

Programmierpraktikum: Datensysteme

01 Kickoff and Introduction

Prof. Dr. Matthias Boehm

Technische Universität Berlin

Berlin Institute for the Foundations of Learning and Data

Big Data Engineering (DAMS Lab)

Announcements / Org



■ #1 Hybrid & Video Recording

- Hybrid lectures (in-person, zoom) with optional attendance

<https://tu-berlin.zoom.us/j/9529634787?pwd=R1ZsN1M3SC9BOU1OcFdmem9zT202UT09>

- Zoom [video recordings](#), links from website

https://mboehm7.github.io/teaching/ws2627_ppds/index.htm



■ #2 Course Registrations

- TU Berlin ISIS / Fak IV Meta registrations
- Bachelor/Master ratio? CS/WINF/others ratio?

31
~45

#3 Faculty IV - Team Awareness and Antidiscrimination

<https://www.tu.berlin/eecs/awan>



■ Goal

- **Low-barrier approachability** for spectrum of awareness and antidiscrimination issues

■ Team

- Irene Hube-Achter (MTSV)
- Matthias Boehm (professors)
- Ceenu George (professors)
- Nadine Karsten (scientific staff)
- Tom Hersperger (students)



■ Mission Statement

- Account for heterogeneity and complexity of modern societies at TU Berlin
- #1 **Treat all persons with fairness and respect**
- #2 **Ensure a safe environment for all**
- #3 **Comply with our duty of care towards others**
- #4 **Actively support the implementation of the above guidelines and contribute**

Contact: private email,

eecs-TB-awareness@win.tu-berlin.de,

or AwAn@dams.tu-berlin.de

Agenda



- Course Organization
- #1 Transactional In-memory Indexing (DAMS)
- #2 Duplicate Detection (D2IP)
- #3 Provenance Tracking in ML Pipelines (DEEM)
- #4 Fairness Auditing for ML Pipelines (DEEM)
- Course **Selection/Enrolment**

Course Organization

Basic Course Organization



■ Language

- Lectures and slides: **English** (German if preferred)
- Communication and presentations: **English/German**
- **Informal language** (first name is fine)
- Offline **Q&A in forum**, answered by teaching assistants

■ Course Format

- **6 ECTS** (4 SWS) bachelor computer science / information systems
- **Every-other-week lectures** (**Mon 4.15pm sharp**, including **Q&A**), **attendance optional**

■ Prerequisites

- Basic programming skills in languages such as **C, C++, Java**, Rust, Python, etc
- Basic understanding of data management SQL / RA (or willingness to fill gaps)

Course Goals and Structure



■ Objectives

- **Apply basic programming skills** to more complex problem (in self-organized team work)
- Technical focus on data management and data systems
- Holistic programming projects: **prototyping, design, versioning, tests, experiments, benchmarks**

■ Grading: Pass/Fail

- **Project Implementation** (project source code) [**45%**]
- **Component and Functional Tests** (test source code) [**10%**]
- **Runtime Experiments** (achieve performance target) [**15%**]
- **Documentation** (design document up to 5 pages / code documentation) [**15%**]
- **Result Presentation** (10min talk) [**15%**]

■ Academic Honesty / No Plagiarism (incl LLMs like ChatGPT)



- **#1 Efficient Join Pipeline Executor**

- Capacity: 16/48
- Organized by **DAMS** group
- Focus on query processing

- **#2 Duplicate Detection**

- Capacity: 16/48
- Organized by **D2IP** group
- Focus on entity resolution

- **#3 Provenance Tracking in ML Pipelines**

- #4 Fairness Auditing for ML Pipelines**

- Capacity: 16/48
- Organized by **DEEM** group
- Focus on ML pipelines

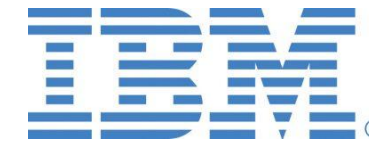
➔ **Admitted Students:**

- 31 + ~15 on ISIS (incl duplicates)
- **Total registrations: up to 48**

#1 Transactional In-memory Indexing (DAMS)

About Me

- **Since 09/2022 TU Berlin, Germany**
 - University professor for Big Data Engineering (DAMS)
- **2018-2022 TU Graz, Austria**
 - BMK endowed chair for data management + research area manager
 - **Data management for data science** (DAMS), **SystemDS & DAPHNE**
- **2012-2018 IBM Research – Almaden, CA, USA**
 - Declarative large-scale machine learning
 - Optimizer and runtime of **Apache SystemML**
- **2007-2011 PhD TU Dresden, Germany**
 - Cost-based optimization of integration flows
 - Time series forecasting / in-memory indexing & query processing



History 1970/1980s Relational Database Systems

Oracle, IBM DB2,
Informix, Sybase
→ MS SQL



Ingres @ UC Berkeley
(Stonebraker et al.,
Turing Award '14)

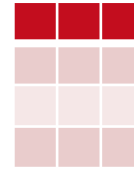
QUEL

System R @ IBM
Research – Almaden
(Jim Gray et al.,
Turing Award '98)

SEQUEL

SQL Standard
(SQL-86)

Tuple Calculus

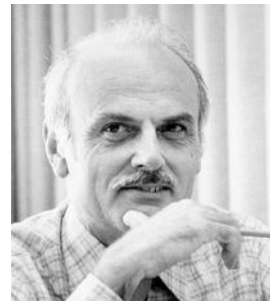


Relational Algebra

Relational Model



- Goal: Data Independence**
(physical data independence)
- Ordering Dependence
 - Indexing Dependence
 - Access Path Dependence



Edgar F. “Ted” Codd @ IBM
Research (**Turing Award '81**)

[E. F. Codd: A Relational Model of
Data for Large Shared Data Banks.
Comm. ACM 13(6), **1970**]



Success of SQL / Relational Model



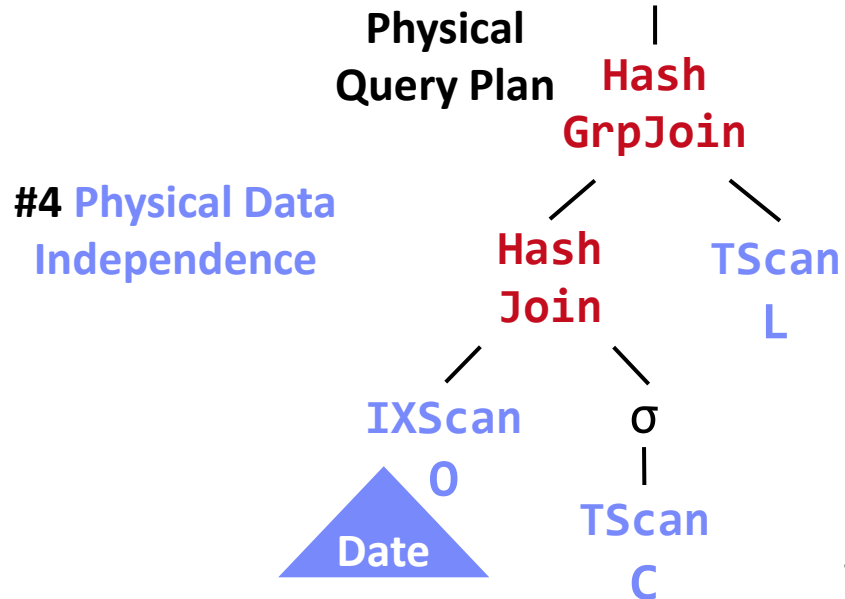
#1 **Declarative:**
what not how

#2 **Flexibility:**
closure property
→ composability

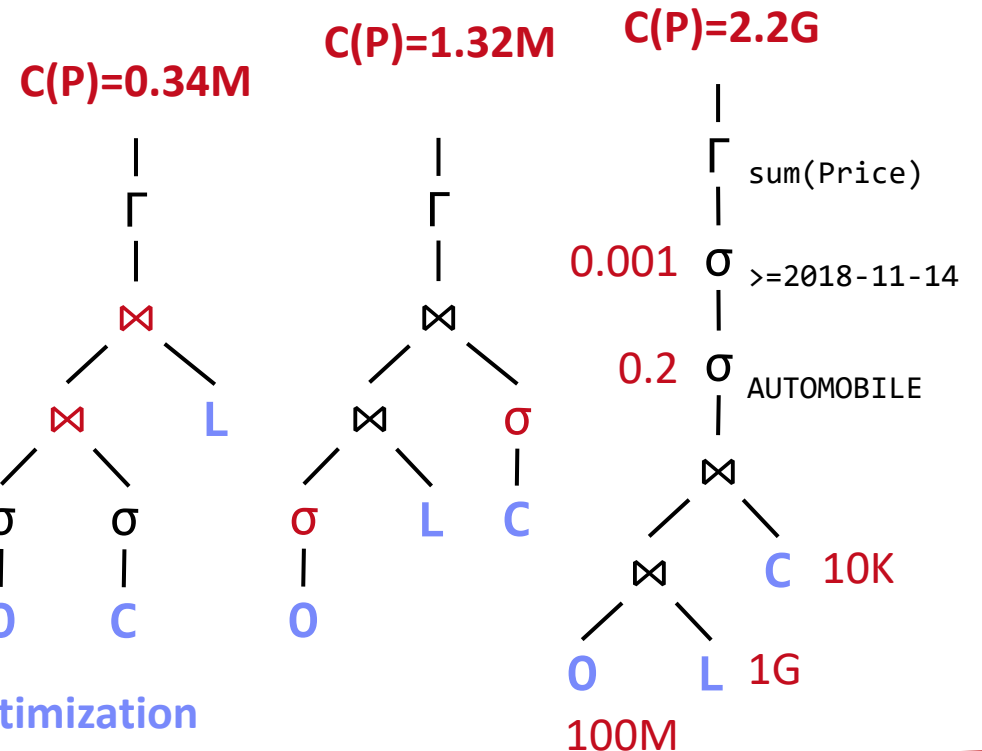
Query:

```
SELECT O_OID, sum(L_Price)
FROM Orders, Lineitem, Customer
WHERE O_OID = L_OID AND O_CID = C_CID
      AND O_Odate >= '2018-11-14'
      AND C_Msegment = 'AUTOMOBILE'
GROUP BY O_OID
```

Logical Query Plans



#3 **Automatic Optimization**



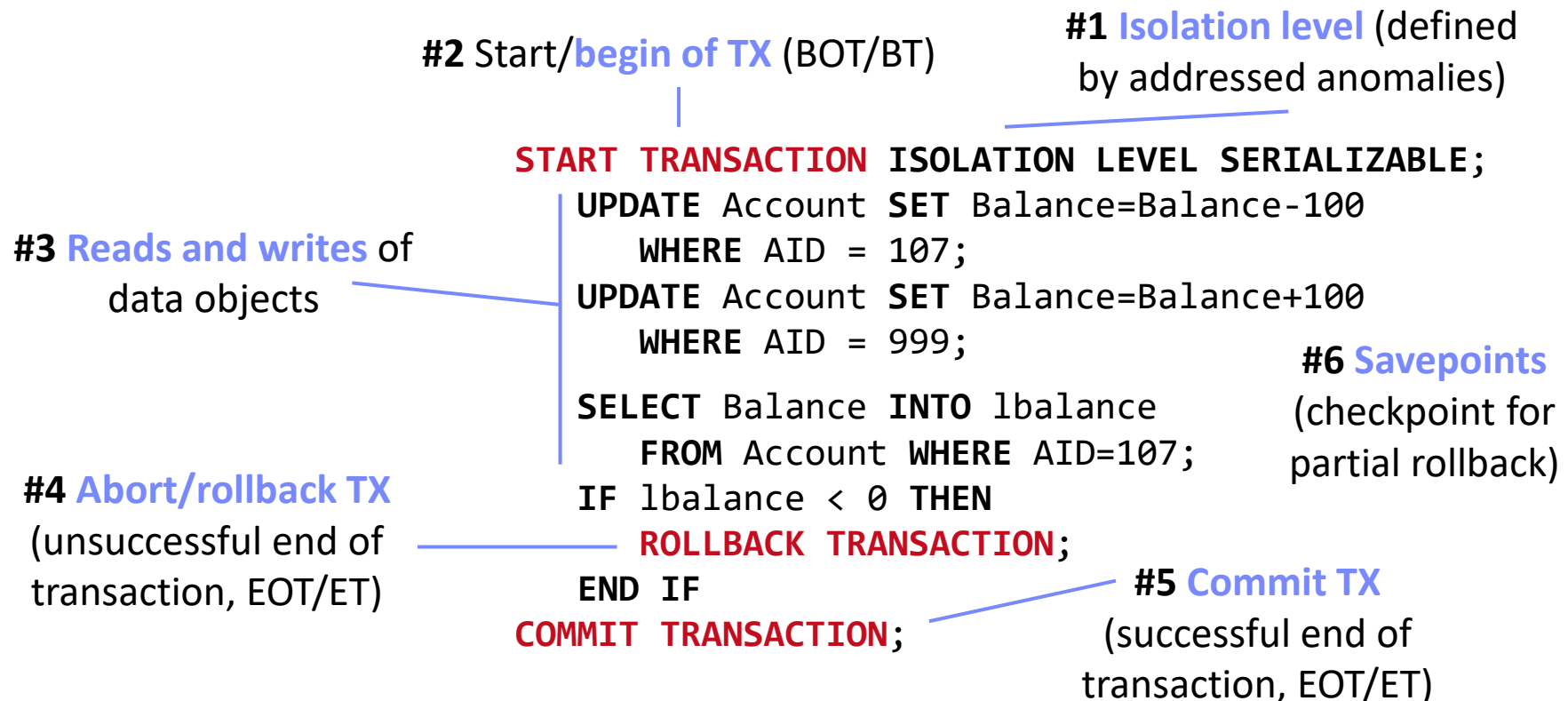
Terminology of Transactions



Database Transaction

- A transaction (TX) is a **series of steps** that brings a database from a **consistent state** into another (not necessarily different) **consistent state**
- ACID properties** (atomicity, consistency, isolation, durability)

Terminology by Example



Overview Programming Project



- **Team**

- **4 person teams** (self-organized team work, but everybody needs to contribute)

- **Task: SIGMOD'09
Programming Contest**

First Annual SIGMOD Programming Contest *Main Memory Transactional Index*

<http://db.csail.mit.edu/sigmod09contest/>

- Transactional, in-memory index for VARCHAR128, INT32, INT64 w/ duplicates
 - C test / performance suites, multi-threaded lookup/scan/insert/delete ops
 - Programming language: **C or C++** recommended, Java possible

- Create a functional implementation of the provided application programming interface (API) that ensures result correctness and high performance for different data types and characteristics
- **API Functions**
server.h

```
// Index Handling
ErrCode create(KeyType type, char *name);
ErrCode drop(char *name);
ErrCode openIndex(const char *name, IdxState **idxState);
ErrCode closeIndex(IdxState *idxState);

// Transaction Handling
ErrCode beginTransaction(TxnState **txn);
ErrCode abortTransaction(TxnState *txn);
ErrCode commitTransaction(TxnState *txn);

// Read and Write Operations
ErrCode get(IdxState *idxState, TxnState *txn, Record *record);
ErrCode getNext(IdxState *idxState, TxnState *txn, Record *record);
ErrCode insertRecord(IdxState *idxState, TxnState *txn, Key *k, const char* payload);
ErrCode deleteRecord(IdxState *idxState, TxnState *txn, Record *record);
```

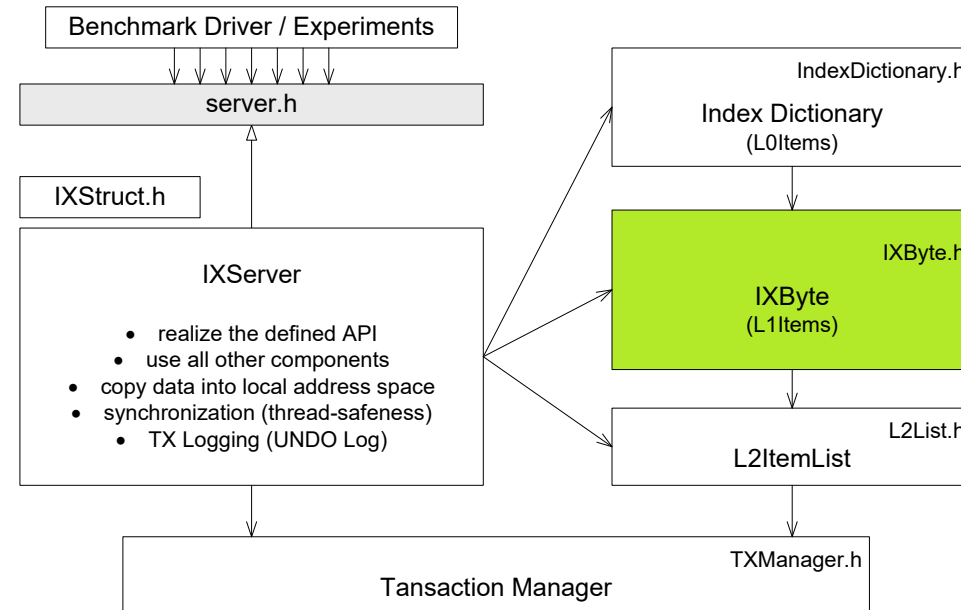
Reference Implementation DEXTER

[Matthias Boehm et al: Efficient In-Memory Indexing with Generalized Prefix Trees. BTW 2011]



■ System Architecture

- Transactional main memory index server
- Manages a set of indices of different data types
- Supports point and range queries as well as inserts and deletes
- Optimistic concurrency control
- TX UNDO log
- Implemented in C



■ Several Subprojects

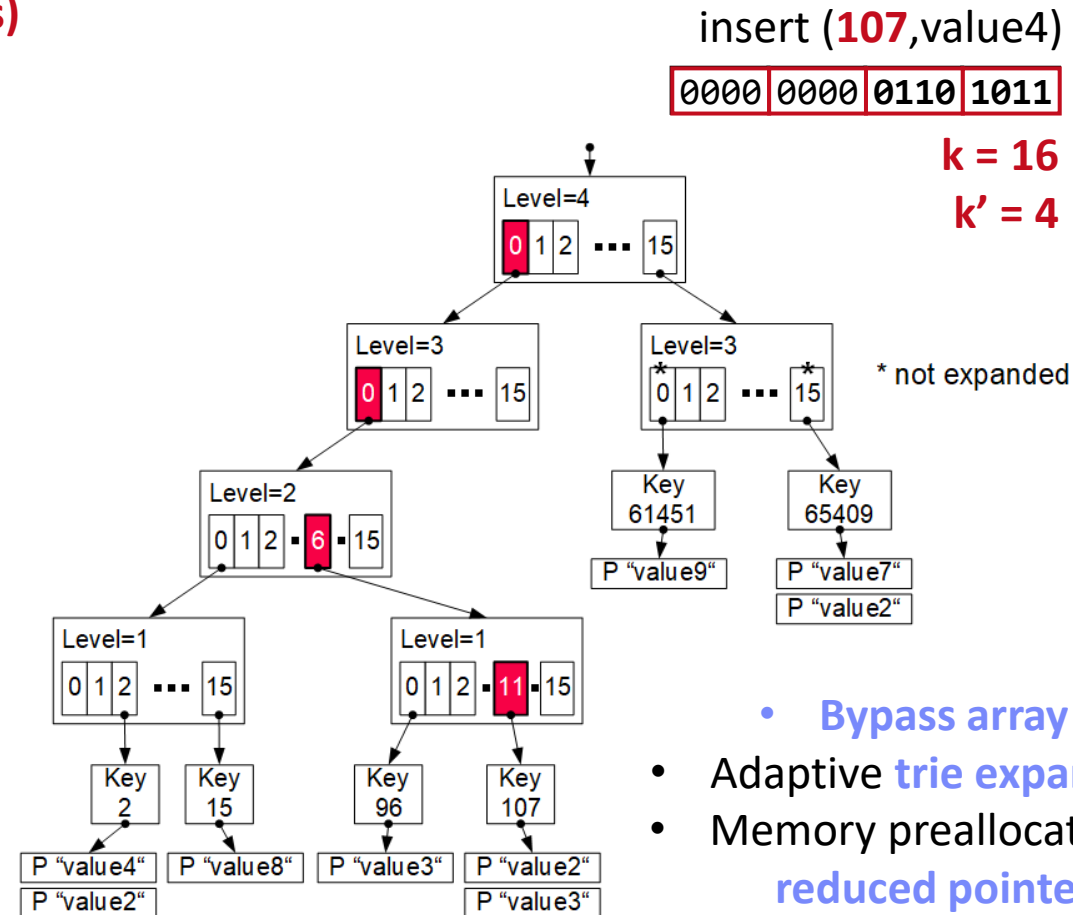
- Core indexing
- Query processing (HPI Future SOC Lab project)
- Mobile devices, GPUs



Reference Implementation DEXTER, cont.

Excursus: Prefix Trees (Radix Trees, Tries)

- **Generalized Prefix Tree**
 - **Arbitrary data types** (byte sequences)
 - **Configurable prefix length k'**
 - Node size: **$s = 2^{k'}$** references
 - Fixed maximum height **$h = k/k'$**
 - Secondary index structure
- **Characteristics**
 - Partitioned data structure
 - **Order-preserving** (for range scans)
 - **Update-friendly**
- **Properties**
 - **Deterministic paths**
 - Worst-case complexity $O(h)$



- Bypass array
- Adaptive trie expansion
- Memory preallocation + reduced pointers

Additional Course Logistics

■ Staff

- **Lecturer:** Prof. Dr. Matthias Boehm
- **Teaching Assistant:** Christina Dionysio, and others if needed

■ Next Dates/Lectures

- **Oct 23:** Course Selection; team preferences, otherwise assignment
- Oct 26: **Background Index Structures**
- Nov 09: **Background Transaction Processing**
- Dec 07: **Experiments and Reproducibility**
- **Jan 25:** Project submissions (**performance target:** 400K TX/second)
- **Feb 01:** Project presentations (10min per team, mandatory attendance)

■ Infrastructure

- Setup your own private Github/Gitlab repository

Each teams gets a mentor
Q&A sessions on demand



Example Speedtest Output:

```
Creating 100 indices
Populating indices 100
Time to populate: 29ms
Testing the indices
Time to test: 1106ms
Testing complete.
    NUM_DEADLOCK: 0
    NUM_TXN_FAIL: 0
    NUM_TXN_COMP: 1,600,000
Overall time to run: 1135ms
```

#2 Duplicate Detection (D2IP)

#3 Provenance Tracking in ML Pipelines (DEEM)

#4 Fairness Auditing for ML Pipelines (DEEM)

Course Selection/Enrolment

Select Your Course



- **#1 Efficient Join Pipeline Executor (DAMS)**
 - Capacity: 16/48
- **#2 Duplicate Detection (D2IP)**
 - Capacity: 16/48
- **#3 Provenance Tracking in ML Pipeline (DEEM)**
 - Capacity: 16/48
- **#4 Fairness Auditing for ML Pipelines (DEEM)**
 - Capacity: 16/48

Thanks

Course Selection/Enrolment
by **Oct 23 EOD**

<https://forms.gle/bgEkEPaKBaTvBZg37>