Postgres Conference Europe - 2022

Breaking the sound barrier

Applications at light speed

Hettie Dombrovskaya

Jan Karremans

October 2022

Introductions

Why stay with this lecture?

You do not want to KIWI

You believe you can do more with less

You understand that apps are not magic

You have learned there are no silver bullets

You are smart?

Hettie Dombrovskaya

Database Architect at DRW, Chicago Il

Local Organizer of Chicago PostgreSQL User Group

39 years with databases

Started before relational databases

There is only one Hettie in Postgres!

Jan Karremans / @Johnnyq72

BACKGROUND

25 years of database technology
15 years of consulting
15 years of management
10 years of technology sales
5 years of community advocacy
5 years of international public speaking

ACCREDITATION

Oracle ACE Alumni
EDB Postgres Advanced Server Professional

Why us?

Hettie and Jan together

Common background and understanding

Diverse careers and unique perspectives

Why this topic at PGConf.eu

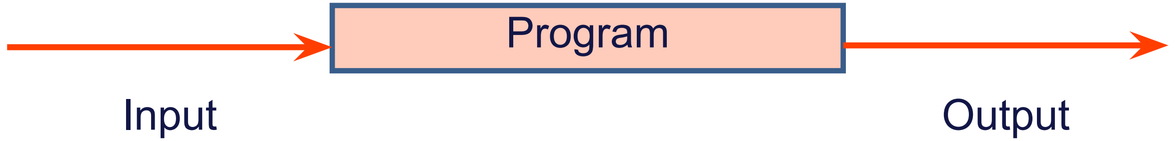
1. Mutual passion to clear up misunderstandings
2. For getting access to the three P's, Postgres is the ideal platform

There is more to discuss than we can hope to achieve!

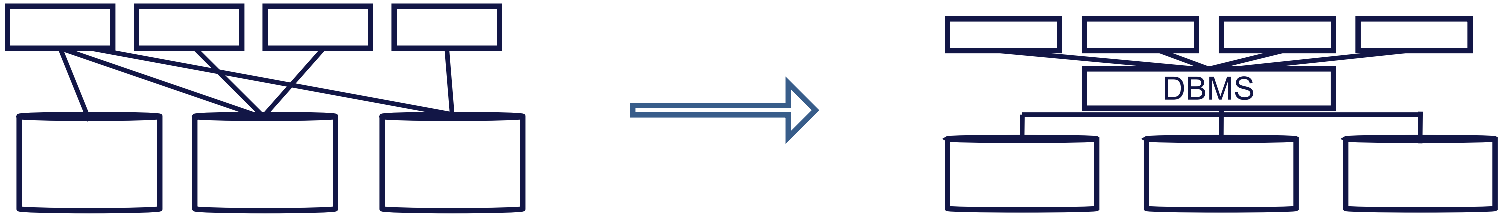
History

IT is a craft

Once upon a time there was a program

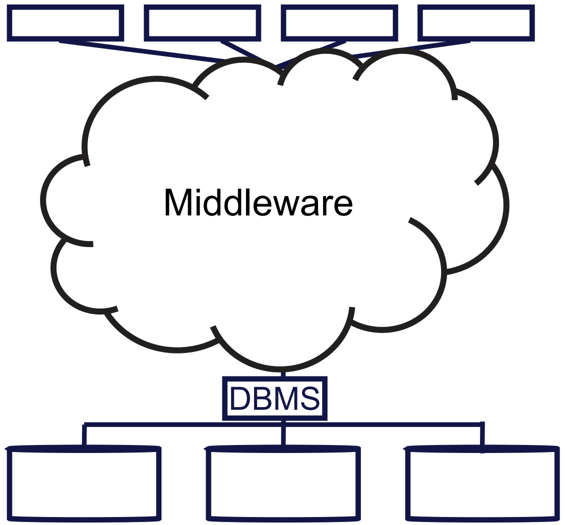


Direct access storage: 70s



DBMS emerged as specialized programs for centralized data management

Then the nineties happened



Business logic - what is it

Wikipedia:
In computer software, business logic is the part of the program that encodes the real-world business rules that determine how data can be created, stored, and changed.

"You end up using more database resources and fewer application server resources in order to achieve the data retrieval results you are looking for."

Database agnostic

Have your application be independent of the database

Possible bias…

- Hoax
- No one switches databases
- You'd be burning money
- Forfeit access to the three P's

"Modern application development"

ORM

Low code

No code

Serverless (okay, that doesn't fit in this list)

You cannot simply put business logic in the middle tier...
...and hope to succeed in scaling your application

Problem

SLOW
Non-idempotent

Database application performance

Everybody wants their application to function efficiently, right?

After all – why choose to use databases in th [illegible] plac [illegible]

Because
The DBMS is [illegible] ***ecializ*** [illegible] ***softw*** [illegible] *e designe* [illegible] *o manag* [illegible] *ata* [illegible] *he* [illegible] ***t efficient way.***

Nevertheless, [illegible] e most [illegible] omm [illegible] mplaint of application developers is ….

Why Nobody is Happy?

Both

imperative programming languages

and declarative query languages

work perfectly

to accomplish the tasks the were designed to accomplish.

The problems start,

when we try to make them

to work together.

EDB™

Is a database actually really slow?

```
SELECT * FROM loans WHERE id=?
```

The query is executed **8,500,000** times during the day, each time taking a few **milliseconds**, with total execution time about **2.5 hours**

Why??

What is object-relational impedance mismatch and why it is bad?

Why should we care?

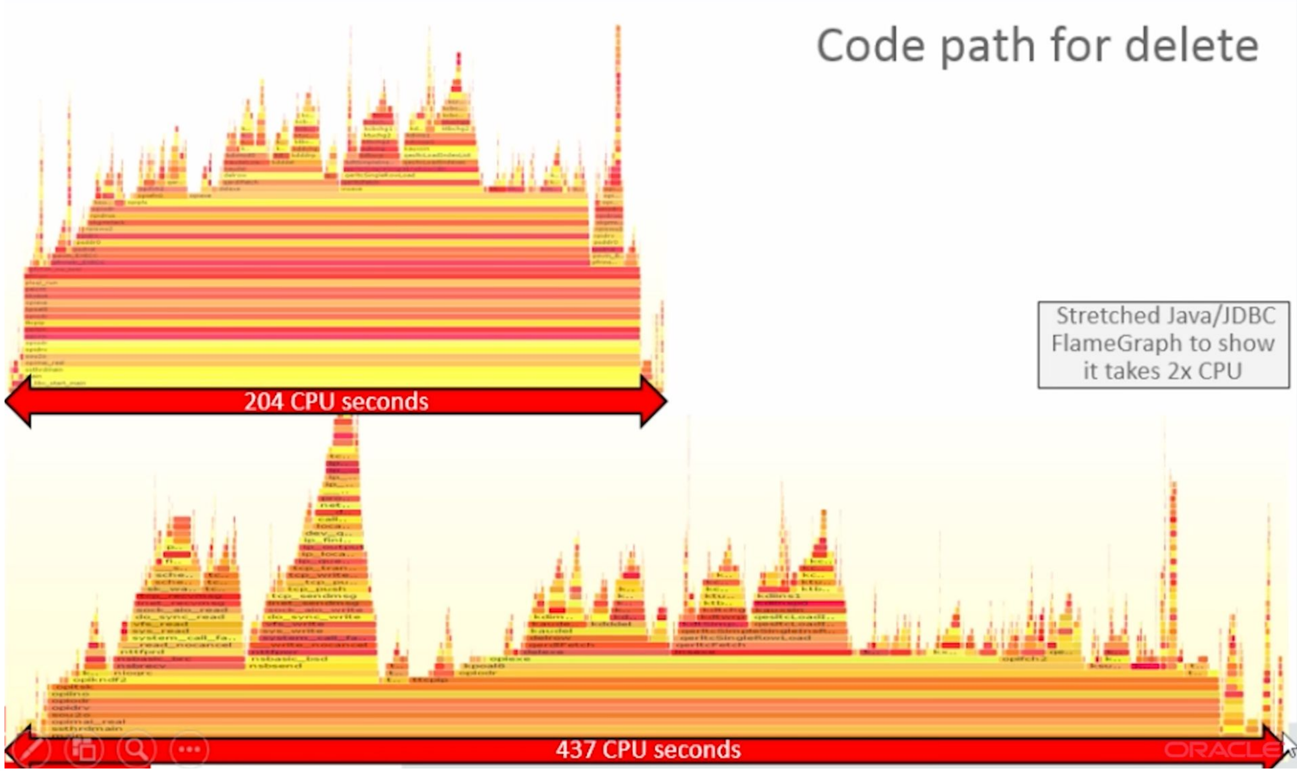
What is ORIM?

When we have Object-Oriented application

- During one screen rendering: objects may belong to different classes with different access methods
- Object-Relational Mapping (ORM): maps a database object to the in-memory application object:
  - solves a problem of abstraction from details of data storing
  - does not provide effective means of manipulation with data sets

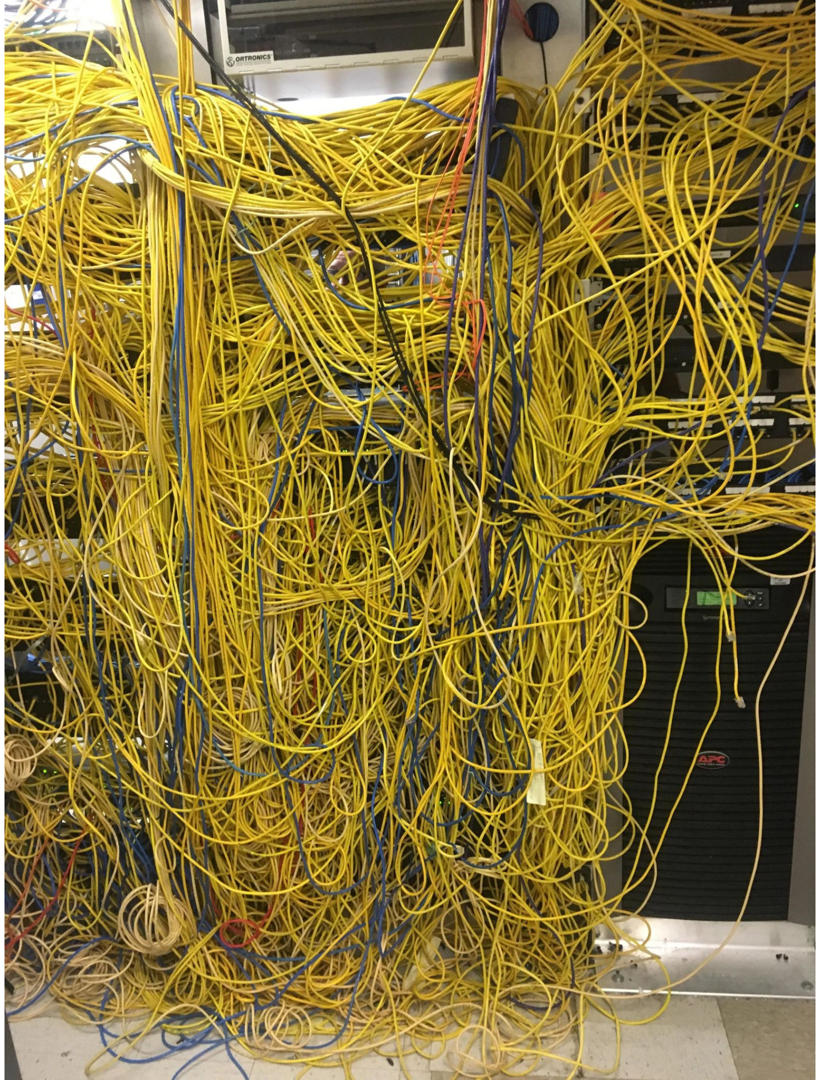
This particular case of impedance mismatch is called *ORIM – object-relational impedance mismatch*

Slow by Slow



Courtesy of ***Toon Koppelaars - RuleGen***
Author of "Applied mathematics for database professionals"

En garde!



Solution

Splitting business logic

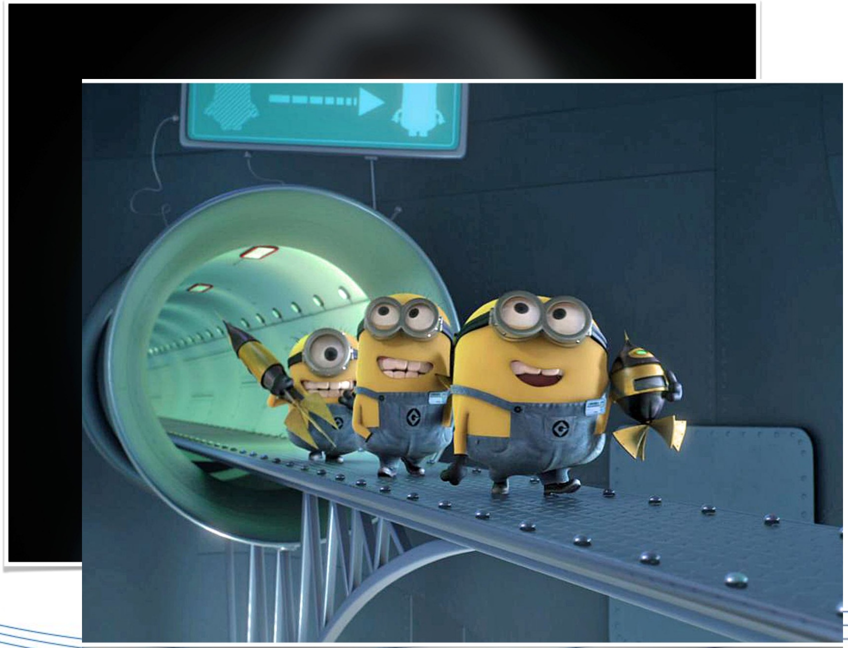
Application business logic

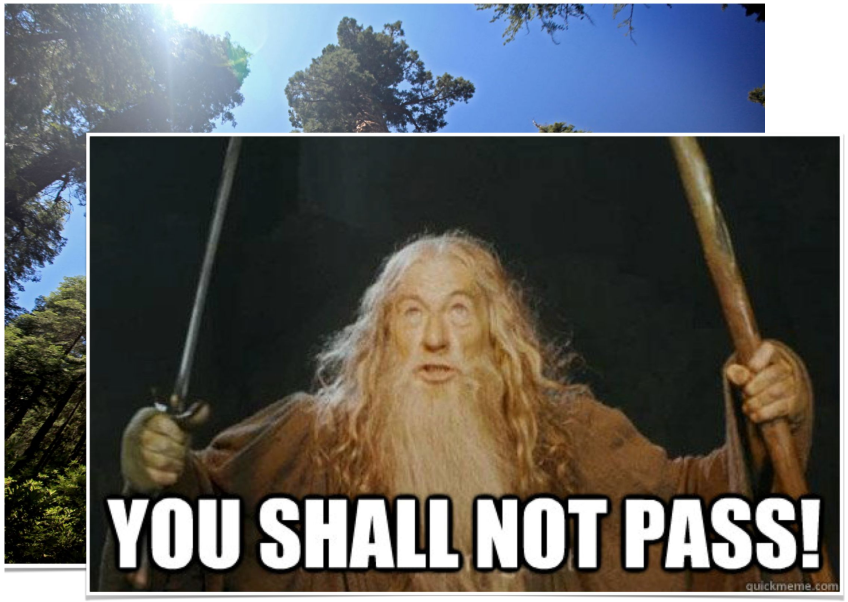
Data business logic

A paradigm shift is pending:

- Switching from focus on applications ('90/'00) to
- Focus on data (now!)

#DBADev





A lesson from my (Jan) past

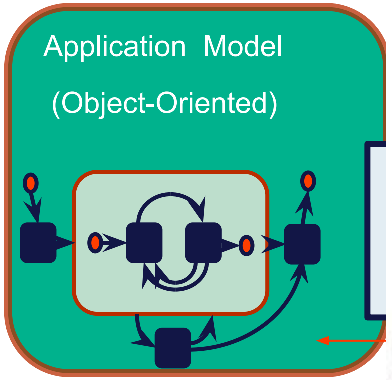
SmartDB

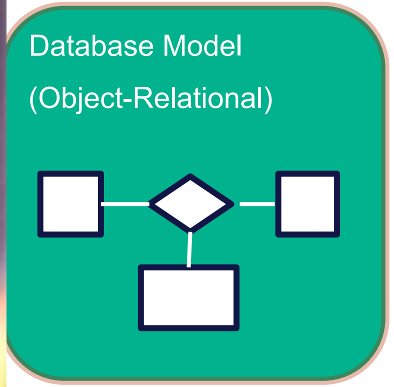
"Slow by slow" vs. set based

Inter-application traffic

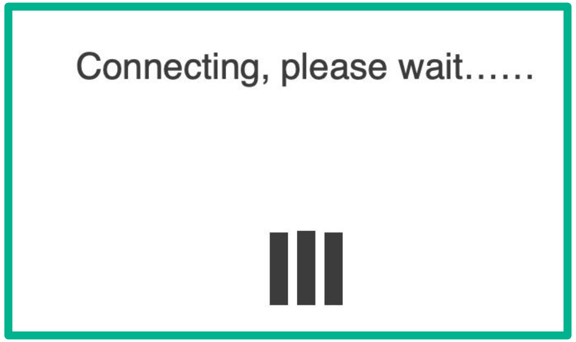
But it is (too) hard to organise

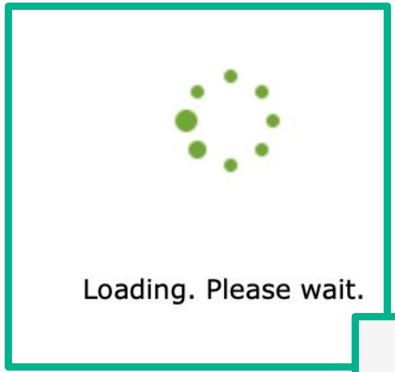
So Hetty came up with something more smooth

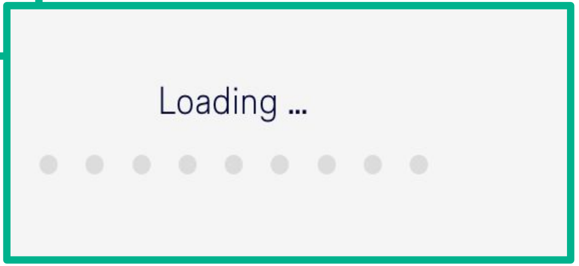

Application Model
(Object-Oriented)


Database Model
(Object-Relational)

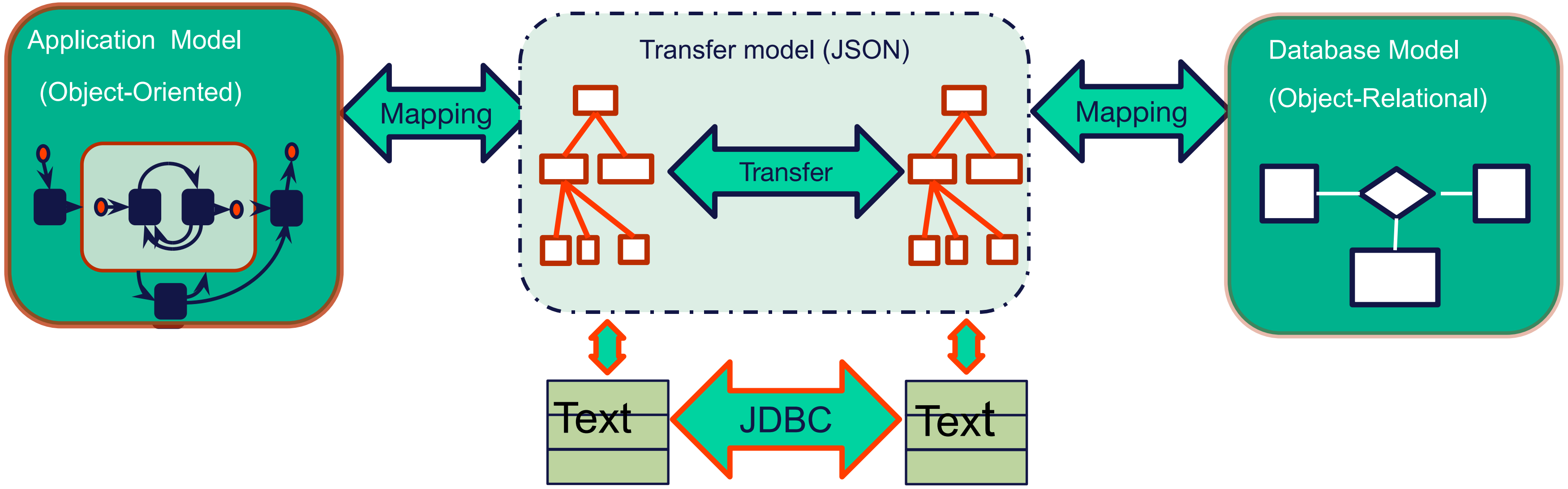
The Result – World-Wide Wait







NORM – No ORM


Application Model
(Object-Oriented)
Mapping
Transfer model (JSON)
Transfer
Mapping
Database Model
(Object-Relational)
Text
JDBC
Text

CONTRACT

- *Both sides (an application and a database) convert internal representations into complex hierarchical object*
- *Contract establishes object structure implemented on both sides*
- *Now, for any application endpoint it takes one database call to transfer data to and from database*

It’s still hard!

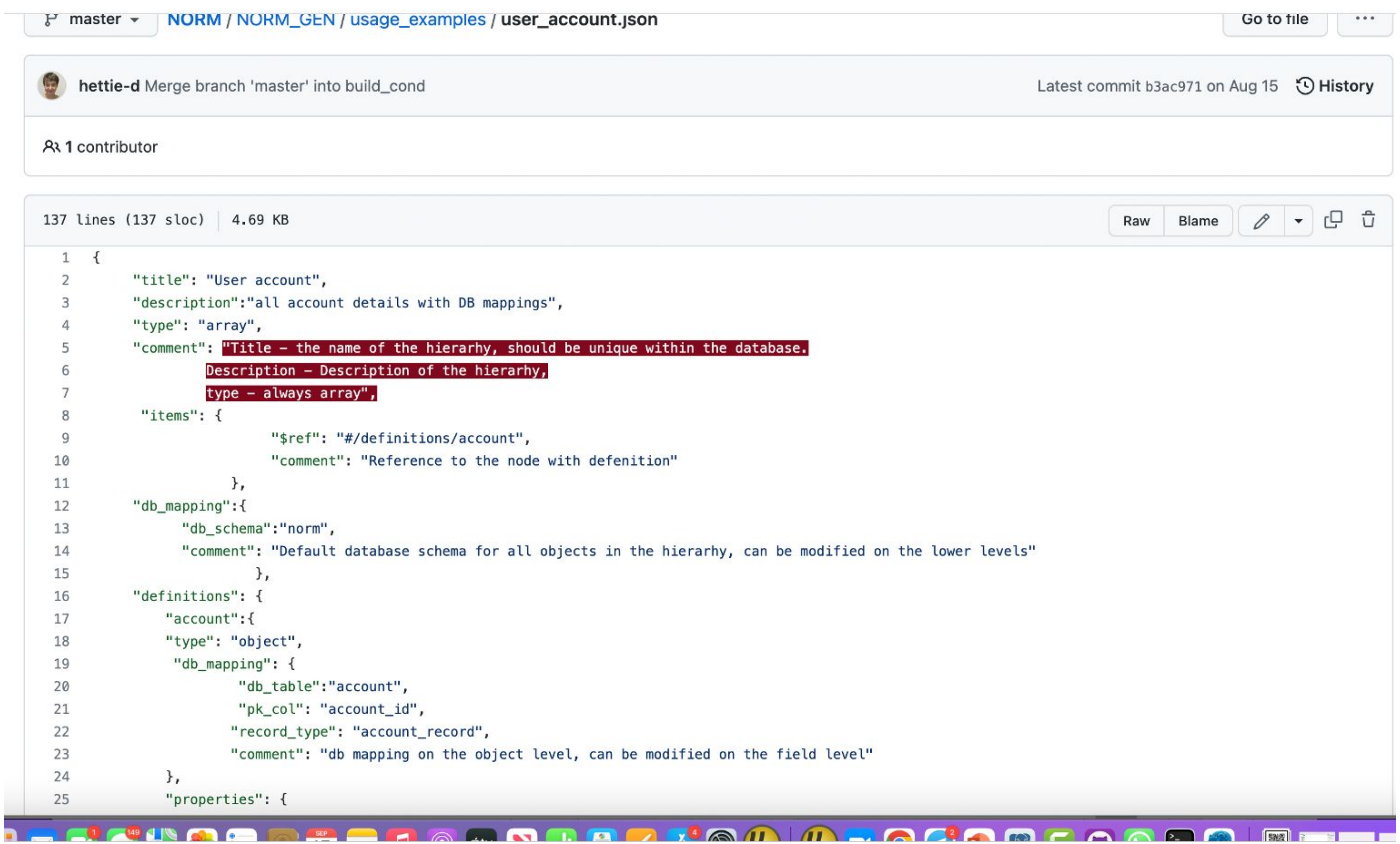
You need a database developer on each project!

Well, now we have some automation!

https://github.com/

hettie-d/NORM/NORM_GEN

EDB™


master
NORM / NORM_GEN / usage_examples / user_account.json
Go to file
hettie-d Merge branch 'master' into build_cond
Latest commit b3ac971 on Aug 15
History
1 contributor
137 lines (137 sloc) | 4.69 KB
Raw
Blame

```
{
      "title": "User account",
      "description":"all account details with DB mappings",
      "type": "array",
      "comment": "Title - the name of the hierarhy, should be unique within the database.
              Description - Description of the hierarhy,
              type - always array",
       "items": {
                        "$ref": "#/definitions/account",
                        "comment": "Reference to the node with defenition"
                   },
      "db_mapping":{
            "db_schema":"norm",
            "comment": "Default database schema for all objects in the hierarhy, can be modified on the lower levels"
                     },
      "definitions": {
          "account":{
          "type": "object",
           "db_mapping": {
                   "db_table":"account",
                   "pk_col": "account_id",
                  "record_type": "account_record",
                  "comment": "db mapping on the object level, can be modified on the field level"
          },
          "properties": {
```

Automation steps

- Parsing JSON schend
- Constructing definitions and applying DB mapping
- Generating UDT
- Generating functions

All without any database developers!

Questions