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)

Last update: Oct 16, 2023

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

DB group

Agenda

- Course Organization
- Background Data Management
- #1 Transactional In-memory Indexing (DAMS)
- #2 Efficient Join Implementations (DIMA)
- 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
  - Weekly/Every-other-week lectures (Mon 4.15pm sharp, including Q&A), attendance optional

- **Prerequisites**
  - Basic programming skills in languages such as C, C++, Java
  - 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)

Sub-Course Offerings

- **#1 Transactional In-memory Indexing**
  - Capacity: 36+/48
  - Organized by DAMS group
  - Broad technical focus
  - Lectures every-other-week

- **#2 Efficient Join Implementations**
  - Capacity: 12/48
  - Organized by DIMA group
  - Focus on low-level systems programming
  - Weekly lectures

➔ **Admitted Students:**
- 48 + 6 waiting list
- 2 with drawn, entire waiting list admitted
- **Total registrations: 52**
  → 13 teams, 4 students each

Background Data Management

History 1970/1980s
Relational Database Systems

SQL Standard
(SQL-**86**)

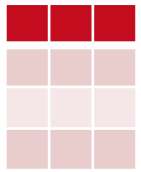
Oracle, IBM DB2,
Informix, Sybase
→ MS SQL

QUEL

SEQUEL

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

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



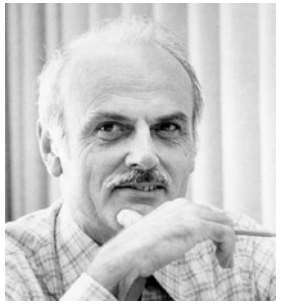
Tuple Calculus

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

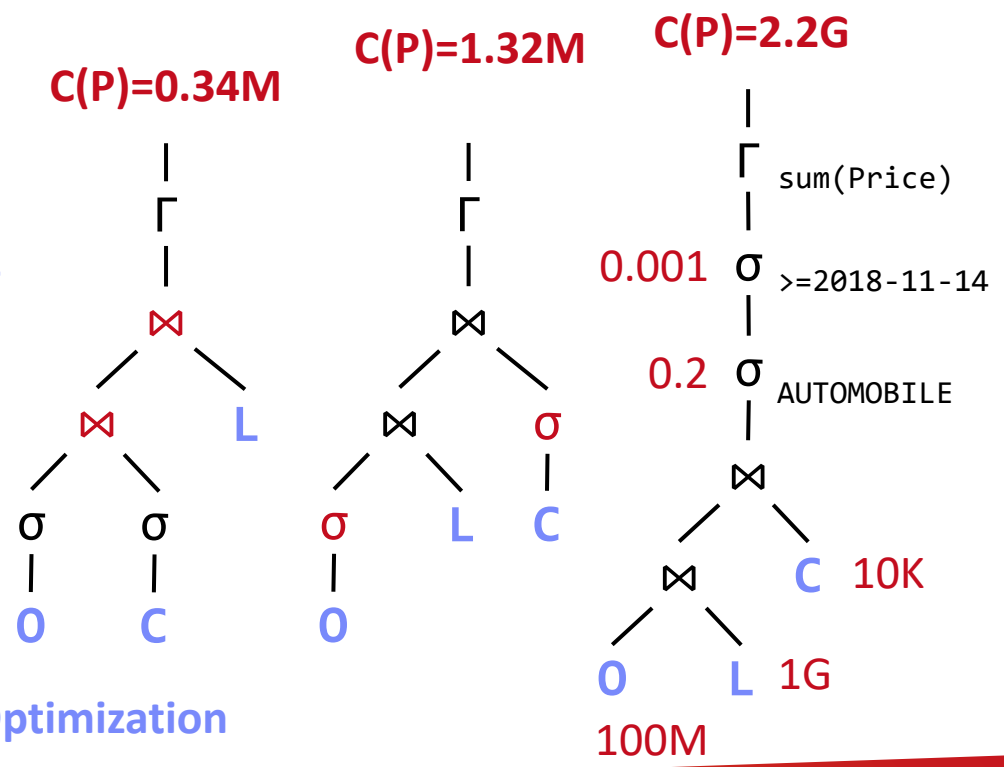
#1 Declarative:
what not how

#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
```

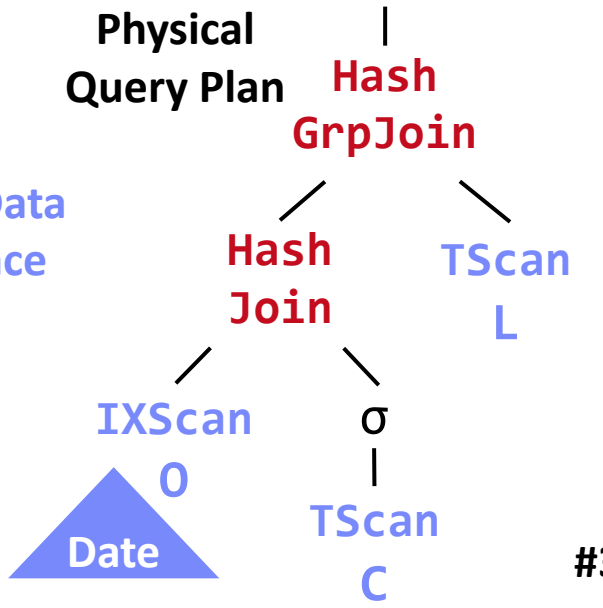
Logical Query Plans



#3 Automatic Optimization

#4 Physical Data Independence

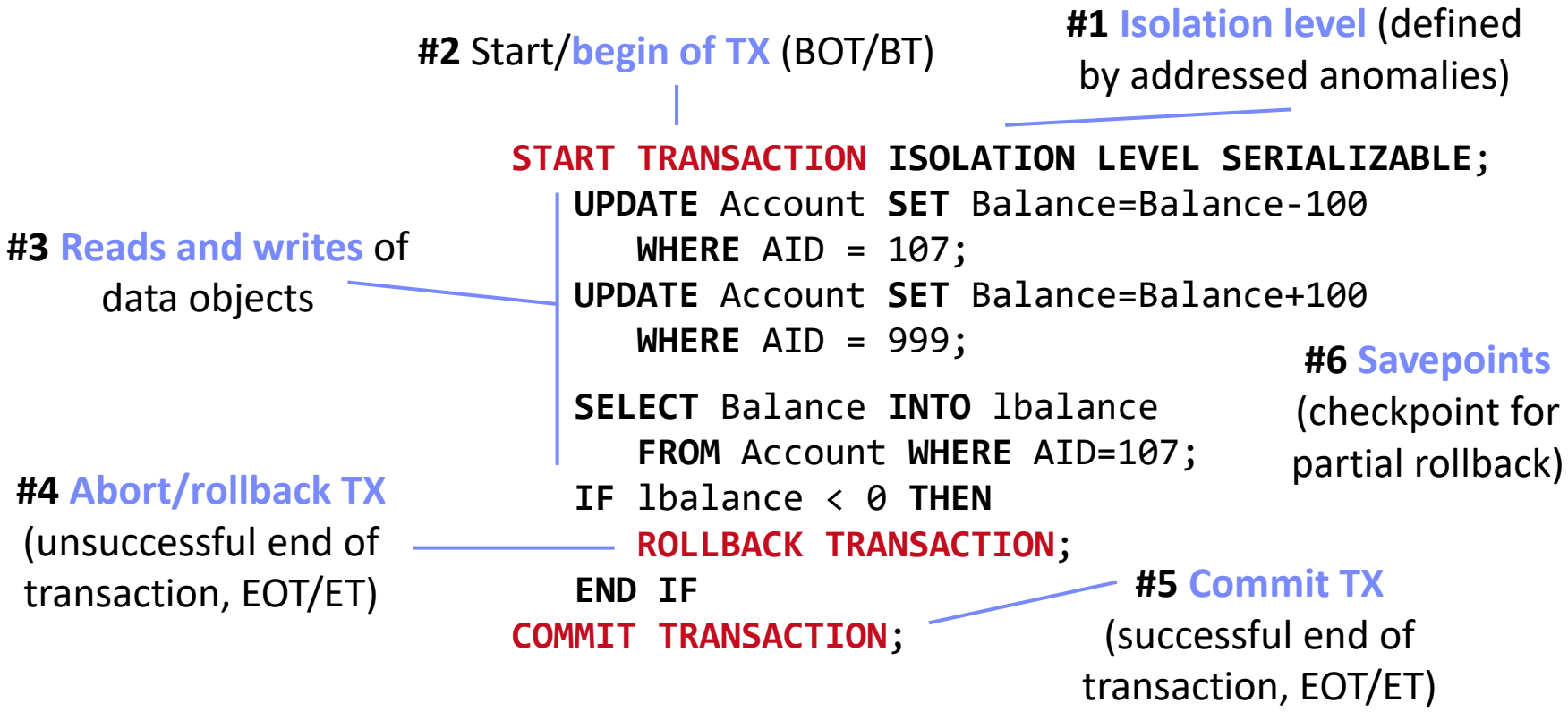
Physical Query Plan



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**



#1 Transactional In-memory Indexing (DAMS)

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

- **Timeline**
  - **Oct 23/Oct 30, 11.59pm:** Team selection and/or assignment
  - **Feb 01, 11.59pm:** Final programming project deadline

API Summary

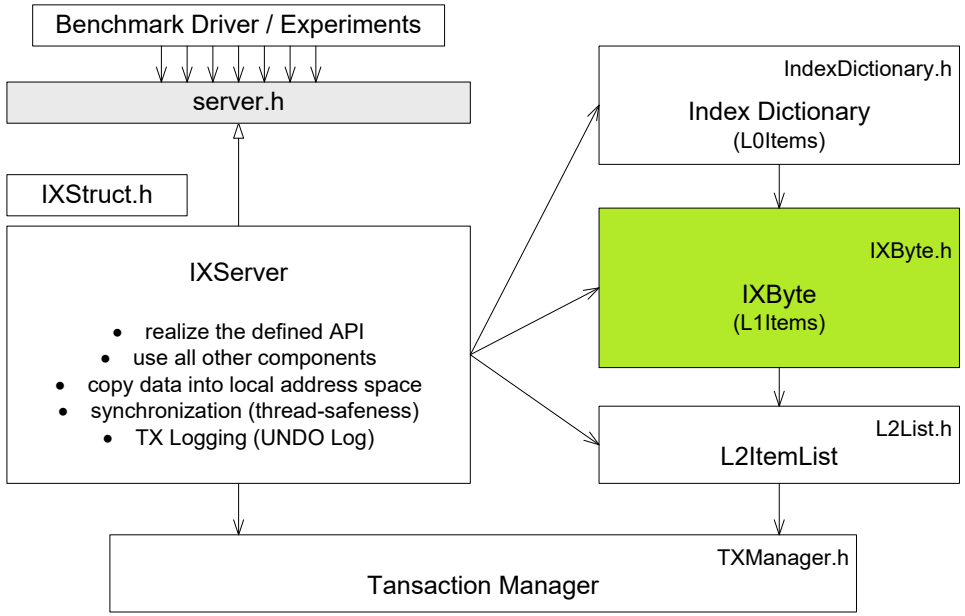
- **Create a functional implementation of the provided application programming interface (API) that ensures result correctness and high performance for different data types and characteristics**
- **API Functions**
  server.h

```
// 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

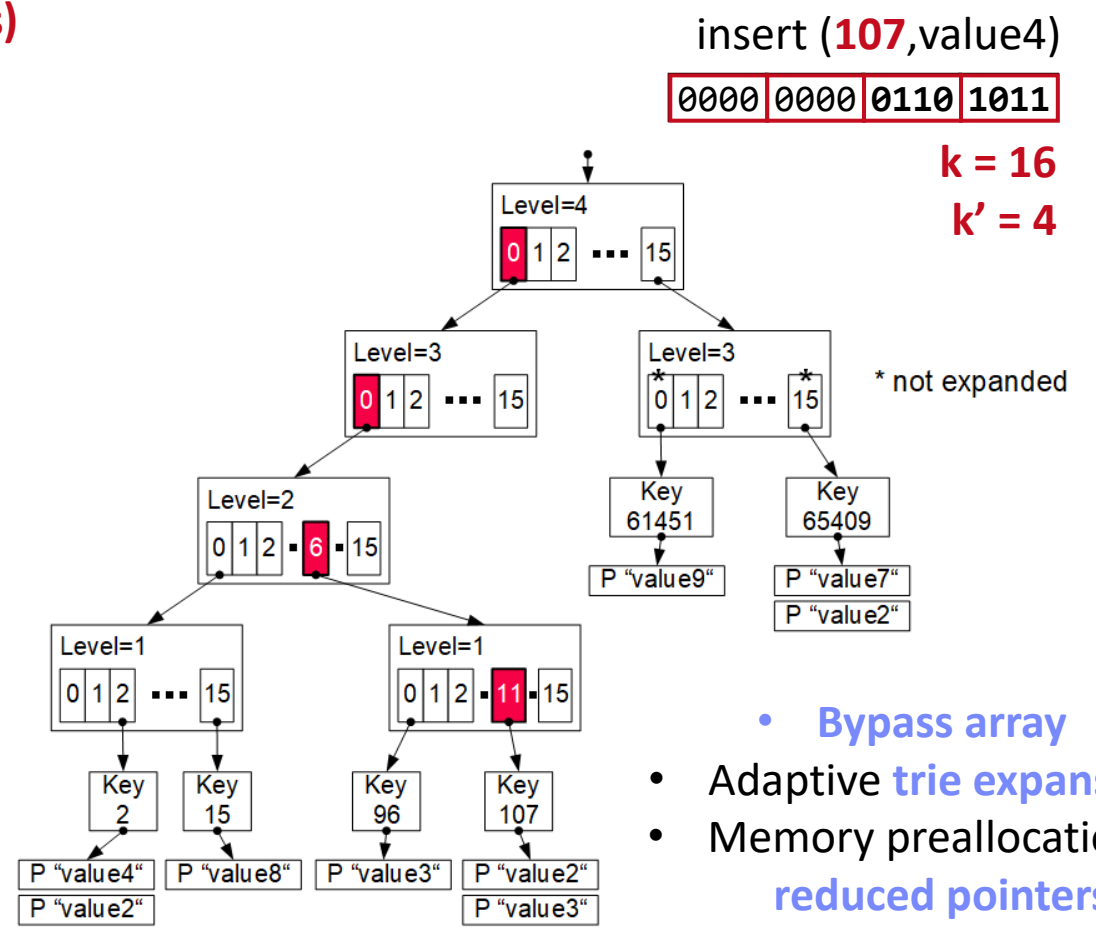


- **Several Subprojects**
  - Core indexing
  - Query processing (HPI Future SOC Lab project)
  - Mobile devices, GPUs

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 Assistants:** Christina Dionysio, David Justen



- **Next Dates/Lectures**
  - Oct 23: Team Selection; otherwise assignment
  - Oct 30: Background Index Structures
  - Nov 13: Background Transaction Processing
  - Nov 27: Experiments and Reproducibility
  - Additional lectures / Q&A sessions on demand
  - **Feb 01:** Project submissions (**performance target:** 400K TXs/second)
  - **Feb 12:** Project presentations (10min per team, mandatory attendance)

- **Infrastructure**
  - Setup your own private Github/Gitlab repository

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 Efficient Join Implementations (DIMA)

Course Selection/Enrolment

Select Your Course

- **#1 Transactional In-memory Indexing (DAMS)**
  - Capacity: 36+/48
- **#2 Efficient Join Implementations (DIMA)**
  - Capacity: 12/48

https://tinyurl.com/ymfkb3s2

Summary & QA

- Course Organization
- Background Data Management
- #1 Transactional In-memory Indexing (DAMS)
- #2 Efficient Join Implementations (DIMA)
- Course Selection/Enrolment

Thanks