Automating Detection of Occurrences of PostgreSQL Database Design Problems

Erki Eessaar

Department of Software Science,

Tallinn University of Technology, Estonia

erki.eessaar@taltech.ee

Outline

- Background and motivation.
- An example of SQL database design problems.
- Introduction of the catalog.
- Conclusions and future work.

Background

- A **good database design** and **following standards/practices** are important prerequisites of **well-maintainable** SQL database-driven software.
- Database can have **technical debt**, which is manifested by the occurrences of **bad smells**.
- Various case studies show problems in the designs of **existing databases**.

Motivation of the work

- The author teaches database design and has to give **quickly**, **thoroughly**, and **frequently** feedback to students.
- **Automating** the process would help a lot.
- The **usefulness** of automation *is not limited* with the teaching domain.

An example of design problems – bad names

```
CREATE TABLE Product_type (
ID SERIAL PRIMARY KEY,
name VARCHAR(100),
description VARCHAR(100),
CONSTRAINT chk CHECK (trim(name)<>''
AND description!~'^[[:space:]]*$'));
```

An example of design problems – inconsistency

```
CREATE TABLE Product_type (
ID SERIAL PRIMARY KEY,
name VARCHAR(100),
description VARCHAR(100),
CONSTRAINT chk CHECK (trim(name)<>''
AND description!~'^[[:space:]]*$'));
```

- Checking textual values in different columns **differently**.
- A constraint is named by user another is named by the system.

An example of design problems – violating the single-responsibility principle

```
CREATE TABLE Product_type (
ID SERIAL PRIMARY KEY,
name VARCHAR(100),
description VARCHAR(100),
CONSTRAINT chk CHECK (trim(name)<>''
AND description!~'^[[:space:]]*$'));
```

An example of design problems – lack of constraints

```
CREATE TABLE Product_type (
ID SERIAL PRIMARY KEY,
name VARCHAR(100),
description VARCHAR(100),
CONSTRAINT chk CHECK
(trim(name)<>''
AND description!~
'^[[:space:]]*$'));
```

- Only the primary key (surrogate key), no additional unique constraints.
- Only the primary key column is mandatory (due to a property of primary keys), all the other are optional (permit NULLs).

Database design must be **complete** meaning among other things that all the relevant constraints about the domain must be enforced in the database.

An example of design problems – potentially incorrect constraints

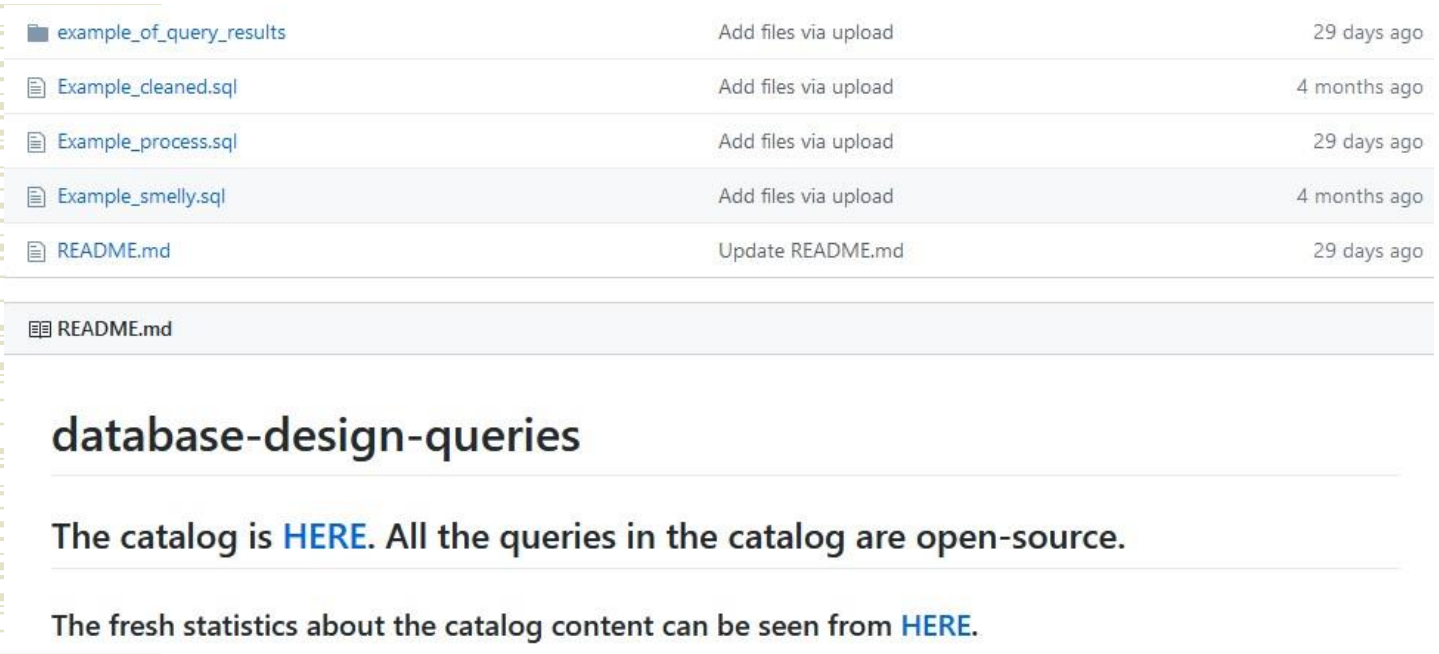
```
CREATE TABLE Product_type (
ID SERIAL PRIMARY KEY,
name VARCHAR(100),
description VARCHAR(100),
CONSTRAINT chk CHECK
(trim(name)<>''
AND description!~
'^[[:space:]]*$'));
```

- Name and description have the same **field size**.
- Checking name and description – why differently?

Database design must be **valid** meaning among other things that all the enforced constraints must be correct and relevant.

The contribution

https://github.com/erki77/database-design-queries



example_of_query_results	Add files via upload	29 days ago
Example_cleaned.sql	Add files via upload	4 months ago
Example_process.sql	Add files via upload	29 days ago
Example_smelly.sql	Add files via upload	4 months ago
README.md	Update README.md	29 days ago

README.md

database-design-queries

The catalog is HERE. All the queries in the catalog are open-source.

The fresh statistics about the catalog content can be seen from HERE.

- Reference to the **browsable catalog** of PostgreSQL system catalog-based SQL queries that can be used to find information about a database, including occurrences of design problems.
- Reference to the page with **up-to-date statistics** of the catalog.
- **An example** of using the queries to analyze the design of a database.

538 queries at the time of creating the presentation.

Browsable catalog

Catalog of PostgreSQL queries for finding information about a PostgreSQL database and its design problems.

Choose collection: Not specified

AND Choose query type: Problem detection — Each row in the result could represent a flaw in the design

AND Choose category: CHECK constraints — Queries of this category provide information about CHECK constraints.

AND Choose data source: Not specified — From where does the query gets its information?

AND Enter string: Search from the name and goal ...

AND Has fixing queries? ☑

Apply filter Reset

Possibility to search

- All the queries about database objects contain a subcondition to exclude from the result information about the system catalog.
- Although the statements use SQL constructs (common table expressions; NOT in subqueries) that could cause performance problems in case of large datasets it shouldn't be a problem in case of relatively small amount of data, which is in the system catalog of a database.
- Statistics about the catalog content and project home in GitHub that has additional information.

There are 18 queries.

Seq nr	Name▲	Goal	Type	Data source	License	...
1	BOOLEAN base table and foreign table columns with a CHECK constraint that involves olnly this column	Find base table and foreign table columns with the Boolean type that has a CHECK constraint that involves only this column. Avoid unnecessary CHECK constraints. The Boolean type contains only two values and there is nothing to check. By creating a check that determines that possible values in the column are TRUE and FALSE, one duplicates the attribute constraint (column has a type). This is a form of duplication.	Problem detection	INFORMATION_SCHEMA+system catalog base tables	MIT License	View
2	Cannot register all legal personal names	Find CHECK constraints on base table or foreign table columns that contain data about personal names and apply unnecessary restrictions to the names, rejecting potentially some legal names.	Problem detection	INFORMATION_SCHEMA+system catalog base tables	MIT License	View

Fragments of a query specification in the catalog

Cannot register all legal personal names

Query goal: Find CHECK constraints on base table or foreign table columns that contain data about personal names and apply unnecessary restrictions to the names, rejecting potentially some legal names.

Notes about the query: The query takes into account only constraints that are associated with exactly one column. Thus, there could be constraints that apply restrictions to multiple columns that still reject legal values. The query takes into account constraints that are associated directly with the column as well as constraints that are associated with the column through a domain. The query does not find CHECK constraints of domains that are not associated with any table. The query considers both column names in English and Estonian.

Query type: Problem detection (Each row in the result could represent a flaw in the design)

Query reliability: Medium (Medium number of false-positive results)

Query license: MIT License

Fixing suggestion: Drop the constraints.

Data source: INFORMATION_SCHEMA+system catalog

SQL query:

```
WITH ck AS (SELECT
o.conname,
(SELECT nspname FROM pg_namespace WHERE oid=c.relnamespace) AS target_schema,
c.relname AS target_table,
c.oid As target_table_oid,
unnest(o.conkey) AS target_col,
substring(pg_get_constraintdef(o.oid),7) AS consrc,
CASE WHEN c.relkind='r' THEN 'BASE TABLE'
WHEN c.relkind='f' THEN 'FOREIGN TABLE'
ELSE 'TABLE' END AS table_type
FROM pg_constraint o INNER JOIN pg_class c ON c.oid = o.conrelid
WHERE o.contype = 'c' AND cardinality(o.conkey)=1),
checks AS (
SELECT ck.target_schema AS table_schema, ck.target_table AS table_name,  table_type, a_target.attname AS column_name, ck.con
FROM ck INNER JOIN pg_attribute a_target ON ck.target_col = a_target.attnum AND ck.target_table_oid = a_target.attrelid AND a
WHERE ck.target_schema NOT IN (SELECT schema_name
FROM INFORMATION_SCHEMA.schemata
WHERE schema_name<>'public' AND
schema_owner='postgres' AND schema_name IS NOT NULL)
```

SQL statements for generating SQL statements that help us to fix the problem

SQL query	Reference
WITH ck AS (SELECT o.conname, (SELECT nspname FROM pg_namespace WHERE oid=c.relnamespace) AS target_schema, c.relname AS target_table, c.oid As target_table_oid, unnest(o.conkey) AS target_col, substring(pg_get_constraintdef(o.oid),7) AS consrc, CASE WHEN c.relkind='r' THEN 'BASE TABLE' WHEN c.relkind='f' THEN 'FOREIGN TABLE' ELSE 'TABLE' END AS table_type FROM pg_constraint o INNER JOIN pg_class c ON c.oid = o.conrelid WHERE o.contype = 'c' AND cardinality(o.conkey)=1), checks AS (SELECT ck.target_schema AS table_schema, ck.target_table AS table_name, table_type, a_target.attname AS column_name, ck.con FROM ck INNER JOIN pg_attribute a_target ON ck.target_col = a_target.attnum AND ck.target_table_oid = a_target.attrelid AND a WHERE ck.target_schema NOT IN (SELECT schema_name FROM INFORMATION_SCHEMA.schemata WHERE schema_name<>'public' AND schema_owner='postgres' AND schema_name IS NOT NULL)) SELECT format('ALTER TABLE %1$I.%2$I DROP CONSTRAINT %3$I;', table_schema, table_name, constraint_name) AS statements FROM checks WHERE column_name~*'(first[_]*name|last[_]*name|given[_]*name|surname|person[_]*name|eesnimi|perenimi|perekonnanimi|isik. AND check_clause~*'(0-9|a-|digit|alpha|alnum)' ORDER BY table_schema, table_name, column_name;	Drop the base table constraint.

Some statistics of the catalog

Type name	Number of queries	Percentage from the total number
Problem detection	385	71.6
General	115	21.4
Sofware measure	38	7.1

Up-to-date statistics is at the statistics homepage.

Currently the catalog has **47** active categories with at least one query.

Number of queries in different categories

Category name	Number of queries	Ranking based on number of queries
User-defined routines	109	1
Naming	107	2
Validity and completeness	105	3
Comfortability of database evolution	86	4
CHECK constraints	75	5

Improved design

```
CREATE TABLE Product_type (
product_type_code SMALLINT NOT NULL,
name VARCHAR(100) NOT NULL,
description VARCHAR(1000),
CONSTRAINT pk_product_type PRIMARY KEY (product_type_code),
CONSTRAINT ak_product_type_name UNIQUE (name),
CONSTRAINT chk_product_type_name CHECK (name!~'^[[:space:]]*$'),
CONSTRAINT chk_product_type_description CHECK
(description!~'^[[:space:]]*$'));
```

About using the queries

- Simple software can be made to execute the queries.
- For instance, during the 2018/2019 fall semester **141 different students** executed the collections of queries based on **159 different databases** more than six thousand times that resulted with almost **1.5 million executions** of individual queries.
- The queries facilitate **quick, frequent, and thorough feedback**.
- The results of many queries point to problem occurrences with **high certainty**, some need review together with the lecturer.
- The earlier a problem is detect the **cheaper** and **easier** it is to fix it.

Conclusions

- We have created and published a catalog of **538** (as of now) queries for identifying occurrences of design problems in PostgreSQL databases.
- Problem detection queries codify problems that in most case can occur in the database of **any SQL DBMS**.
- There is a set of **generic problems** that can manifest itself in models, application source code as well as database design.
  - Should be stressed in the educational process!

Future work

- Identify **additional** design problems.
- Write SQL queries to **detect** these.
- Reduce **technical debt** in the queries.
- Analyze **larger** databases.
- Translate the queries for **other SQL DBMSs**.
- Identify design problems in databases that are based on data model **other than SQL**.

Thank you for your attention!

Questions?

Erki Eessaar – Erki.Eessaar@taltech.ee

The reference to the catalog:
https://github.com/erki77/database-design-queries