Advanced SQL

Summer 2017

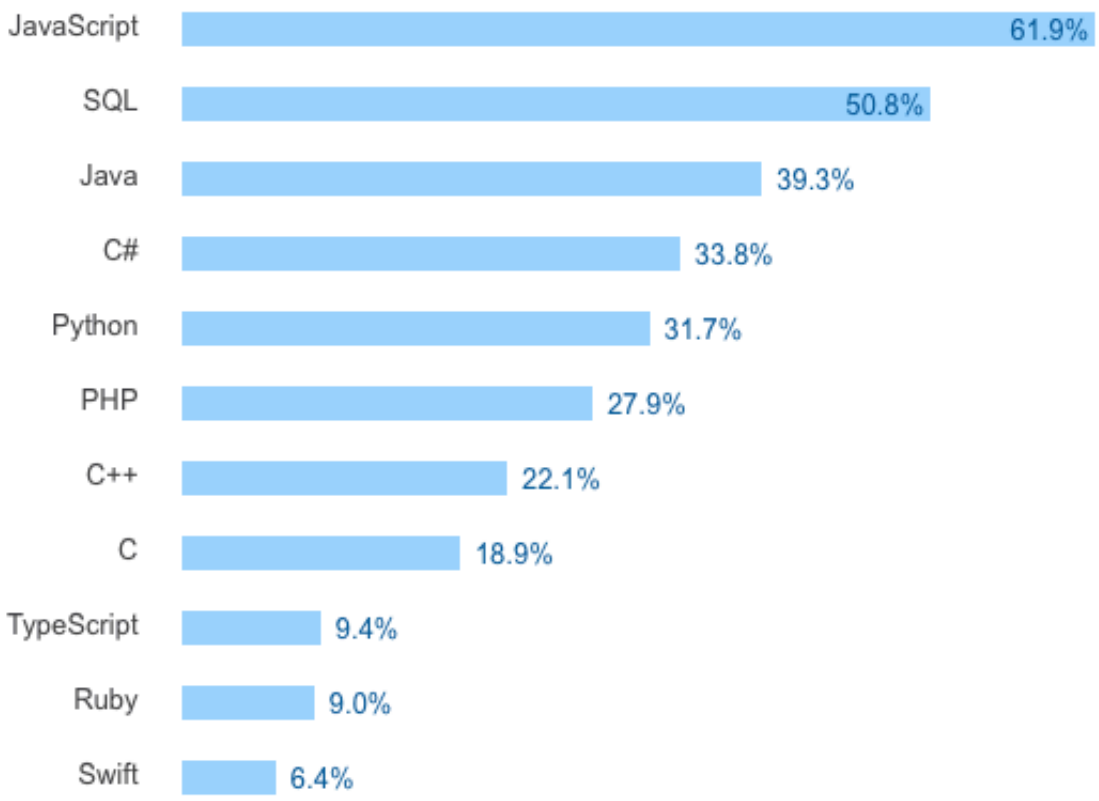
Torsten Grust
Universität Tübingen, Germany

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

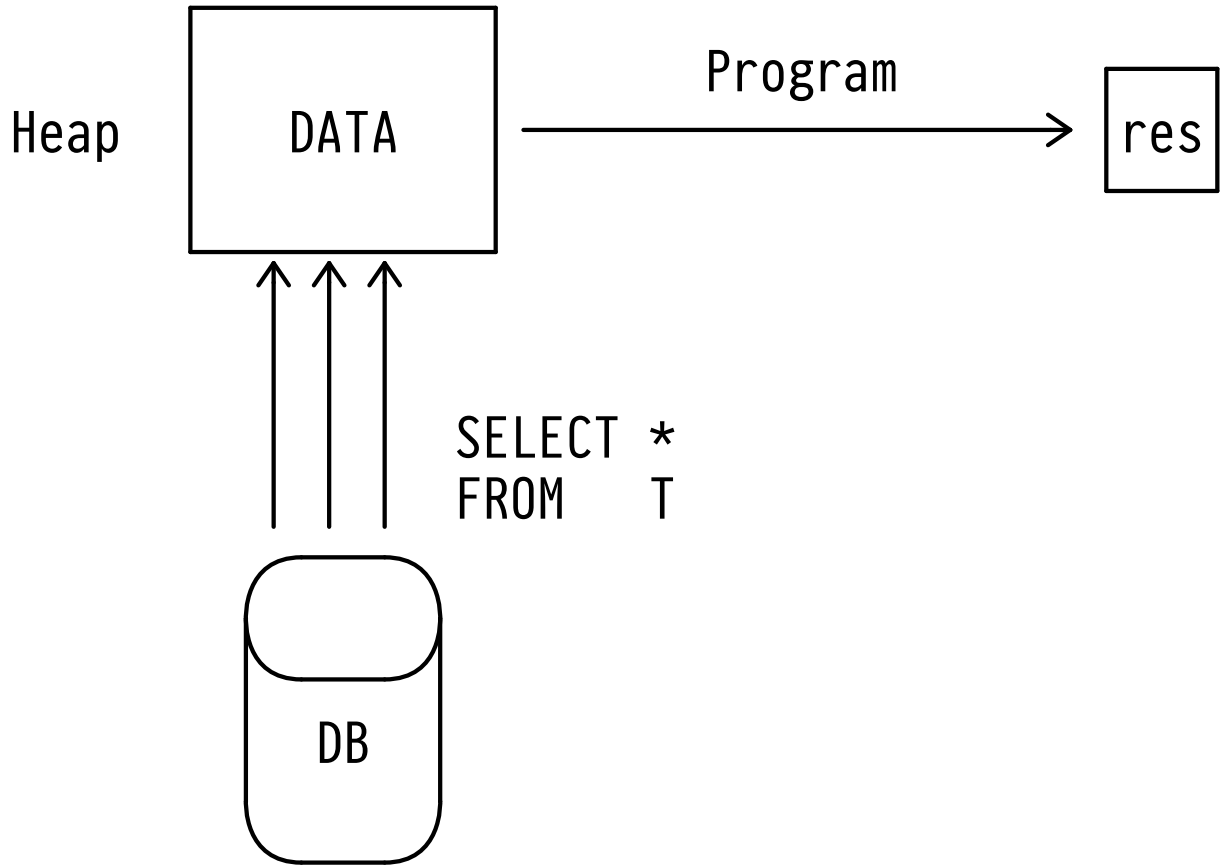
Stack Overflow Developer Survey (March 2017)



Most Popular Technologies — Programming Languages[1]

[1] https://stackoverflow.com/insights/survey/2017

Operating the Database System as a Dumbed Down Table Storage

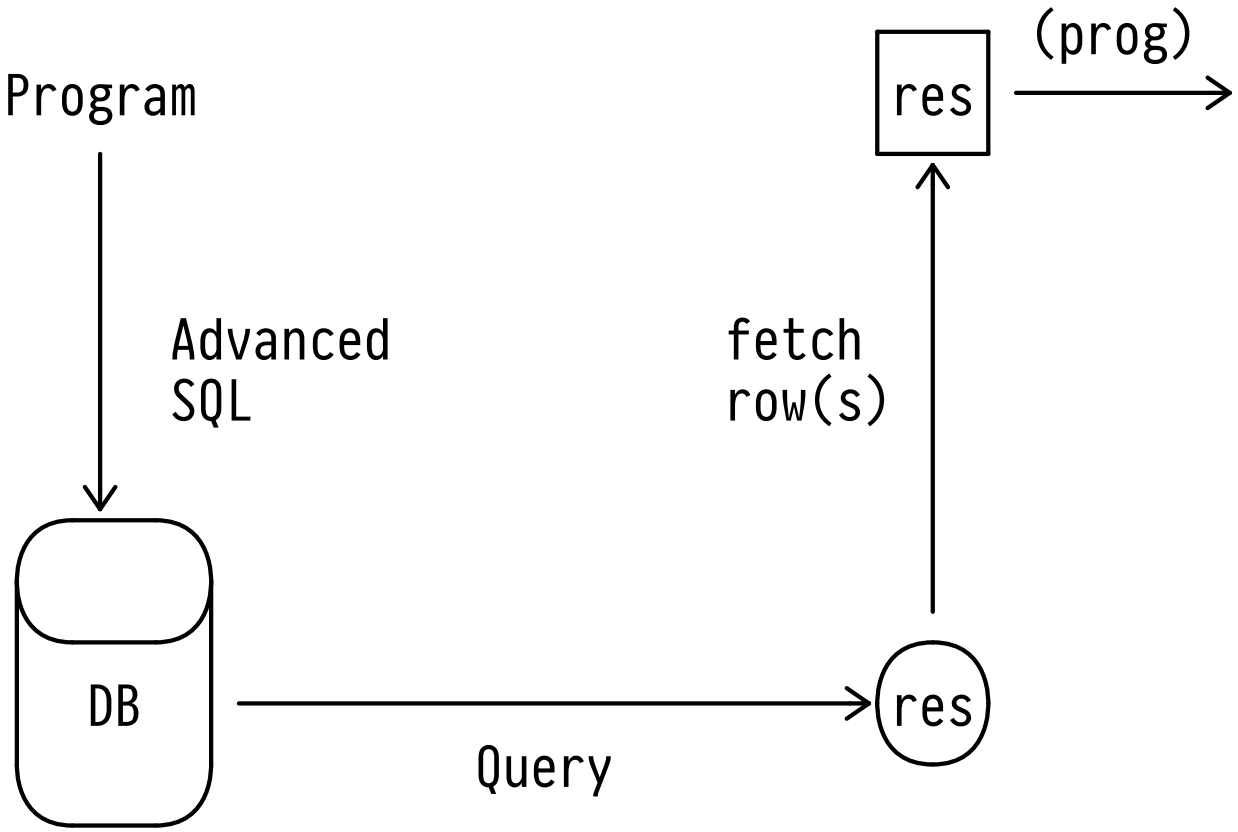


👎 Program- and Heap-Centric Operation of Database System

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.

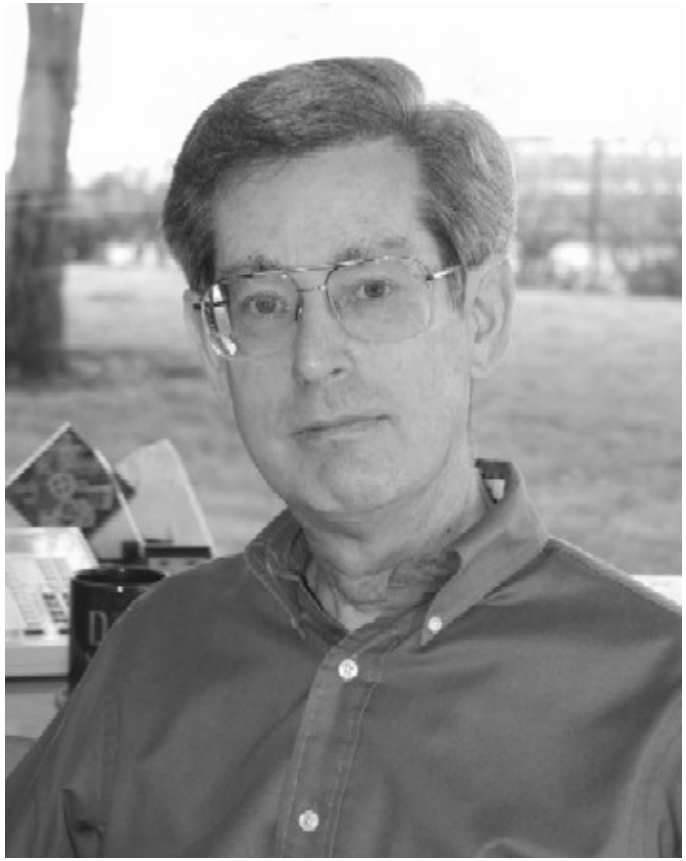
Moving Computation Close to the Data



👍 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).



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 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

- 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 9.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.[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

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

Administrativa

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

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

End-Term Exam

- 90-min **written exam** on July 25, 10:00–12:00 (Room A301).
- 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) via Github .
- 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 additional **SQL code**
- **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

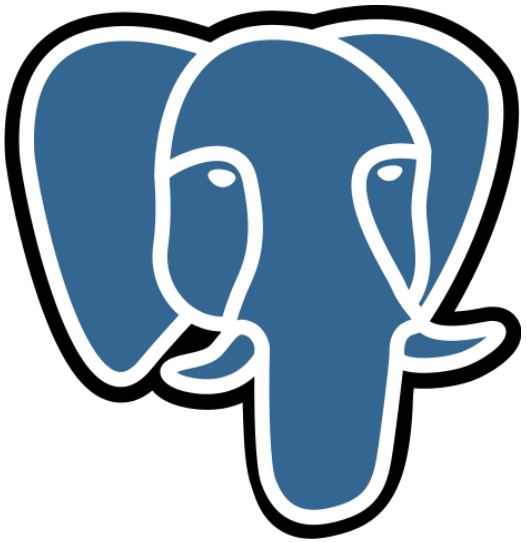
- a variety of scientific papers,
- textbook excerpts,
- blog and mailing list postings, Stack Exchange Q&As,[4]
- SQL references/standards,
- experience, and best practices.

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 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.[5]

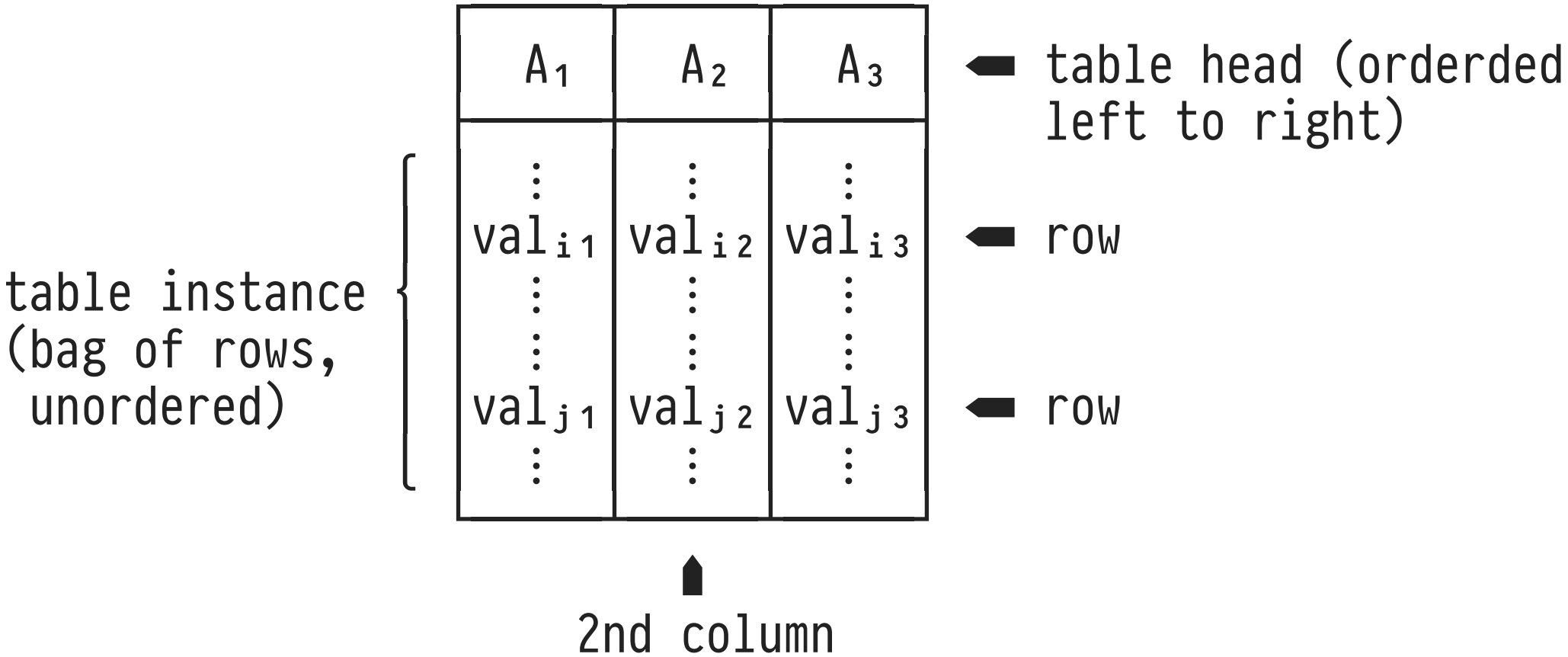
Let us only spend a few moments/slides to recollect the **data model fundamentals** and to synchronize on terminology.

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

[5] Please see *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
⋮	⋮	⋮
$val_{j\,1}$	$val_{j\,2}$	NULL
⋮	⋮	⋮

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

- On table creation, the ith column is assigned a unique **column name** A_i and **column data type** τ_i.
- **Cell values** $val_{j\,i}$, for *any* 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
$\vdots$	$\vdots$	$\vdots$
$val_{j\,1}$	$val_{j\,2}$	$val_{j\,3}$
$\vdots$	$\vdots$	$\vdots$

- SQL tables are in **first normal form (1NF):** all column data types τ_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

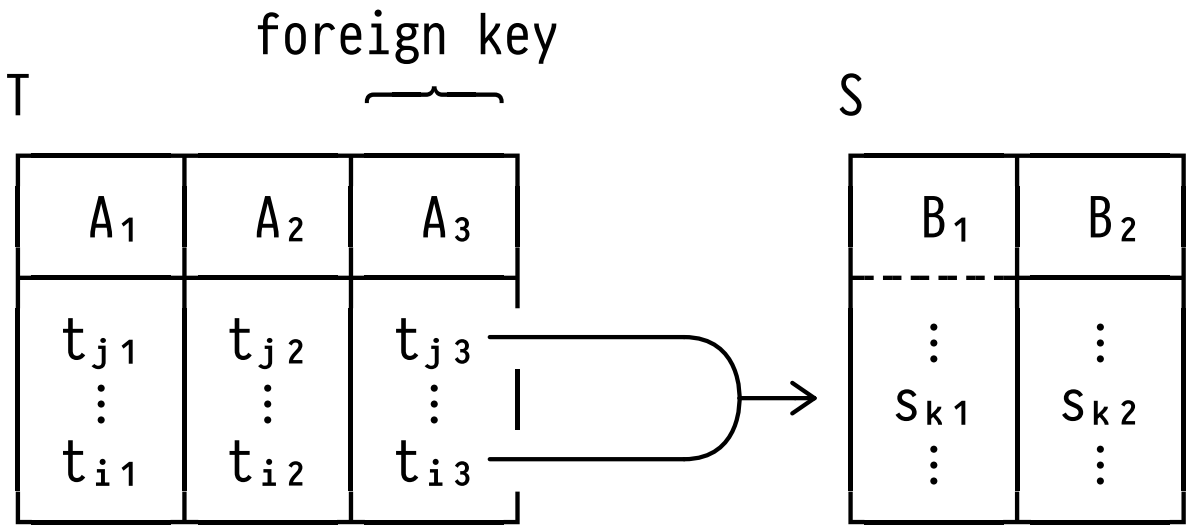
convention in these slides:
---- marks key columns

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