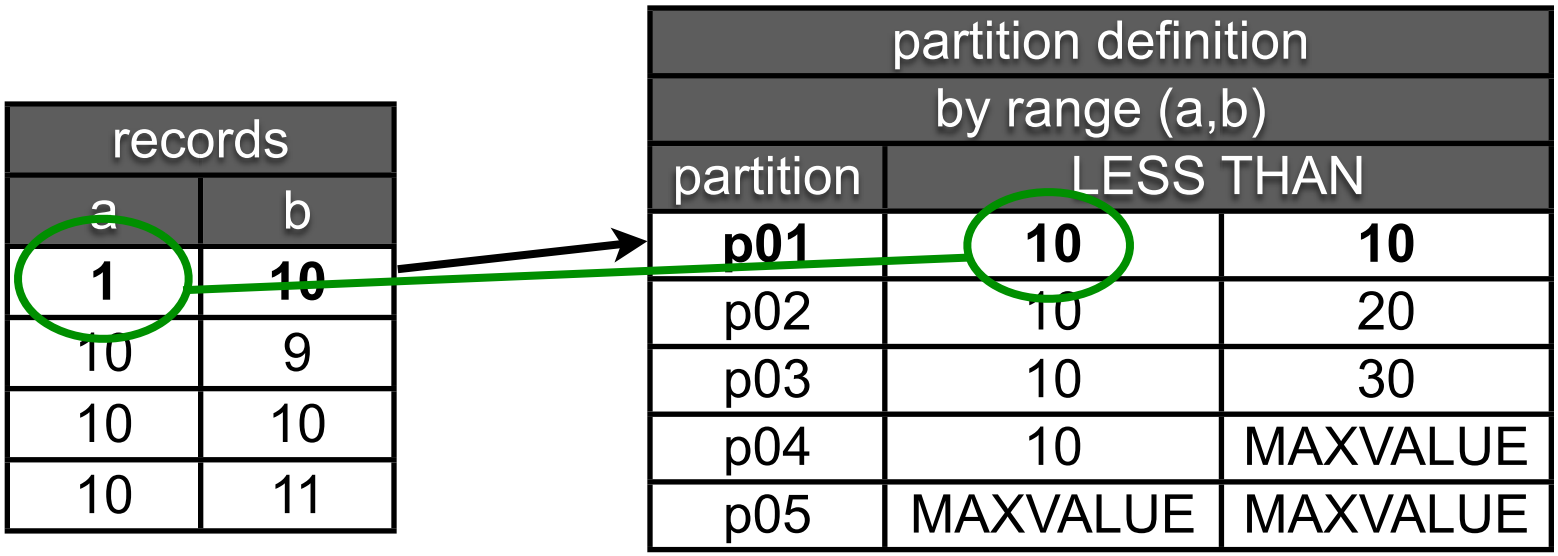

```
SHOW CREATE TABLE t
       Table: t
Create Table: CREATE TABLE `t` (
  `dt` date DEFAULT NULL
) ENGINE=MyISAM DEFAULT
CHARSET=latin1
/*!50500 PARTITION BY RANGE
COLUMNS(dt)
(PARTITION p01 VALUES LESS THAN
('2007-01-01') ENGINE = MyISAM,
[…]
```

MySQL 5.5 - 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));
```

records	
a	b
1	10
10	9
10	10
10	11

partition definition by range (a,b)		
partition	LESS THAN	
p01	10	10
p02	10	20
p03	10	30
p04	10	MAXVALUE
p05	MAXVALUE	MAXVALUE

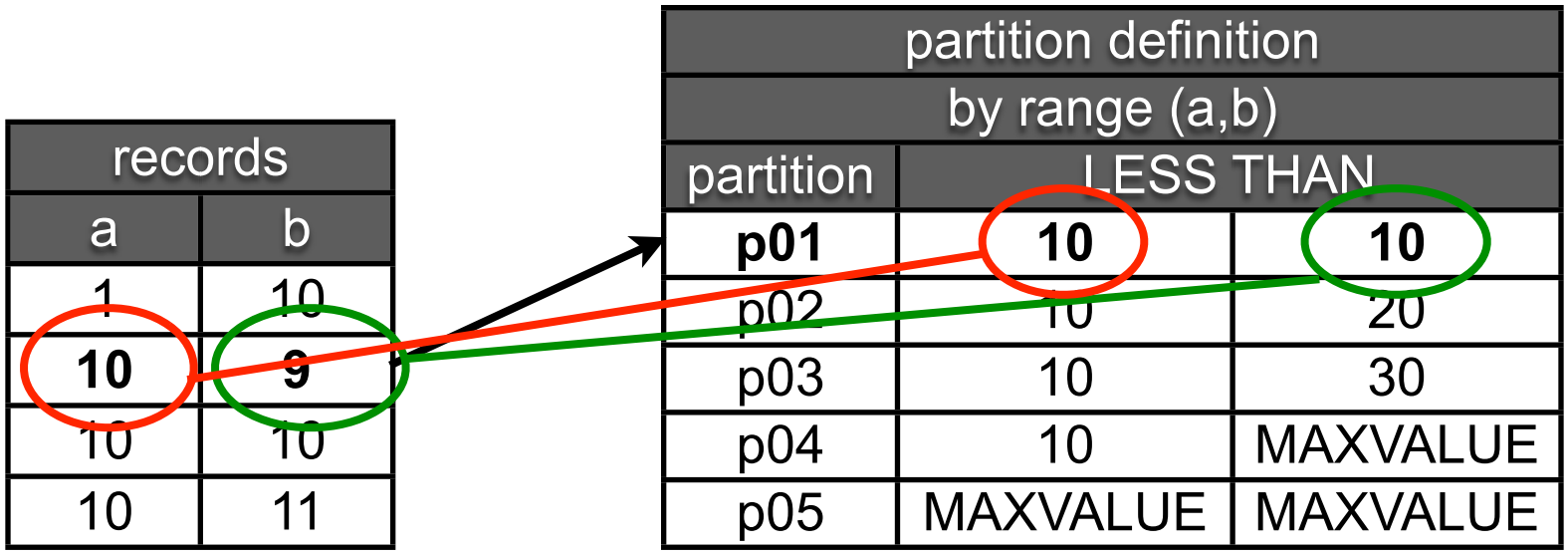


(1,10) < (10,10) ?

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

(1 < 10)
OR
((1 = 10) AND (10 < 10))

TRUE



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

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

(10 < 10)
OR
((10 = 10) AND (9 < 10))

TRUE

records	
a	b
1	10
10	9
10	**10**
10	11

partition definition by range (a,b)		
partition	LESS THAN	
p01	**10**	**10**
p02	10	20
p03	10	30
p04	10	MAXVALUE
p05	MAXVALUE	MAXVALUE

(10,10) < (10,10) ?

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

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

FALSE

records	
a	b
1	10
10	9
10	**10**
10	11

partition definition by range (a,b)		
partition	LESS THAN	
p01	10	10
p02	**10**	**20**
p03	10	30
p04	10	MAXVALUE
p05	MAXVALUE	MAXVALUE

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

(a < 10)
OR
((a = 10) AND (b < 20))

(10 < 10)
OR
((10 = 10) AND (10 < 20))

TRUE

```
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()

More new features

COMING SOON

PLANNED FEATURES - DISCUSS

- ALTER TABLE t1 EXCHANGE PARTITION p WITH TABLE t2

- SELECT * FROM T (PARTITION p) WHERE ...

Read more

- **articles**

  http://dev.mysql.com

- **official docs**

  http://dev.mysql.com/doc

- **my blog**

  http://datacharmer.blogspot.com

Thank you!

Giuseppe Maxia
http://datacharmer.blogspot.com
datacharmer@sun.com

MySQL 5.1 partitions

BONUS SLIDES

MySQL 5.1 partitions

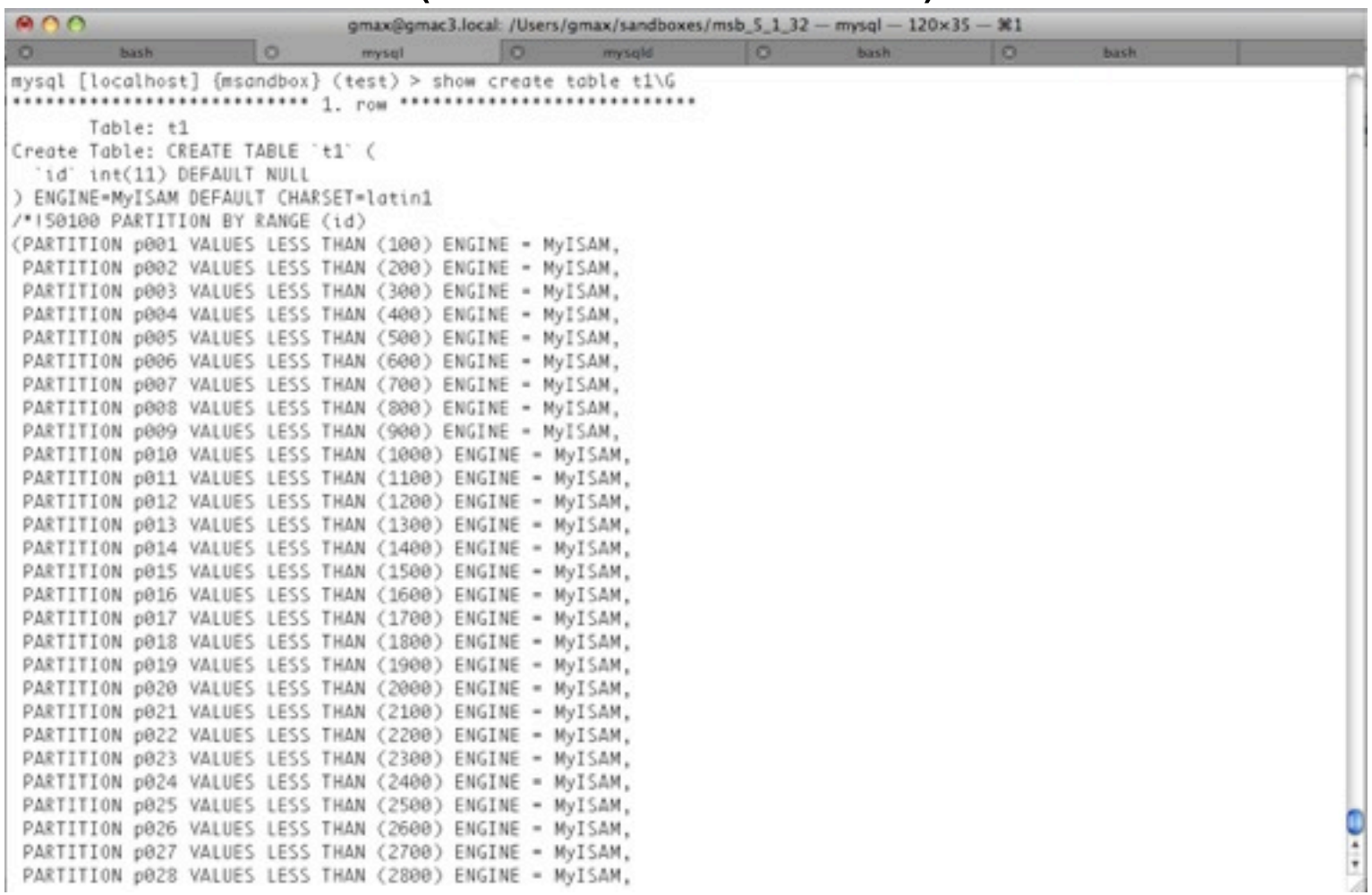
ANNOYANCES

Annoyances with partitions

- CREATE TABLE STATEMENT hard to read

Annoyances with partitions

- hard to read (**FIXED!! in 5.1.31**)



Annoyances with partitions

- CREATE TABLE only keeps the result of the expression for each partition.
- (you can use the information_schema to ease the pain in this case)

Annoyances with partitions

```
CREATE TABLE t1 (
 d date
) ENGINE=InnoDB
PARTITION BY RANGE (to_days(d))
(
 PARTITION P1 VALUES
  LESS THAN (to_days('1999-01-01')),
 PARTITION P2 VALUES
 LESS THAN (to_days('2000-01-01'))
)
```

Annoyances with partitions

```
CREATE TABLE `t1` (
   `d` date DEFAULT NULL
) ENGINE=InnoDB DEFAULT
CHARSET=latin1
/*!50100 PARTITION BY RANGE (to_days
(d))
(PARTITION P1 VALUES LESS THAN
(730120) ENGINE = InnoDB,
 PARTITION P2 VALUES LESS THAN
(730485) ENGINE = InnoDB) */
```

Fixing annoyances with partitions

```
select partition_name part,
partition_expression expr,
partition_description val
from information_schema.partitions
where table_name='t1';
+------+------------+--------+
| part | expr       | val    |
+------+------------+--------+
| P1   | to_days(d) | 730120 |
| P2   | to_days(d) | 730485 |
+------+------------+--------+
```

fixing annoyances with partitions

```
select partition_name part ,
partition_expression expr, from_days
(partition_description) val from
information_schema.partitions where
table_name='t1';
+------+-------------+-------------+
| part | expr        | val         |
+------+-------------+-------------+
| P1   | to_days(d)  | 1999-01-01  |
| P2   | to_days(d)  | 2000-01-01  |
+------+-------------+-------------+
```