Advanced SQL

①

Welcome & Setup

Winter 2025/26

Torsten Grust
Universität Tübingen, Germany

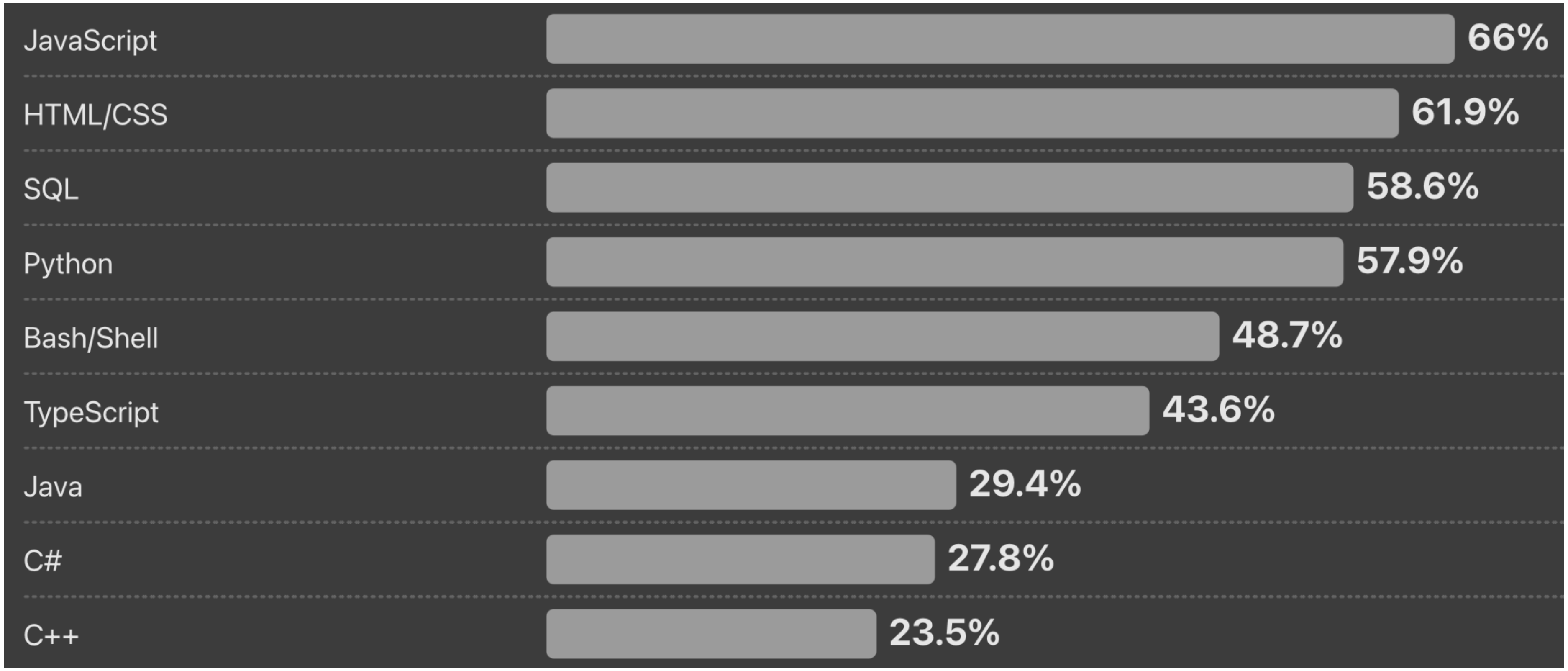
1 ¦ Welcome…

... 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!

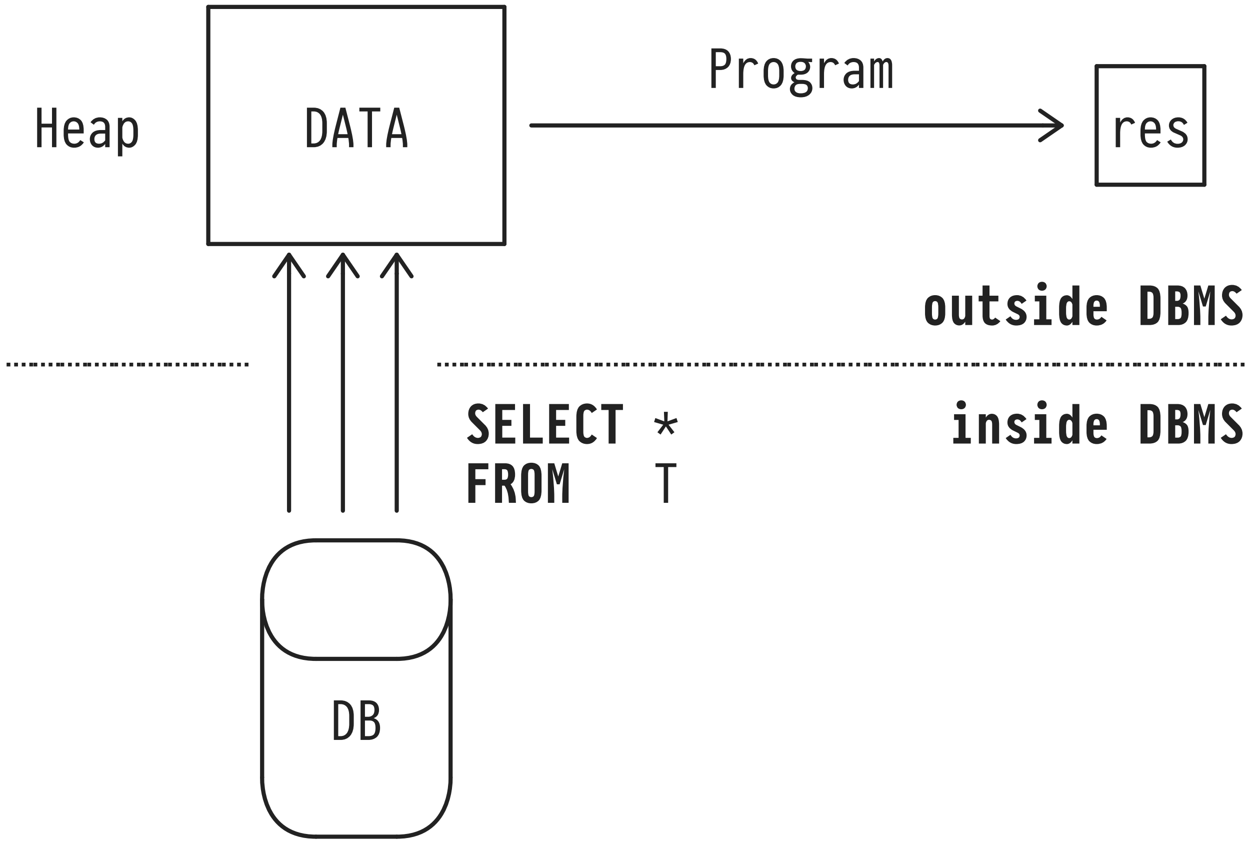
Stack Overflow Developer Survey (June 2025)



Most Popular Technologies — Programming Languages[1]

[1] https://survey.stackoverflow.co/2025/

Operating the Database System as a Dumbed Down Table Storage



```
SELECT *
FROM   T
```

👎 Program- and Heap-Centric Operation of Database System

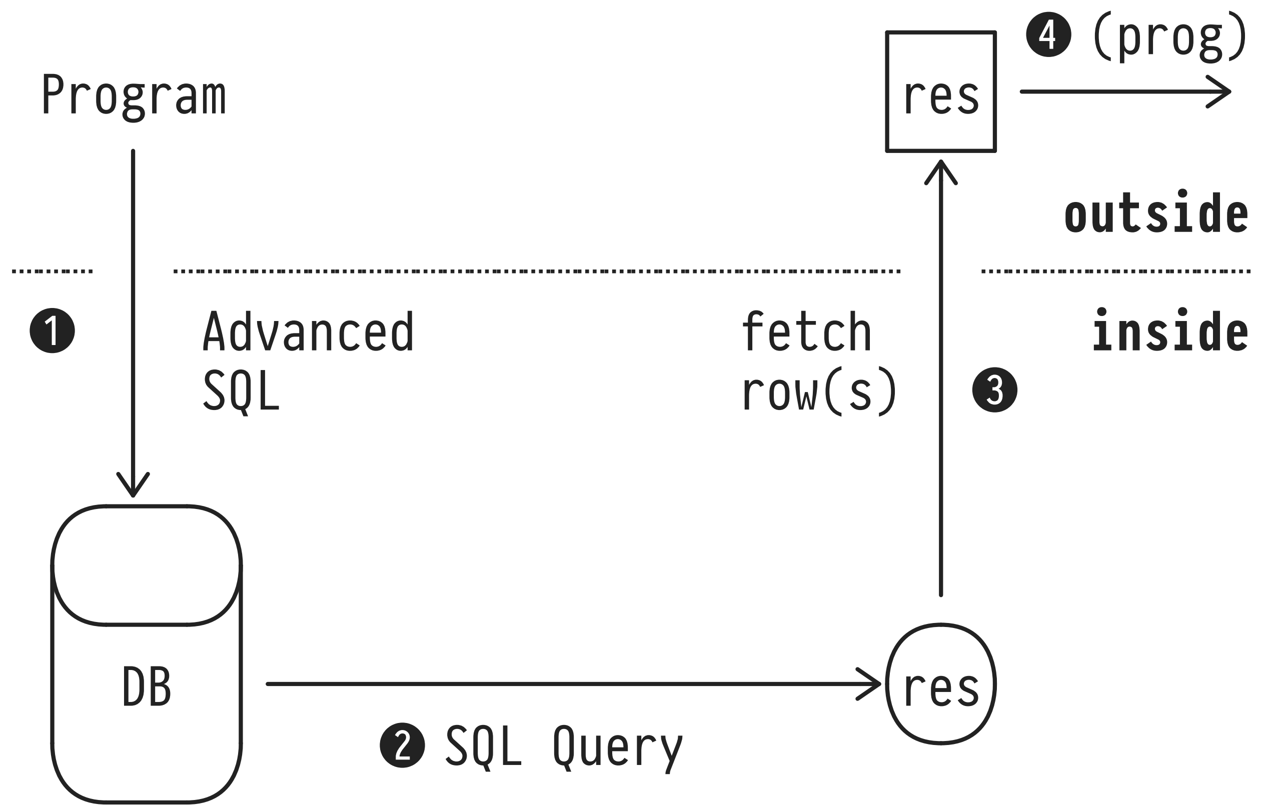
Operating the Database System as a Dumbed Down Table Storage

- **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

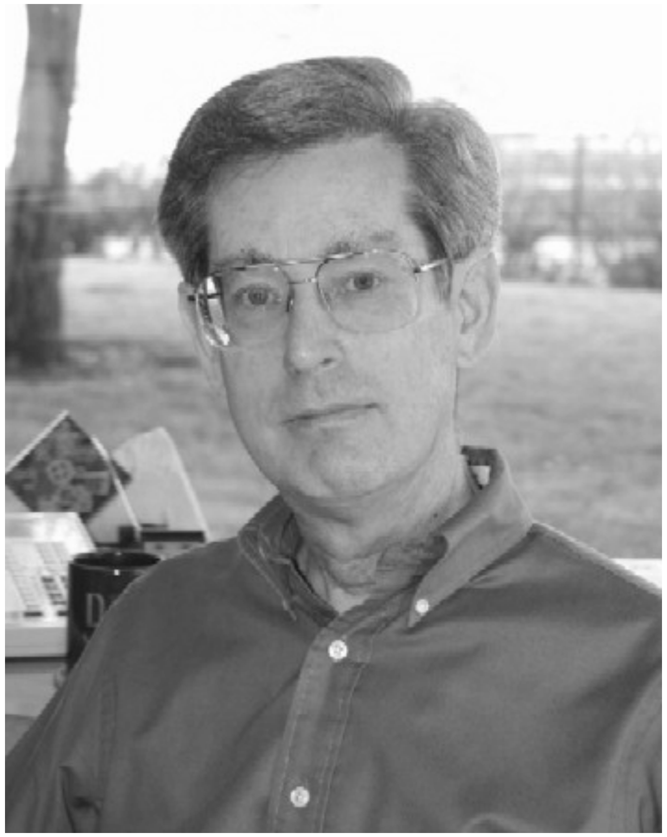


👍 Data- and Query-Centric Operation of Database System

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 ❶.**
- **Let the database system operate** over (high-volume) data in native DBMS format, supported by index structures ❷.
- **Fetch the—typically few or even single—result row**(**s**) into the PL heap ❸, perform lightweight in-heap post-processing (only if needed) ❹.

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 pronounciations.)

- First commercial implementations in the late 1970s/early 1980s. By 1986, the ANSI/ISO standardization process begins.

- Since then, SQL has been in active development and remains the ***“Intergalactic Dataspeak”***.[2]

[2] 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
2023	SQL:2023		graph processing

- 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 18.*x* implements the core of SQL:2011/2016.

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 (*i.e.*, across the ·········· divide) 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-projects-departments* and TPC-H examples.[3]

[3] 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	*Visiting Researcher*, 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

- Web: https://db.cs.uni-tuebingen.de/grust
- Office: WSI, Sand 13, Room B318
- Bluesky 🦋: @teggy.org

- Best bet is to catch me on the DB group's Discord

Administrivia

Weekday/Time	Slot	Room
Thursday, 10:15–11:45	Lecture	Sand 1, A301
Tuesday, 14:15–15:45	Tutorial	Sand 14, C215

- ⚠️ **No** lectures/tutorials on
  - Tue, Oct 21
  - Thu, Oct 30 (Bavarian Database Day)
  - ~~(In this winter semester, we may miss Tue, Jan 20, 2026 and Thu, Jan 22, 2026—details will follow.)~~

End-Term Exam (6 ECTS)

- **Written exam** on Thu, Feb 12, 2026, 10:00 (Room F119).
- Score $\geqslant \frac{2}{3}$ of the overall assignment points to be admitted to the exam.

Weekly Assignments & Tutorial Sessions

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

Organized and run by **Tim Fischer** and **Björn Bamberg:**

- Web: https://db.cs.uni-tuebingen.de/team/
- Offices: WSI, Sand 13, Rooms B314 and B315
- Find Tim and Björn on Discord

Assignments start once we have collected the first batch of interesting material, probably on Friday, October 24.

Slides and Pieces of SQL (and Lecture Videos)

- These **slides** (PDF), **SQL code fragments,** and **sample data** will be uploaded to a Github repository:

  https://github.com/DBatUTuebingen-Teaching/asql-ws2526

- For the 2020 edition of the course, I have produced **lecture videos:**
  - 58 videos, ≈ 30-min fragments.
  - Playlist on YouTube: tinyurl.com/AdvSQL-2020
  - ⚠️ Since 2020, the course has moved on—material was added/superseded/shuffled, the **DBMS has been replaced.**
  - We do aim to make your/our time in A301 worthwhile.

Discord

During this summer semester, the **Advanced SQL Discord** is *the* course hub:

https://db.cs.uni-tuebingen.de/discord

- **Registration** (do it!): /verify with your e-mail address
- Questions and answers (do *no* post complete solutions)
- Download additional code examples (e.g., SQL snippets)
- General discussion
- Quick turnaround (responses often within minutes)

Course Homepage

db.cs.uni-tuebingen.de/teaching/ws2526/advanced-sql/

- **Organizational matters**
  Curriculum. General announcements regarding the lecture, exams, or dates. (Less important this semester.)

- **Contact information**
  Turn to Discord 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,[4]
- 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.

[4] https://dba.stackexchange.com is worth a look

Get Your Hands Dirty: Install DuckDB!

The RDBMS **DuckDB** will be the primary tool in this course:

duckdb.org, version 1.4 (October 2025: 1.4.1)

- Implements an extensive SQL dialect, is highly performant, open to contributions, 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.[5]

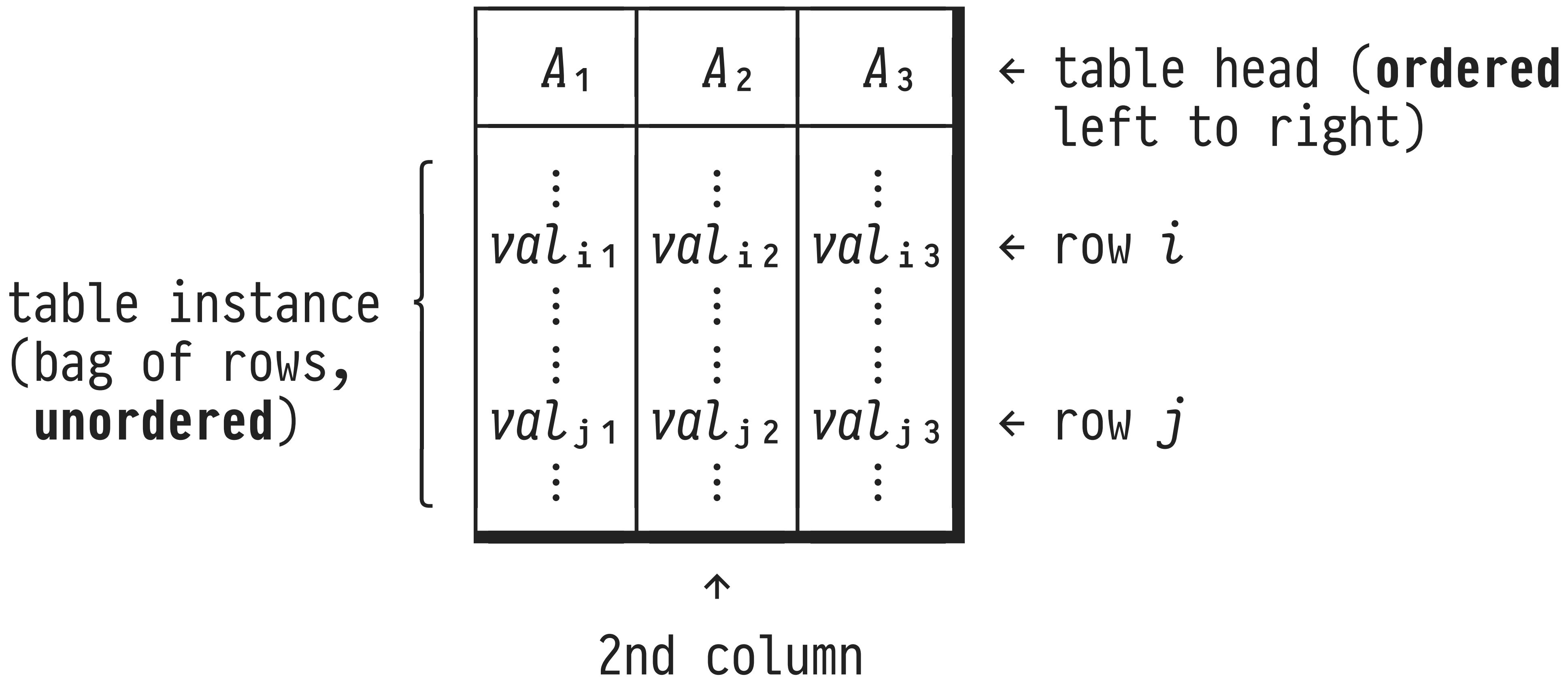
Let us only spend a few moments/slides to recollect the **data model fundamentals** and to synchronize on terminology.

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

[5] Please see the course Tabular Database Systems (*TaDa*) for such an introduction.

Tables

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



A1
A2
A3
← table head (ordered left to right)
val i1
val i2
val i3
← row i
val j1
val j2
val j3
← row j
table instance (bag of rows, unordered)
2nd column

Columns, Types, Cells, NULL

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

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

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

First Normal Form (1NF)

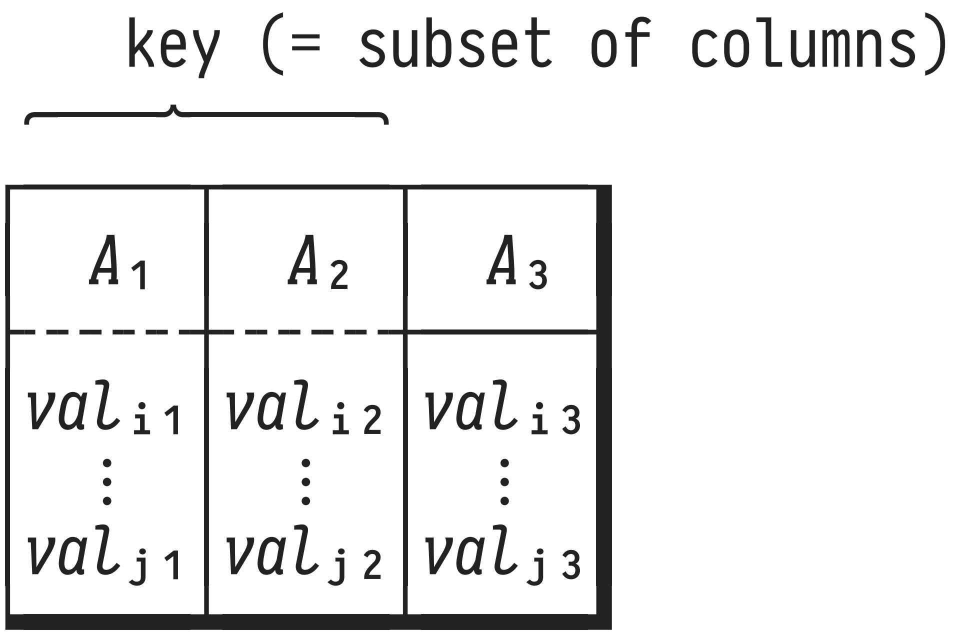
A_1	A_2	A_3
$\vdots$	$\vdots$	$\vdots$
val_{j1}	val_{j2}	val_{j3}
$\vdots$	$\vdots$	$\vdots$

- SQL tables are in **first normal form (1NF):** all column data types τ_i are **atomic.**
- In particular, val_{ji} may *not* be a table again.[6]
- In modern SQL, we will see how *row values* (or: *structs*), *arrays*, *maps*, and types like JSON water down strict 1NF.

[6] Such data nesting is admitted by *non-first normal form* (NFNF, NF²) data models.

Keys: Value-Based Row Identification

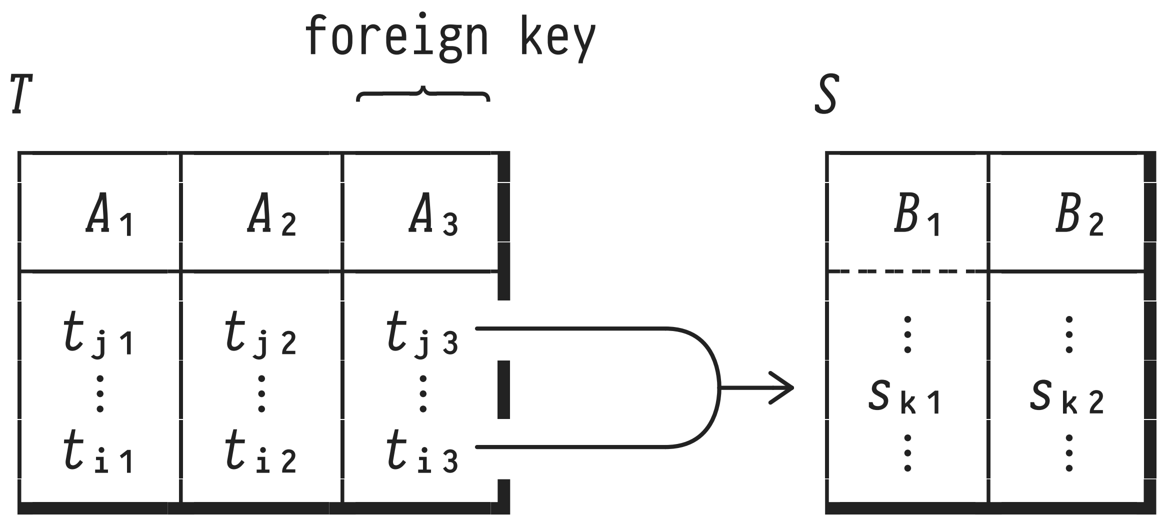
convention in these slides: →
---- marks key columns



A_1	A_2	A_3
val_{i1}	val_{i2}	val_{i3}
⋮	⋮	⋮
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 $A_1 = c_1$ AND $A_2 = c_2$ identifies at most one row.
- Convention: key columns A_1,A_2 are leftmost in the schema, notation: $\underline{A_1\ A_2}\ A_3$.

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 T.
- In general, $\{A_3\}$ is *not* a key in T ($t_{j3} = t_{i3}$ is OK).