We Know ORM Is Bad, Now What?

Hettie Dombrovskaya
Senior Cloud consultant
EDB

Swiss PG Day 2022

EDB

NORM: No ORM Framework

Now – Automated!

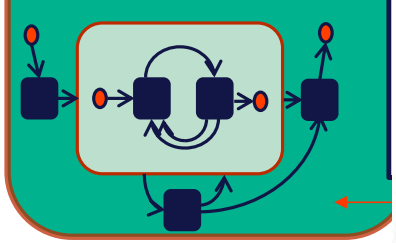
EDB

Quick Quizz

- What's ORM?
  - Object Relational Mapper
- Why developers use ORM?
  - To abstract from database specifics
- Why ORM is ~~bad~~ good?
  - Rapid development
- Why ORM **is bad**?
  - Inefficient data access patterns

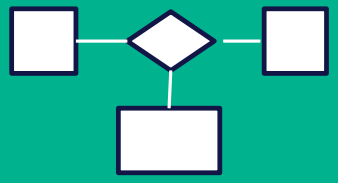
Application Model

(Object-Oriented)

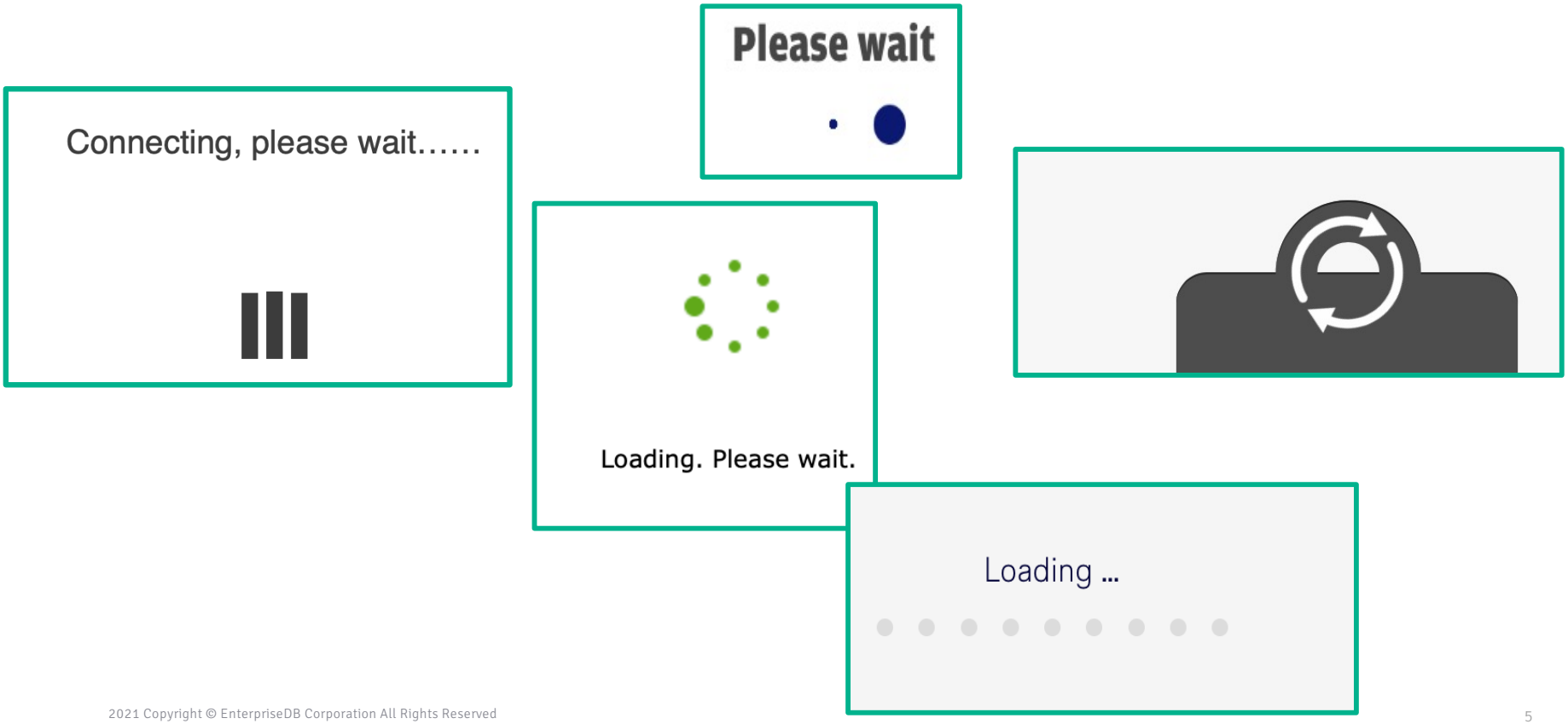


Database Model

(Object-Relational)



The Result – World-Wide Wait



Why Waiting is Bad?

Amazon: 0.1 sec increase response time - 1% sales loss.

50 % visitors abandon the site, which is not loaded within 3 sec

79% visitors will never return again

More Hardware?...

How much RAM you may need?

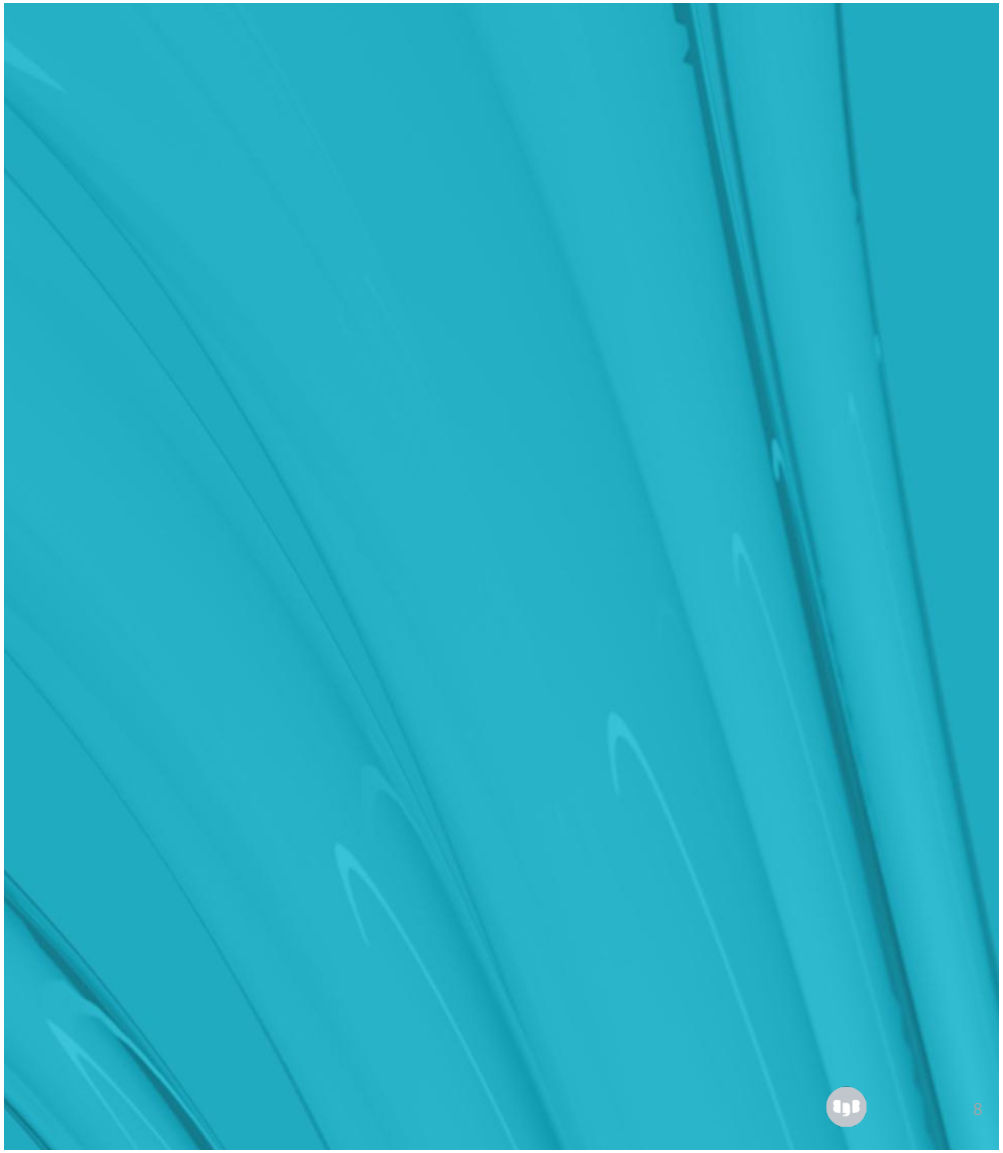
500 GB? 1 TB? 4 TB???

What if we are in the cloud?

- Amazon instances

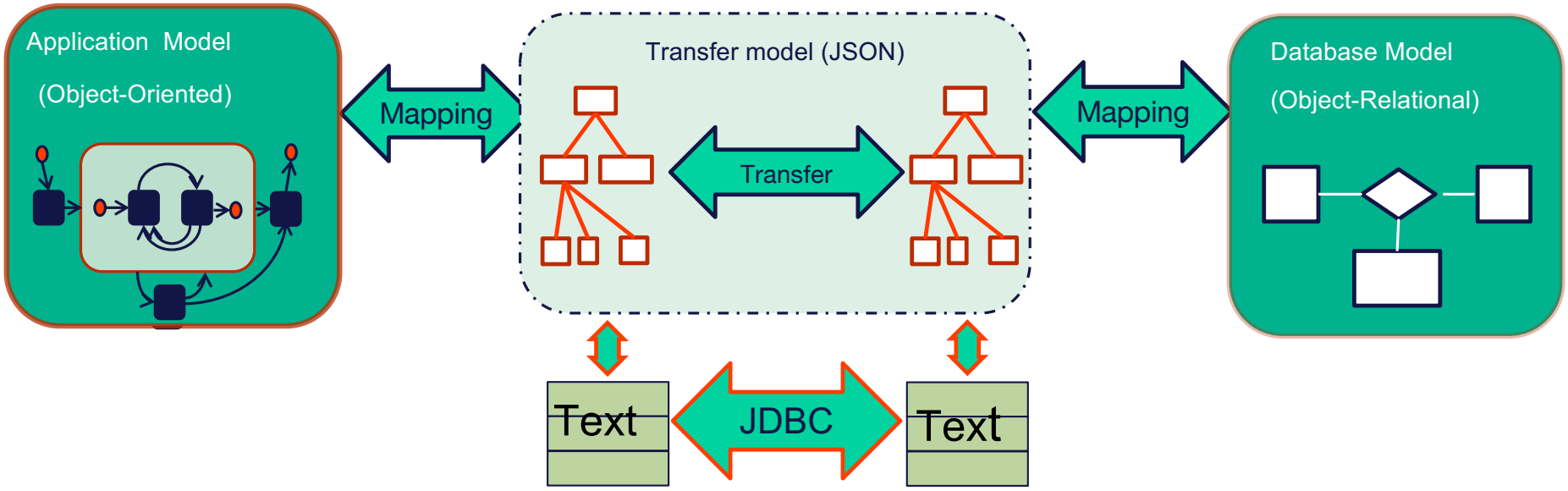
4	17h47m23s	107,350,233 Details	0ms	15s457ms	0ms	d£)INSERT INTO "table_xxxxxxx_v3" ("tbl_xxxxxxx_id", "tbl_xxxxxxx_yid", "xxxxxe_id", "xxxxxd_id", "xxxxxr_id", "pssssss_bbbl_xxxxxxx_id", "pssssss_bbbl_xxxxxxx_yid", "ppppppp_xxxxxe_id", "ppppppp_xxxxxd_id", "ppppppp_xxxxxr_id", "hhhhh_ccccccc_at", "utututututut", "atata_ssssssss") VALUES (?, ? ?, ?, ?, ?, ?, ?, ?, ?, ?) ON CONFLICT ON CONSTRAINT "table_xxxxxxx_v3_pkey" DO UPDATE SET "hhhhh_ccccccc_at" = eseseses."hhhhh_ccccccc_at", "utututututut" = eseseses."utututututut "table_xxxxxxx_v3"."hhhhh_ccccccc_at" <> "eseseses"."hhhhh_ccccccc_at"; Examples User(s) involved

We know what’s the right solution..



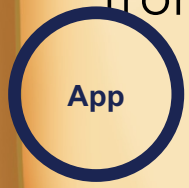
Thinking sets!

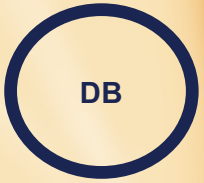
Our Solution - NORM



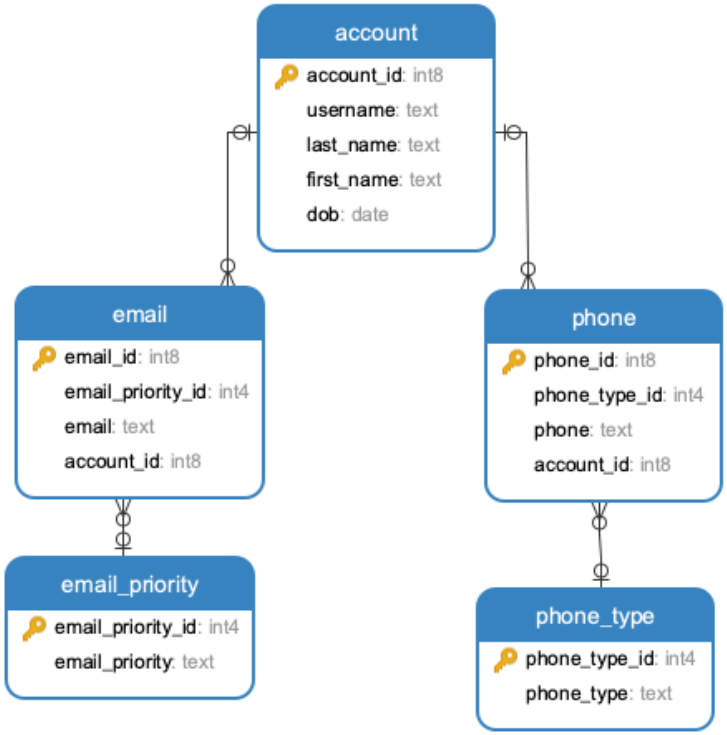
CONTRACT

- 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

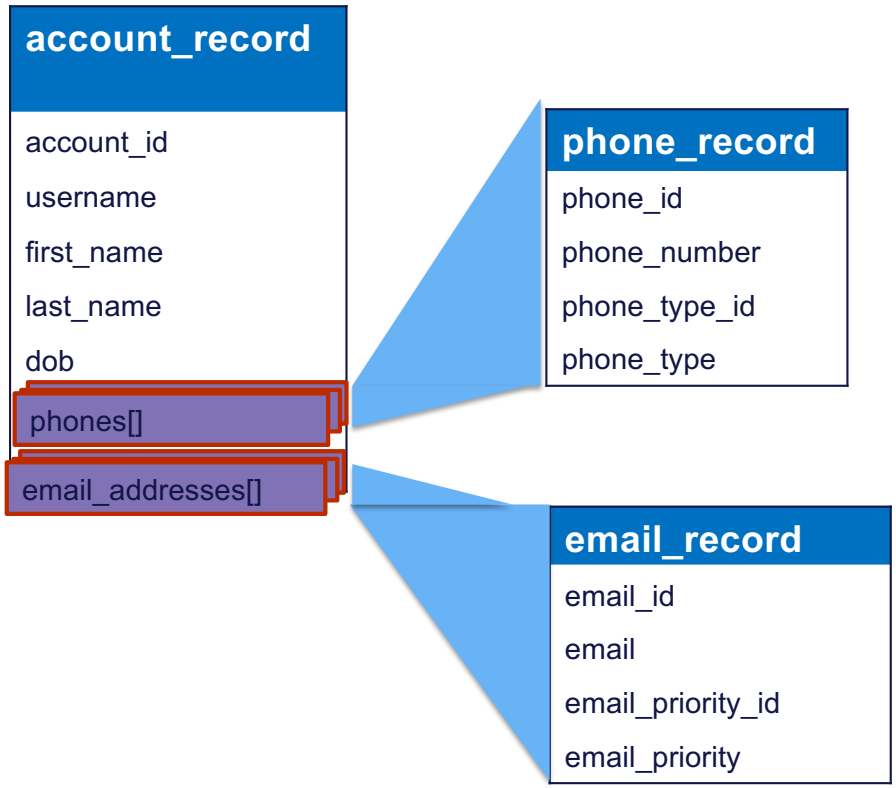




DB Schema



Objects Mapping (Transport Object)



Transfer Object in JSON Format

```
{
”account id”: ”1”,
”username”: ”aliceacct1”,
”last_name”: ”johns”,
“firs_name”:”alice”,
”dob” : ”1996-04-01” ”,
”phones” : [
      { ”phone_id”: 1,
        ”phone_number” : ”2021234567” ,
        ”phone_type_id”:”2”, ”phone type”: ”cell” }
      ],
”email addresses”:[
     {”email_address_id”:1 ,
      ”email address”:alicejons@gmail.com,
      “email_priority_id”: “1”,
      “email_priority”:”primary” }
      ]}
```


Refresher

Why not storing JSON?!

- Single hierarchy
- The search is slower

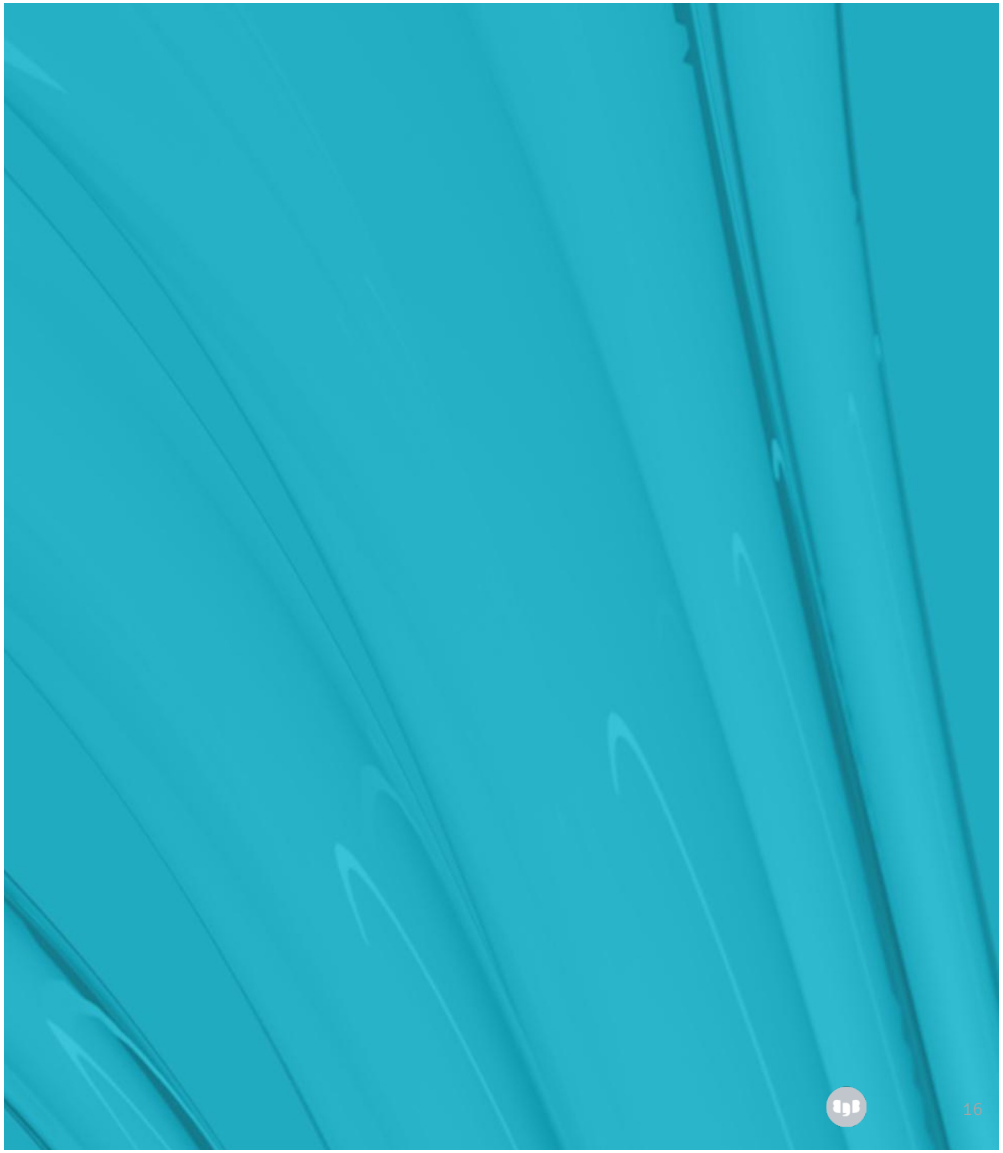
Why not returning JSON?!

- By using JSON as a return type for functions we are losing the strong type dependencies

  That's why we organize functions in "packages":

  we want to ensure that all functions from one "package" return the same object

Things look good for a while...



master NORM / sql / account_pkg.sql Go to file ...

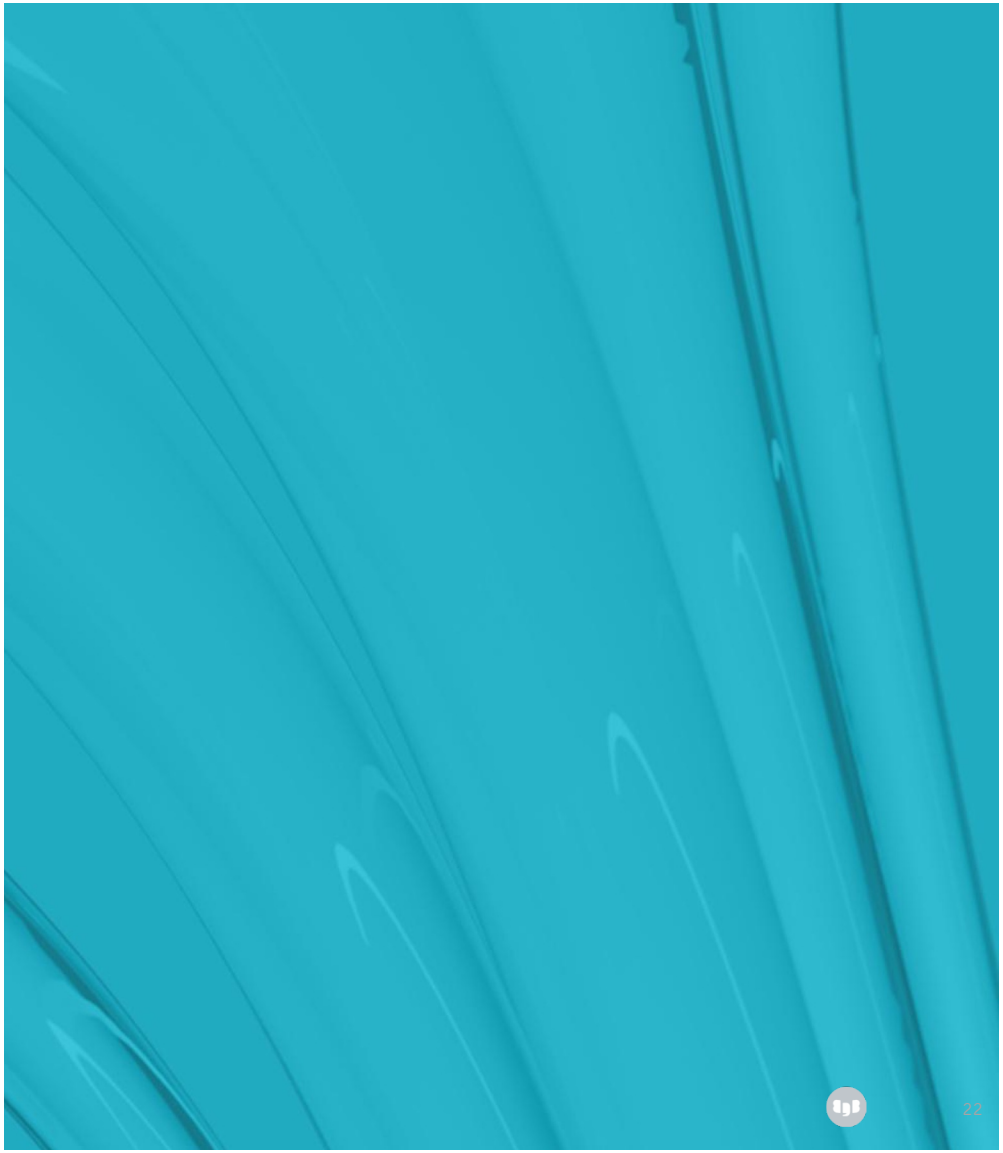
hettie-d account_update function and more usage examples are added Latest commit a6983ff on Apr 30, 2020 History

1 contributor

341 lines (316 sloc) | 10.5 KB Raw Blame

```
1  set search_path to norm;
2
3  drop type if exists phone_record cascade ;
4  create type phone_record as (
5                   phone_id  bigint,
6                   phone_number  text,
7                   phone_type_id int,
8                   phone_type   text);
9  drop type if exists email_record cascade;
10 create type email_record as(
11                  email_id bigint,
12                      email  text,
13                  email_priority_id int,
14                  email_priority text);
15  drop type if exists account_record cascade;
16  create type account_record as (
17                account_id  bigint,
18                username text,
19                first_name text,
20                last_name text,
21                dob date,
```

Any Problem With That?



Yes!

“We need a database developer to do this!”

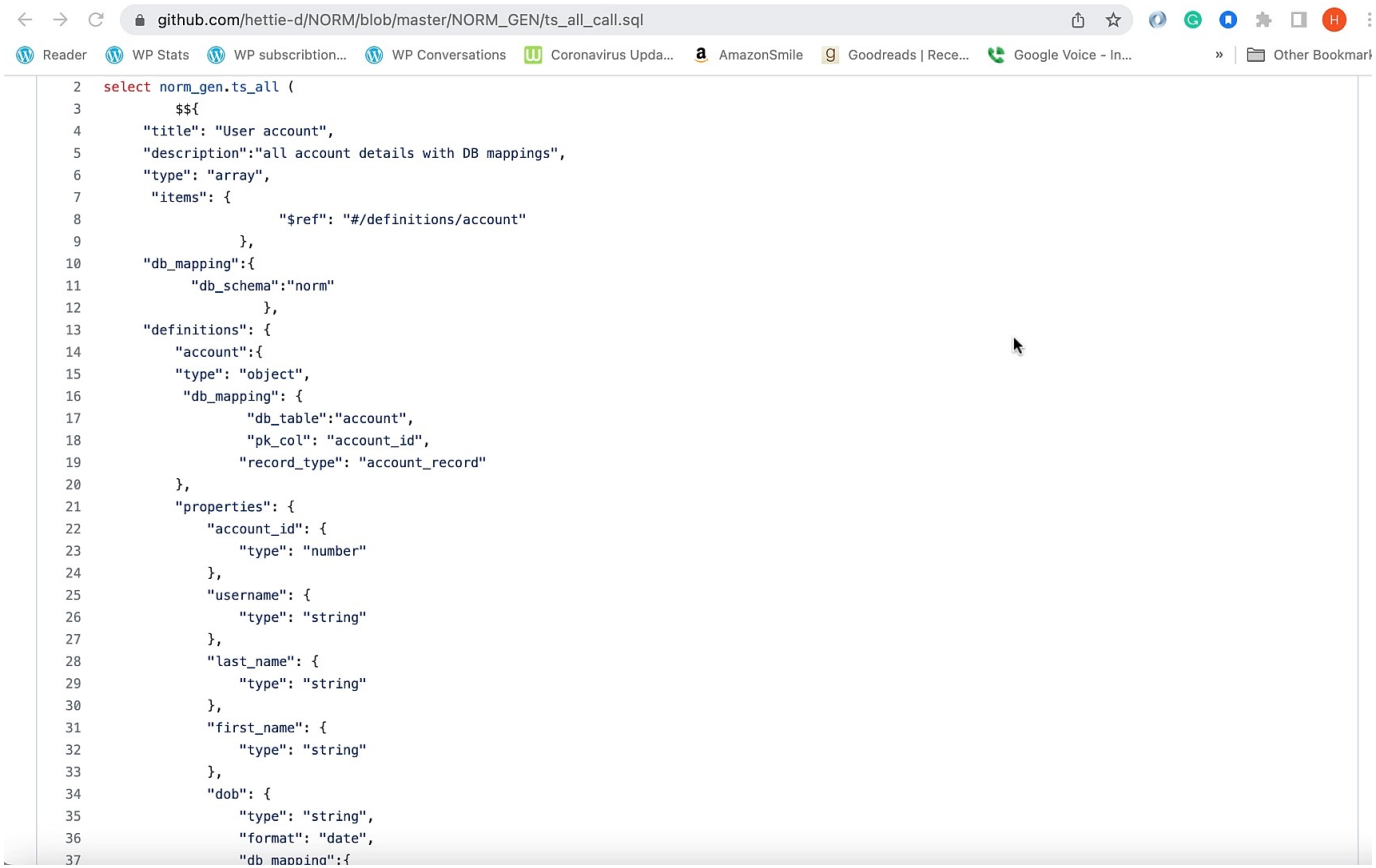
With an ORM, you do not need to do anything special!

Let’s Go Back to JSON:

```
{
”account id”: ”1”,
”username”: ”aliceacct1”,
”last_name”: ”johns”,
“firs_name”:”alice”,
”dob” : ”1996-04-01” ”,
”phones” : [
      { ”phone_id”: 1,
        ”phone_number” : ”2021234567” ,
        ”phone_type_id”:”2”, ”phone type”: ”cell” }
      ],
”email addresses”:[
     {”email_address_id”:1 ,
      ”email address”:alicejons@gmail.com,
      “email_priority_id”: “1”,
      “email_priority”:”primary” }
      ]}
```


That’s great, but that’s not a contract!

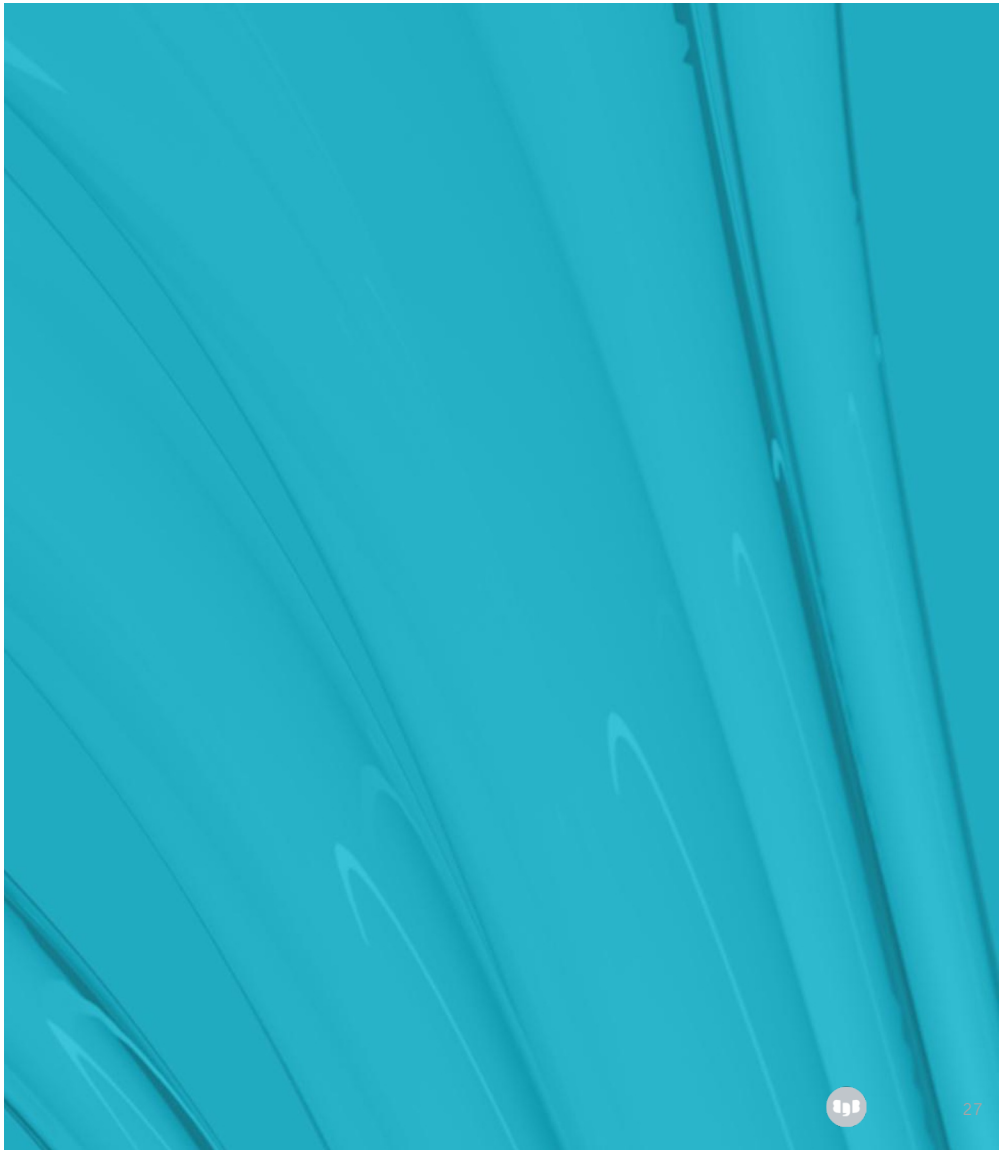
Using JSON Schema to Represent the Contract




```
select norm_gen.ts_all (
        $${
    "title": "User account",
    "description":"all account details with DB mappings",
    "type": "array",
     "items": {
                    "$ref": "#/definitions/account"
                },
    "db_mapping":{
         "db_schema":"norm"
                },
    "definitions": {
        "account":{
        "type": "object",
         "db_mapping": {
                "db_table":"account",
                "pk_col": "account_id",
               "record_type": "account_record"
        },
        "properties": {
            "account_id": {
                "type": "number"
            },
            "username": {
                "type": "string"
            },
            "last_name": {
                "type": "string"
            },
            "first_name": {
                "type": "string"
            },
            "dob": {
                "type": "string",
                "format": "date",
                "db_mapping":{
```

Automating functions generation

https://github.com/hettie-d/NORM/NORM_GEN



Metadata Tables

create_norm_gen_tables.sql

transfer_schema

transfer_schema_object

transfer_schema_key

All of them are automatically populated by the “contract analyzer”

Parsing JSON Schema

process_schema_pkg.sql

save_transfer_schema (json, Boolean) – initial parsing, saving json, assigning the schema_id

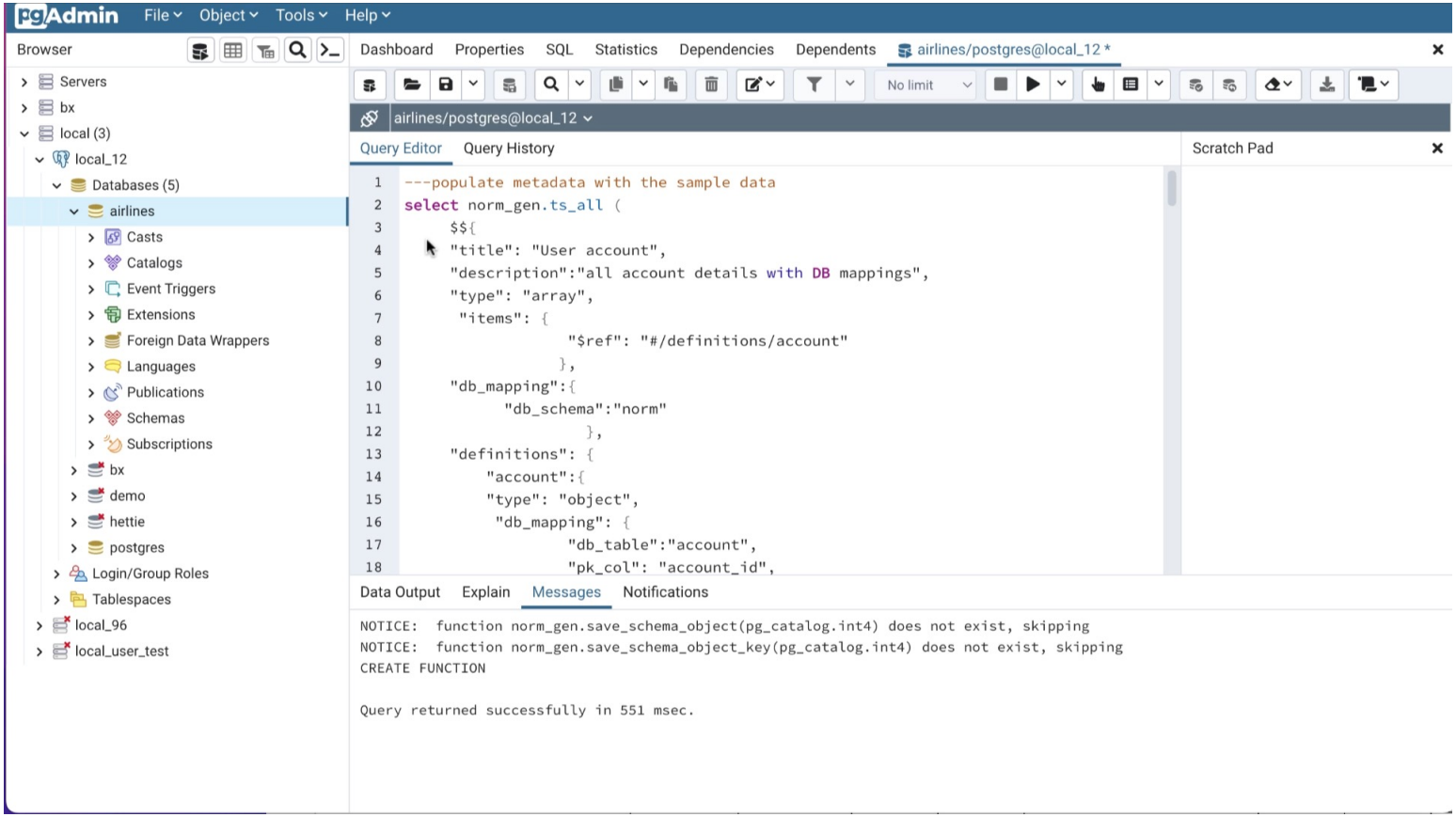
save_schema_object (int) - parse and save details of transfer objects for a given schema

save_schema_key (int) – parse and save details of individual keys, updates parent references for objects

update_db_type (int) – sets up all key types which are not explicitly defined

ts_all.sql runs them all

Example



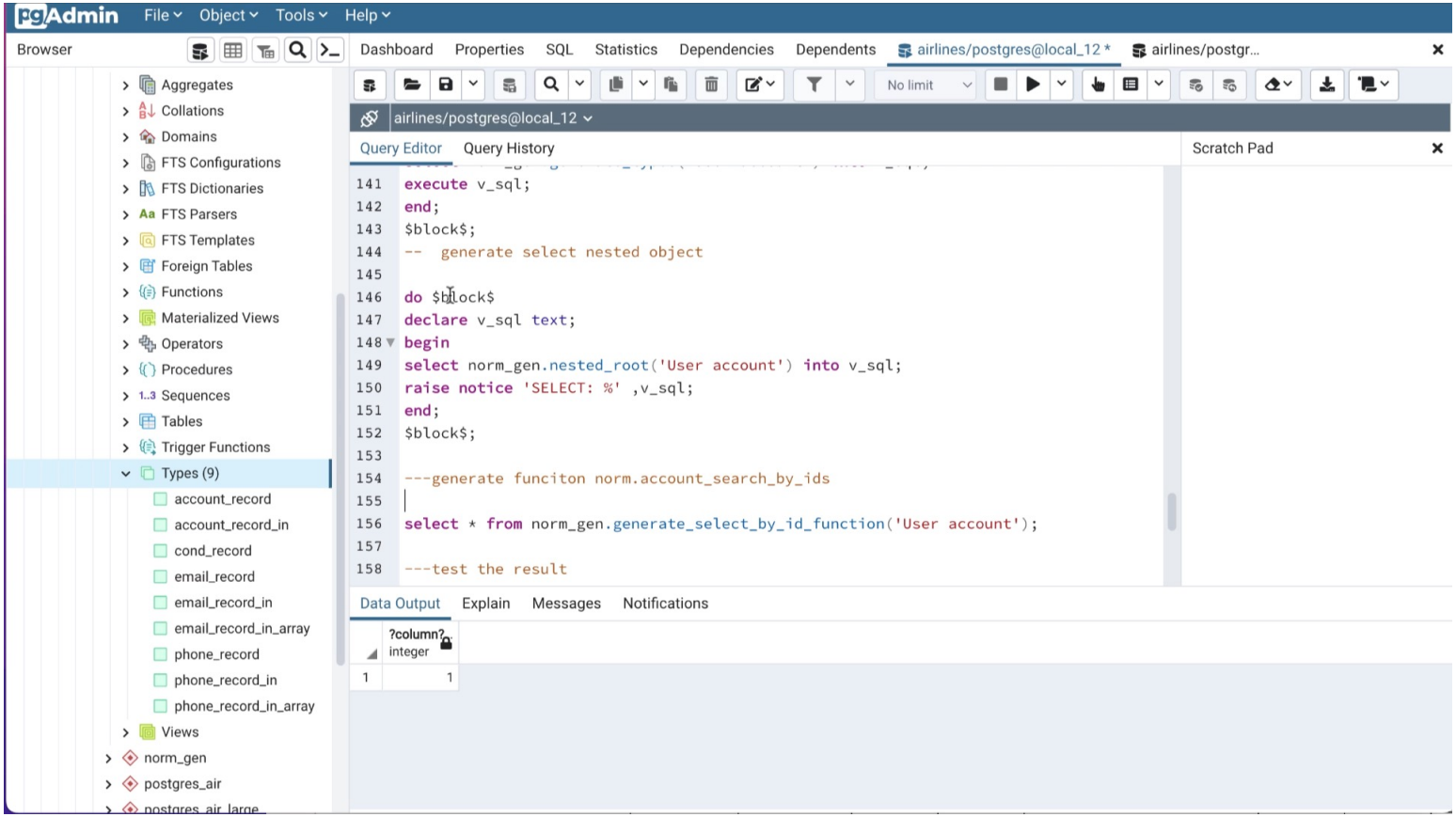
Next steps

Now, all functions can be generated, no database developer needed!

We generate

- SELECT of the nested object for a given hierarchy
- SEARCH by ID
- GENERIC SEARCH

Example



Universal NORM-TO-DB Converter

This function is custom-built, and also can be used as a template.
Requires additional “in” types.

insert:

```
select * from norm.account_to_db ($$[{
  "username":"johnsmithaccount",
  "first_name":"john",
  "last_name":"smith",
  "dob":"1991-04-01",
   "phones":[{"phone_number":"3123334556", "phone_type_id":"1"}]
 }]
$$::json)
```

update:

```
select * from norm.account_to_db ($$[{
   "account_id":1,
   "username":"aliceacct2"}]
 $$::json
)
```


Universal NORM-TO-DB Converter

This function is also custom-built, and also can be used as a template.

update embedded objects:

```
select * from account_to_db($$[{
  "account_id":"3",
  "emails":[{"email":"new.email@hotmail.com",
                        "email_priority_id":"1"}]
   }]
   $$::json)
```

delete:

```
select * from account_to_db($$[{
  "account_id":"1",
  "phones":[{"phone_id":"2", "command":"delete"}]
  }]
 $$::json)
```


Additional Considerations and Summary

How this is different from other ORMs?

- "Thinking sets"
- Mapping complex objects
- Doing DB work inside DB

Other programming languages

- Details may vary (no need for converting to text, etc.)
- Active Record requires additional work, because it reads the DB catalog directly

Storing JSON vs. building JSON

- Search
- Multiple hierarchies

Questions?

EDB