# PostgreSQL Object Oriented Concepts Applied to Real Business Problems

Achilleas Mantzios

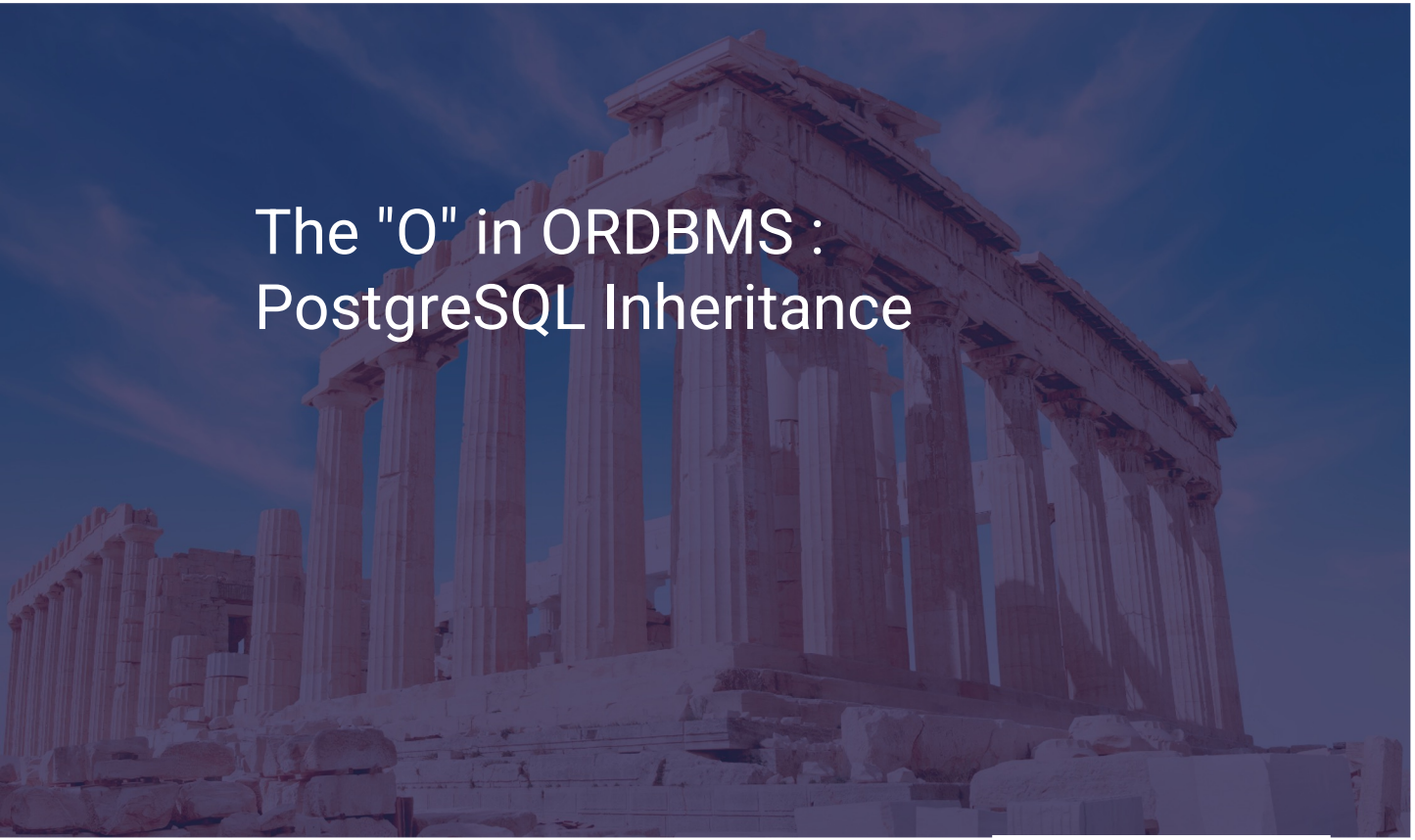
DBA, Team leader, Dynacom Tankers

Athens, Greece 2023

PERCONA UNIVERSITY

In partnership with FerretDB

# The "O" in ORDBMS : PostgreSQL Inheritance



Athens, Greece 2023

PERCONA UNIVERSITY

In partnership with FerretDB

# Inheritance Basics

```
CREATE TABLE vehicle (
     id SERIAL PRIMARY KEY,
     name text NOT NULL,
     length_m real,
     speed_kmh real
);
```

```
CREATE TABLE car (
     fuel varchar(20) NOT NULL,
     awd boolean NOT NULL DEFAULT 'f'
) INHERITS (vehicle);
```

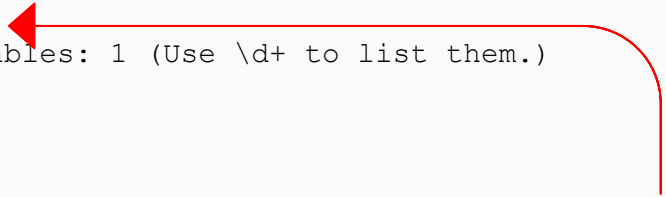
```
CREATE TABLE rally_car (
     boost_pressure real NOT NULL,
     category TEXT NOT NULL CHECK (category IN ('WRC','DRAGSTER','F1'))
) INHERITS (car);
```

# Inheritance Basics

```
percona_univ=# \d vehicle
                              Table "public.vehicle"
 Column    |  Type   | Collation | Nullable |               Default
-----------+---------+-----------+----------+-------------------------------------
id         | integer |           | not null | nextval('vehicle_id_seq'::regclass)
name       | text    |           | not null |
length_m   | real    |           |          |
speed_kmh | real    |           |          |
Indexes:
   "vehicle_pkey" PRIMARY KEY, btree (id)
Number of child tables: 1 (Use \d+ to list them.)
```

# Inheritance Basics

```
percona_univ=# \d car
                                    Table "public.car"
 Column    |          Type           | Collation | Nullable |                Default
-----------+-------------------------+-----------+----------+---------------------------------------
id         | integer                 |           | not null | nextval('vehicle_id_seq'::regclass)
name       | text                    |           | not null |
length_m   | real                    |           |          |
speed_kmh  | real                    |           |          |
fuel       | character varying(20)   |           | not null |
awd        | boolean                 |           | not null | false
Inherits: vehicle
Number of child tables: 1 (Use \d+ to list them.)
```



Where is the UNIQUE Index on id ?

# Inheritance Basics

```
percona_univ=# \d rally_car
                                    Table "public.rally_car"
    Column     |          Type          | Collation | Nullable |                Default
---------------+------------------------+-----------+----------+---------------------------------------
id             | integer                |           | not null | nextval('vehicle_id_seq'::regclass)
name           | text                   |           | not null |
length_m       | real                   |           |          |
speed_kmh      | real                   |           |          |
fuel           | character varying(20)  |           | not null |
awd            | boolean                |           | not null | false
boost_pressure | real                   |           | not null |
category       | text                   |           | not null |
Check constraints:
   "rally_car_category_check" CHECK (category = ANY (ARRAY['WRC'::text, 'DRAGSTER'::text, 'F1'::text]))
Inherits: car
```

As with car, the UNIQUE key is missing. Those should be created independently.

# Inheritance Basics

```
percona_univ=# INSERT INTO rally_car(name,length_m,speed_kmh,fuel,awd,boost_pressure,category) VALUES
('Subaru Impreza',4.4,250,'benzin','true',2.0,'WRC');
INSERT 0 1
percona_univ=# select * from rally_car ;
id |      name      | length_m | speed_kmh |  fuel   | awd | boost_pressure | category
---+----------------+----------+-----------+---------+-----+----------------+----------
 1 | Subaru Impreza |      4.4 |       250 | benzin | t   |              2 | WRC
(1 row)

percona_univ=# select * from car ;
id |      name      | length_m | speed_kmh |  fuel   | awd
---+----------------+----------+-----------+---------+-----
 1 | Subaru Impreza |      4.4 |       250 | benzin | t
(1 row)

percona_univ=# select * from vehicle;
id |      name      | length_m | speed_kmh
---+----------------+----------+-----------
 1 | Subaru Impreza |      4.4 |       250
(1 row)

percona_univ=# select tableoid::regclass,* from vehicle;
tableoid  | id |      name      | length_m | speed_kmh
----------+----+----------------+----------+-----------
rally_car |  1 | Subaru Impreza |      4.4 |       250
(1 row)
```


# Inheritance Basics

```
percona_univ=# ALTER TABLE car ADD vin CHAR(17) NOT NULL UNIQUE DEFAULT 'xxxxxxxxxxxxxxxx';
ALTER TABLE
percona_univ=# \d car
                                    Table "public.car"
 Column    |          Type          | Collation | Nullable |                Default
-----------+------------------------+-----------+----------+---------------------------------------
id         | integer                |           | not null | nextval('vehicle_id_seq'::regclass)
name       | text                   |           | not null |
length_m   | real                   |           |          |
speed_kmh  | real                   |           |          |
fuel       | character varying(20)  |           | not null |
awd        | boolean                |           | not null | false
vin        | character(17)          |           | not null | 'xxxxxxxxxxxxxxxx'::bpchar
Indexes:
   "car_vin_key" UNIQUE CONSTRAINT, btree (vin)
Inherits: vehicle
Number of child tables: 1 (Use \d+ to list them.)
```

# Inheritance Basics

```
percona_univ=# \d rally_car
                                  Table "public.rally_car"
    Column     |         Type          | Collation | Nullable |                Default
---------------+-----------------------+-----------+----------+---------------------------------------
id             | integer               |           | not null | nextval('vehicle_id_seq'::regclass)
name           | text                  |           | not null |
length_m       | real                  |           |          |
speed_kmh      | real                  |           |          |
fuel           | character varying(20) |           | not null |
awd            | boolean               |           | not null | false
boost_pressure | real                  |           | not null |
category       | text                  |           | not null |
vin            | character(17)         |           | not null | 'xxxxxxxxxxxxxxxxx'::bpchar
Check constraints:
   "rally_car_category_check" CHECK (category = ANY (ARRAY['WRC'::text, 'DRAGSTER'::text, 'F1'::text]))
Inherits: car

percona_univ=# ALTER TABLE rally_car ADD CONSTRAINT rally_car_vin_key UNIQUE(vin);
ALTER TABLE
percona_univ=#
```

# Inheritance Basics

```
percona_univ=# INSERT INTO rally_car(name,length_m,speed_kmh,fuel,awd,boost_pressure,category,vin) VALUES
('Subaru Impreza',4.4,250,'benzin','true',2.0,'WRC','JF1SH5LW49G009417');
INSERT 0 1
percona_univ=# select tableoid::regclass,* from car;
 tableoid   | id |      name      | length_m | speed_kmh |  fuel   | awd |        vin
------------+----+----------------+----------+-----------+---------+-----+-------------------
 rally_car  |  1 | Subaru Impreza |      4.4 |       250 | benzin | t   | xxxxxxxxxxxxxxxxx
 rally_car  |  3 | Subaru Impreza |      4.4 |       250 | benzin | t   | JF1SH5LW49G009417
(2 rows)

percona_univ=# UPDATE car SET vin='JF1SH5LW49G009417' WHERE id = 1;
ERROR:  duplicate key value violates unique constraint "rally_car_vin_key"
DETAIL:  Key (vin)=(JF1SH5LW49G009417) already exists.
percona_univ=#
```

# Inheritance Basics

```
percona_univ=# INSERT INTO car(name,length_m,speed_kmh,fuel,awd,vin)
VALUES ('Chevrolet Captiva ',4.3,170,'benzin','false','1GNEK13ZX3R298984');
INSERT 0 1
percona_univ=# select tableoid::regclass,* from vehicle;
tableoid  | id |        name        | length_m | speed_kmh
----------+----+--------------------+----------+-----------
car       |  5 | Chevrolet Captiva  |      4.3 |       170
rally_car |  1 | Subaru Impreza     |      4.4 |       250
rally_car |  3 | Subaru Impreza     |      4.4 |       250
(3 rows)

percona_univ=# select tableoid::regclass,* from car;
tableoid  | id |        name        | length_m | speed_kmh |  fuel   | awd |        vin
----------+----+--------------------+----------+-----------+---------+-----+-------------------
car       |  5 | Chevrolet Captiva  |      4.3 |       170 | benzin | f   | 1GNEK13ZX3R298984
rally_car |  1 | Subaru Impreza     |      4.4 |       250 | benzin | t   | xxxxxxxxxxxxxxxx
rally_car |  3 | Subaru Impreza     |      4.4 |       250 | benzin | t   | JF1SH5LW49G009417
(3 rows)
percona_univ=# select * from rally_car;
id |      name      | length_m | speed_kmh |  fuel   | awd | boost_pressure | category |        vin
---+----------------+----------+-----------+---------+-----+----------------+----------+-----------------
 1 | Subaru Impreza |      4.4 |       250 | benzin | t   |              2 | WRC      | xxxxxxxxxxxxxxxx
 3 | Subaru Impreza |      4.4 |       250 | benzin | t   |              2 | WRC      | JF1SH5LW49G009417
(2 rows)
```

# Inheritance Basics

Does not have to have a tree structure like e.g. subclasess in Java. Could be a DAG (Directed Acyclic Graph). Think Java Interfaces.

```
percona_univ=# CREATE table restaurant (
id SERIAL PRIMARY KEY,
name text NOT NULL UNIQUE,
restaurant_class VARCHAR NOT NULL CHECK (restaurant_class in ('Lux','Common','Street Food'))
);
CREATE TABLE
percona_univ=# CREATE TABLE caravan_canteen (
id INTEGER NOT NULL DEFAULT nextval('vehicle_id_seq'::regclass),
food_type VARCHAR(50) NOT NULL CHECK (food_type IN ('Asian','Greek','Balkan','British','TexMex','Other'))
)
INHERITS (restaurant, vehicle);
NOTICE:  merging multiple inherited definitions of column "id"
NOTICE:  merging multiple inherited definitions of column "name"
NOTICE:  merging column "id" with inherited definition
CREATE TABLE
percona_univ=#
```

# Inheritance Basics

```
percona_univ=# \d caravan_canteen
                                  Table "public.caravan_canteen"
     Column      |         Type          | Collation | Nullable |                Default
-----------------+-----------------------+-----------+----------+---------------------------------------
id               | integer               |           | not null | nextval('vehicle_id_seq'::regclass)
name             | text                  |           | not null |
restaurant_class | character varying     |           | not null |
length_m         | real                  |           |          |
speed_kmh        | real                  |           |          |
food_type        | character varying(50) |           | not null |
Check constraints:
   "caravan_canteen_food_type_check" CHECK (food_type::text = ANY (ARRAY['Asian'::character varying,
'Greek'::character varying, 'Balkan'::character varying, 'British'::character varying,
'TexMex'::character varying, 'Other'::
character varying]::text[]))
   "restaurant_restaurant_class_check" CHECK (restaurant_class::text = ANY (ARRAY['Lux'::character
varying, 'Common'::character varying, 'Street Food'::character varying]::text[]))
Inherits: restaurant,
          vehicle
```

# Inheritance Basics

```
percona_univ=# INSERT INTO caravan_canteen (name,restaurant_class,length_m, speed_kmh,food_type)
VALUES('O gyros tis plateias','Street Food',6,140,'Greek');
INSERT 0 1
percona_univ=# select tableoid::regclass,* from restaurant;
   tableoid     | id |         name          | restaurant_class
----------------+----+-----------------------+------------------
caravan_canteen |  7 | O gyros tis plateias | Street Food
(1 row)

percona_univ=# select tableoid::regclass,* from vehicle;
   tableoid     | id |         name          | length_m | speed_kmh
----------------+----+-----------------------+----------+-----------
car             |  5 | Chevrolet Captiva     |      4.3 |       170
caravan_canteen |  7 | O gyros tis plateias |        6 |       140
rally_car       |  1 | Subaru Impreza        |      4.4 |       250
rally_car       |  3 | Subaru Impreza        |      4.4 |       250
(4 rows)

percona_univ=# select tableoid::regclass,* from caravan_canteen;
   tableoid     | id |         name          | restaurant_class | length_m | speed_kmh | food_type
----------------+----+-----------------------+------------------+----------+-----------+-----------
caravan_canteen |  7 | O gyros tis plateias | Street Food      |        6 |       140 | Greek
(1 row)
```

# Caveats

- PostgreSQL will most probably do the right thing, but be careful about the semantics of your DDL, DML
  - SELECTs, UPDATEs, DELETEs can work on the whole hierarchy
    - use ONLY keyword to SELECT, UPDATE, DELETE strictly on the physical table
  - INSERTs work on the specified physical table only
    - (use rules for the fancy stuff)
  - For the rest DML and DDL consult the docs
- Indexes are NOT global along the whole hierarchy
  - this means that Uniqueness cannot be inherited
  - this also means that other tables cannot have Foreign Keys to the whole hierarchy
- Foreign Keys are not inherited
  - all tables in the hierarchy must explicitly declare the FK to another table
  - as seen above, other tables cannot have Foreign Keys to the whole hierarchy
    - no nice workaround for this one, could use triggers (should cover both ways, i.e. in both sides of the FK), measure performance.

# Uses of inheritance

- Partitioning
- Multi-tenancy
- Data Departmentalization / Consolidation
- Data Segregation

Alternatives :
- views
- physical partitioning and reconsolidation in a separate central warehouse
- etc

The above suffer from :
- they are bandaids not native to the DB technology
- need serious implementation
- source code adaptation (not always easy with closed source apps, ERPs, CRMs, etc)
- read-only on the top level (e.g. in the cases of views, or separate read-only warehouse)

Inheritance provides a natural way of solving the above data modeling/organizational/architectural problems right in the source of the database.

# Uses of inheritance

- System / DBA tasks

The coolest way I found for OIDs conversion was by using inheritance. Kudos to Robert Bernier @ Percona for his insightful and smart blog on this very topic!

# Data Departmentalization / Consolidation

Let us imagine our company makes and sells :

- Oil olive and olive based products
- Milk and dairy products

So our company, let's call it *Medifood SA*, operates as a parent company, having two subsidiaries (*MediMilk SA* and *MediOlive SA*). The parent company wishes to have the two subsidiaries operating independently but still wants to control all bank traffic and incoming/outgoing invoices centrally. In short the top financial managers of the Medifood want to have full access to the data of both MediMilk and MediOlive and have consolidated statistics.
Plus there is an activity managed and provided directly by the parent company having to do with quality assurance and certification services.
Let us assume 100,000s+ of code, of the same app/DB used by the all three companies. We don't want to rewrite the code as this would take a long time. Let's assume three instances of the app, one for each company.

**Inheritance to the rescue!**

# Data Departmentalization / Consolidation

```
CREATE SCHEMA medifood;
CREATE SCHEMA medimilk;
CREATE SCHEMA mediolive;

CREATE TABLE medifood.sales (
      id BIGSERIAL NOT NULL PRIMARY KEY,
      product TEXT NOT NULL,
      units_sold INTEGER NOT NULL,
      product_price FLOAT NOT NULL
);

CREATE TABLE medimilk.sales() INHERITS (medifood.sales);
CREATE TABLE mediolive.sales() INHERITS (medifood.sales);

ALTER TABLE medimilk.sales ADD CONSTRAINT sales_pkey PRIMARY KEY(id);
ALTER TABLE mediolive.sales ADD CONSTRAINT sales_pkey PRIMARY KEY(id);
```

# Data Departmentalization / Consolidation

We use the search_path setting / feature in PostgreSQL. We adjust the three apps as follows :

- Top parent MediFood app
  - `set search_path = '"$user", medifood, public';`
- MediMilk app
  - `set search_path = '"$user", medimilk, medifood, public';`
- MediOlive
  - `set search_path = '"$user", mediolive,medifood, public';`

**Top parent MediFood app :**

```
insert into sales (product,units_sold,product_price)
VALUES('Cert for dairy health',120,100);
```

**MediMilk app :**

```
insert into sales (product,units_sold,product_price) VALUES('Goat Milk',5000,2.5);
```

**MediOlive :**

```
insert into sales (product,units_sold,product_price) VALUES('Olive skin care',1000,10);
```

**Statements stay exactly the same!!!**

# Data Departmentalization / Consolidation

```
set search_path = '"$user", medimilk, medifood, public';
select sum(units_sold*product_price) from sales;
 sum
-------
12500

set search_path = '"$user", mediolive, medifood, public';
select sum(units_sold*product_price) from sales;
 sum
-------
10000

set search_path = '"$user", medifood, public';
select sum(units_sold*product_price) from ONLY sales;
 sum
-------
12000

select sum(units_sold*product_price) from sales;
 sum
-------
34500
```

# Data Departmentalization / Consolidation

And now some PostgreSQL magic …

```
select COALESCE(subsidiary,'----------Total') as subsidiary, sum
FROM
      (
            select tableoid::regclass::text as subsidiary,
            sum(units_sold*product_price) from sales
            group by rollup (subsidiary)
            ORDER BY subsidiary NULLS LAST
      ) as qry;
  subsidiary    |  sum
----------------+-------
medimilk.sales  | 12500
mediolive.sales | 10000
sales           | 12000
----------Total | 34500
(4 rows)
```

# DDL Management

And some more PostgreSQL magic : one of the basic things to take away from this talk.

Inherited tables also receive all future DDL run to their parent(s).

Management of DDL becomes so much easier!

# DDL Management

```
alter table medifood.sales ADD region varchar(50);
\d medimilk.sales
                                        Table "medimilk.sales"
   Column     |         Type          | Collation | Nullable |                   Default
--------------+-----------------------+-----------+----------+-----------------------------------------
-
id            | bigint                |           | not null | nextval('medifood.sales_id_seq'::regclass)
product       | text                  |           | not null |
units_sold    | integer               |           | not null |
product_price | double precision      |           | not null |
region        | character varying(50) |           |          |
…

\d mediolive.sales
                                        Table "mediolive.sales"
   Column     |         Type          | Collation | Nullable |                   Default
--------------+-----------------------+-----------+----------+-----------------------------------------
-
id            | bigint                |           | not null | nextval('medifood.sales_id_seq'::regclass)
product       | text                  |           | not null |
units_sold    | integer               |           | not null |
product_price | double precision      |           | not null |
region        | character varying(50) |           |          |
…
```

# Inheritance is one cool PostgreSQL Feature!

Athens, Greece 2023
PERCONA UNIVERSITY

# Thanks!

Contact :

Achilleas Mantzios

mantzios.achill@gmail.com

In partnership with FerretDB