Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Develop your own Database 2019/2020

Week 1

Outlook

Hasso Plattner Institut

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

Weekly Meetings

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

Hasso Plattner Institut

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

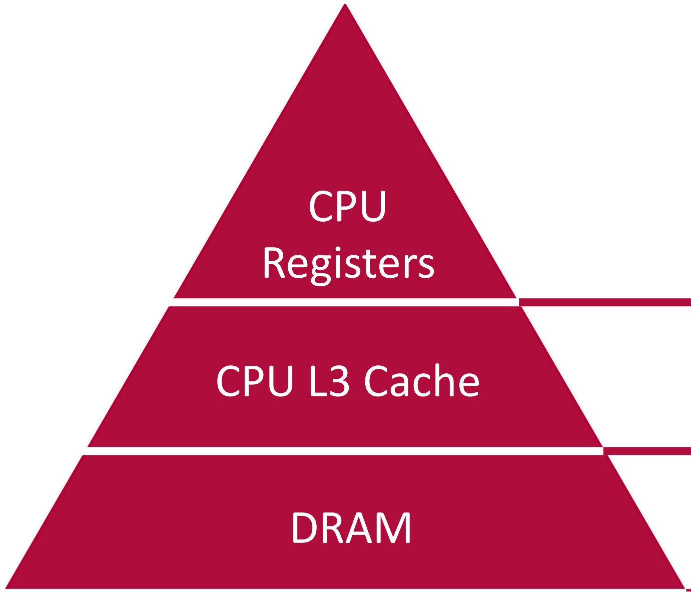
Hasso Plattner Institut

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?



	Access Time	If a nanosecond was a second, similar to retrieving the data from...
CPU Registers	1 ns	your memory (1 sec)
CPU L3 Cache	10 ns	your desk (10 sec)
DRAM	100 ns	the next office (01:40 min)

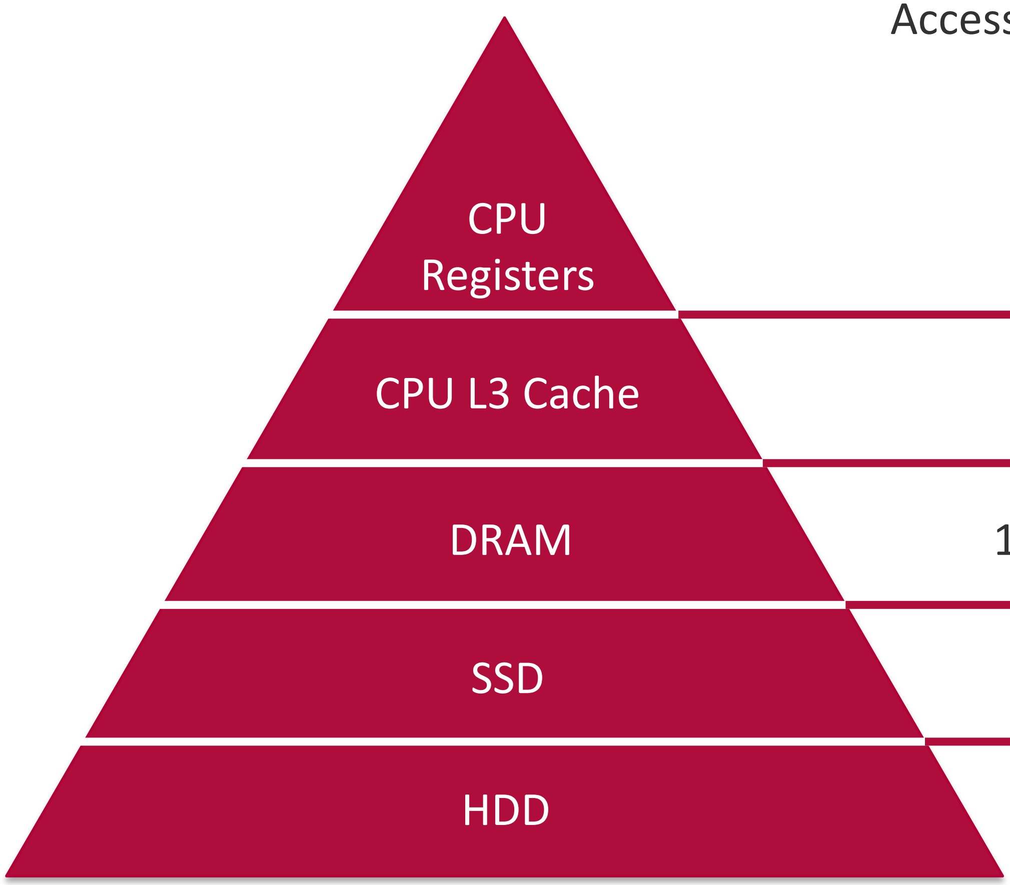
Why In-Memory?

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

HPI Hasso Plattner Institut

http://people.eecs.berkeley.edu/~rcs/research/interactive_latency.html
http://www.montana.edu/cpa/news/wwwpb-archives/yuth/pigeon.html

Why In-Memory?



	Access Time	If a nanosecond was a second, similar to retrieving the data from...
CPU Registers	1 ns	your memory (1 sec)
CPU L3 Cache	10 ns	your desk (10 sec)
DRAM	100 ns	the next office (01:40 min)
SSD	16 µs	Düsseldorf, by train (04:27 h)
HDD	3 ms	Australia, by pigeon (34 days)

HPI Hasso Plattner Institut

http://people.eecs.berkeley.edu/~rcs/research/interactive_latency.html
http://www.montana.edu/cpa/news/wwwpb-archives/yuth/pigeon.html

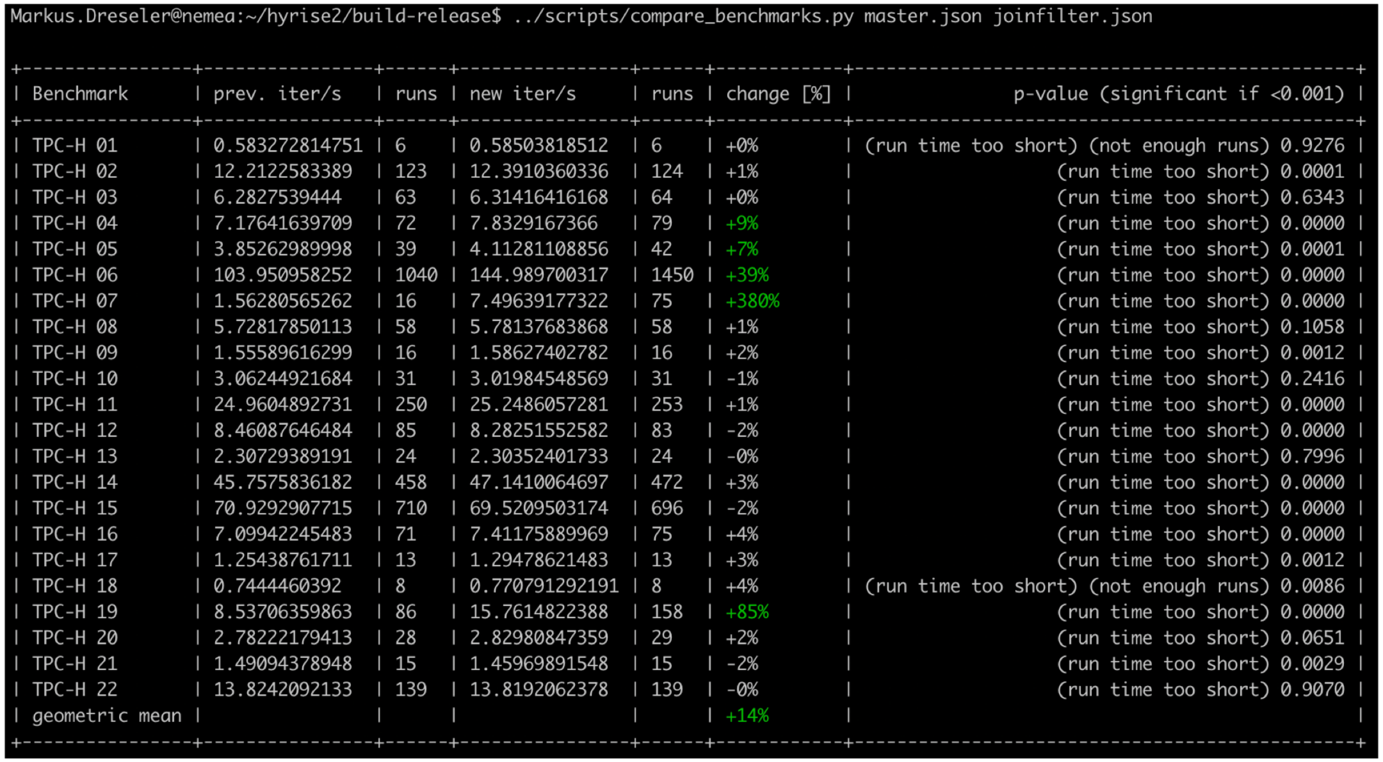
Why write our own database at all?

- 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

HPI Hasso Plattner Institut

Why write our own database at all?

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



```
Markus.Dreseler@nemea:~/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       | 12.2122583389 | 123  | 12.3910360336 | 124  | +1%        |                  (run time too short) 0.0001 |
| TPC-H 03       | 6.2827539444  | 63   | 6.31416416168 | 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       | 103.950958252 | 1040 | 144.989700317 | 1450 | +39%       |                  (run time too short) 0.0000 |
| TPC-H 07       | 1.56280565262 | 16   | 7.49639177322 | 75   | +380%      |                  (run time too short) 0.0000 |
| TPC-H 08       | 5.72817850113 | 58   | 5.78137683868 | 58   | +1%        |                  (run time too short) 0.1058 |
| TPC-H 09       | 1.55589616299 | 16   | 1.58627402782 | 16   | +2%        |                  (run time too short) 0.0012 |
| TPC-H 10       | 3.06244921684 | 31   | 3.01984548569 | 31   | -1%        |                  (run time too short) 0.2416 |
| TPC-H 11       | 24.9604892731 | 250  | 25.2486057281 | 253  | +1%        |                  (run time too short) 0.0000 |
| TPC-H 12       | 8.46087646484 | 85   | 8.28251552582 | 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  | 47.1410064697 | 472  | +3%        |                  (run time too short) 0.0000 |
| TPC-H 15       | 70.9292907715 | 710  | 69.5209503174 | 696  | -2%        |                  (run time too short) 0.0000 |
| TPC-H 16       | 7.09942245483 | 71   | 7.41175889969 | 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    | 0.770791292191 | 8   | +4%        | (run time too short) (not enough runs) 0.0086 |
| TPC-H 19       | 8.53706359863 | 86   | 15.7614822388 | 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       | 1.49094378948 | 15   | 1.45969891548 | 15   | -2%        |                  (run time too short) 0.0029 |
| TPC-H 22       | 13.8242092133 | 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

Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

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

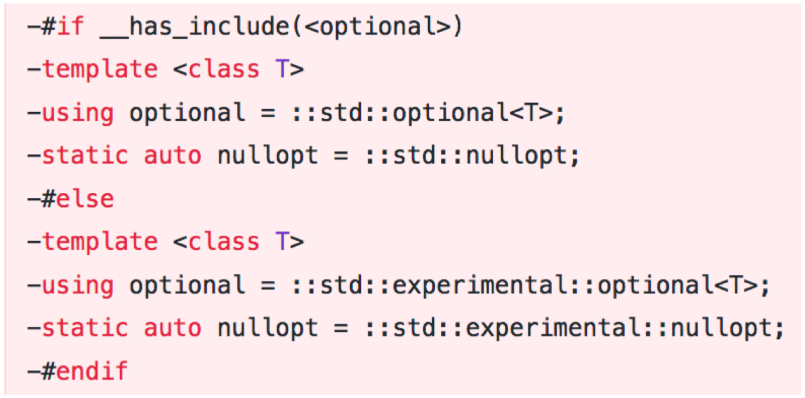
1. Set up your build environment
2. Implement a single segment
3. Group segments into a chunk
4. Append data to a chunk
5. Group chunks into a table
6. Store tables in a StorageManager

Setting up your Environment

- Demo (git clone, install, cmake, make test -j)

Up-to-Date Build Setup

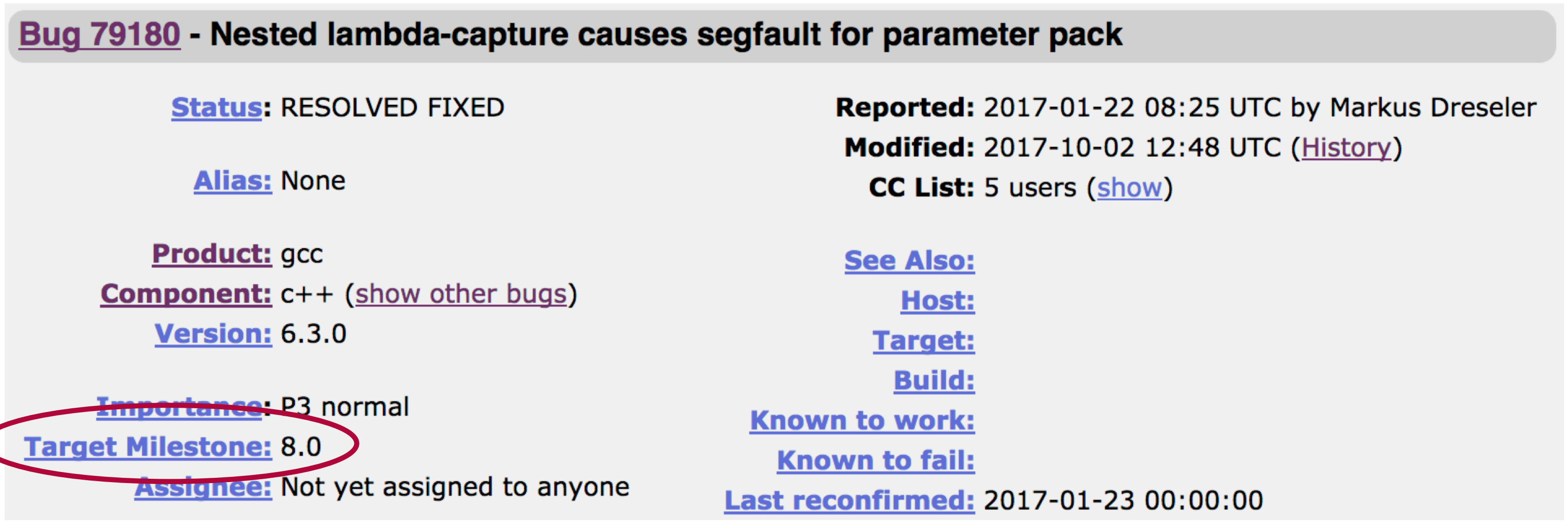
- Why do we require current compiler and library versions?
- First reason: New C++17 features are great, but building up technical debt for workarounds is not:



```
-#if __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

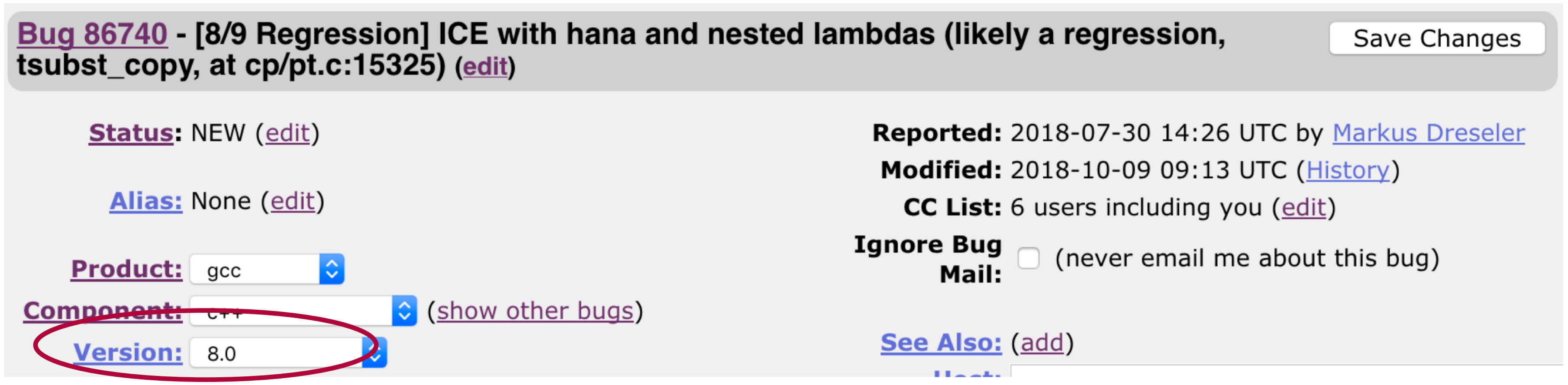


Bug 79180 - Nested lambda-capture causes segfault for parameter pack

Status:	RESOLVED FIXED	**Reported:**	2017-01-22 08:25 UTC by Markus Dreseler
		Modified:	2017-10-02 12:48 UTC (History)
Alias:	None	**CC List:**	5 users (show)
Product:	gcc	**See Also:**	
Component:	c++ (show other bugs)	**Host:**	
Version:	6.3.0	**Target:**	
		Build:	
Importance:	P3 normal	**Known to work:**	
Target Milestone:	8.0	**Known to fail:**	
Assignee:	Not yet assigned to anyone	**Last reconfirmed:**	2017-01-23 00:00:00

Up-to-Date Build Setup

- Once we had gcc 8…



Bug 86740 - [8/9 Regression] ICE with hana and nested lambdas (likely a regression, tsubst_copy, at cp/pt.c:15325) (edit)

Save Changes

Status: NEW (edit)

Alias: None (edit)

Product: gcc

Component: c++ (show other bugs)

Version: 8.0

Reported: 2018-07-30 14:26 UTC by Markus Dreseler

Modified: 2018-10-09 09:13 UTC (History)

CC List: 6 users including you (edit)

Ignore Bug Mail: ☐ (never email me about this bug)

See Also: (add)

The Opossum Table Model

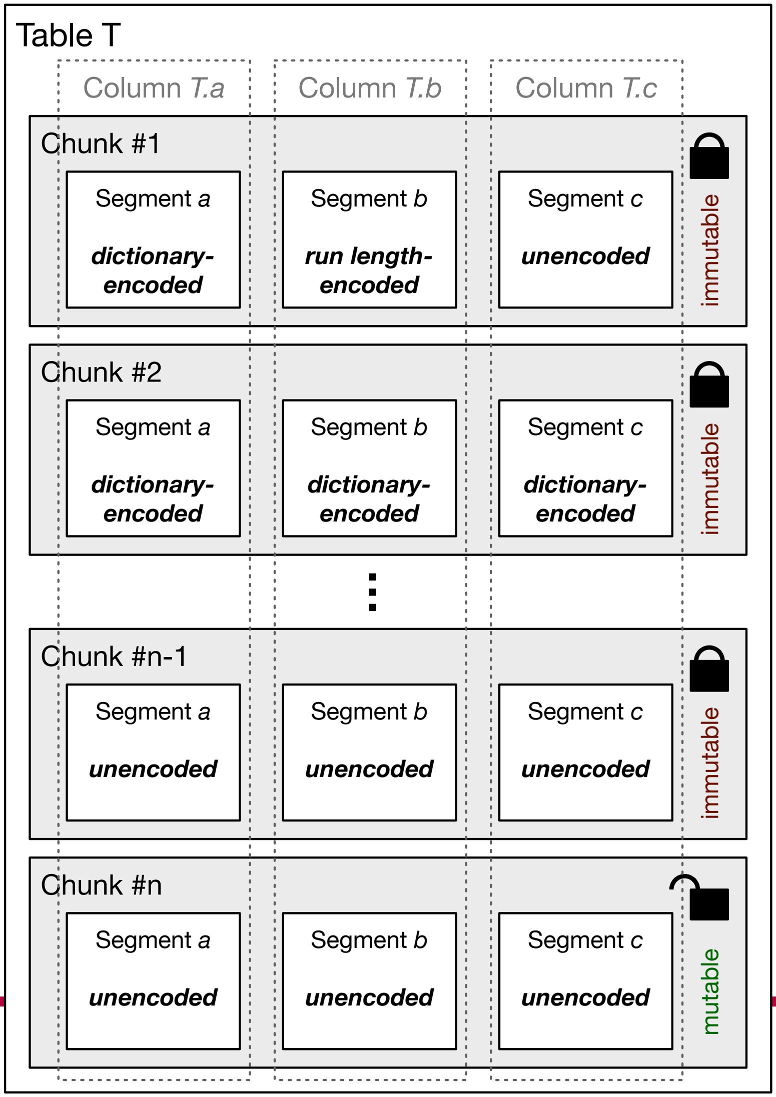


Table T
Column *T.a*
Column *T.b*
Column *T.c*
Chunk #1
Segment *a* ***dictionary-encoded***
Segment *b* ***run length-encoded***
Segment *c* ***unencoded***
immutable
Chunk #2
Segment *a* ***dictionary-encoded***
Segment *b* ***dictionary-encoded***
Segment *c* ***dictionary-encoded***
immutable
⋮
Chunk #n-1
Segment *a* ***unencoded***
Segment *b* ***unencoded***
Segment *c* ***unencoded***
immutable
Chunk #n
Segment *a* ***unencoded***
Segment *b* ***unencoded***
Segment *c* ***unencoded***
mutable

HPI Hasso Plattner Institut

Document Walkthrough

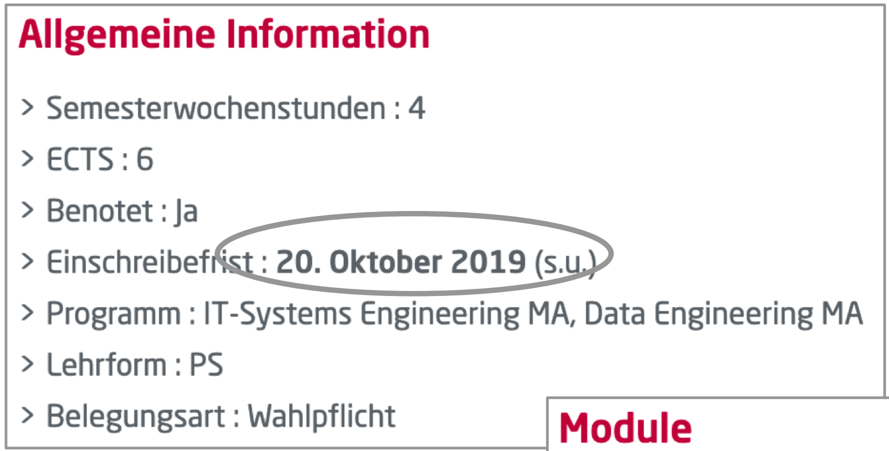
Build your own Database – Week 1

Organizational Stuff

About Correctness

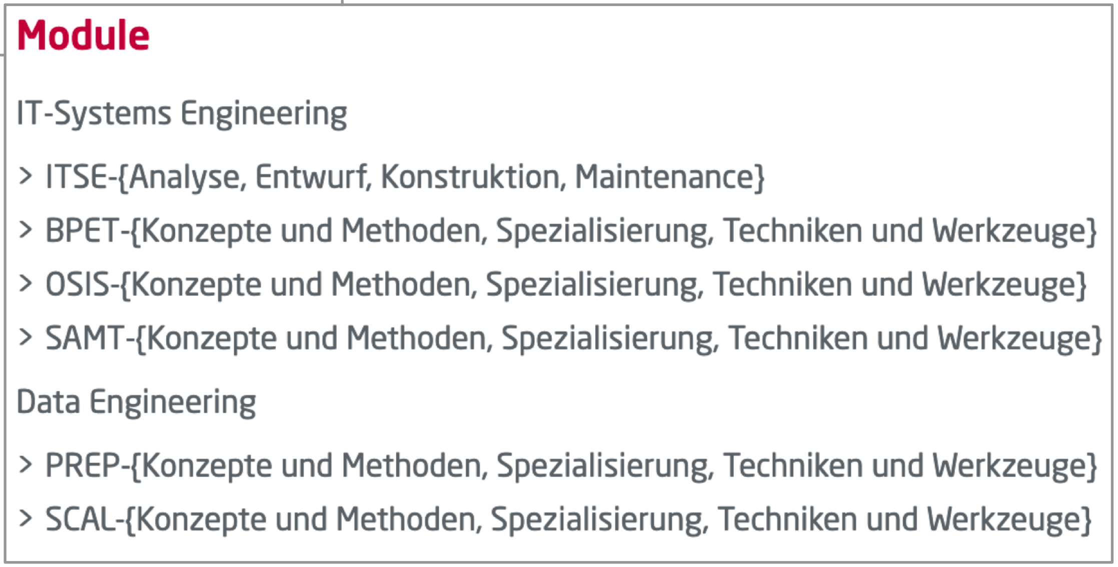
- For the sprints, we are using a stripped Hyrise code base
- Some things look slightly different in the master, but we believe that this is a better start
- We have tested that everything works the way we expect it to, but this does not mean that everything is perfect
- If something looks wrong, or 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

Kriterium	Gewichtung
Sprint 1-3	30 %
Gruppenphase	60 %
Aktive Mitarbeit	10 %

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

• 29 Oct	Code Sprint 1
• 5 Nov	Review Sprint 1
• 12 Nov	Code Sprint 2 (tbc)
• 19 Nov	Review Sprint 2
• 26 Nov	Code Sprint 3
• 3 Dec	Review Sprint 3
(Group phase)	
• 5 Feb	Final Presentation, First Code Group Phase
• tbd	Review and Final Code 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