10 Common Mistakes Developers Make When Writing SQL

Charly Batista
Percona
charly.batista@percona.com

PERCONA
LIVE EUROPE
AMSTERDAM

Charly Batista

- Senior Support Engineer at Percona.
- MySQL and PostgreSQL expert.
- Working with development and databases for over 20 years.
- Speaker at Percona Live and meetups about database and development.

Agenda

- Do not be scared of writing SQL.
- The infamous SELECT *.
- Do you really know how NULL works?
- Is this column integer or varchar?
- The order of the columns doesn't matter, does it?
- We have a lot of memory, let's use it!!
- We need to paginate the result.
- Do not underestimate the power of character encoding.
- Connection pool.
- Let's talk about batch...

Bonus...

- Database transactions.
- The power of the prepared statements.
- The index is our friend!

Do Not Be Scared of Writing SQL

SQL isn't rocket science ;-)

SQL is Our Friend

- SQL is simple and easy to use.
- It can give a very good readability.
- Easier to tune and improve performance.
- Have you ever tried to map a complex join?
- How about batch processing?

Tweak Hundreds of Annotations, or...

```
@NamedEntityGraph(
  name = "post-entity-graph-with-comment-users",
  attributeNodes = {
    @NamedAttributeNode("subject"),
    @NamedAttributeNode("user"),
    @NamedAttributeNode(value = "comments", subgraph = "comments-subgraph"),
  },
  subgraphs = {
    @NamedSubgraph(
      name = "comments-subgraph",
      attributeNodes = {
        @NamedAttributeNode("user")
      }
    )
  }
)
@Entity
public class Post {

    @OneToMany(mappedBy = "post")
    private List<Comment> comments = new ArrayList<>();
    //...
}
```

```
select
    p.id,
    p.subject,
    p.user_id,
    c.post_id,
    c.id,
    c.id,
    c.post_id,
    c.reply,
    c.user_id,
    u.id,
    u.email,
    u.name
from
    post p
    join comment c on p.id = c.post_id
    join user u    on p.user_id = u.id
where p.id = ?
```

Do You Really Need All Columns?

*The infamous SELECT **

- It can prevent optimizations.
  - Index only scans / covered index.
- It can bloat the DB memory footprint.
- Can also bloat the app server memory footprint with massive ResultSet's.
- Let's not forget about our wires… Network will also suffer!

Do You Really Know How NULL Works?

You may be surprised!!

- Even though **JDBC** maps **SQL NULL** to **Java null** they are not the same.
- NULL is also called UNKNOWN.
- SQL “NULL = NULL” is false (not the same as null == null in Java).
- Arithmetic operations with NULL may not results in what you expect!

Is This Column Integer or Varchar?

Data type mismatch in predicates may ruin your day!

- Be careful with the Statements and the Prepared Statements.
- Also when joining tables do not use columns with different data types.
- It may cause data loss (truncate, implicit conversion, etc).
- It can cause the database to avoid an index.

The Order of the Columns Doesn't Matter, Does it?

Not all indexes are the same, nor are the databases!

- The order of the columns is very important for composed indexes.
- The columns are evaluated from the left most in the index creation order.
- The order you declare the columns in the where clause doesn’t impact.
- Cardinality and operators are very important.

Let’s Talk About Java!

It’s time!!!

We Have a Lot of Memory, Let's Use it!

Please, do not process data in Java memory when it can be done by the DB

- NEVER load two tables from separate queries into maps and join them in Java memory.
- Do not order, aggregate, execute nasty maths if it can be done by the DB.
- It is, for most of the cases, easier to write correctly in SQL than in Java.
- The database will probably be faster than your algorithm.
- Remember those wire plugged to your server? They will thank you!

We Need to Paginate the Result

More often than you think!

- The database is your friend and has nice features:
  - LIMIT .. OFFSET, TOP .. START AT, OFFSET .. FETCH
- Remember the memory? How about the wires? They will all love you!
  - In order to be able to sort and paginate, we have to fetch the complete result set into app memory.
- Ordering and paginating on DB is way faster than pagination in memory.
- We can avoid early row lookup and improve the performance even more.

Do Not Underestimate the Power of Character Encoding

My database speaks many languages

- Make sure that the application and DB are using same charset.
- Encoding mismatch can:
  - Completely mess your app “view”.
  - Prevent you from using a specific language or symbols.
  - Cause data loss.
- It can also cause odd errors that are hard to debug:
  - *ORA-01461: can bind a LONG value only for insert into a LONG column.*

Connection Pool

It may be obvious but many people do not use one!

- Standard. It is present in the JDBC specification (3.0).
- Can greatly increase the performance of the app, while reducing overall resource usage.
- Reduced connection creation time.
- Controlled resource usage.

Let's Talk About Batch...

The more the better

- It’s way faster than single inserts.
- JDBC knows batching! Create a batch INSERT with a single SQL statement and multiple bind value sets.
- Try to load the data during off peak time.
- If the above is not possible try to commit smaller batches to:
  - Keep the UNDO log small.
  - Avoid locks and race condition.

Demonstration

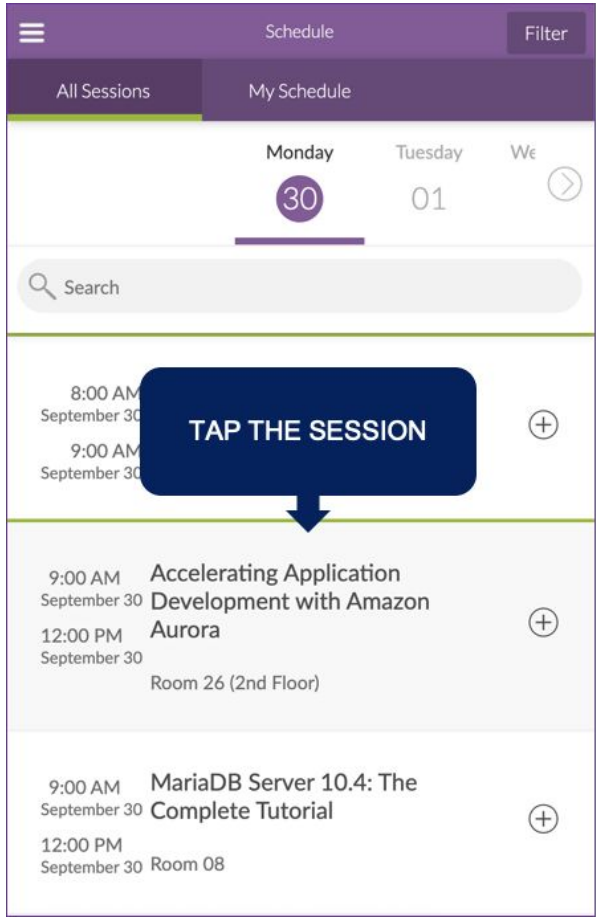
Bonus

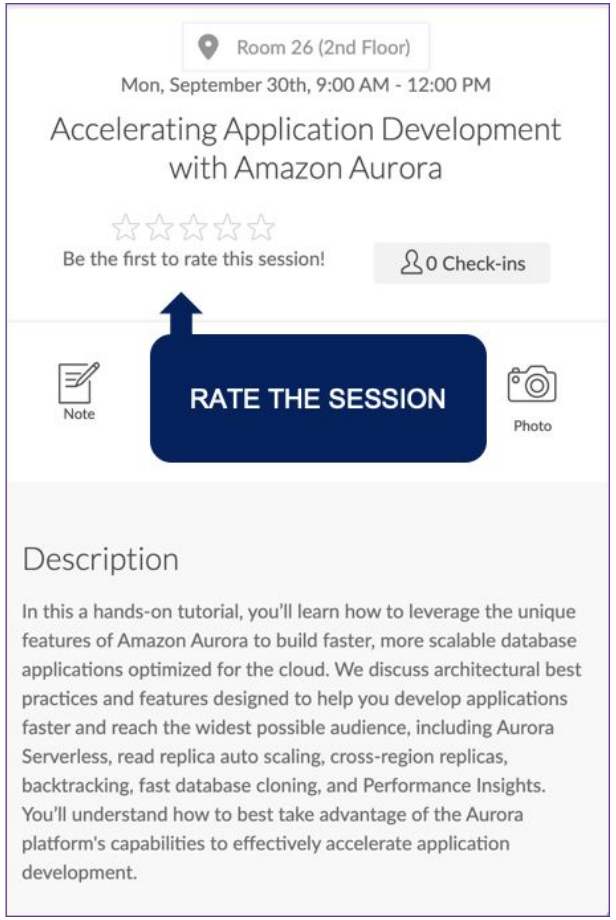
Do we have time?

Bonus...

- Database Transactions.
- The power of the Prepared Statements.
- The index is our friend!

Rate My Session



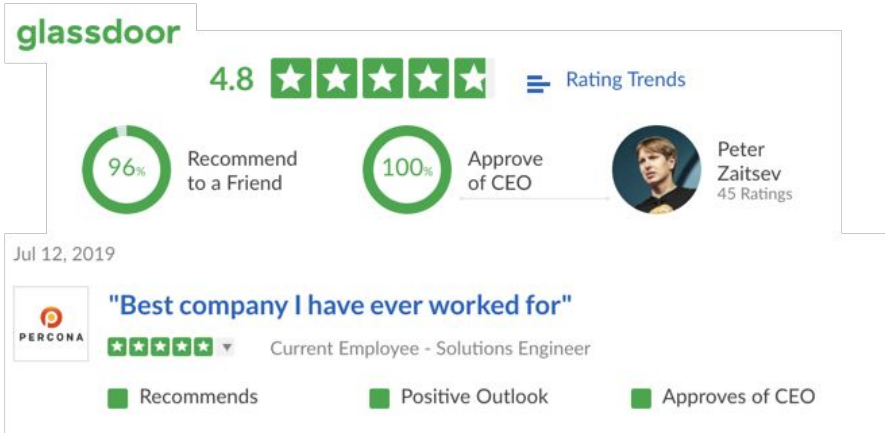


We're Hiring!

Percona's open source database experts are true superheroes, improving database performance for customers across the globe.

Our staff live in nearly 30 different countries around the world, and most work remotely from home.

Discover what it means to have a Percona career with the smartest people in the database performance industries, solving the most challenging problems our customers come across.



Any Questions?