

# Advanced SQL

---

Summer 2020

Torsten Grust  
Universität Tübingen, Germany

# 1 | Welcome...

---

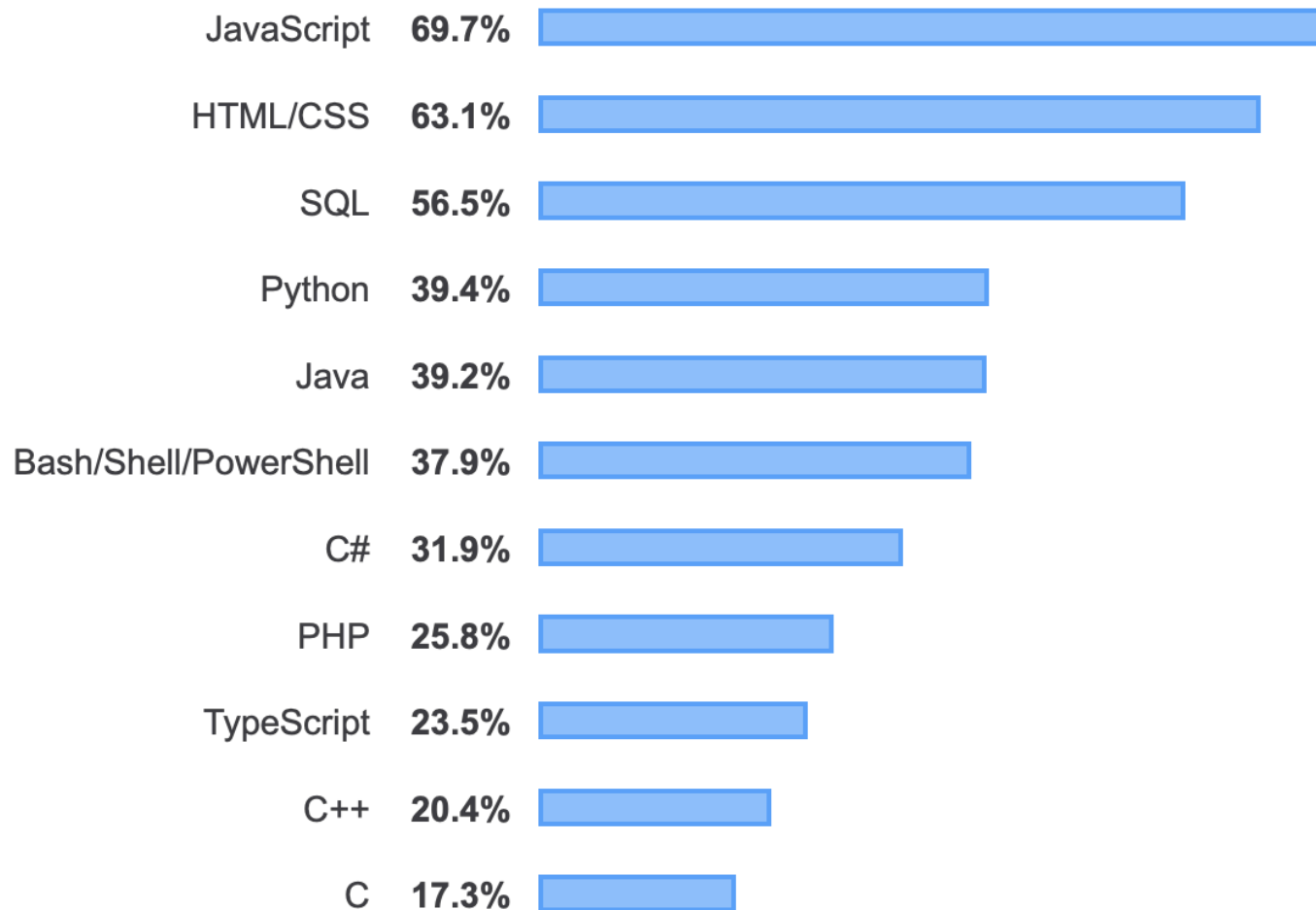
... to this exploration of **advanced aspects of SQL**. Your current mental image of SQL will change during this course (mine surely did already).

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

SQL is a remarkably rich and versatile **declarative database and programming language**. Let's take a deep dive together!

# Stack Overflow Developer Survey (March 2019)

---

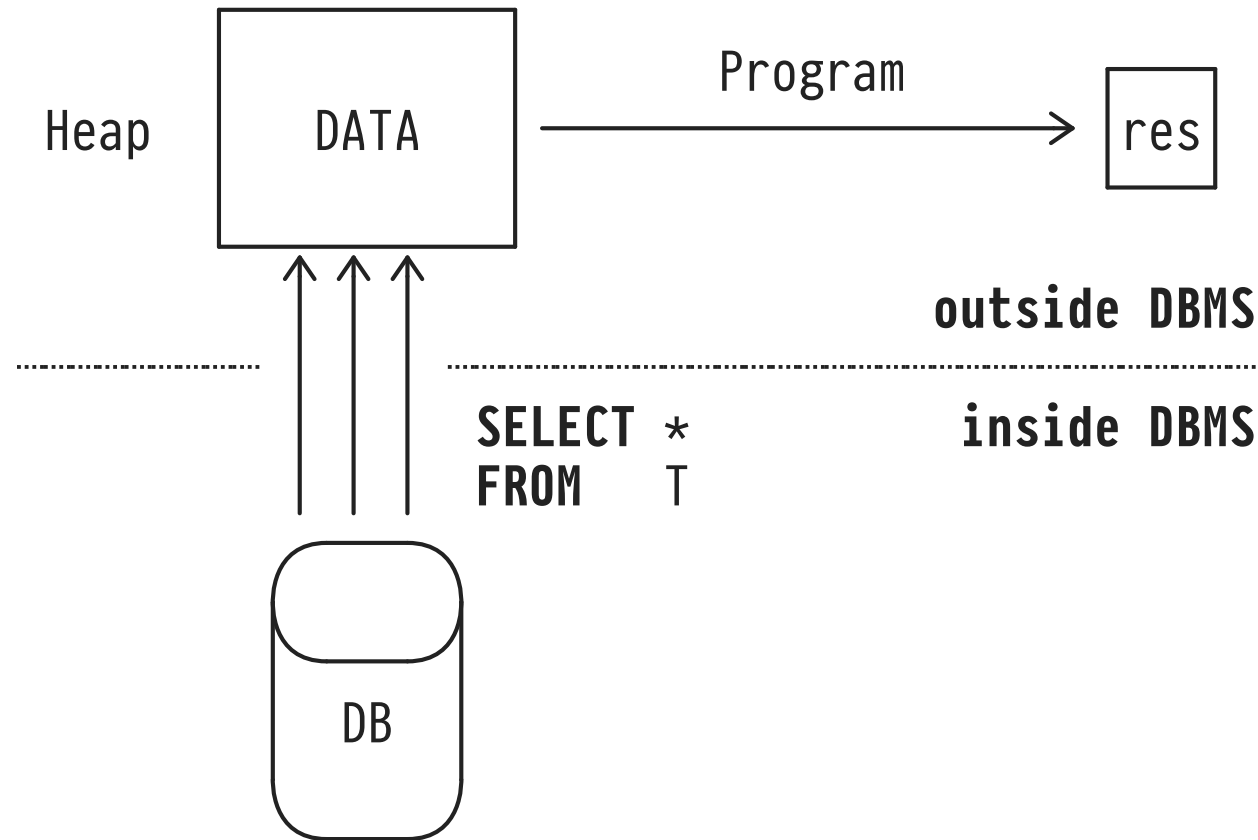


Most Popular Technologies — Programming Languages<sup>1</sup>

<sup>1</sup> <https://stackoverflow.com/insights/survey/2019>

# Operating the Database System as a Dumbed Down Table Storage

---



👉 Program- and Heap-Centric Operation of Database System

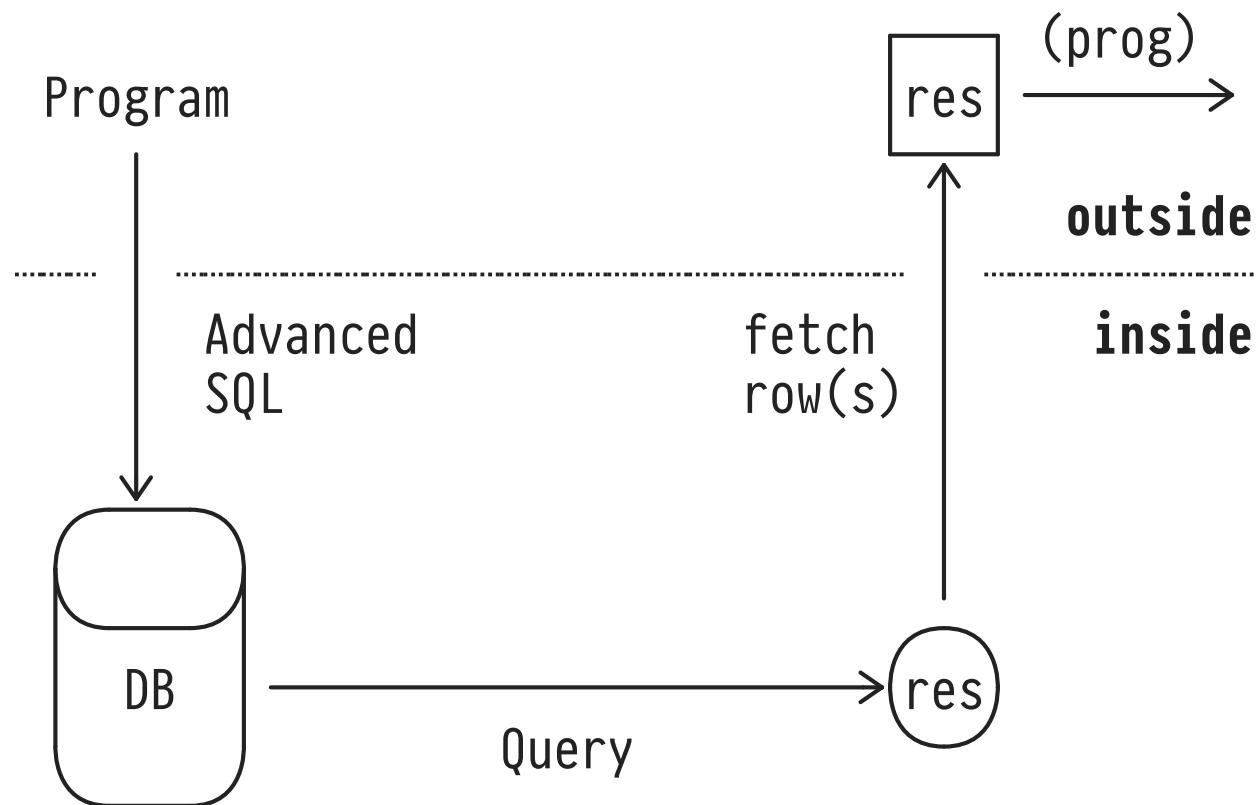
## Operating the Database System as a Dumbed Down Table Storage

---

- **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” /,ɛskjuːˈɛl/ and “*sequel*” /ˈsiːkwəl/ are okay pronunciations.)
- First commercial implementations in the late 1970s/early 1980s. By 1986, the ANSI/ISO standardization process begins.
- Since then, SQL has been in under active development and remains the “*Intergalactic Dataspeak*”.<sup>2</sup>

<sup>2</sup> Mike Stonebraker, inventor of Ingres (1972, precursor of Postgres, PostgreSQL)

## SQL Standards

---

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

- SQL standards are multi-1000 page documents. *Conformance levels* have been defined to give DBMS implementors a chance to catch up.
- IBM DB2 implements subsets of SQL-92 and SQL:2003. PostgreSQL 12.x implements most of core SQL:2011.

### 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.<sup>3</sup>

<sup>3</sup> The *drosophila melanogaster* of database research.

## Torsten Grust?


---

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

- E-Mail: [Torsten.Grust@uni-tuebingen.de](mailto:Torsten.Grust@uni-tuebingen.de)
- Twitter: [@Teggy](https://twitter.com/Teggy) (*Professor, likes database systems, programming languages, and SC Freiburg ツ*)
- WSI, Sand 13, Room B318

## Administrativa



---

- To help keep the ongoing COVID-19  pandemic at bay, there will be **no in-class lectures**, at least until mid-June 2020 (*i.e.*, until the whitsun break).
- *In case* we will finish the semester in the class room, the following are the assigned lecture hall and time slots:

Weekday/Time	Slot	Room
Thursday, 10:15–11:45	Lecture	Sand 14, C215
Tuesday, 14:15–15:45	Tutorial	Sand 14, C215

## Lecture Videos, Slides, and Pieces of SQL



---

- I will post **lecture videos** ( $\approx$  15-min fragments) to a dedicated playlist on YouTube . Those videos will
  - walk through the slides,
  - develop, run, and discuss SQL code snippets,
  - run live PostgreSQL experiments, and
  - expand on slide material.
- These **slides** (PDF), **SQL code fragments**, and **sample data** will be uploaded to a Github  repository:

[github.com/DBatUTuebingen/asql-ss20](https://github.com/DBatUTuebingen/asql-ss20) 




## Weekly Assignments & Tutorial Sessions

---

- We will distribute, collect, and grade **weekly assignments** (Tuesday→Tuesday) via Github .
- You work on these in **teams of two**. Hand-in again via .

Organized and run by **Christian Duta**:

- E-Mail: [Christian.Duta@uni-tuebingen.de](mailto:Christian.Duta@uni-tuebingen.de)
- WSI, Sand 13, Room B315

Assignments start once we have collected the first batch of interesting material, probably by the end of April. **Live, interactive tutorials**    will be announced in time.

## Forum

---

During this lunatic summer semester, the **Advanced SQL forum** is *the* course hub and more important than ever:

[forum-db.informatik.uni-tuebingen.de/c/ss20-asql](https://forum-db.informatik.uni-tuebingen.de/c/ss20-asql) 

- ⚠ **Registration** (mandatory) and announcements
- ❓ Questions and answers (do *not* post complete solutions)
- ☁️ Download additional code examples (e.g., SQL)
- 💬 General discussion
- 🕒 Quick turnaround (responses often within minutes)



## End-Term Exam

---

⚠ Exactly when and how we can run the Advanced SQL end-term exam is subject to contact restrictions and regulations of U Tübingen. Our current plans (as of April 20):

- 90-min **written exam** on Thursday, July 23, 10:00.
- 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.
- You may bring a DIN A4 double-sided *cheat sheet*.
- Passing earns you 6 ECTS.

## Course Homepage

---

[db.inf.uni-tuebingen.de/teaching/AdvancedSQLSS2020.html](http://db.inf.uni-tuebingen.de/teaching/AdvancedSQLSS2020.html) 

- **Organizational matters**

Curriculum. General announcements regarding the lecture, exams, or dates. Please surf by regularly. Thank you!

- **Contact information**

Turn to the forum first. But feel free to send e-mail if you seek specific help/need to discuss personal issues with us.

## 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,<sup>4</sup>
- SQL references/standards,
- Markus Winand's excellent web site [modern-sql.com](#),
- experience, and best practices.

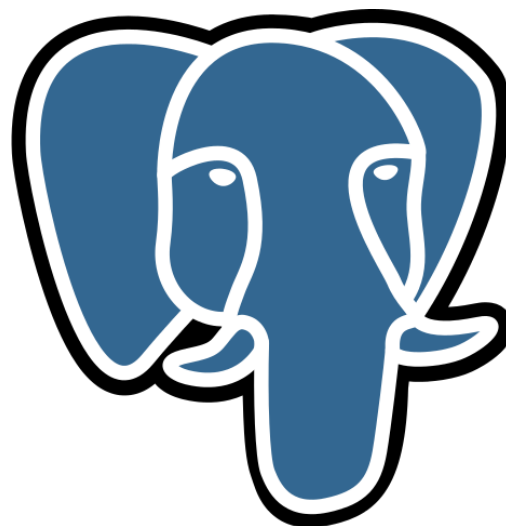
There is plethora of books on SQL Hacks, Quizzes, Puzzles, (Anti-)Patterns, Performance Tweaks, and Idioms. If we will use sources like these, we will name them.

<sup>4</sup> <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](https://www.postgresql.org), version 12.x assumed (11.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.

## 4 : SQL's Tabular Data Model

---

This course will *not* provide an introduction to SQL's **tabular data model** or the language itself.<sup>5</sup>

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

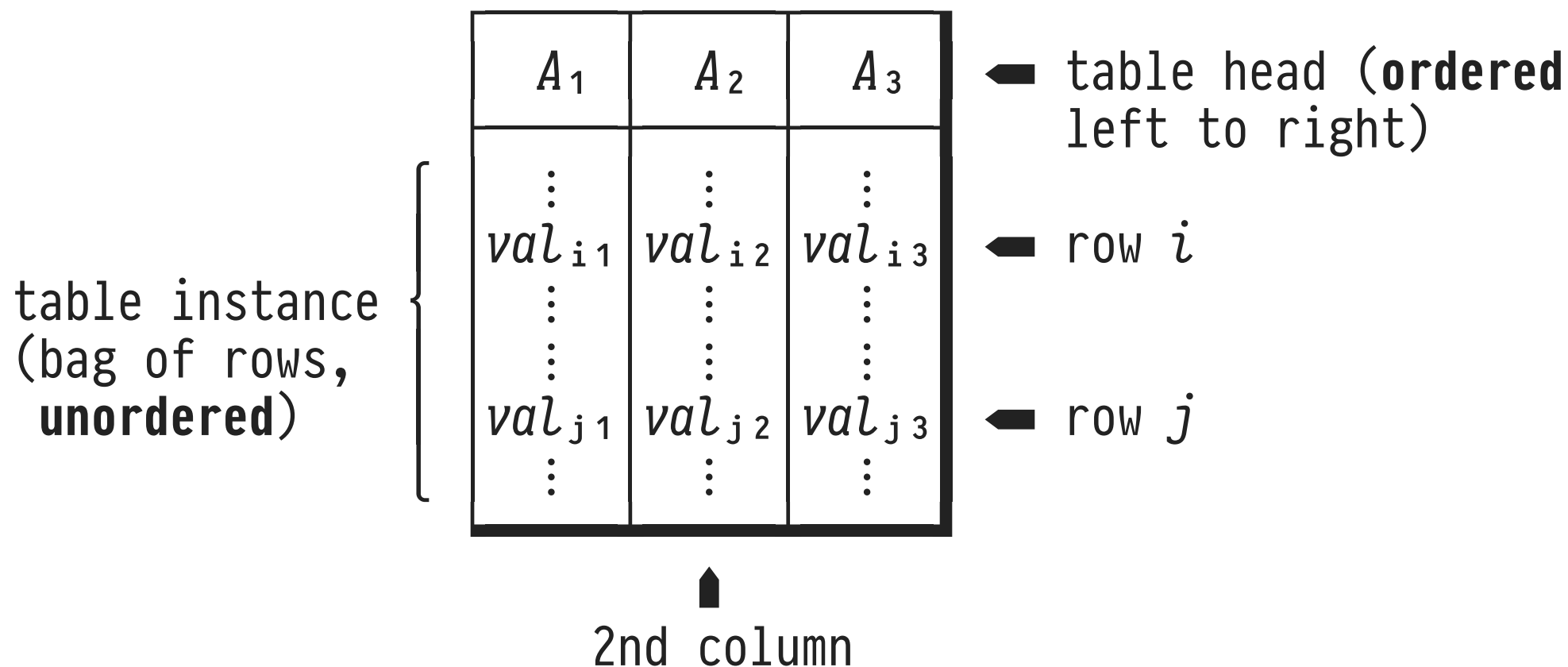
We will do the same with **SQL language fundamentals** right after.

<sup>5</sup> Please see [Database Systems 1](#) for such an introduction.

## Tables

---

In a SQL-based database instance, *all* data is organized in **tables**:



## Columns, Types, Cells, NULL

---

$A_1$	$A_2$	$A_3$
$\vdots$ $val_{j1}$ $\vdots$	$\vdots$ $val_{j2}$ $\vdots$	$\vdots$ NULL $\vdots$

◀  $A_i :: \tau_i, \quad i \in \{1,2,3\}$

- On table creation, the  $i^{\text{th}}$  column is assigned a unique **column name**  $A_i$  and **column data type**  $\tau_i$ .
- **Cell values**  $val_{ji}$ , for *any* row  $j$ , are of data type  $\tau_i$ .
- Each data type  $\tau_i$  features a unique **NULL** value. Value  $val_{ji}$  may be **NULL** unless column  $A_i$  explicitly forbids it.

## First Normal Form (1NF)

---

$A_1$	$A_2$	$A_3$
$\vdots$ $val_{j\ 1}$ $\vdots$	$\vdots$ $val_{j\ 2}$ $\vdots$	$\vdots$ $val_{j\ 3}$ $\vdots$

- 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.<sup>6</sup>
- In modern/real-world SQL, we will see how *row values*, *arrays*, and data types like JSON water down strict 1NF.

<sup>6</sup> Such data nesting is admitted by *non-first normal form* (NFNF, NF<sup>2</sup>) data models.



## Keys: Value-Based Row Identification

---

key (= subset of columns)

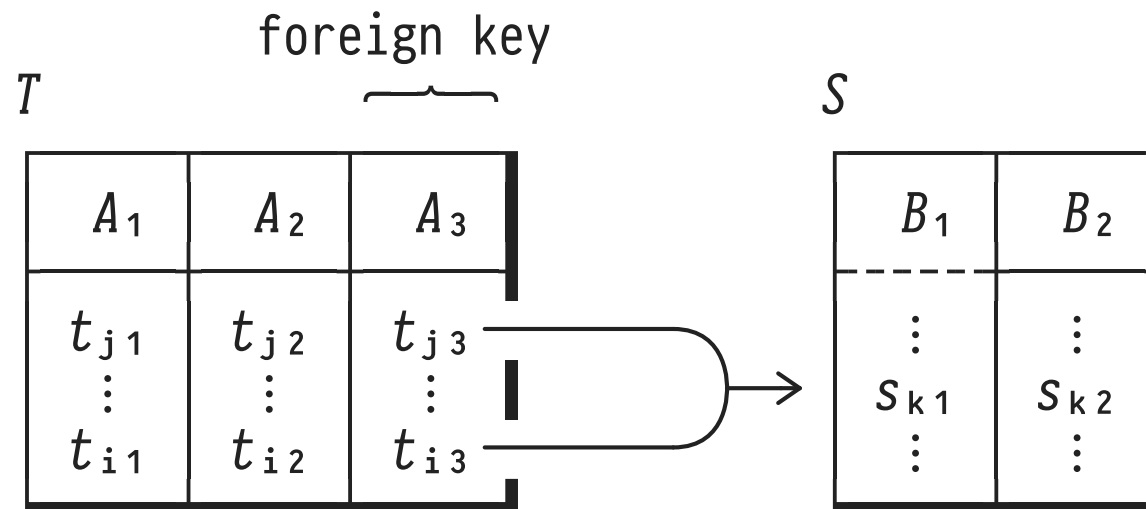
convention in these slides: ➡  
 ---- marks key columns

$A_1$	$A_2$	$A_3$
$val_{i1}$	$val_{i2}$	$val_{i3}$
$\vdots$	$\vdots$	$\vdots$
$val_{j1}$	$val_{j2}$	$val_{j3}$

- If **key**  $\{A_1, A_2\}$  has been declared, we are guaranteed that  $(val_{i1}, val_{i2}) \neq (val_{j1}, val_{j2})$  for any  $i \neq j$ .
- Predicate  $A_1 = c_1 \text{ AND } A_2 = c_2$  identifies at most one row.
- Convention: key columns  $A_1, A_2$  are leftmost in the schema, notation:  $A_1 A_2$   $A_3$ .

## Foreign Keys: Identifying Rows in Other Tables

---



- If **foreign key**  $T(A_3) \rightarrow S(B_1)$  has been declared, for any value  $t_{j3}$  a matching value  $s_{k1}$  is guaranteed to exist (⚠ no “dangling pointers”). If row  $s_{k1}$  is deleted, we need to compensate.
- In general,  $\{A_3\}$  is *not* a key in  $T$  ( $t_{j3} = t_{i3}$  is OK).