Advanced SQL

Summer 2017

Torsten Grust
Universität Tübingen, Germany
1 | Welcome...

... to this exploration of advanced aspects of SQL. Your current mental image of SQL will change during this course (mine surely did already).

The value — in terms of scientific insight as well as 💰 — of knowing the ins and outs of SQL can hardly be overestimated.

SQL is an incredibly rich and versatile declarative database and programming language. Let's take a deep dive together!
Stack Overflow Developer Survey (March 2017)

Most Popular Technologies — Programming Languages

JavaScript: 61.9%
SQL: 50.8%
Java: 39.3%
C#: 33.8%
Python: 31.7%
PHP: 27.9%
C++: 22.1%
C: 18.9%
TypeScript: 9.4%
Ruby: 9.0%
Swift: 6.4%

1 https://stackoverflow.com/insights/survey/2017
Operating the Database System as a Dumbed Down Table Storage

Program- and Heap-Centric Operation of Database System
• **Move tables** — i.e., almost all columns/rows — from database system (DBMS) storage into programming language (PL) heap.

• Count on the PL heap to be able to hold all required row data (otherwise try to chunk or stream data).

• Map rows to PL data structures, then **perform in-heap computation** to obtain result.
Moving Computation Close to the Data

Data- and Query-Centric Operation of Database System
Moving Computation Close to the Data

- **Express complex computation** in terms of the advanced constructs offered by the SQL database language, ship query to DBMS.

- **Let the database system operate** over (high-volume) data in native DBMS format, supported by index structures.

- **Fetch the — typically few or even single — result row(s)** into the PL heap, perform lightweight in-heap post-processing (only if needed).
2 | The Origins of SQL

Don Chamberlin

Ray Boyce († 1974)
The Origins and of SQL

- Development of the language started in 1972, first as SQUARE, from 1973 on as SEQUEL (Structured English Query Language). In 1977, SEQUEL became SQL because of a trademark dispute. (Thus, both “S-Q-L” /ˌɛskjuːˈɛl/ and “sequel” /ˈsiːkwəl/ are okay pronunciations.)


- Since then, SQL has been in under active development and remains the “Intergalactic Dataspeak”.\(^2\)

\(^2\) Mike Stonebraker, inventor of Ingres (1972, precursor of Postgres, PostgreSQL)
**SQL Standards**

<table>
<thead>
<tr>
<th>Year</th>
<th>Name</th>
<th>Alias</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>1986</td>
<td>SQL-86</td>
<td>SQL-87</td>
<td>first ANSI-standardized version</td>
</tr>
<tr>
<td>1989</td>
<td>SQL-89</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1992</td>
<td>SQL-92</td>
<td>SQL2</td>
<td>integrity constraints</td>
</tr>
<tr>
<td>1999</td>
<td>SQL:1999</td>
<td>SQL3</td>
<td>major revision, ⚠ orthogonality</td>
</tr>
<tr>
<td>2003</td>
<td>SQL:2003</td>
<td></td>
<td>⚠️ recursive queries, PL/SQL, rows/arrays</td>
</tr>
<tr>
<td>2006</td>
<td>SQL:2006</td>
<td></td>
<td>XML support, window functions, sequences</td>
</tr>
<tr>
<td>2008</td>
<td>SQL:2008</td>
<td></td>
<td>XQuery support</td>
</tr>
<tr>
<td>2011</td>
<td>SQL:2011</td>
<td></td>
<td>TRUNCATE, MERGE, improved CASE/WEN</td>
</tr>
<tr>
<td>2016</td>
<td>SQL:2016</td>
<td></td>
<td>temporal data types/operations, row pattern matching, JSON support</td>
</tr>
</tbody>
</table>

- SQL standards are multi-1000 page documents. *Conformance levels* have been defined to give DBMS implementors a chance to catch up.

3 | This Course

- We will explore the wide variety of **query and procedural constructs** in SQL.

- How much **computation can we push** into the DBMS and thus towards the data?

- Where are the **limits of expressiveness** and pragmatics?

- Have fun along the way! 😎

  We will discuss **offbeat applications of SQL** beyond employees–departments and TPC-H examples.³

³ The *drosophila melanogaster* of database research.
### Torsten Grust?

<table>
<thead>
<tr>
<th>Time Frame</th>
<th>Affiliation/Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>1989-1994</td>
<td>Diploma in Computer Science, TU Clausthal</td>
</tr>
<tr>
<td>1994-1999</td>
<td>Promotion (PhD), U Konstanz</td>
</tr>
<tr>
<td>2000</td>
<td>Visiting Researcher, IBM (USA)</td>
</tr>
<tr>
<td>2000-2004</td>
<td>Habilitation, U Konstanz</td>
</tr>
<tr>
<td>2004-2005</td>
<td>Professor Database Systems, TU Clausthal</td>
</tr>
<tr>
<td>2005-2008</td>
<td>Professor Database Systems, TU München</td>
</tr>
<tr>
<td>since 2008</td>
<td>Professor Database Systems, U Tübingen</td>
</tr>
</tbody>
</table>

- **E-Mail:** Torsten.Grust@uni-tuebingen.de
- **Twitter:** @Teggy (Professor, likes database systems, programming languages, and LEGO ヽ)
- **WSI, Sand 13, Room B318**
### Administrativa

<table>
<thead>
<tr>
<th>Weekday/Time</th>
<th>Slot</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tuesday, 10:15-11:45</td>
<td>Lecture</td>
<td>Sand 14, C215</td>
</tr>
<tr>
<td>Thursday, 14:15-15:45</td>
<td>Tutorial</td>
<td>Sand 1, A301</td>
</tr>
</tbody>
</table>

- **⚠ No lectures/tutorials on**
  - Thursday, April 20 (tutorials start on April 27)
  - Thursday, May 25
  - Tuesday, June 6
  - Thursday, June 8
  - Thursday, June 15
Administrativa

End-Term Exam

- 90-minute **written exam** on Tuesday, July 25 2017, 10:00–12:00 (*room tbd*).
- You may bring a DIN A4 double-sided *cheat sheet*.
- Passing earns you 6 ECTS.

Weekly Assignments

- We will distribute, collect, and grade **weekly assignments** (Tuesday→Tuesday).
- Score $\geq \frac{2}{3}$ of the overall assignment points to be admitted to the exam and earn bonus points in the end-term exam.
Weekly Assignments & Tutorials

1. Expand on lecture material
2. Develop additional code, run additional examples, ...
3. Discuss solutions to weekly assignments

Organized and run by Christian Duta:

- E-Mail: Christian.Duta@uni-tuebingen.de
- WSI, Sand 13, Room B315

Assignments and tutorials will start in the second week of the semester once we have collected the first batch of interesting material.
Course Homepage

db.inf.uni-tuebingen.de/teaching/AdvancedSQLSS2017.html

- Download **slides** (PDF)
  Slide set developed while the semester runs — please be aware of bugs and report them. Thank you!
- (Download **assignments**)
- **Contact information**
  Just drop by our offices (Sand 13), send e-mail first if you seek specific help/require longer attention.
- Please visit page regularly ("...assignment unsolvable as given...", "...no lecture on...").
Material

This course is not based on a single textbook but based on

- a variety of scientific papers,
- textbook excerpts,
- blog and mailing list postings, Stack Exchange Q&As,\(^4\)
- SQL references/standards,
- experience, and best practices.

There is plethora of books on SQL Hacks, Quizzes, Puzzles, (Anti-)Patterns, Performance Tweaks, and Idioms. If we will use sources like these, we will name them.

\(^4\) [http://dba.stackexchange.com/questions/tagged/sql](http://dba.stackexchange.com/questions/tagged/sql) is worth a look
Get Your Hands Dirty: Install PostgreSQL!

PostgreSQL will be the primary tool in this course:

postgresql.org, version 9.6 assumed (9.x probably OK)

- Implements an extensive SQL:2011 dialect, is extensible as well as open to inspection, and generally awesome.
- Straightforward to install/use on macOS, Windows, Linux.
This course will not provide an introduction to SQL's tabular data model or the language itself.\footnote{Please see \textit{Database Systems 1} for such an introduction.}

Let us only spend a few moments/slides to recollect the data model fundamentals and to synchronize on terminology.

We will do the same with SQL language fundamentals right after.
In a SQL-based database instance, all data is organized in tables:

<table>
<thead>
<tr>
<th>A₁</th>
<th>A₂</th>
<th>A₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>⋮</td>
<td>⋮</td>
<td>⋮</td>
</tr>
<tr>
<td>valᵢ₁</td>
<td>⋮</td>
<td>valᵢ₃</td>
</tr>
<tr>
<td>⋮</td>
<td>⋮</td>
<td>⋮</td>
</tr>
<tr>
<td>valⱼ₁</td>
<td>valⱼ₂</td>
<td>valⱼ₃</td>
</tr>
</tbody>
</table>

- **Table Head**: (ordered left to right)
- **Row**: unordered

Table instance (bag of rows, unordered)

↑

2nd column
### Columns, Types, Cells, NULL

<table>
<thead>
<tr>
<th>A₁</th>
<th>A₂</th>
<th>A₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>val₁</td>
<td>val₂</td>
<td>NULL</td>
</tr>
</tbody>
</table>

- On table creation, the i^{th} column is assigned a unique column name Aᵢ and column data type τᵢ.
- Cell values valₖᵢ, for any j, are of data type τᵢ.
- Each data type τᵢ features a unique NULL value. Value valₖᵢ may be NULL unless column Aᵢ explicitly forbids it.
First Normal Form (1NF)

- SQL tables are in **first normal form (1NF):** all column data types $\tau_i$ are **atomic.**
- In particular, $val_{j_i}$ may **not** be a table again.\(^6\)
- In modern/real-world SQL, we will see how row values, arrays, and data types like JSON water down strict 1NF.

\(^6\) Such data nesting is admitted by **non-first normal form (NFNF, NF^2)** data models.
Keys: Value-Based Row Identification

<table>
<thead>
<tr>
<th></th>
<th>A₁</th>
<th>A₂</th>
<th>A₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>val₁</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>⋮</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>valₗ₁</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>valₗ₂</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>valₗ₃</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- If **key** \{A₁,A₂\} has been declared, we are guaranteed that \((\text{val}_i₁, \text{val}_i₂) \neq (\text{val}_j₁, \text{val}_j₂)\) for any \(i \neq j\).
- Predicate \(A₁ = c₁ \text{ AND } A₂ = c₂\) identifies at most one row.
- Convention: key columns \(A₁,A₂\) are leftmost in the schema, notation: \(A₁A₂A₃\).
Foreign Keys: Identifying Rows in Other Tables

- If foreign key $T(A_3) \rightarrow S(B_1)$ has been declared, for any value $t_{j_3}$ a matching value $s_{k_1}$ is guaranteed to exist (⚠ no “dangling pointers”). If row $s_{k_1}$ is deleted, we need to compensate.
- In general, $\{A_3\}$ is not a key in $T$, i.e., $t_{j_3} = t_{i_3}$ is OK.