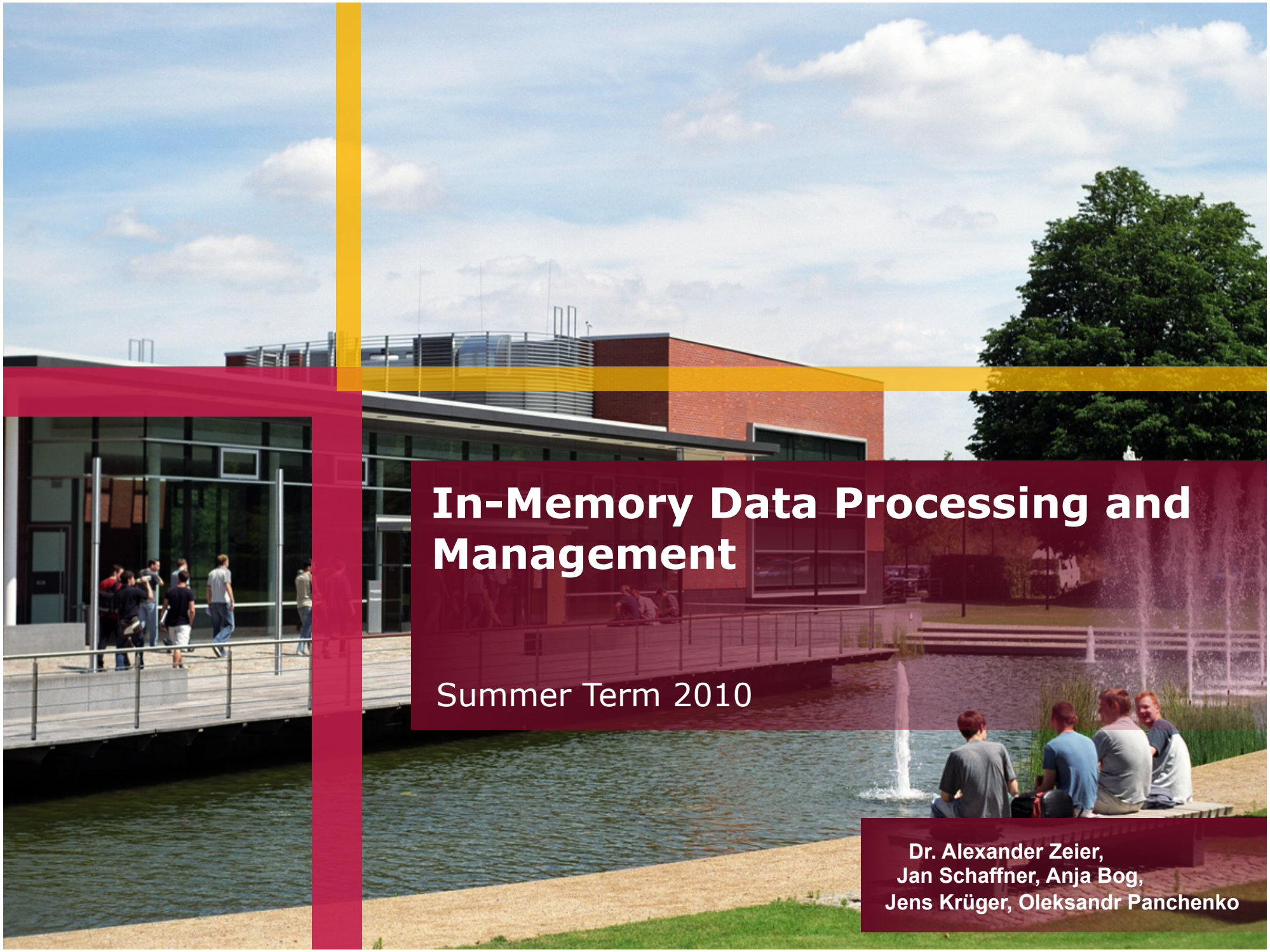

In-Memory Data Processing and Management

Summer Term 2010

Dr. Alexander Zeier,
Jan Schaffner, Anja Bog,
Jens Krüger, Oleksandr Panchenko

Seminar topics

Topics by Jens Krüger

- Data structures for In-Memory Databases (IMDB)
  - F1: Databases organizes data either row- or column-wise
  - F2: With regards to the workload, the one or the other is advantageous.
  - Q: What is the optimal structure? Where is the tipping point?
- Variants of in-memory column stores.
  - F1: Column stores can be implemented either by using a tuple ID or the direct array offset.
  - F2: TupleID's introduce an overhead but enable sorting of columns independently.
  - Q: What is the best implementation in a mixed workload environment?

Topics by Jan Schaffner

- Build a simulation environment
  - Challenges: Balance both overlap and bytes scanned
  - Should be able to run both greedy heuristics and fancy machine learning algorithms
- Build an application on top of the Rock framework
  - Pick your favorite scenario
  - Do something „presentable“
- Build an on-line visualization of the Rock cluster state
  - Show active EC2 nodes and tenant layout
  - Visualize current query workload, tenant overlap, failures, migrations, merges, …

Topics by Jens Krüger

- Leveraging compression in In-Memory Databases (IMDB)
  - F1: The challenge of IMDB's: The main memory is the new bottleneck!
  - F2: Compression reduces the data amount and can be used for improving query execution (late materialization).
  - Q: To what extend can compression improve a IMDB?
- Data organization in Write-optimized Stores (WOS)
  - F1: Even if kept completely in main memory logging to disk is still necessary
  - F2: Due to data compression the IMDB is separated into a compressed read-optimized store and a write-optimized buffer
  - Q: What is the optimal physical data representation of this buffer? Rows vs. Columns & compressed vs lightweight compression.

Topics by Anja Bog

- Data Model Evolution in OLTP and OLAP
  - Historical development of specific data models
    - Classification of scenarios to apply different data models
    - Explore and compare the data models employed
  - State-of-the-art data schemes for OLTP and OLAP
    - Argumentation behind today's specialized models
    - Normalization vs. denormalization
    - Materialized aggregates
- Database Self-Tuning Capabilities
  - Explore mechanisms behind self-tuning of databases
    - Analyze issues and find a fitting solution
    - Mechanisms how and when to adapt the database
  - Focus on physical database structures, i.e. dynamic creation and deletion of e.g. Indexes, Views, Partitions, etc.

Topics by Jens Krüger

- Optimizing the compaction process
  - F1: Due to data compression at a certain point in time the buffer has to be merged with the already compressed data.
  - F2: There are several strategies to execute this process
  - Q: What is the optimal strategy under certain workloads?
- Indices for IMDB's
  - F1: Indices can improve certain types of queries
  - F2: But: maintaining index structures introduces an overhead.
  - Q: Which cache-aware index variant is optimal for certain queries?
- Order-preserving data dictionaries
  - F1: Order-preserving dictionaries can be used for late materialization during query execution
  - Q: Is the overhead of maintaining the sort order justified?

Topics by Oleksandr Panchenko

- Execution of SQL queries with a great number of WHERE-conditions
  - Some scenarios require a great number of WHERE-conditions.
  - Current database query optimizers disregard properties of conditions, such as, distribution, influence of each condition on the result set, relations between conditions.
  - The goal is elaborating a technique which analyses the properties of conditions for constructing query trees. Of course, for one selected scenario.
- XPath interface for querying traces stored in SQlite
  - Some scenarios require other structural languages than SQL (e.g., XPath for XML documents / trees).
  - The goal is an exemplary implementation of an XPath interface to structural data which is stored in a relational table.