

# 5 things developers do that drive me crazy

And what I did to survive

## About me

- Øyvind Isene
- Consultant, work for Sysco in Oslo
- Oracle ACE
- @OyvindIsene on Twitter
- DBA, but haven't restored a database for some time
- Don't miss patching, best reason to love the cloud



## The last 5 years

- Displaced among developers as a DBA
- Optimization aka tuning
- Troubleshooting in general
- Looking for trouble, before it goes to production
- Trying to explain the mindset of DBAs to developers (unsuccessfully)

## I actually like development

- APEX and PL/SQL is great fun
- R, Python, Scala ...
- Other stuff with focus on the data
- Development is not just coding

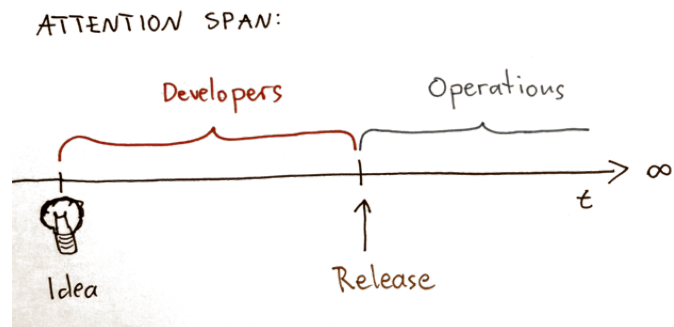
## Agenda and goal for this presentation

- Complain about developers
- Justify the complaints
- Show you what to do since they don't care

## Not that I try to start a flame war

... but if they could just try to understand how it is to get the responsibility for some crap when they are all gone.

- Today's development - tomorrow's legacy
- Likely to stay there for a long time ( $\infty \approx 20$  years)
- Out of sight, out of mind



# Them

- Creative
- Like shiny things
- Impatient - they've already got a new idea
- Wonder if ITIL is the work of the Devil



## A persistent storage?

Some developers simply refuse to relate to the database

Rambling around about schema-on-read, persistent storage etc.

Tempting to ask them to swim in a data lake

## Us

- Don't like being called 2 a.m - boring is good
- Too many systems and users to please
- Little time for manual tasks
- Worries about security, scalability, and uptime

## Doesn't have to be that way

- Operations have something to learn from developers
- Focus on one tasks once in a while
- Do more coding (shell scripts do count!)
- Be more creative - automate!

## Can developers learn something from DBAs?

- Part of reality is unknown to them
- Devs are usually less exposed to the entire architecture
- DBAs know more about the nature of databases - what works and what doesn't
- DevOps *can* be a good idea

## 1. No data model

Sit down for a think first

## What were they thinking?

- Probably not at all
- A data model proves that you have spent time understanding your universe
- This is the time to meditate over subjects and connections
- Abstract expressions need clarifications

## A way to communicate

- Use it to communicate with ~~stakeholders~~ normal people
- Especially with those who don't share your daily language
- If you can't model it, you might not have understood it
- Fail early - discover misconceptions early
- An important part of documentation

- Avoid crappy data getting in to the database
- Less work on data cleansing later
- Make sure they are not using the database as a bitbucket





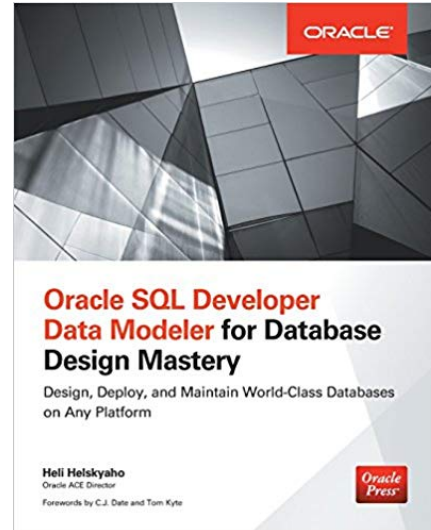
## When ORM is not the best option

- Sometimes an object to relational mapping (ORM) tool is not the best option
- If the ORM tools decides the data model what do you get?
- "Will it scale?"
- "May I write you some PL/SQL?"

## One persistent myth

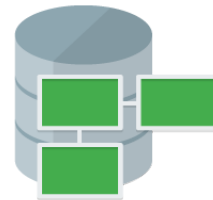
- Constraints lead to bad performance
- Not true - the optimizer use extra information to create a better plan

- Templates
- Source control
- Design rules



## What can you do?

- Simply demand a data model before you do system acceptance
- Look for the usual stuff
  - Primary, unique, and foreign keys
  - Other constraints
  - Decent normalization
- Help them with *SQL Developer Data Modeler*



## 2. Forget about security

Two things can ruin my day: talking about security and licenses

- But has been on the agenda for a while
- Sounds boring to many people
- Security is the new HA



## Security is a feature

- Perhaps a boring subject, but necessary
- Does the application collect sensitive data?
- This can be identified during data modeling

→ Security must be a proper activity in the project

# Authentication

Being authenticated means the system trust who you are, nothing more



## Authorization

Now you are *allowed* to do something...

At minimum agree on a few roles that give access

## SSO / one single source for authentication

Sign-On / login with a password or similar only once

If you authenticate several times against the same user base (e.g. Active Directory), that is not SSO.

## Sensitive data and personally identifiable information

- GDPR, anyone?
- No, you can't have all production data in dev and test
- Data masking is easier if you start out with identifying what needs to be masked

→ identify sensitive data before you start

## Audit logs

- Very useful after data theft
- Audit only access to the most critical data

→ another reason to identify sensitive data

# Encryption

This is for data at rest or in transit to avoid data being read by unauthorized part.

You can't show encrypted data to users, so you have to decrypt it at some point...



## SQL injection

Just because they don't want to know about bind variables.

Bonus: If you do this right the database performs better too!



## Sometimes a burden

- You don't make more friends by denying access
- Should not be a burden only for the DBA
- Possibly a burden for the proxy aka project manager
- But the DBA probably worry more about a security break

## Where to implement security

- As close as possible to the data
- Doesn't hurt to implement in layers
- But an application can get thrown away

→ Don't hide a wide open database behind fire walls only



## What can you do?

- Again, ask for a data model
- Request an assessment about sensitive data
- Repeating review during implementation
- Be alert when they ask for elevated privileges
- Standard procedure to implement security in different projects
- Look into *Oracle Database Security Assessment Tool*

### 3. Not using the *Oracle* documentation

RTFM FFS

## It is OK to google, but...

- Googling SELECT and ending up in a manual for another database is not cool
- Usually easier to lookup the syntax in a bookmarked manual
- It is about studying - not always getting a short answer

## Database Concepts

- At least the chapters relevant to design and development (Part I, II, III and VI)
- Discover why certain certain patterns do not scale
- Understand better critical parts to making the database consistent

## Database Development Guide

- Should be obvious for developers
- Important things in database applications
  - Correct datatype
  - Locking
  - Indexing
  - PL/SQL

## Database New Features Guide

- You can't google what you don't know
- Discover new cool stuff that simplifies your application

## PL/SQL Packages and Types Reference

- Shows packages that can replace your code
- Gives you ideas for utilizing the database better

## SQL Language Reference

- Much faster then googling and finding the wrong version
- Discover all the features in Oracle SQL
- A few gems have been around for many years, but not used much

→ using advanced SQL can speed up code by orders of magnitude



## PL/SQL Language Reference

- Maybe you prefer a book?
- Useful reference for syntax

## SQL Tuning Guide

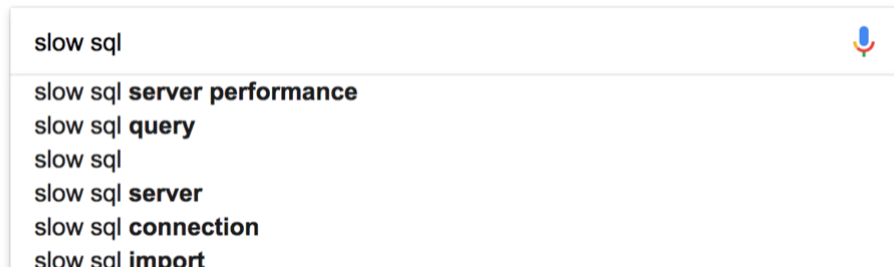
- When you want to figure out performance on your own
- Performance as a feature - design for it
- There is a chapter *Designing and Developing for Performance*

## Other specialized guides

- *Data Warehousing Guide*
- *Data Mining Concepts*
- *JSON Developer's Guide*
- *Spatial and Graph Developer's Guide*

## Can you trust what Google finds you?

- Of course not
- How can you know if you haven't read the documentation or learned it from a trustworthy source?



## What can you do?

- Have links ready to send out to the team and new members
- [docs.google.com](https://docs.google.com) is a must
- Give short introductions to the documentation set
- Explain the most important manuals
- Recommend good books, blogs and online training material

## 4. Confusing DevOps with operations

Full stack developer?!

## It takes more than building a container

- Administration is stressful (sometimes)
- Availability
- Network
- N-tier architecture
- End users
- Patching & upgrades
- Backup & Restore

## DevOps is...

*... a software engineering culture and practice that aims at unifying software development (Dev) and software operation (Ops). ... to strongly advocate automation and monitoring at all steps of software construction, from integration, testing, releasing to deployment and infrastructure management. (Wikipedia)*

If this is done right I'm 90% happy.

But I suspect not all talking about DevOps go the extra mile.



## *In production usually means*

- People depend on it - if it's useful
- Data being collected - usually has some value, we don't want to lose it
- We discover all the bugs we didn't find by testing before release

## Not enough testing

Lots of tests, but it fails in production because...

### TESTING







- \* Unit ✓
- \* Integration ✓
- \* Functional ✓
- \* Volume X too much work

## Can they run their own applications?

- Small hiccups accumulate over time and causes big problems
- We only ask for
  - No manual intervention in production
  - Meaningful error messages
  - Tested for real world scenarios

## Attention span redux


- A developer can usually forget what is in production
- What if he was forced to do a review once a year?
- CV-driven development - what is cool this year?
  - Frameworks
  - Methods
  - New buzzwords
- Legacy systems usually last for years


  Menu    Account  Country/Region  Call

## Oracle Technology Network

Welcome to the world's largest community of developers, admins, and architects using industry-standard technologies in combination with Oracle products.



[Join today or learn more](#)







What's New 



Oracle Code: Free event series for developers is coming to your town. [Register now!](#)

Free Cloud Platform Trial

[Java Developers](#)  


[Database Admins and Developers](#)  


[System Admins and Developers](#)  


[Solution Architects](#)  


**Working**

**Not working**

## 5. Little understanding of basic concepts

Embarrassing simple ideas

# Latency

- It starts with *late*
- Simple math
- If you repeat a small operation many times it will take a lot of time
- This explains why an index is not always the solution

## Classic beginner's latency trap

- Beginner learns to execute SELECT in database, retrieves N rows.
- For each row, generate a new SQL and send N UPDATES to the database
- At least  $N + 1$  roundtrips
- When all can be done with a slightly more complex UPDATE once.



## Speed of light

- About 300 000 km/s
- 1000 km between you and your server; how many round trips in 5 seconds?

## Response time

- The user wants an answer in 5 seconds or less
- How much work can you do in available time?
  - Sorting 500 GB?
  - 2000 roundtrips between application and database?

## No such thing as a best practice

- If one solution served everybody we will automate it
- Plenty of bad practices
- Some good practices

## No idea about volume

- Sorting small data is very fast
- Sorting large data spills to disk - slow
- Shuffling lots of data between servers take time

Slightly more advanced concepts

but if you work with databases...

# Transactions

- *Consistent* answers to queries
- Vendors implement this differently
- Logical sequence of update that belong together
- Brings the database from one state to another
- Know when to *commit*...

## Locking

- Data are locked on row level
- Oracle never locks data to read it
- Writers never block readers
- A writer is only blocked by another writer, on the same row

## Multi-versioning

- Your query gets a consistent answer with respect to the time it started
- Queries are never blocked by updates
- Oracle can serve many sessions that started at different point in time

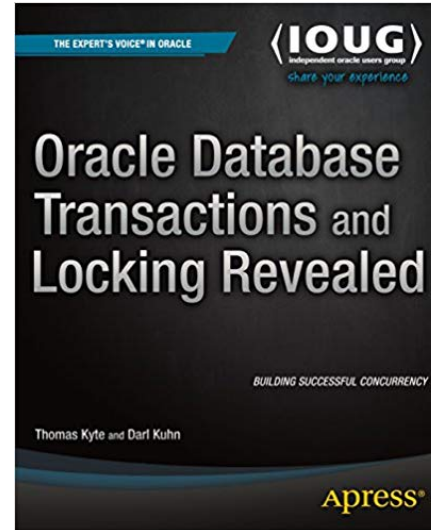


## Redo and Undo

- Is it too much to ask for that developers learn about undo and redo?
- For large applications they need to know about the role they have

## What can you do?

- Measure everything
- Prove where time is spent: outside the database
- Locking problems mean they probably don't understand transactions
- Ask them to read the *Oracle Concepts Guide* and start a conversation



If you lose it

Why not start coding?

## The world of autonomous databases

- The database will heal itself
- Surely it will deal with developers too
- There is a high demand of people who know
  - Security
  - Data science / AI / ML
  - How to brew beer

Cheers