

Advanced SQL

Summer 2022

Torsten Grust
Universität Tübingen, Germany

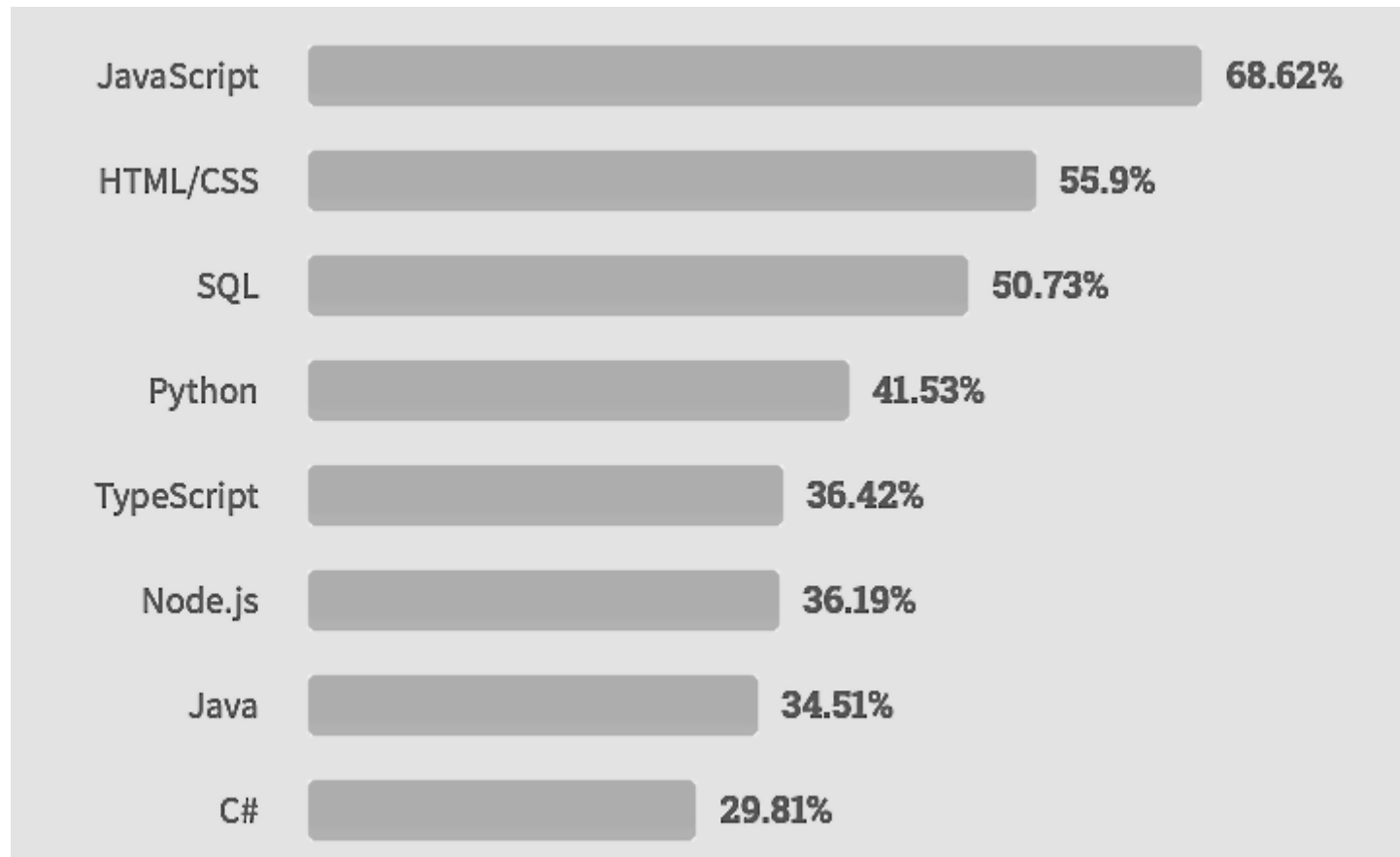
1 | Welcome...

... 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 

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

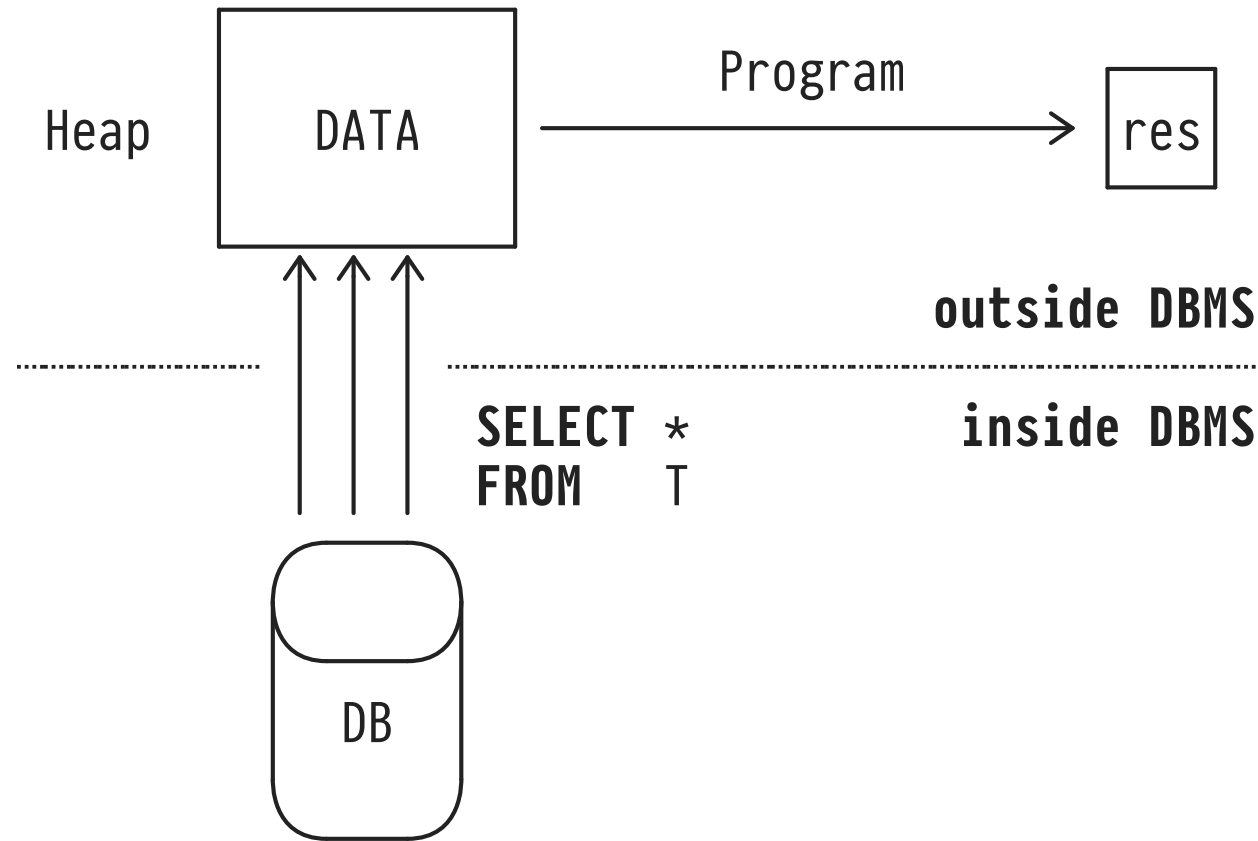
Stack Overflow Developer Survey (March 2021)



Most Popular Technologies — Programming Languages¹

¹ <https://insights.stackoverflow.com/survey/2021>

Operating the Database System as a Dumbed Down Table Storage

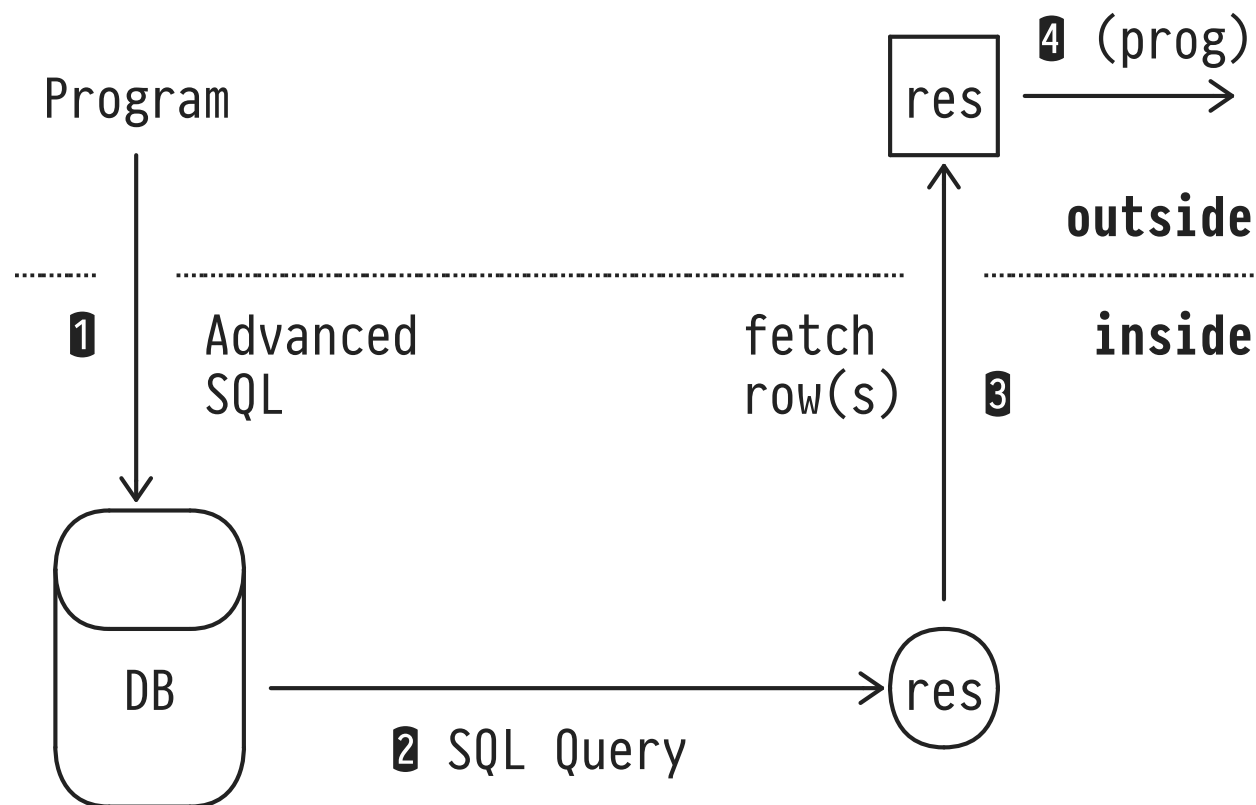


👉 Program- and Heap-Centric Operation of Database System

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 ①**.
- **Let the database system operate** over (high-volume) data in native DBMS format, supported by index structures **②**.
- **Fetch the—typically few or even single—result row(s)** into the PL heap **③**, perform lightweight in-heap post-processing (only if needed) **④**.

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***”.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 14.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) 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	<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
- Twitter: [@Teggy](#) (*Professor, likes database systems, programming languages, and SC Freiburg ツ*)
- WSI, Sand 13, Room B318

Administrivia

Weekday/Time	Slot	Room
Thursday, 10:15–11:45	Lecture	Sand 13, A301
Wednesday, 14:15–15:45	Tutorial	Sand 13, A301

-  **No** lectures/tutorials on
 - Thu, May 26 (Ascension Day)
 - Wed/Thu, June 8/9 (Whitsun break)
 - Thu, June 16 (Corpus Christi)



End-Term Exam (6 ECTS)

- **Written exam** on Thu, July 28, 8:30–10:00 (Rooms N10+N11).
- 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.



Slides, and Pieces of SQL (and Lecture Videos)

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

github.com/DBatUTuebingen/asql-ss22 

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

Weekly Assignments & Tutorial Sessions

- We will distribute, collect, and grade **weekly assignments** (Friday→Friday) 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
- WSI, Sand 13, Room B315

Assignments start once we have collected the first batch of interesting material, probably on Friday, April 29.

Forum

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

forum-db.informatik.uni-tuebingen.de/c/ss22-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)

Course Homepage

db.inf.uni-tuebingen.de/teaching/AdvancedSQLSS2022.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,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/questions/tagged/postgresql> is worth a look

Get Your Hands Dirty: Install PostgreSQL!

PostgreSQL will be the primary tool in this course:



postgresql.org, version 14.x (we run 14.1)

- 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.⁵

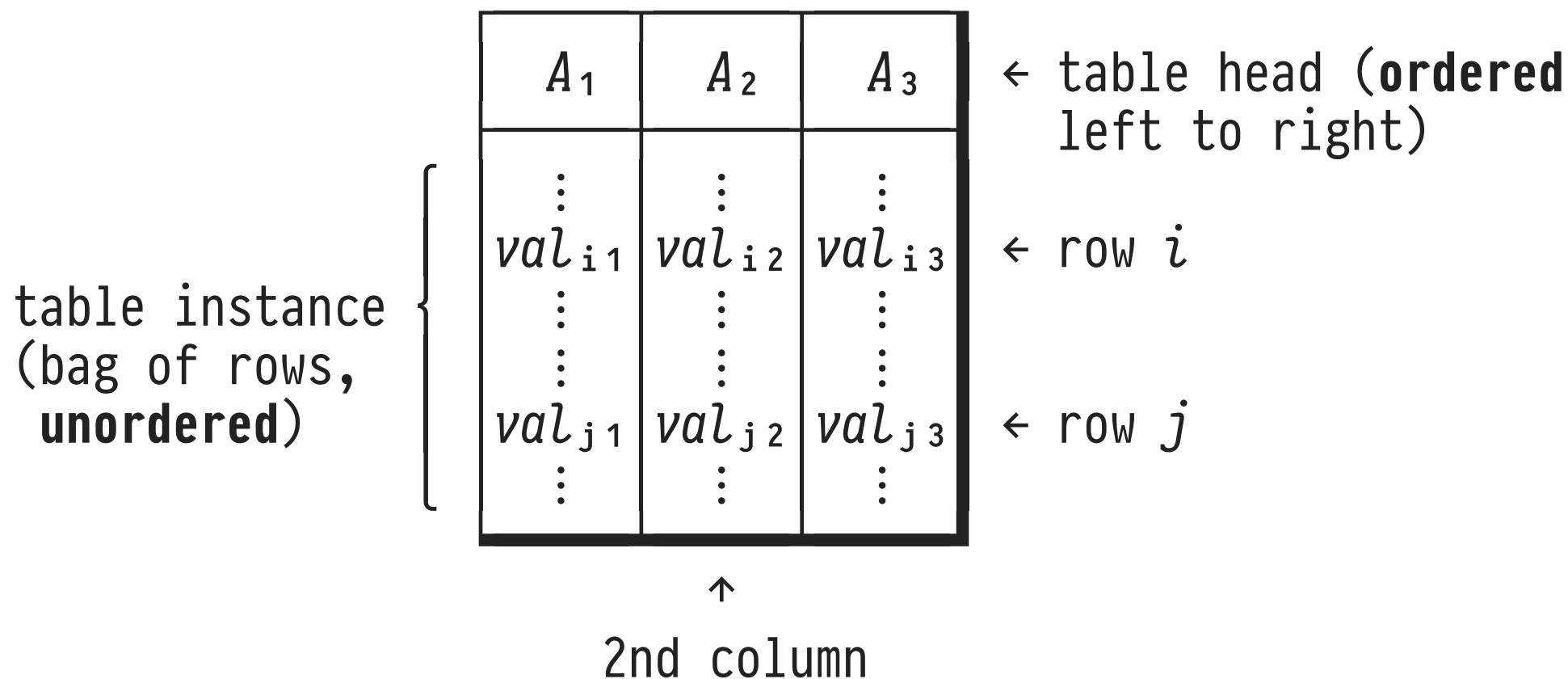
Let us only spend a few moments/slides to recollect the **data model fundamentals** and to synchronize on terminology.

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

⁵ Please see the course [DB1](#) 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

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

- On table creation, the i^{th} column is assigned a unique **column name** A_i and **column data type** τ_i .
- **Cell values** val_{ji} , for *any* row j , are of data type τ_i .
- Each data type τ_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_{j1} \vdots	\vdots val_{j2} \vdots	\vdots val_{j3} \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/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, NF²) data models.

Keys: Value-Based Row Identification

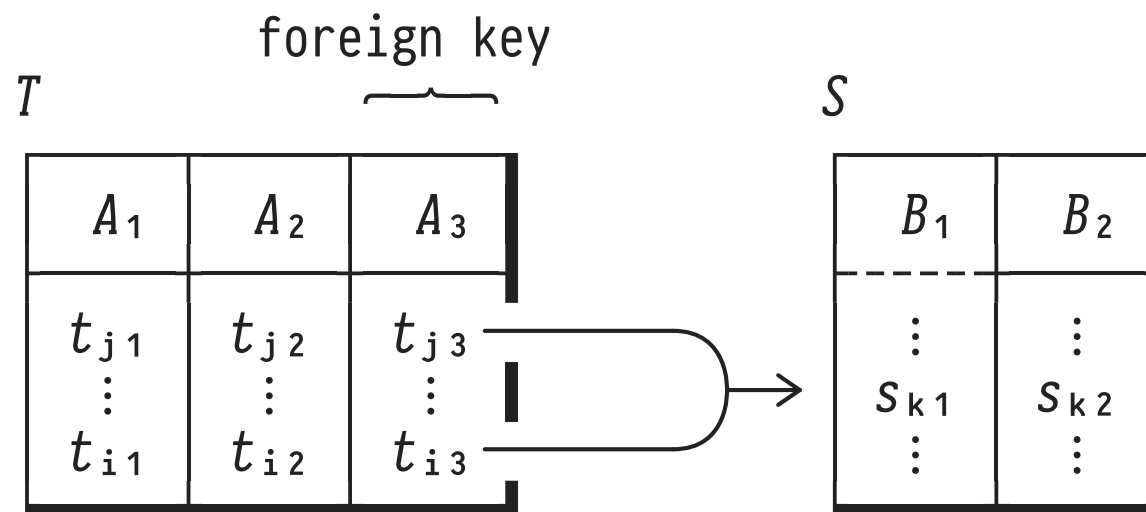
key (= subset of columns)

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