MySQL Enhancement Proposal: Range Type

Tabs vs. Spaces

Varsha Ragavendran
Nisha Sharma

Davood Anbarnam
Ayman Abualsunun
Anton Sitkovets

Glib Sitiugin
Kevin Arindaeng

Overview

- Introduction
- Proposed Enhancement and Motivation
- Approaches of realizing the new feature
- Impacts on existing subsystems
- Testing Plans
- Effects of Concurrency
- Benefits of Enhancement
- Risks and Limitations
- Conclusion
- Lessons Learned

Introduction

- Enhancing MySQL to support Range Types
  - allows users to set a range for each column within a table

- Avoids using two columns to define start and end values of a range interval

Proposed Enhancement and Motivation

Ranged Types: A data type that represents a range of values

- Reserving rooms
- Employee scheduling
- Buying something within a certain price range

Proposed Enhancement and Motivation

http://www.sql-workbench.net/dbms_comparison.html

Feature	Oracle	Postgres	SQL Server	IBM DB2	MySQL	M
Data Types						
Range types (*)		Yes	No	No	No	N
User defined d		Yes	No (*)	Yes	No	N
Domains (*)	No	Yes	(Yes) (*)	No	No	N
Distinct types (*)	No	No	No	Yes	No	N

A data type that represents a range of values, e.g.:
all values from 1 through 100
The dates from 2014-01-01 to 2014-01-08

Proposed Enhancement and Motivation

MySQL Numeric and Date/Time Datatypes (Examples):

	Integer	Fixed-Point	Floating Point	Date and Time
	INT	DECIMAL(5,2)	FLOAT(5,3)	DATETIME
Example	123456	999.99	-999.999	1000-01-01 00:00:00

Proposed Enhancement and Motivation

MySQL Ranged Types (Examples):

	Integer	**Fixed-Point**	**Floating Point**	**Date and Time**
	INT**RANGE**	DECIMAL**RANGE**(5,2)	FLOAT**RANGE**(5,3)	DATETIME**RANGE**
Ex.	[100,200]	(500.01,900.99)	[-123.45,678.91)	[1000-01-01 00:00:00, 2017-11-29,13:00:00)

() = Does not include (exclusive)

[]= Includes (inclusive)

Proposed Enhancement and Motivation

```
CREATE TABLE work_sched
(
    employeeId INT
   ,during DATETIMERANGE
);
```

```
INSERT INTO work_sched
VALUES
(
    1,
   [2017-11-29 09:00:00,
   2017-11-29,17:00:00]
);
```

Proposed Enhancement and Motivation

```
SELECT * from work_sched where during =

'[2017-11-29 09:00:00,
2017-11-29,17:00:00]'::DATETIMERANGE
```

```
SELECT * from work_sched where during <@

'[2017-11-29 08:00:00,
2017-11-29,20:00:00]'::DATETIMERANGE
```

Where "<@" means "range is contained by"

Stakeholders

- End-User
- ORACLE
- Competitors (DB2, Amazon RDS)
- Us

First Approach: Tuples

$$\{ id1 \mapsto e1, id2 \mapsto e2, id3 \mapsto e3, \ldots \}$$

ei $\triangleq$ ith element of range

- Id is generated, i.e. user does not provide one

Second Approach : Hybrid

$$[a, b] \triangleq \{ x \mid a <= x <= b \}$$

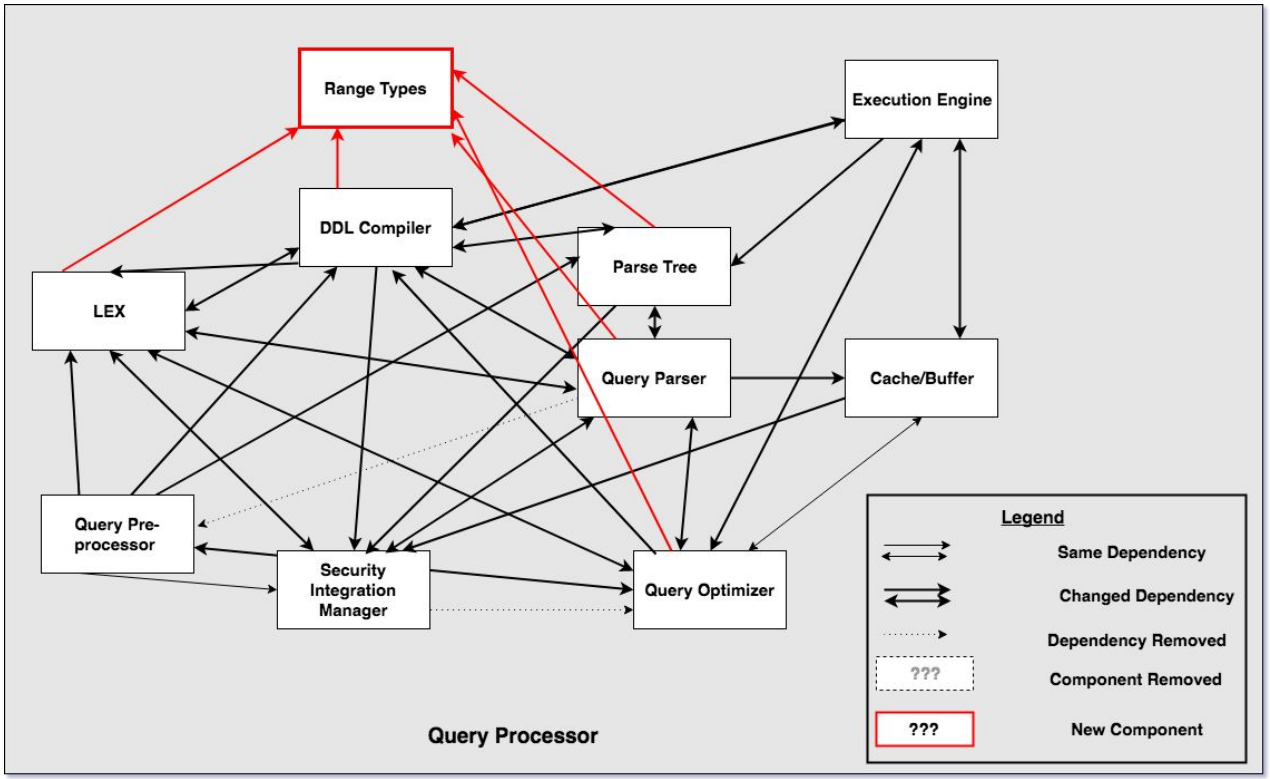
OR

$$\{ id1 \mapsto e1, id2 \mapsto e2, id3 \mapsto e3, \ldots \}$$

Comparing the Approaches

- Approach 1 - Tuples
  - \+ General model
  - \+ Relatively Simple
  - \- Large Ranges are an Issue (Scalability)
- Approach 2 - Adaptive
  - \+ Choice
  - \+ Greater Efficiency with Intervals
  - \- More work on designer's end
- **Final Choice : Approach 2**

Impacts on subsystems



Plans for Testing

- Compare the functional and nonfunctional test coverage
  - To account for: performance & resource utilization
  - Utilizing: Random Query Generator

Effects of concurrency

- No effect on concurrency control
- MySQL supports locks of different granularities
  - row locks
  - table locks
- Locks don’t deal with individual cells

Id	User	Host	db	Command	Time	State	Info
2	root	localhost:52714	sampledb	Sleep	79		NULL
3	root	localhost:52715	sampledb	Query	0	starting	show processlist
4	root	localhost:52779	sampledb	Sleep	99		NULL
5	root	localhost:52780	sampledb	Query	74	Waiting for table metadata lock	insert into tbl(col) value...

Value/Benefits of Enhancement

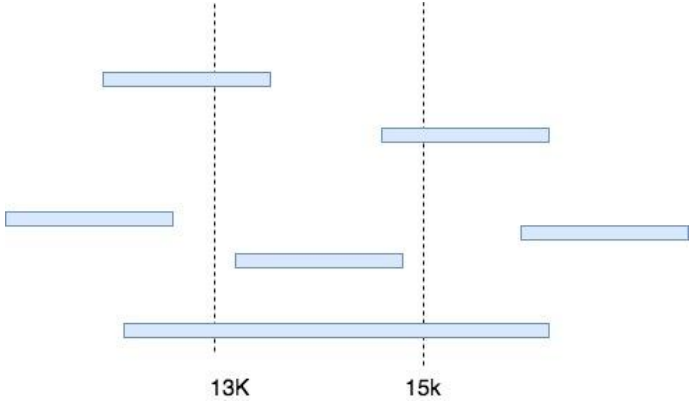
- Ranges are everywhere.
- Many use cases that require ranges of values including price, scheduling and measurement data.
- Implementing a range type reduces complexity of range value querying
- Makes searching for overlapping data within a range much easier for the programmer
- `select '[2014/01/01, 2014/01/31]'::datetimerange;`

Value/Benefits of Enhancement Example

Shopping for a Used Car within a Min/Max budget

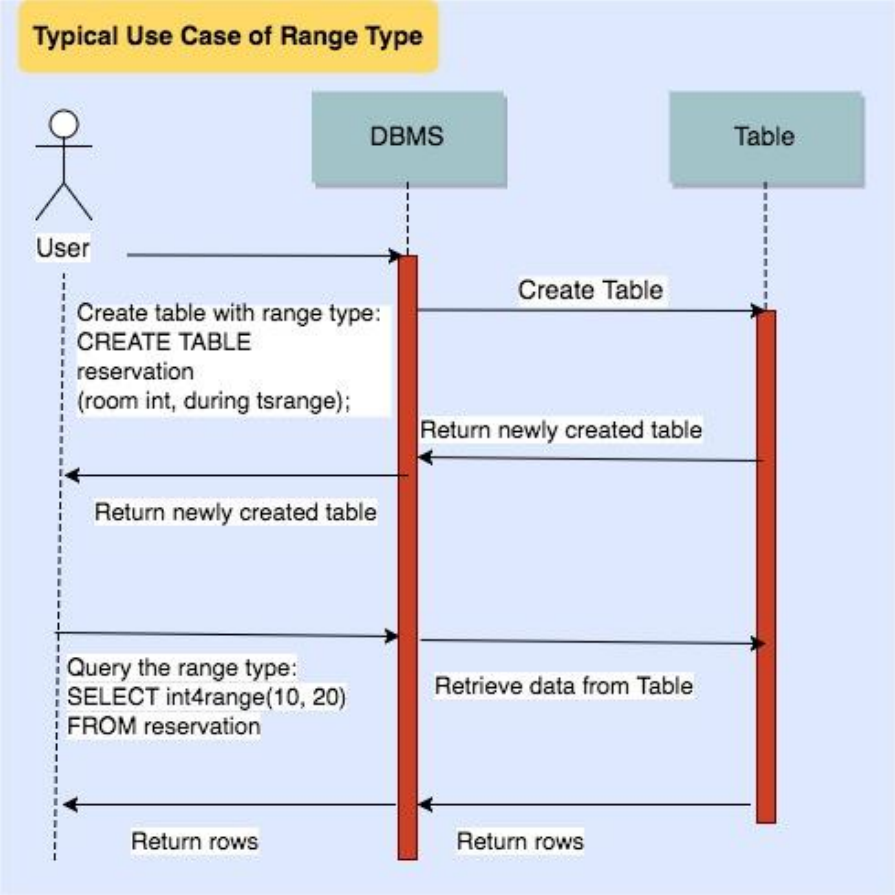
```
SELECT *
FROM cars
WHERE
  (
    cars.min_price ≤ 13000 AND
    cars.min_price ≤ 15000 AND
    cars.max_price ≥ 13000 AND
    cars.max_price ≤ 15000
  ) OR
  (
    cars.min_price ≤ 13000 AND
    cars.min_price ≤ 15000 AND
    cars.max_price ≥ 13000 AND
    cars.max_price ≥ 15000
  ) OR
  (
    cars.min_price ≥ 13000 AND
    cars.min_price ≤ 15000 AND
    cars.max_price ≥ 13000 AND
    cars.max_price ≤ 15000
  ) OR
  (
    cars.min_price ≥ 13000 AND
    cars.min_price ≤ 15000 AND
    cars.max_price ≥ 13000 AND
    cars.max_price ≥ 15000
  )
ORDER BY cars.min_price;
```

VS.




```
SELECT *
FROM cars
WHERE cars.price_range && int4range(13000, 15000, '[]')
ORDER BY lower(cars.price_range);
```

Use Cases



Risks and Limitations

- Since a new subsystem is added there might be
  - some unexpected conflicts and overlapping with existing subsystems
- Only limited to Numeric and DateTime data types
  - With others not so optimal

Conclusions

- Mysql with range types makes it easy to use.
- Doable with the given second approach.
- Doesn’t affect concurrency.
- Minimal changes to other subsystems.

Lessons Learned

Normal MySQL

MySQL + Range Types

Question Period