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 13, 2024

Agenda

- Course Organization
- Background Data Management
- #1 Disk-based B-Trees (DAMS)
- #2 Duplicate Detection (D2IP)
- #3 Provenance Tracking in 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, 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)

Sub-Course Offerings

- **#1 Disk-based B-Trees**
  - Capacity: 48/80
  - Organized by **DAMS** group
  - Focus on index structures
  - Lectures every-other-week in **H 0111**
- **#2 Duplicate Detection**
  - Capacity: 16/80
  - Organized by **D2IP** group
  - Focus on entity resolution
- **#2 Provenance Tracking in ML Pipelines**
  - Capacity: 16/80
  - Organized by **D2IP** group
  - Focus on entity resolution

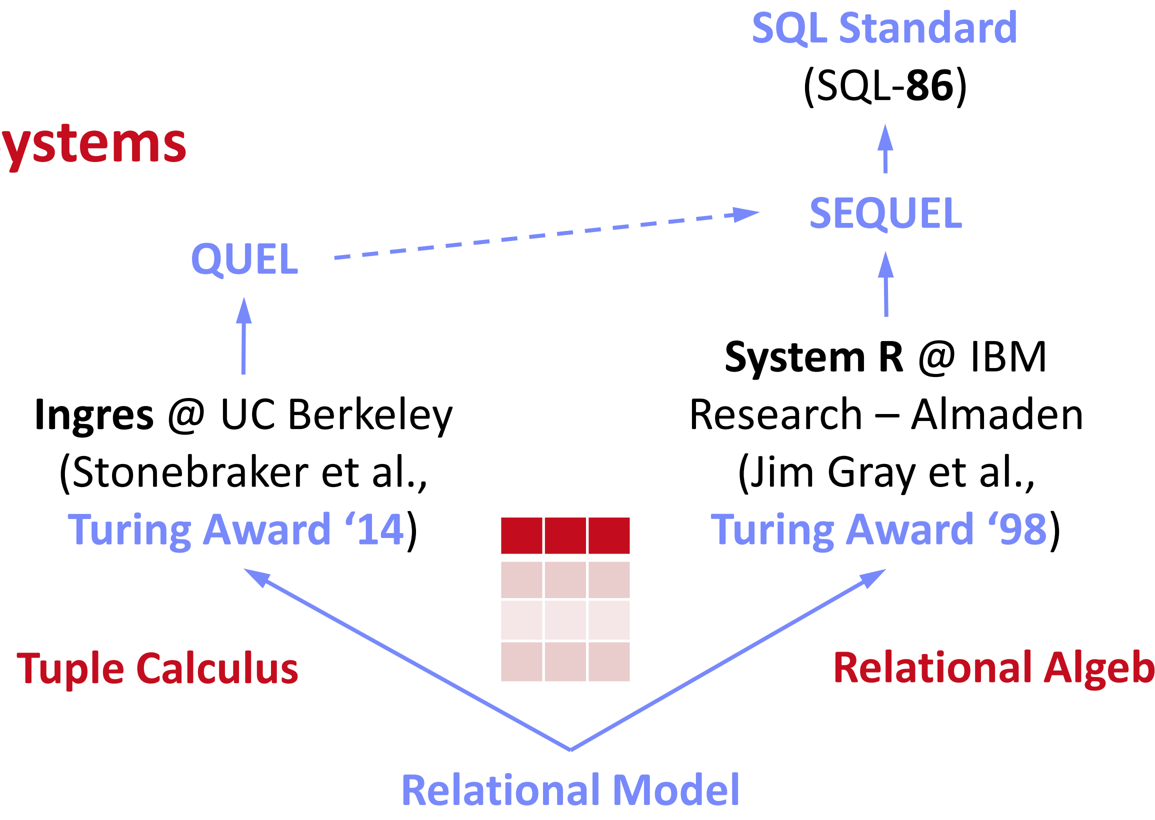
→ **Admitted Students:**

- 5 + 48 on ISIS (incl duplicates)
- **Total registrations: up to 80**

  → 20 teams, 4 students each

Background Data Management

History 1970/1980s
Relational Database Systems



SQL Standard (SQL-**86**)

SEQUEL

QUEL

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

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

Tuple Calculus

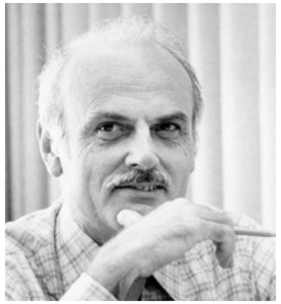
Relational Algebra

Relational Model

Oracle, IBM DB2, Informix, Sybase → MS SQL

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

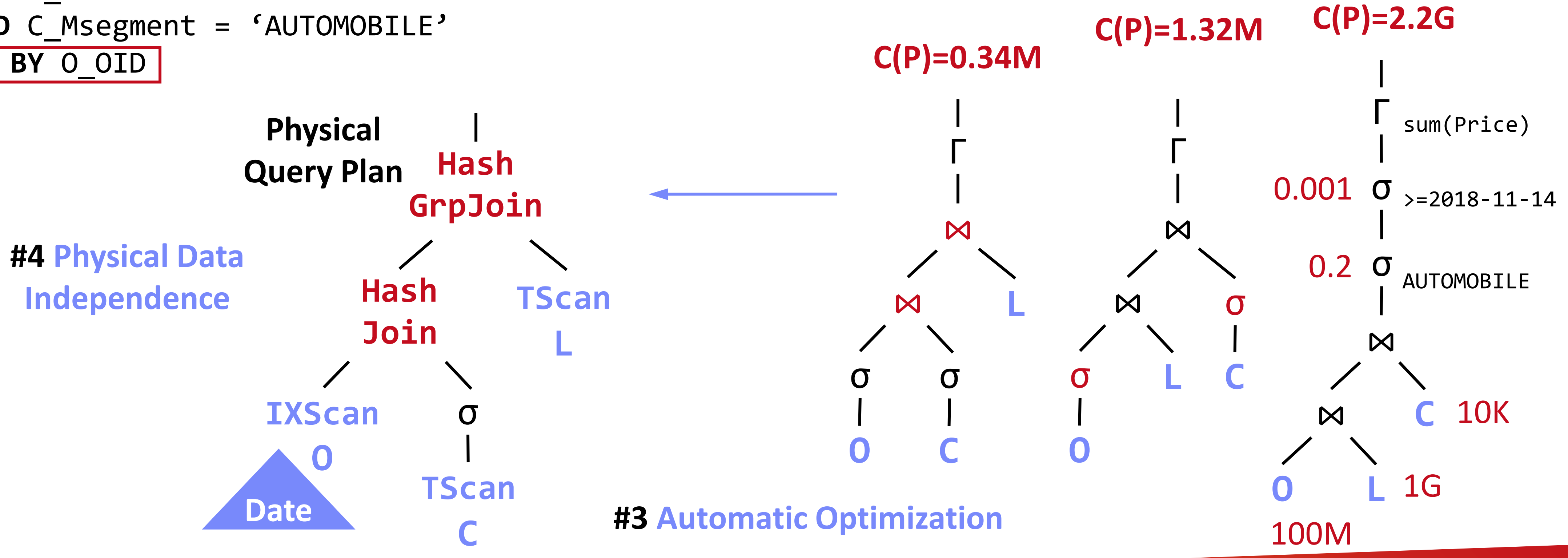
#1 Declarative:
what not how

#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



#4 Physical Data Independence

#3 Automatic Optimization

#1 Disk-based B-Trees (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

DB group

Additional Course Logistics

- **Staff**
  - **Lecturer:** Prof. Dr. Matthias Boehm
  - **Teaching Assistants:** Christina Dionysio, Ramon Schöndorf

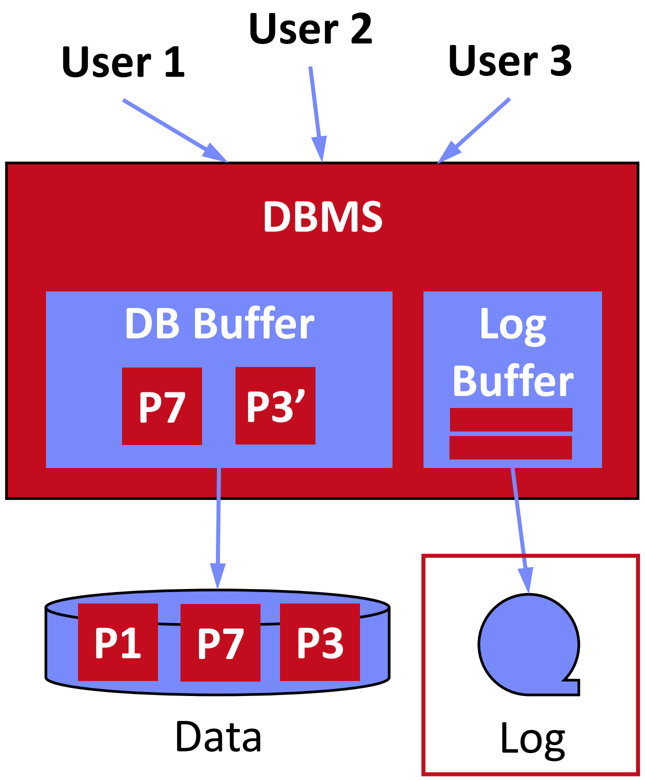
Each teams gets a mentor
Q&A sessions on demand

- **Next Dates/Lectures**
  - Oct 21: Course Selection; team preferences, otherwise assignment
  - Oct 28: **Background Index Structures**
  - Nov 11: **Background Buffer Pool**
  - Nov 28: **Background Transaction Processing**
  - Dec 12: **Experiments and Reproducibility**
  - **Jan 27:** Project submissions (**performance target:** 20K transactions/second)
  - **Feb 03:** Project presentations (10min per team, mandatory attendance)

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

Overview Database (Transaction) Log

- **Database Architecture**
  - **Page-oriented storage** on disk and in memory (DB buffer)
  - Dedicated **eviction algorithms**
  - Modified in-memory pages marked as dirty, flushed by cleaner thread
  - **Log:** append-only TX changes
  - Data/log often placed on different devices and periodically archived (backup + truncate)



- **Write-Ahead Logging (WAL)**
  - The log records of changes to some (dirty) data page must be on **stable storage before the data page** (UNDO - atomicity)
  - **Force-log on commit** or full buffer (REDO - durability)
  - **Recovery:** forward (REDO) and backward (UNDO) processing
  - Log sequence number (LSN)

[C. Mohan, Donald J. Haderle, Bruce G. Lindsay, Hamid Pirahesh, Peter M. Schwarz: ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging. **TODS 1992**]

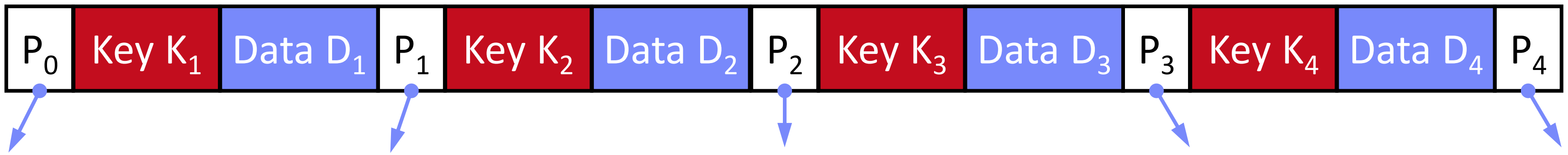
B-Tree Overview

[Rudolf Bayer, Edward M. McCreight: Organization and Maintenance of Large Ordered Indices. **Acta Inf. (1) 1972**]

- **History B-Tree**
  - **B**ayer and McCreight 1972, **B**lock-based, **B**alanced, **B**oeing Labs
  - **Multiway tree** (node size = page size); designed for DBMS
  - Extensions: **B+-Tree/B*-Tree** (data only in leafs, double-linked leaf nodes)
- **Definition B-Tree (k, h)**
  - All paths from root to leafs have equal length h
  - All nodes (except root) have **[k, 2k]** key entries
  - All nodes (except root, leafs) have **[k+1, 2k+1]** successors
  - Data is a record or a reference to the record (RID)

$$\lceil \log_{2k+1}(n+1) \rceil \le h \le \left\lceil \log_{k+1}\left(\frac{n+1}{2}\right)\right\rceil + 1$$

All nodes adhere to max constraints



P_0 | Key K_1 | Data D_1 | P_1 | Key K_2 | Data D_2 | P_2 | Key K_3 | Data D_3 | P_3 | Key K_4 | Data D_4 | P_4

k=2

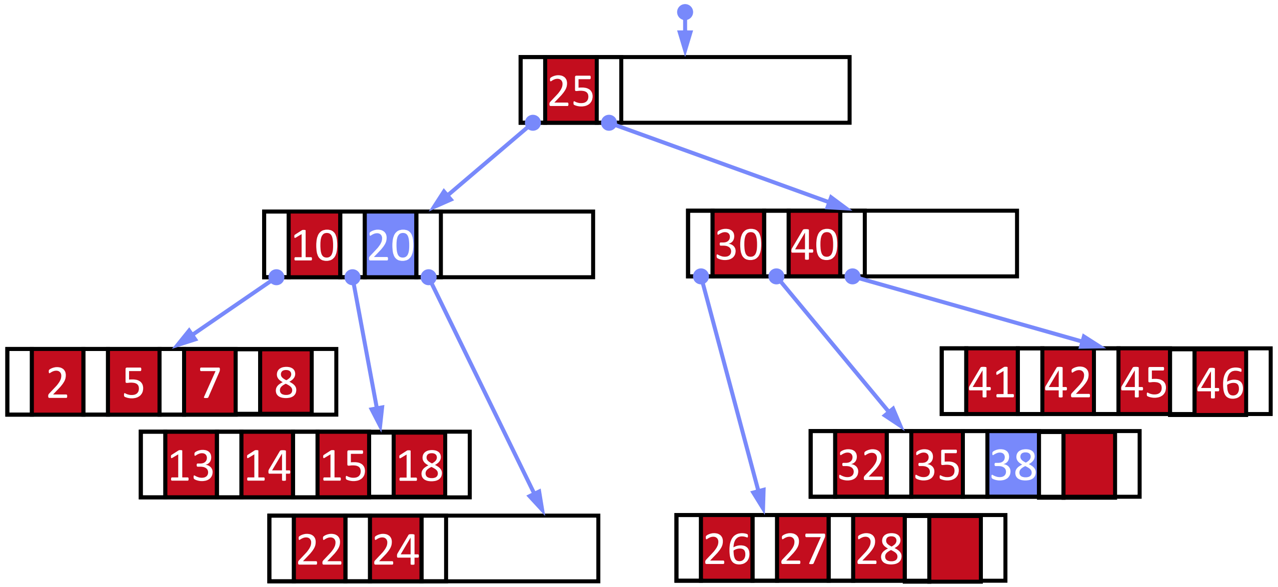
Subtree w/ keys ≤ K_1

Subtree w/ K_2 < keys ≤ K_3

pointers are page IDs

B-Tree Overview – Search

- **Example B-Tree k=2**
  - Get 38 → D38
  - Get 20 → D20
  - Get 6 → NULL



- **Lookup Q_K within a node**
  - Scan / binary search keys for Q_K, if K_i=Q_K, return D_i
  - If node does not contain key
    - If leaf node, abort search w/ NULL (not found), otherwise
    - Decent into subtree Pi with $K_i < Q_K \leq K_{i+1}$
- **Range Scan $Q_{L<K<U}$**
  - Lookup Q_L and call next K while K<Q_U (keep current position and node stack)

Query Processing – Iterator Model

[Goetz Graefe: Volcano - An Extensible
and Parallel Query Evaluation System.
IEEE Trans. Knowl. Data Eng. 1994]

Scalable (small memory)

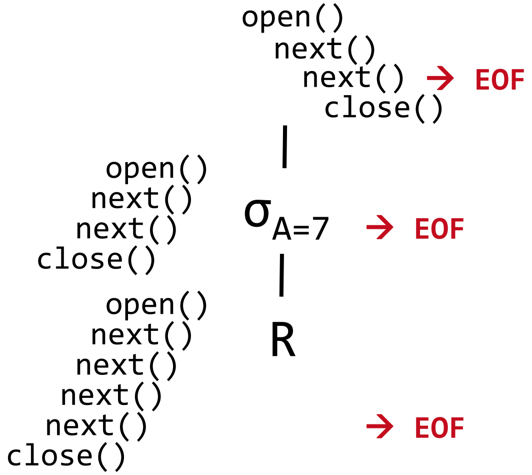
High CPI measures

- **Volcano Iterator Model**
  - **Open-Next-Close** (ONC) interface
  - Query execution from root node (pull-based) → **Pipelined**
- **Example**
  $\sigma_{A=7}(R)$

```
void open() { R.open(); }

void close() { R.close(); }

Record next() {
  while( (r = R.next()) != EOF )
    if( p(r) ) //A==7
      return r;
  return EOF;
}
```



PostgreSQL: Init(), GetNext(), ReScan(), MarkPos(), RestorePos(), End()

- **Blocking Operators**
  - Sorting, grouping/aggregation, build-phase of (simple) hash joins

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 concurrent operations
  - **Programming language:** **C or C++** recommended, **Java or Rust**

  - **WiSe 23/24:** in-memory indexing w/ perf target 400K TXN/second
  - **WiSe 24/25:** **disk-based b-tree w/ perf target 20K TXN/second**
    no VARCHAR and fixed payload length

```
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
```

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, size_t pageSize);
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); //guarantee durability!

// 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);
```

#2 Duplicate Detection (D2IP)

#3 Provenance Tracking in ML Pipelines (DEEM)

Course Selection/Enrolment

Select Your Course

- **#1 Disk-based B-Trees (DAMS)**
  - Capacity: 48/80
- **#2 Duplicate Detection (D2IP)**
  - Capacity: 16/80
- **#3 Provenance Tracking in ML Pipeline (DEEM)**
  - Capacity: 16/80

https://forms.gle/HFvzPCHHpcyZis8KA

Summary & QA

- Course Organization
- Background Data Management
- #1 Disk-based B-Trees (DAMS)
- #2 Duplicate Detection (D2IP)
- #3 Provenance Tracking in ML Pipelines (DEEM)
- Course Selection/Enrolment by Oct 21 EOD

Thanks

https://forms.gle/HFvzPCHHpcyZis8KA