

Who's Afraid of the Big Bad SQL?

Kim Berg Hansen


 @kibeha

 <http://kibeha.dk>

BASEL | BERN | BRUGG | BUKAREST | DÜSSELDORF | FRANKFURT A.M. | FREIBURG I.B.R. | GENÈVE
HAMBURG | KOPENHAGEN | LAUSANNE | MANNHEIM | MÜNCHEN | STUTT GART | WIEN | ZÜRICH

trivadis

About me

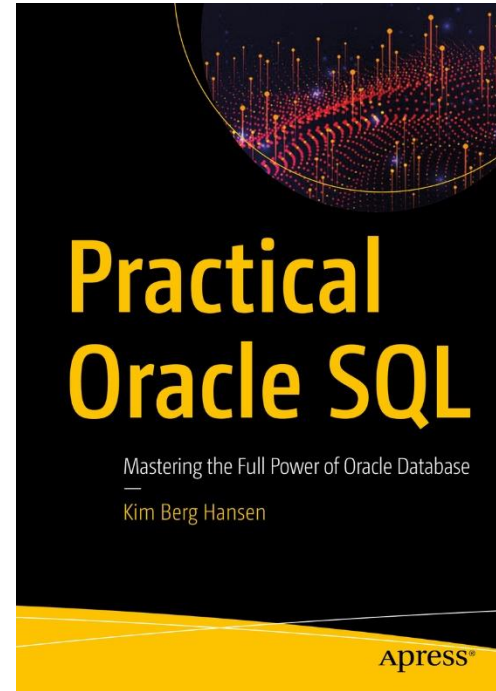
- Danish geek
- SQL & PL/SQL developer since 2000
- Developer at Trivadis since 2016 <https://www.trivadis.com>
- Oracle Certified Expert in SQL
- Oracle ACE Director 
- SQL quizmaster <https://devgym.oracle.com>
- Blogger <https://kibeha.dk>
- Likes to cook and read sci-fi
- Member of Danish Beer Enthusiasts

 @kibeha



Author of "Practical Oracle SQL"

- Not a SQL-101 book
- Not a reference manual replacement
- For developers knowing basic SQL-92 syntax but wanting to advance further
- More elaborate examples relating to daily life as very simple examples are difficult to relate to work
- Useful SQL features that aren't widely used - but should be
- More background in an interview in NoCOUG Journal:
[http://nocoug.org/Journal/NoCOUG Journal 202002.pdf#page=4](http://nocoug.org/Journal/NoCOUG%20Journal%20202002.pdf#page=4)
- The book:
<https://www.apress.com/gp/book/9781484256169>
<https://www.amazon.com/Practical-Oracle-SQL-Mastering-Database/dp/1484256166>





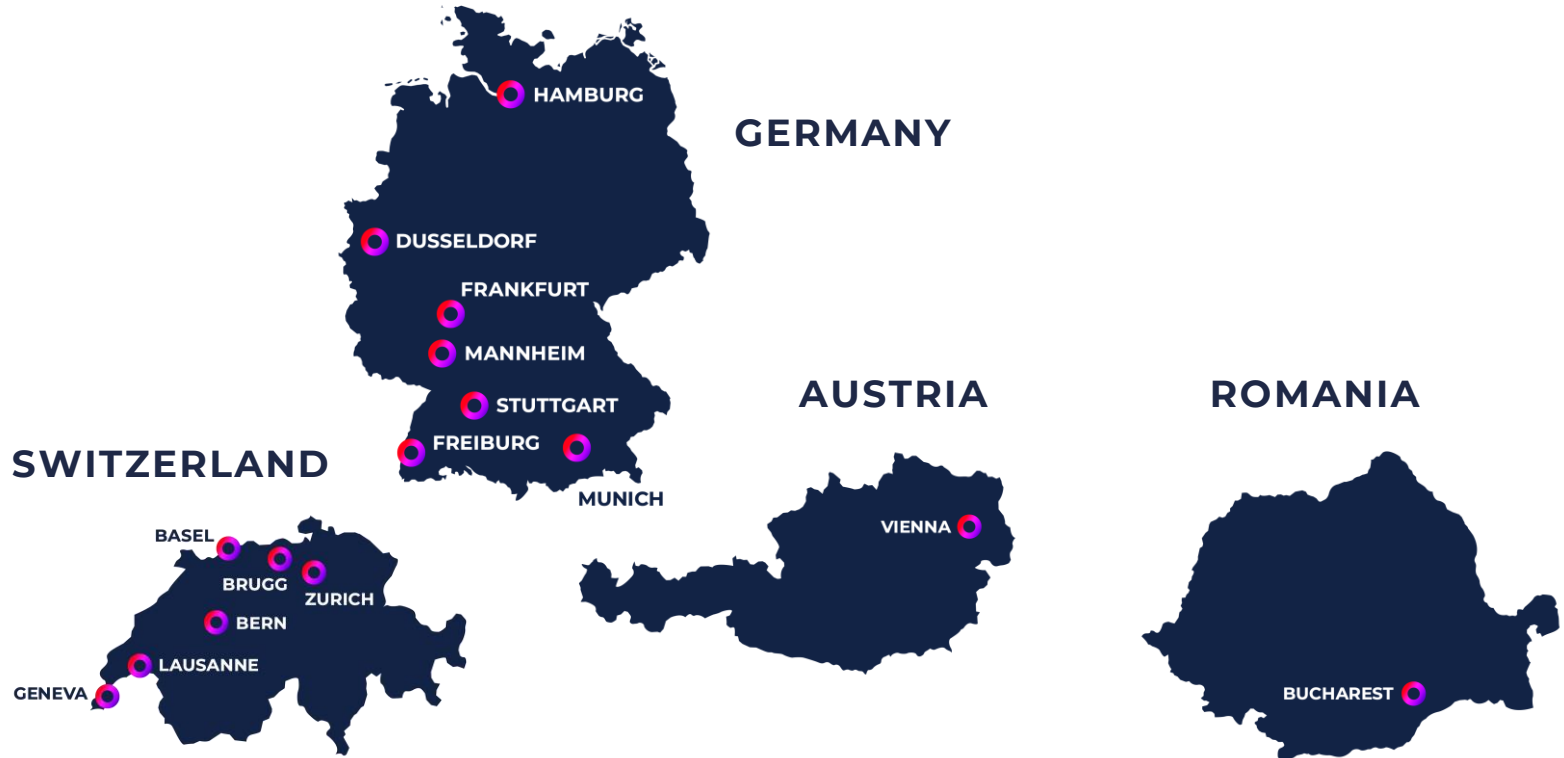
Mentor and Speaker Hub

Our goal is to *connect* speakers with mentors to assist in *preparing* technical sessions and *improving* presentation skills

Interested? Read more and get in touch

<https://mashprogram.wordpress.com>

5 TRIVADIS - PART OF ACCENTURE



Agenda

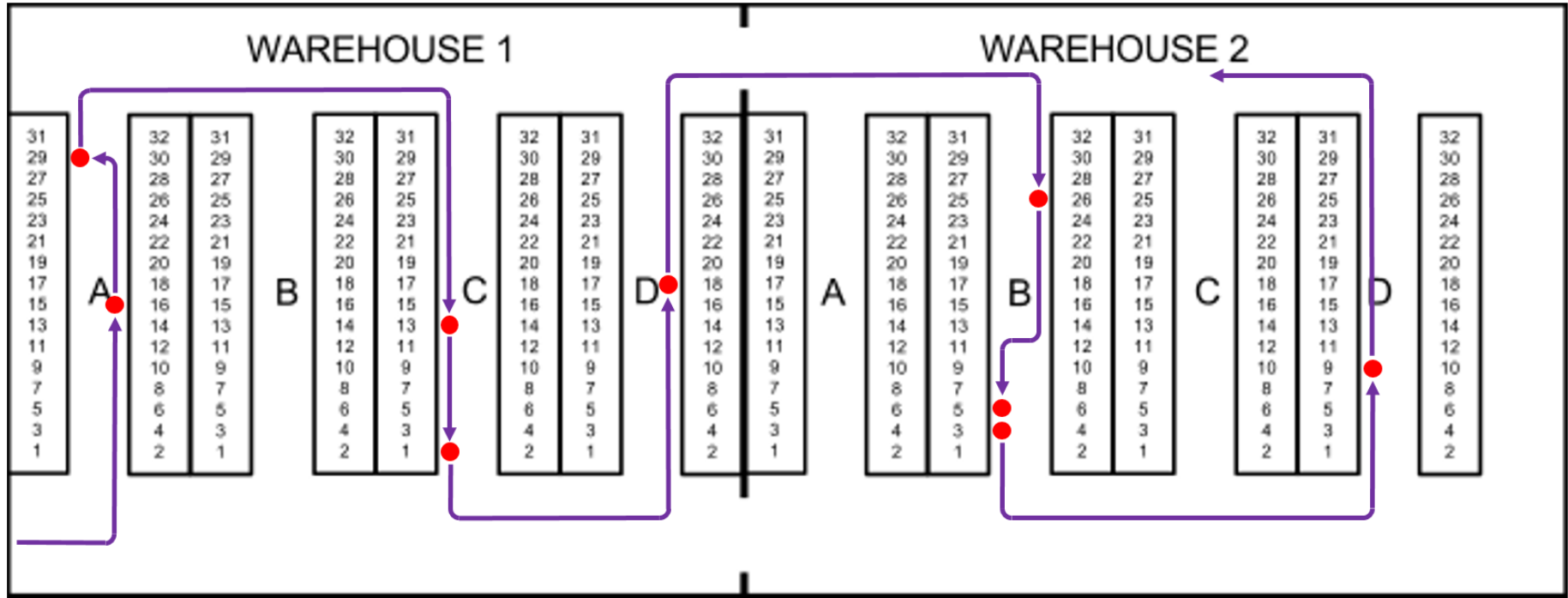
- Task: FIFO picking
- Three demos:
 - Naïve Hibernate
 - Improved Hibernate
 - SQL with analytic functions
- Comparison
- Conclusion

Task: FIFO picking

First-In-First-Out picking

- A FIFO warehouse picking list
 - I have an order from a customer containing a set of orderlines that need to be picked from the warehouse
 - For each of the products in the orderlines I need to pick from the locations in the warehouse where the oldest purchased inventory is (the First-In-First-Out principle)
 - And the resulting picking list needs to be output in an optimal driving order for the operator so he doesn't waste time by driving his picking trolley back and forth in the warehouse

Desired result



What seems to be needed

- How can I?
 - Retrieve orderlines
 - Loop over them
 - For each product do a query of inventory
 - Find the oldest until sufficient quantity reached
 - Fort the resultant collection in the desired driving order
- Let's try it and find out...

Naïve Hibernate

Naïve Hibernate - DEMO

- Mapping tables to Java objects / entities with Hibernate
- Retrieve orderlines of the desired order
- For each orderline:
 - Retrieve inventory of that product by purchase-date
 - Add inventory to picklist array
 - Break loop when enough quantity has been picked
- Sort the saved array of inventories for good picking route
- Output sorted array



Improved Hibernate

Improved Hibernate - DEMO

- Same base code as Naïve Hibernate demo
- But added annotations / join specifications to improve generated SQL



SQL with analytic functions

SQL with analytic functions - DEMO

- Packaged stored procedure
- Opens output REF CURSOR with a single SQL statement
- SQL includes all FIFO and routing logic
- Java just calls procedure, fetches from the cursor, outputs the result



Comparison

Test results

- For each of the three versions I did:
 - Flush shared pool
 - 1 warmup execution
 - 5 test executions and take average milliseconds of those 5


	Avg ms	Calls	Blocks	Rows
Naïve Hibernate	~400	36	150	56
Improved Hibernate	~305	4	35	20
Analytic SQL	~300	1	31	9

- Improved Hibernate version is not that bad for this simple case
- But number of calls can be significant with higher latency
 - This test was VirtualBox DB on same laptop - practically no network latency at all
- And amount of data sent from DB to client can be significantly different

- Remember too, that in order to make improved Hibernate version, database knowledge is needed
- If you have sufficient knowledge for that, then you also have sufficient knowledge to write good SQL

Conclusion

Who's Afraid of the Big Bad SQL?

- Frameworks often advertise "make app without SQL"
- SQL thought of as "hard"
- Developers feel out of control with SQL - it's too much "magic"
- Are you afraid of the **Big Bad SQL?** 

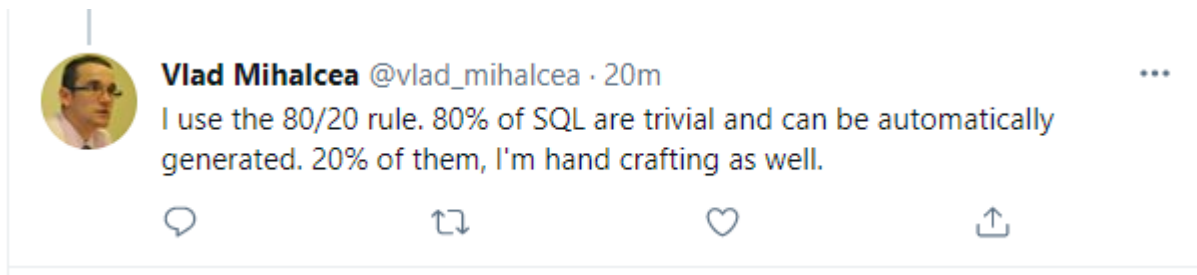


I'm not!

- SQL isn't really harder
 - SQL can solve most tasks
 - SQL is very efficient
 - SQL saves coding time
 - SQL saves resources
-
- It's **Great Good SQL!** 😊



Mixing is allowed



<https://vladmihalcea.com/>
[@vlad_mihalcea](#)

- You can use frameworks as code generators to save time instead of typing trivial code
- Then use your time more profitably on SQL for the non-trivial code
- Using the framework does not excuse you from knowing SQL
- Knowing SQL allows you to use the framework well
- Using the framework without knowing SQL can easily lead you astray

Q & A



Questions & Answers

This presentation

https://bit.ly/bigbadsql_pptx

Source code repo

<https://github.com/kibeha/big-bad-sql>

✉ kim.berghansen@trivadis.com

🐦 [@kibeha](https://twitter.com/kibeha)

📡 <http://kibeha.dk>

