Develop your own Database 2019/2020

Week 1
Outlook

1. High-Level Overview
2. First Work Package
3. Organizational Stuff
What can you expect?

• Better understand how in-memory databases work
• Learn how to familiarize yourself with a larger code base
• Gain experience in systems development
• Improve your C++(2a) skills
• Work in small teams on a larger project

If this sounds interesting to you, you are in the right room.
Timeline

1. Sprint
   Simple Table Functionality (Segments, Chunks, Types)

2. Sprint
   Dictionary Compression

3. Sprint
   First Operators (Scan, Sort, ...)

Group Phase
   It’s all about performance
   • Optimizer Rules
   • Self-Tuning
   • and more...

Final Presentation
Timeline

• In addition to introducing you to the architecture, the first two sprints aim at
  – refreshing your C++ knowledge
  – getting you up to speed with our code style, test setup, and expectations

• If you and C++ are on a first-name basis, this might appear a bit slow - please bear with us
What do we expect?

• Fruitful discussions about why we do things the way we do
• Active participation in the group work and our meetings
What do we hope for?

1. Generate interest in our research
2. Continue to work with you in Master’s theses, Hiwi jobs, ...

If anyone is interested right away, please contact us.
Who are we?

Markus Dreseler
- Non-Volatile Memory

Jan Kossmann
- Self-Driving Databases

Martin Boissier
- Data Aging & Tiering

Thomas Bodner
- Cloud-based Databases

Stefan Halfpap
- Replication
Introducing Opossum
Introducing Opossum

- Opossum is the (1) prototypical, (2) columnar (3) in-memory database that we will build during the first three sprints
- Prototypical: We do not plan for Opossum to be used in a productive environment
- Columnar: We exclusively use columnar orientation for data
- In-Memory: All data that we work with is stored in RAM
Why In-Memory?

<table>
<thead>
<tr>
<th>CPU Registers</th>
<th>1 ns</th>
<th>your memory (1 sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU L3 Cache</td>
<td>10 ns</td>
<td>your desk (10 sec)</td>
</tr>
<tr>
<td>DRAM</td>
<td>100 ns</td>
<td>the next office (01:40 min)</td>
</tr>
</tbody>
</table>

Access Time

If a nanosecond was a second, similar to retrieving the data from...

- 1 ns: your memory (1 sec)
- 10 ns: your desk (10 sec)
- 100 ns: the next office (01:40 min)

http://people.eecs.berkeley.edu/~rcs/research/interactive_latency.html
### Why In-Memory?

<table>
<thead>
<tr>
<th>Layer</th>
<th>Access Time</th>
<th>If a nanosecond was a second, similar to retrieving the data from...</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Registers</td>
<td>1 ns</td>
<td>your memory (1 sec)</td>
</tr>
<tr>
<td>CPU L3 Cache</td>
<td>10 ns</td>
<td>your desk (10 sec)</td>
</tr>
<tr>
<td>DRAM</td>
<td>100 ns</td>
<td>the next office (01:40 min)</td>
</tr>
<tr>
<td>SSD</td>
<td>16 µs</td>
<td>Düsseldorf, by train (04:27 h)</td>
</tr>
</tbody>
</table>

- [http://people.eecs.berkeley.edu/~rcs/research/interactive_latency.html](http://people.eecs.berkeley.edu/~rcs/research/interactive_latency.html)
Why In-Memory?

Access Time | If a nanosecond was a second, similar to retrieving the data from...
---|---
1 ns | your memory (1 sec)
10 ns | your desk (10 sec)
100 ns | the next office (01:40 min)
16 µs | Düsseldorf, by train (04:27 h)
3 ms | Australia, by pigeon (34 days)

CPU Registers
CPU L3 Cache
DRAM
SSD
HDD

http://people.eecs.berkeley.edu/~rcs/research/interactive_latency.html
Why write our own database at all?

• For research, we need a database that has reasonable performance but is easier to modify than product databases.

• Leaving out things like authentication and error handling makes the database leaner, thus easier to understand and maintain.

• Re-building Hyrise takes <2s, with a commercial database it comes close to an hour.
Why write our own database at all?

• Focus on the things we need, for example, fast benchmarking:

```
MARKUS.DRESLER@HMEA.ORG:~/HYRISE2/BUILD-RELEASE$./SCRIPTS/COMPARE_BENCHMARKS.PY MASTER.JSON JOINFILTER.JSON

     Benchmark | prev. iter/s | runs | new iter/s | runs | change [%] | p-value (significant if <0.001) |
---------------|--------------|------|------------|------|------------|----------------------------------|
TPC-H 01      | 0.583272814751 | 6   | 0.58503818512 | 6   | +0%        | (run time too short) (not enough runs) 0.9276 |
TPC-H 02      | 12.2122583899 | 123 | 12.3910830366 | 124 | +1%        | (run time too short) 0.0001 |
TPC-H 03      | 6.2827539444  | 63  | 6.3141641618  | 64  | +0%        | (run time too short) 0.6343 |
TPC-H 04      | 7.17641639709 | 72  | 7.8329167366  | 79  | +9%        | (run time too short) 0.0000 |
TPC-H 05      | 3.85262989998 | 39  | 4.11281108856 | 42  | +7%        | (run time too short) 0.0001 |
TPC-H 06      | 103.95088295  | 1040| 144.98790317 | 1450| +39%       | (run time too short) 0.0000 |
TPC-H 07      | 1.56280565262 | 16  | 7.4963917722  | 75  | +380%      | (run time too short) 0.0000 |
TPC-H 08      | 5.72817850113 | 58  | 5.7813768368  | 58  | +1%        | (run time too short) 0.1058 |
TPC-H 09      | 1.55589616299 | 16  | 1.5862740278  | 16  | +2%        | (run time too short) 0.0012 |
TPC-H 10      | 3.0624421684  | 31  | 3.01984548569 | 31  | -1%        | (run time too short) 0.2416 |
TPC-H 11      | 24.9608492731 | 250 | 25.2486057281 | 253 | +1%        | (run time too short) 0.0000 |
TPC-H 12      | 8.46087646484 | 85  | 8.2825155282  | 83  | -2%        | (run time too short) 0.0000 |
TPC-H 13      | 2.30729389191 | 24  | 2.30352401733 | 24  | +0%        | (run time too short) 0.7996 |
TPC-H 14      | 45.7575836182 | 458 | 47.140064697 | 472 | +3%        | (run time too short) 0.0000 |
TPC-H 15      | 70.9292987175 | 710 | 69.520953174 | 696 | -2%        | (run time too short) 0.0000 |
TPC-H 16      | 7.09942245483 | 71  | 7.4117588969 | 75  | +4%        | (run time too short) 0.0000 |
TPC-H 17      | 1.25438761711 | 13  | 1.29478621483 | 13  | +3%        | (run time too short) 0.0012 |
TPC-H 18      | 0.7444460392 | 8   | 0.770971292191 | 8 | +4% | (run time too short) (not enough runs) 0.0086 |
TPC-H 19      | 8.53706359863 | 86  | 15.761482338 | 158 | +85%       | (run time too short) 0.0000 |
TPC-H 20      | 2.78222179413 | 28  | 2.82980847359 | 29  | +2%        | (run time too short) 0.0651 |
TPC-H 21      | 1.49054738948 | 15  | 1.4960891548  | 15  | -2%        | (run time too short) 0.0029 |
TPC-H 22      | 13.8240921333 | 139 | 13.8192062378 | 139 | -0%        | (run time too short) 0.9070 |

geometric mean | | | | | | +14% |
```
Status Quo

- Hyrise has grown significantly and can slowly be considered a real database
  - Just as in industry, you will have to work your way into a grown (but well maintained) code base
  - We will help you by proposing group projects that are digestible chunks
- Compared to commercial databases, our query latency is within 5x; sometimes, we are actually faster
Build your own Database – Week 1

First Work Package
Description

- You can find the description of the work package online:
  - https://hpi.de/plattner/teaching/winter-term-201920/develop-your-own-database.html
First tasks

1. Set up your build environment
2. Implement a single segment
3. Group segments into a chunk
4. Append data to a chunk
5. Group chunks into a table
6. Store tables in a StorageManager
Setting up your Environment

- Demo (git clone, install, cmake, make test -j)
Up-to-Date Build Setup

• Why do we require current compiler and library versions?

• First reason: New C++17 features are great, but building up technical debt for workarounds is not:

```cpp
#ifndef __has_include(<optional>)
    template <class T>
    using optional = std::optional<T>;
    static auto nullopt = std::nullopt;
#else
    template <class T>
    using optional = std::experimental::optional<T>;
    static auto nullopt = std::experimental::nullopt;
#endif
```
Up-to-Date Build Setup

- Second reason: Even compilers are not infallible

**Bug 79180 - Nested lambda-capture causes segfault for parameter pack**

- **Status:** RESOLVED FIXED
- **Alias:** None
- **Product:** gcc
- **Component:** c++ (show other bugs)
- **Version:** 6.3.0
- **Importance:** P3 normal
- **Target Milestone:** 8.0
- **Assignee:** Not yet assigned to anyone

- **Reported:** 2017-01-22 08:25 UTC by Markus Dreseler
- **Modified:** 2017-10-02 12:48 UTC (History)
- **CC List:** 5 users (show)
- **See Also:**
  - **Host:**
  - **Target:**
  - **Build:**
  - **Known to work:**
  - **Known to fail:**
  - **Last reconfirmed:** 2017-01-23 00:00:00
Up-to-Date Build Setup

- Once we had gcc 8...
The Opossum Table Model

Table T

Chunk #1
- Segment a: dictionary-encoded
- Segment b: run length-encoded
- Segment c: unencoded

Chunk #2
- Segment a: dictionary-encoded
- Segment b: dictionary-encoded
- Segment c: dictionary-encoded

... (more chunks)

Chunk #n-1
- Segment a: unencoded
- Segment b: unencoded
- Segment c: unencoded

Chunk #n
- Segment a: unencoded
- Segment b: unencoded
- Segment c: unencoded
Document Walkthrough
Organizational Stuff
About Correctness

• For the sprints, we are using a stripped Hyrise code base

• Some things look slightly different in the master, but we believe that this is a better start

• We have tested that everything works the way we expect it to, but this does not mean that everything is perfect

• If something looks wrong, or if you have any issues about the course itself, please do not hesitate to talk to us
Einschreibung und -fristen, Leistungserfassungsprozess, Vertiefungsgebieteinordnung

**Allgemeine Information**
- Semesterwochenstunden: 4
- ECTS: 6
- Benotet: Ja
- Einschreibefrist: **20. Oktober 2019** (s.u.)
- Programm: IT-Systems Engineering MA, Data Engineering MA
- Lehrform: PS
- Belegungsart: Wahlpflicht

**Module**

IT-Systems Engineering
- ITSE-{Analyse, Entwurf, Konstruktion, Maintenance}
- BPET-{Konzepte und Methoden, Spezialisierung, Techniken und Werkzeuge}
- OSIS-{Konzepte und Methoden, Spezialisierung, Techniken und Werkzeuge}
- SAMT-{Konzepte und Methoden, Spezialisierung, Techniken und Werkzeuge}

Data Engineering
- PREP-{Konzepte und Methoden, Spezialisierung, Techniken und Werkzeuge}
- SCAL-{Konzepte und Methoden, Spezialisierung, Techniken und Werkzeuge}
Einschreibung und -fristen, Leistungserfassungsprozess, Vertiefungsgebieteinordnung

<table>
<thead>
<tr>
<th>Kriterium</th>
<th>Gewichtung</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sprint 1-3</td>
<td>30 %</td>
</tr>
<tr>
<td>Gruppenphase</td>
<td>60 %</td>
</tr>
<tr>
<td>Aktive Mitarbeit</td>
<td>10 %</td>
</tr>
</tbody>
</table>
Piazza

- Most likely, there will be remaining questions about the architecture or the implementation
- Waiting for a week is not an option
- Your classmates may have the same question or be able to help you
Piazza

• We use Piazza to answer questions, communicate, and organize the class:
  • https://piazza.com/hpi.uni-potsdam.de/fall2019/dyod/home
  • Please use common sense in how much of your implementation you should share
Groups

• We would like for you to work in groups of three

• Feel free to start working on the first sprint now

• Please wait with forming groups until you have received your confirmation by the Studienreferat (Monday?)

• You can also use Piazza to find team members

• For your submission, please send us an email with the names of your group members, a link to your repository, and the SHA-1 hash of your final commit
# Deliverables

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>29 Oct</td>
<td>Code Sprint 1</td>
</tr>
<tr>
<td>5 Nov</td>
<td>Review Sprint 1</td>
</tr>
<tr>
<td>12 Nov</td>
<td>Code Sprint 2</td>
</tr>
<tr>
<td>19 Nov</td>
<td>Review Sprint 2</td>
</tr>
<tr>
<td>26 Nov</td>
<td>Code Sprint 3</td>
</tr>
<tr>
<td>3 Dec</td>
<td>Review Sprint 3</td>
</tr>
<tr>
<td>5 Feb</td>
<td>Final Presentation, First Code Group Phase</td>
</tr>
<tr>
<td>tbd</td>
<td>Review and Final Code Group Phase</td>
</tr>
</tbody>
</table>

*(Group phase)*

*Harte Code-Deadline klar kommunizieren: einigen Gruppen war nicht bekannt, dass der Termin der Final Presentation auch die Code-Deadline ist.*
Next Week

• Deep Dive into some of the used C++ concepts and beyond
  – Templates
  – Smart Pointers
  – RAII