SDMS Lab 3

Scalable Datamanagment Systems @ TU Darmstadt - 2025-01-31

Jonathan Schild, Tilo Gaulke, Adrian Lutsch, Nils Boeschen

Overview

In this lab, you will build a Snowflake-style OLAP DBMS prototype.

1. Column-wise table chunk storage
2. Iceberg-style metadata layer
3. Storage Engine: inserts, updates, deletes
4. Query Engine: Table Scans, Filters, Joins, Aggregations
5. (Bonus/Benchmark) Table Re-Clustering

Dependencies

Tasks 3 to 5 build on tasks 1 and 2.

Task 1: Column-wise table chunk storage

Tasks

- Implement reading table chunks from binary files
- Implement writing table chunks to binary files

Data Structure

- Tables are stored and processed in horizontal partitions – “chunks”.
- Tables are stored and processed column-wise
  - `pub type TableChunk = Vec<Column>`
  - `pub type Column = Vec<Value>`

File Format

EBNF:

```
file         = header, data
header       = magic_number, rows, columns, column_infos
column_infos = { type_id, start_index } (* repeated exactly columns times *)
data         = { column } (* exactly columns times *)
```

- One chunk = one file
- `rows`, `columns`, `type_id`, and `start_index` are `u64` stored in little-endian order
- `column` is the serialized form of the data inside a column.
  - u32, u64 and i32 stored in little-endian order
  - Strings stored as the length of the string as u64 followed by the UTF-8-encoded string
- `rows` is the number of rows in the chunk (= length of the vectors)
- `columns` is the number of columns in the chunk
- `start_index` is the first byte of the column data inside the file

Example: SDMS Iceberg Datafile

```
Hex View  00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F

00000000  53 44 4D 53 19 03 4A 53  02 00 00 00 00 00 00 00  SDMS..JS........
00000010  02 00 00 00 00 00 00 00  03 00 00 00 00 00 00 00  ................
00000020  38 00 00 00 00 00 00 00  01 00 00 00 00 00 00 00  8...............
00000030  C0 00 00 00 00 00 00 00  47 00 00 00 00 00 00 00  ........G.......
00000040  F3 86 8F 98 C3 A4 F3 95  A3 99 16 59 F3 84 A1 BD  ...........Y....
00000050  D4 87 C5 88 74 42 52 75  7E 1E 61 14 07 10 F0 9A  ....tBRu~.a.....
00000060  B2 A2 29 72 F0 A8 BB B0  2E 3D 0C 18 30 1B 39 3D  ..)r.....=..0.9=
00000070  08 F4 8E BF 8D F2 8E A4  A5 14 4F F3 95 81 AA 42  ..........O....B
00000080  63 5B 33 5E 40 CE B7 31  00 00 00 00 00 00 00 20  c[3^@..1.......
00000090  60 02 61 3B F0 BE A3 AB  60 F1 BC 99 A1 F1 A9 B5  `.a;....`.......
000000A0  B3 0B D4 99 3E 3F 61 1C  7B 27 3D CE 93 F1 AF 8B  ....>?a.{'=.....
000000B0  8A 6D 25 5C 6E 2E 11 0D  64 66 F3 9F AF A9 58 1E  .m%\n...df....X.
000000C0  75 4F 87 76 4D 4A F6 F6                           uO.vMJ..
```

Listing 1: SDMS Iceberg datafile with 2 columns, varchar and uint

SDMS Iceberg: Header - Magic Bytes

```
Hex View  00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F

00000000  53 44 4D 53 19 03 4A 53  02 00 00 00 00 00 00 00  SDMS..JS........
00000010  02 00 00 00 00 00 00 00  03 00 00 00 00 00 00 00  ................
00000020  38 00 00 00 00 00 00 00  01 00 00 00 00 00 00 00  8...............
00000030  C0 00 00 00 00 00 00 00  47 00 00 00 00 00 00 00  ........G.......
00000040  F3 86 8F 98 C3 A4 F3 95  A3 99 16 59 F3 84 A1 BD  ...........Y....
00000050  D4 87 C5 88 74 42 52 75  7E 1E 61 14 07 10 F0 9A  ....tBRu~.a.....
00000060  B2 A2 29 72 F0 A8 BB B0  2E 3D 0C 18 30 1B 39 3D  ..)r.....=..0.9=
00000070  08 F4 8E BF 8D F2 8E A4  A5 14 4F F3 95 81 AA 42  ..........O....B
00000080  63 5B 33 5E 40 CE B7 31  00 00 00 00 00 00 00 20  c[3^@..1.......
00000090  60 02 61 3B F0 BE A3 AB  60 F1 BC 99 A1 F1 A9 B5  `.a;....`.......
000000A0  B3 0B D4 99 3E 3F 61 1C  7B 27 3D CE 93 F1 AF 8B  ....>?a.{'=.....
000000B0  8A 6D 25 5C 6E 2E 11 0D  64 66 F3 9F AF A9 58 1E  .m%\n...df....X.
000000C0  75 4F 87 76 4D 4A F6 F6                           uO.vMJ..
```

Figure 1: SDMS Iceberg datafile: Magic Bytes

SDMS Iceberg: Header - Rows

```
Hex View  00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F

00000000  53 44 4D 53 19 03 4A 53  02 00 00 00 00 00 00 00  SDMS..JS........
00000010  02 00 00 00 00 00 00 00  03 00 00 00 00 00 00 00  ................
00000020  38 00 00 00 00 00 00 00  01 00 00 00 00 00 00 00  8...............
00000030  C0 00 00 00 00 00 00 00  47 00 00 00 00 00 00 00  ........G.......
00000040  F3 86 8F 98 C3 A4 F3 95  A3 99 16 59 F3 84 A1 BD  ...........Y....
00000050  D4 87 C5 88 74 42 52 75  7E 1E 61 14 07 10 F0 9A  ....tBRu~.a.....
00000060  B2 A2 29 72 F0 A8 BB B0  2E 3D 0C 18 30 1B 39 3D  ..)r.....=..0.9=
00000070  08 F4 8E BF 8D F2 8E A4  A5 14 4F F3 95 81 AA 42  ..........O....B
00000080  63 5B 33 5E 40 CE B7 31  00 00 00 00 00 00 00 20  c[3^@..1.......
00000090  60 02 61 3B F0 BE A3 AB  60 F1 BC 99 A1 F1 A9 B5  `.a;....`.......
000000A0  B3 0B D4 99 3E 3F 61 1C  7B 27 3D CE 93 F1 AF 8B  ....>?a.{'=.....
000000B0  8A 6D 25 5C 6E 2E 11 0D  64 66 F3 9F AF A9 58 1E  .m%\n...df....X.
000000C0  75 4F 87 76 4D 4A F6 F6                           uO.vMJ..
```

Figure 2: SDMS Iceberg datafile: Rows

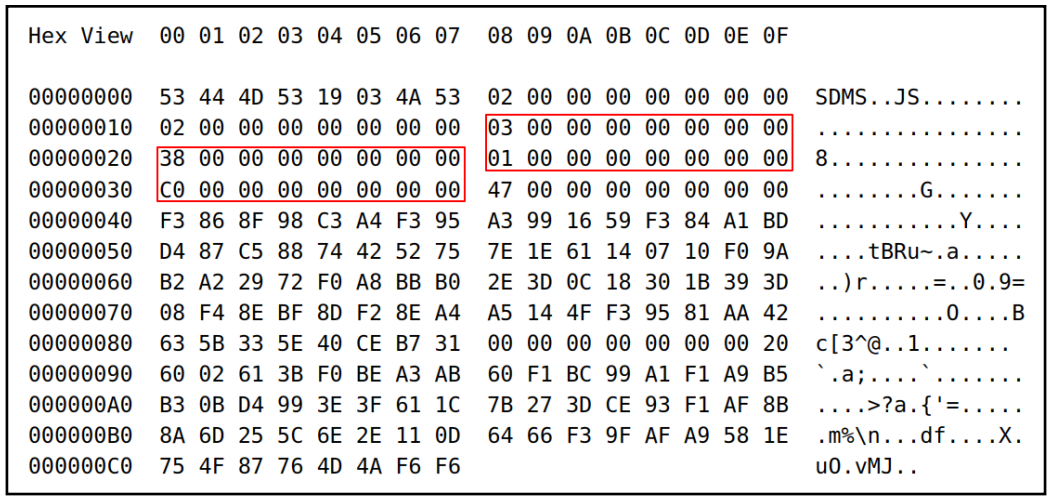
SDMS Iceberg: Header - Columns

```
Hex View  00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F

00000000  53 44 4D 53 19 03 4A 53  02 00 00 00 00 00 00 00  SDMS..JS........
00000010  02 00 00 00 00 00 00 00  03 00 00 00 00 00 00 00  ................
00000020  38 00 00 00 00 00 00 00  01 00 00 00 00 00 00 00  8...............
00000030  C0 00 00 00 00 00 00 00  47 00 00 00 00 00 00 00  ........G.......
00000040  F3 86 8F 98 C3 A4 F3 95  A3 99 16 59 F3 84 A1 BD  ...........Y....
00000050  D4 87 C5 88 74 42 52 75  7E 1E 61 14 07 10 F0 9A  ....tBRu~.a.....
00000060  B2 A2 29 72 F0 A8 BB B0  2E 3D 0C 18 30 1B 39 3D  ..)r.....=..0.9=
00000070  08 F4 8E BF 8D F2 8E A4  A5 14 4F F3 95 81 AA 42  ..........O....B
00000080  63 5B 33 5E 40 CE B7 31  00 00 00 00 00 00 00 20  c[3^@..1.......
00000090  60 02 61 3B F0 BE A3 AB  60 F1 BC 99 A1 F1 A9 B5  `.a;....`.......
000000A0  B3 0B D4 99 3E 3F 61 1C  7B 27 3D CE 93 F1 AF 8B  ....>?a.{'=.....
000000B0  8A 6D 25 5C 6E 2E 11 0D  64 66 F3 9F AF A9 58 1E  .m%\n...df....X.
000000C0  75 4F 87 76 4D 4A F6 F6                           uO.vMJ..
```

Figure 3: SDMS Iceberg datafile: Columns

SDMS Iceberg: Header - Column Info



```
Hex View  00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F

00000000  53 44 4D 53 19 03 4A 53  02 00 00 00 00 00 00 00  SDMS..JS........
00000010  02 00 00 00 00 00 00 00  03 00 00 00 00 00 00 00  ................
00000020  38 00 00 00 00 00 00 00  01 00 00 00 00 00 00 00  8...............
00000030  C0 00 00 00 00 00 00 00  47 00 00 00 00 00 00 00  ........G.......
00000040  F3 86 8F 98 C3 A4 F3 95  A3 99 16 59 F3 84 A1 BD  ...........Y....
00000050  D4 87 C5 88 74 42 52 75  7E 1E 61 14 07 10 F0 9A  ....tBRu~.a.....
00000060  B2 A2 29 72 F0 A8 BB B0  2E 3D 0C 18 30 1B 39 3D  ..)r.....=..0.9=
00000070  08 F4 8E BF 8D F2 8E A4  A5 14 4F F3 95 81 AA 42  ..........O....B
00000080  63 5B 33 5E 40 CE B7 31  00 00 00 00 00 00 00 20  c[3^@..1.......
00000090  60 02 61 3B F0 BE A3 AB  60 F1 BC 99 A1 F1 A9 B5  `.a;....`.......
000000A0  B3 0B D4 99 3E 3F 61 1C  7B 27 3D CE 93 F1 AF 8B  ....>?a.{'=.....
000000B0  8A 6D 25 5C 6E 2E 11 0D  64 66 F3 9F AF A9 58 1E  .m%\n...df....X.
000000C0  75 4F 87 76 4D 4A F6 F6                           uO.vMJ..
```

Figure 4: SDMS Iceberg datafile: Column Info

SDMS Iceberg: Header - Column Info TypeID

```
Hex View  00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F

00000000  53 44 4D 53 19 03 4A 53  02 00 00 00 00 00 00 00  SDMS..JS........
00000010  02 00 00 00 00 00 00 00  03 00 00 00 00 00 00 00  ................
00000020  38 00 00 00 00 00 00 00  01 00 00 00 00 00 00 00  8...............
00000030  C0 00 00 00 00 00 00 00  47 00 00 00 00 00 00 00  ........G.......
00000040  F3 86 8F 98 C3 A4 F3 95  A3 99 16 59 F3 84 A1 BD  ...........Y....
00000050  D4 87 C5 88 74 42 52 75  7E 1E 61 14 07 10 F0 9A  ....tBRu~.a.....
00000060  B2 A2 29 72 F0 A8 BB B0  2E 3D 0C 18 30 1B 39 3D  ..)r.....=..0.9=
00000070  08 F4 8E BF 8D F2 8E A4  A5 14 4F F3 95 81 AA 42  ..........O....B
00000080  63 5B 33 5E 40 CE B7 31  00 00 00 00 00 00 00 20  c[3^@..1.......
00000090  60 02 61 3B F0 BE A3 AB  60 F1 BC 99 A1 F1 A9 B5  `.a;....`.......
000000A0  B3 0B D4 99 3E 3F 61 1C  7B 27 3D CE 93 F1 AF 8B  ....>?a.{'=.....
000000B0  8A 6D 25 5C 6E 2E 11 0D  64 66 F3 9F AF A9 58 1E  .m%\n...df....X.
000000C0  75 4F 87 76 4D 4A F6 F6                           uO.vMJ..
```

Figure 5: SDMS Iceberg datafile: Column Info TypeID

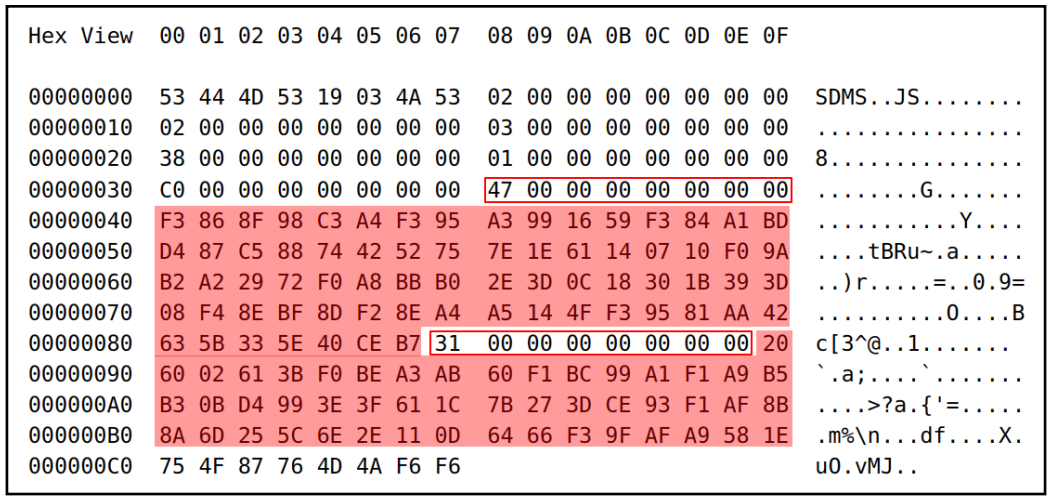
SDMS Iceberg: Header - Column Info Start

```
Hex View  00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F

00000000  53 44 4D 53 19 03 4A 53  02 00 00 00 00 00 00 00  SDMS..JS........
00000010  02 00 00 00 00 00 00 00  03 00 00 00 00 00 00 00  ................
00000020  38 00 00 00 00 00 00 00  01 00 00 00 00 00 00 00  8...............
00000030  C0 00 00 00 00 00 00 00  47 00 00 00 00 00 00 00  ........G.......
00000040  F3 86 8F 98 C3 A4 F3 95  A3 99 16 59 F3 84 A1 BD  ...........Y....
00000050  D4 87 C5 88 74 42 52 75  7E 1E 61 14 07 10 F0 9A  ....tBRu~.a.....
00000060  B2 A2 29 72 F0 A8 BB B0  2E 3D 0C 18 30 1B 39 3D  ..)r.....=..0.9=
00000070  08 F4 8E BF 8D F2 8E A4  A5 14 4F F3 95 81 AA 42  ..........O....B
00000080  63 5B 33 5E 40 CE B7 31  00 00 00 00 00 00 00 20  c[3^@..1.......
00000090  60 02 61 3B F0 BE A3 AB  60 F1 BC 99 A1 F1 A9 B5  `.a;....`.......
000000A0  B3 0B D4 99 3E 3F 61 1C  7B 27 3D CE 93 F1 AF 8B  ....>?a.{'=.....
000000B0  8A 6D 25 5C 6E 2E 11 0D  64 66 F3 9F AF A9 58 1E  .m%\n...df....X.
000000C0  75 4F 87 76 4D 4A F6 F6                           uO.vMJ..
```

Figure 6: SDMS Iceberg datafile: Column Info Start

SDMS Iceberg: Header - Varchar



```
Hex View  00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F

00000000  53 44 4D 53 19 03 4A 53  02 00 00 00 00 00 00 00  SDMS..JS........
00000010  02 00 00 00 00 00 00 00  03 00 00 00 00 00 00 00  ................
00000020  38 00 00 00 00 00 00 00  01 00 00 00 00 00 00 00  8...............
00000030  C0 00 00 00 00 00 00 00  47 00 00 00 00 00 00 00  ........G.......
00000040  F3 86 8F 98 C3 A4 F3 95  A3 99 16 59 F3 84 A1 BD  ...........Y....
00000050  D4 87 C5 88 74 42 52 75  7E 1E 61 14 07 10 F0 9A  ....tBRu~.a.....
00000060  B2 A2 29 72 F0 A8 BB B0  2E 3D 0C 18 30 1B 39 3D  ..)r.....=..0.9=
00000070  08 F4 8E BF 8D F2 8E A4  A5 14 4F F3 95 81 AA 42  ..........O....B
00000080  63 5B 33 5E 40 CE B7 31  00 00 00 00 00 00 00 20  c[3^@..1.......
00000090  60 02 61 3B F0 BE A3 AB  60 F1 BC 99 A1 F1 A9 B5  `.a;....`.......
000000A0  B3 0B D4 99 3E 3F 61 1C  7B 27 3D CE 93 F1 AF 8B  ....>?a.{'=.....
000000B0  8A 6D 25 5C 6E 2E 11 0D  64 66 F3 9F AF A9 58 1E  .m%\n...df....X.
000000C0  75 4F 87 76 4D 4A F6 F6                           uO.vMJ..
```

Figure 7: SDMS Iceberg datafile: Varchar

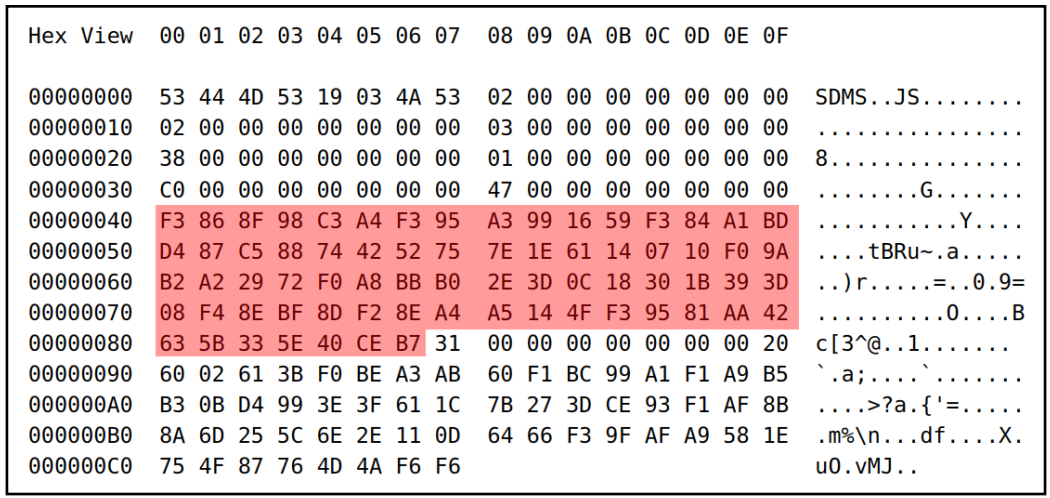
SDMS Iceberg: Header - Varchar Length

```
Hex View  00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F

00000000  53 44 4D 53 19 03 4A 53  02 00 00 00 00 00 00 00  SDMS..JS........
00000010  02 00 00 00 00 00 00 00  03 00 00 00 00 00 00 00  ................
00000020  38 00 00 00 00 00 00 00  01 00 00 00 00 00 00 00  8...............
00000030  C0 00 00 00 00 00 00 00  47 00 00 00 00 00 00 00  ........G.......
00000040  F3 86 8F 98 C3 A4 F3 95  A3 99 16 59 F3 84 A1 BD  ...........Y....
00000050  D4 87 C5 88 74 42 52 75  7E 1E 61 14 07 10 F0 9A  ....tBRu~.a.....
00000060  B2 A2 29 72 F0 A8 BB B0  2E 3D 0C 18 30 1B 39 3D  ..)r.....=..0.9=
00000070  08 F4 8E BF 8D F2 8E A4  A5 14 4F F3 95 81 AA 42  ..........O....B
00000080  63 5B 33 5E 40 CE B7 31  00 00 00 00 00 00 00 20  c[3^@..1.......
00000090  60 02 61 3B F0 BE A3 AB  60 F1 BC 99 A1 F1 A9 B5  `.a;....`.......
000000A0  B3 0B D4 99 3E 3F 61 1C  7B 27 3D CE 93 F1 AF 8B  ....>?a.{'=.....
000000B0  8A 6D 25 5C 6E 2E 11 0D  64 66 F3 9F AF A9 58 1E  .m%\n...df....X.
000000C0  75 4F 87 76 4D 4A F6 F6                           uO.vMJ..
```

Figure 8: SDMS Iceberg datafile: Varchar Length

SDMS Iceberg: Header - Varchar String



```
Hex View  00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F

00000000  53 44 4D 53 19 03 4A 53  02 00 00 00 00 00 00 00  SDMS..JS........
00000010  02 00 00 00 00 00 00 00  03 00 00 00 00 00 00 00  ................
00000020  38 00 00 00 00 00 00 00  01 00 00 00 00 00 00 00  8...............
00000030  C0 00 00 00 00 00 00 00  47 00 00 00 00 00 00 00  ........G.......
00000040  F3 86 8F 98 C3 A4 F3 95  A3 99 16 59 F3 84 A1 BD  ...........Y....
00000050  D4 87 C5 88 74 42 52 75  7E 1E 61 14 07 10 F0 9A  ....tBRu~.a.....
00000060  B2 A2 29 72 F0 A8 BB B0  2E 3D 0C 18 30 1B 39 3D  ..)r.....=..0.9=
00000070  08 F4 8E BF 8D F2 8E A4  A5 14 4F F3 95 81 AA 42  ..........O....B
00000080  63 5B 33 5E 40 CE B7 31  00 00 00 00 00 00 00 20  c[3^@..1.......
00000090  60 02 61 3B F0 BE A3 AB  60 F1 BC 99 A1 F1 A9 B5  `.a;....`.......
000000A0  B3 0B D4 99 3E 3F 61 1C  7B 27 3D CE 93 F1 AF 8B  ....>?a.{'=.....
000000B0  8A 6D 25 5C 6E 2E 11 0D  64 66 F3 9F AF A9 58 1E  .m%\n...df....X.
000000C0  75 4F 87 76 4D 4A F6 F6                           uO.vMJ..
```

Figure 9: SDMS Iceberg datafile: Varchar String

SDMS Iceberg: Header - UInt

```
Hex View  00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F

00000000  53 44 4D 53 19 03 4A 53  02 00 00 00 00 00 00 00  SDMS..JS........
00000010  02 00 00 00 00 00 00 00  03 00 00 00 00 00 00 00  ................
00000020  38 00 00 00 00 00 00 00  01 00 00 00 00 00 00 00  8...............
00000030  C0 00 00 00 00 00 00 00  47 00 00 00 00 00 00 00  ........G.......
00000040  F3 86 8F 98 C3 A4 F3 95  A3 99 16 59 F3 84 A1 BD  ...........Y....
00000050  D4 87 C5 88 74 42 52 75  7E 1E 61 14 07 10 F0 9A  ....tBRu~.a.....
00000060  B2 A2 29 72 F0 A8 BB B0  2E 3D 0C 18 30 1B 39 3D  ..)r.....=..0.9=
00000070  08 F4 8E BF 8D F2 8E A4  A5 14 4F F3 95 81 AA 42  ..........O....B
00000080  63 5B 33 5E 40 CE B7 31  00 00 00 00 00 00 00 20  c[3^@..1.......
00000090  60 02 61 3B F0 BE A3 AB  60 F1 BC 99 A1 F1 A9 B5  `.a;....`.......
000000A0  B3 0B D4 99 3E 3F 61 1C  7B 27 3D CE 93 F1 AF 8B  ....>?a.{'=.....
000000B0  8A 6D 25 5C 6E 2E 11 0D  64 66 F3 9F AF A9 58 1E  .m%\n...df....X.
000000C0  75 4F 87 76 4D 4A F6 F6                           uO.vMJ..
```

Figure 10: SDMS Iceberg datafile: UInt

SDMS Iceberg: Header - UInt Data

```
Hex View  00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F

00000000  53 44 4D 53 19 03 4A 53  02 00 00 00 00 00 00 00  SDMS..JS........
00000010  02 00 00 00 00 00 00 00  03 00 00 00 00 00 00 00  ................
00000020  38 00 00 00 00 00 00 00  01 00 00 00 00 00 00 00  8...............
00000030  C0 00 00 00 00 00 00 00  47 00 00 00 00 00 00 00  ........G.......
00000040  F3 86 8F 98 C3 A4 F3 95  A3 99 16 59 F3 84 A1 BD  ...........Y....
00000050  D4 87 C5 88 74 42 52 75  7E 1E 61 14 07 10 F0 9A  ....tBRu~.a.....
00000060  B2 A2 29 72 F0 A8 BB B0  2E 3D 0C 18 30 1B 39 3D  ..)r.....=..0.9=
00000070  08 F4 8E BF 8D F2 8E A4  A5 14 4F F3 95 81 AA 42  ..........O....B
00000080  63 5B 33 5E 40 CE B7 31  00 00 00 00 00 00 00 20  c[3^@..1.......
00000090  60 02 61 3B F0 BE A3 AB  60 F1 BC 99 A1 F1 A9 B5  `.a;....`.......
000000A0  B3 0B D4 99 3E 3F 61 1C  7B 27 3D CE 93 F1 AF 8B  ....>?a.{'=.....
000000B0  8A 6D 25 5C 6E 2E 11 0D  64 66 F3 9F AF A9 58 1E  .m%\n...df....X.
000000C0  75 4F 87 76 4D 4A F6 F6                           uO.vMJ..
```

Figure 11: SDMS Iceberg datafile: UInt Data

Task Files

Task	File
Serialization & De-serialization	`src/storage/data.rs`

Task 2: Iceberg-style metadata layer

Implement an Apache-Iceberg-style metadata layer.

Data files are only added and deleted, never changed.

Layers

1. Catalog - Contains metadata of all tables of the DB
2. TableMetadata - Contains metadata of a table, has a version
3. Manifest - Description of changes to a table, has a version and consists of
    1. added files
    2. column-wise statistics for added files
    3. deleted files

Important functionality

1. Store (versioned) information on added and deleted files
2. Find files that contain overlap with a filter criterion

Layers in Detail

- 1 Catalog : N TableMetadata
- 1 TableMetadata : N Manifest (versions)
- 1 Manifest : N Added Files
- 1 Manifest : N FileStats (one for each added file)
- 1 Manifest : M Deleted Files
- 1 FileStats : N ColumnStats

```
pub struct Manifest {
    version: Version,
    added: Vec<FileHandle>,
    deleted: Vec<FileHandle>,
    stats: Vec<FileStats>,
}
```

Comparison of Values

We expect that trying to compare different value types results in a panic.

Examples

```
Value::Int(10) < Value::Int(30) == true
Value::Int(10) => Value::Int(30) == false
Value::UInt(10) < Value::Int(30) => panic!()
```

Task

Implement `PartialOrd` and `Ord` for `Value` in `value_cmp.rs`

Finding relevant files

Each new table version can delete and add files

Tasks

- Determine all valid files for a version of the table
- Determine all valid files for a version of the table that contain relevant values for answering a range query
  - Use the column statistics stored in the metadata
  - Support point and range predicates
  - Support conjunctive predicates of the same type over multiple columns

Task Files

Task	File
Comparison Logic for Value	`src/value_cmp.rs`
Range Checks for Column Statistics	`src/iceberg/stats.rs`
Contains Check for one Version	`src/iceberg/manifest.rs`
Contains Check for Table, Manifests for a Specific Version, Files for a Specific Version	`src/iceberg/table_metadata.rs`

Task 3: Storage Engine

The storage engine is responsible for inserting, updating, and deleting parts of tables.

It integrates the functionality of the storage and metadata layer and calculates statistics for chunks inserted into the table.

The engine has a state: it saves which table is currently being changed and a manifest containing the current changes.

```
pub struct SdmsIcebergEngine {
    pub catalog: Catalog,
    manifest: Manifest,
    table_id: Option<usize>,
    pub storage: FileBasedStorage,
}
```

Usage of Storage Engine

1. Choose table: `engine.start_table_modification(table_id);`
2. Run changes: `engine.insert(table_chunk_1); engine.insert(table_chunk_2);`
3. Apply changes: `engine.commit();`

Expected behavior

- `insert`, `update`, `delete`, `delete_chunks`, and `commit` are expected to return `Err(DatabaseError::EngineError)` if no table is chosen (`table_id == None`)
- `start_table_modification` is expected to return `Err(DatabaseError::EngineError)` if `table_id != None`
- `update` and `delete` always add the files they change to the deleted files and store the result in new files.
- If all rows of a chunk are deleted, no empty file will be created

Task Files

Task	File
insert, update, delete, delete_chunks, calculate_statistics	`src/engine/db_engine.rs`

Task 4: Query Engine

Idea

Implement chunk-at-a-time columnar operators for the OLAP engine!

TableChunk

- Before: Volcano model that emits full rows, one-at-a-time, e.g. [1, “Customer1”, “Frankfurt”]
- Now: Chunk-at-a-time execution that consumes and emits multiple rows of data in columnar layout `TableChunk = Vec<Vec<Value>>`:
  - e.g. a chunk of customer’s could look like [[1,2,3], [“Customer1”,“Customer2”,“Customer3”], …]

```rust
pub trait Operator {
    fn open(&mut self);
    fn next(&mut self) -> Option<TableChunk>;
    fn close(&mut self);
}
// A chunk of a table, in column layout.
pub type TableChunk = Vec<Vec<Value>>;
```

Filter

```
impl ColumnFilter{
    pub fn new(child: Box<dyn Operator>,
               filters: HashMap<usize, (Value,Value)>) -> Self {}}
```

- Filter a chunk, return only values that belong to qualifying rows
- `filter` is a hash map from column idxs to range of Values that satisfy the filter for that column
- Here, we only consider conjunctive predicates (`AND`), i.e. all filter predicates in different columns need to be true for a row to be qualifying
- e.g. for this filter: {0: (2,15)}, and this chunk from the child operator [[1,2,3], [“Customer1”,“Customer2”,“Customer3”]], the `ColumnFilter` should emit [[2,3], [“Customer2”,“Customer3”]]
- Preserve the order of rows inside the child’s chunks.

Aggregation

```
pub type AggFunc = Box<dyn Fn(&Vec<Value>) -> Value>;
impl ColumnAggregate{
    pub fn new(child: Box<dyn Operator>,
               aggregates: HashMap<usize, AggFunc>) -> Self {}}
```

- Do simple sum/max/min aggregates per column
- `aggregates` is a hash map from column idxs to an aggregation function for a Vector of Values
- e.g. for this aggregates map {0: sum}, and these chunks as the only ones from the child operator [[1,2,3], [“Customer1”,“Customer2”,“Customer3”]] and [[4,5], [“Customer4”,“Customer5”]], the `ColumnAggregate` should emit [[15]]
- The same aggregation function is to be used inside chunk values, as well as between chunk aggregates!
- Preserve the order of and project down to aggregated columns, e.g. if there are aggregates on columns 3 and 1, return a Vector with the aggregates of column 1, then column 3.

Join

```
impl ColumnEqJoin{
    pub fn new(child0: Box<dyn Operator>, child1: Box<dyn Operator>,
               join_column_idxs: (usize,usize)) -> Self {}}
```

- Join based on equality predicate between specified columns
- For each chunk in child1, return the concatenated columns of both child operator inputs
  - e.g. for a full scan of this data as child0 [[1,2,3], [“Customer1”,“Customer2”,“Customer3”]]
  - … the following chunk emitted by child1: [[“A”,“B”,“C”], [2,3,4]]
  - … and join_column_idxs = (0,1)
  - the next call should emit: [[2,3], [“Customer2”,“Customer3”], [“A”,“B”,], [2,3]]
- In this project, you can again assume that the first child will not have duplicate join keys
  - i.e. a **hashjoin** could build a hash table for the chunks from that side
- Preserve the order of rows of each of child1′s chunks (between chunks and inside chunks).

TableScan

```
impl ColumnTableScan {
    pub fn new(files: Vec<FileHandle>, col_project: Columns, storage: FileBasedStorage) ->
Self{}}
```

- Scan the given files, reading only the projected columns
  - one `TableChunk` for each file in `files`
- Keep an empty vector for columns that are not read
- e.g. for the following chunk data saved in a file [[1,2,3], [“Customer1”,“Customer2”,“Customer3”]], a `ColumnTableScan` with col_project = [1] should return the following data for the respective file: [[], [“Customer1”,“Customer2”,“Customer3”]]
- Have a look at `FileBasedStorage::read_file` and `DataFile::parse_columns` from previous task

Task Files

Task	File
Operator Implementation	`src/engine/operators.rs`

Bonus/Benchmark: Task 5: Table Re-Clustering

Idea

Given a database and different workloads that are frequently executed, re-cluster the underlying data files for better pruning!

- Completion of this task only (possibly) provides bonus points!

```
pub enum QueryPlan {
    TableScan(TableID),
    Filter(Box<QueryPlan>, Vec<(ColumnID,DomainSize,FilterSelectivity)>),
    Aggregate(Box<QueryPlan>, Vec<ColumnID>),
    Join(Box<QueryPlan>, Box<QueryPlan>, (ColumnID,ColumnID)),
}
impl Optimizer {
  pub fn re_cluster(
        engine: &mut SdmsIcebergEngine,
        workload: Vec<QueryPlan> -> Result<(), DatabaseError> {}}
```

- `QueryPlan` is similar to an operator plan, except it only contains interesting query information

Approach & Hints

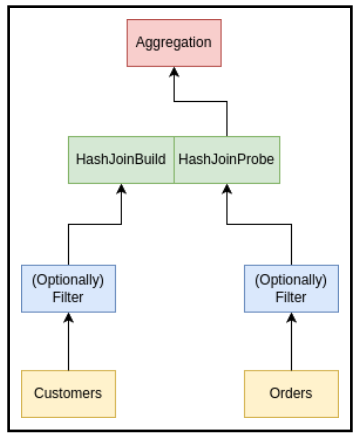
- You are free to do any valid re-clusterings. You can gain information based on the given engine and workloads:

1. Potentially analyze the given workloads
   - Hint: have a look at filtered columns (plus potentially the given domain sizes)
2. Potentially analyze the data
   - Hint: execute a table scan of the engine's data (plus think about how to sort data)
3. Implement re-clustering logic for the data
   - The re-clustering itself should be done by deleting all chunks (SdmsIcebergEngine::delete_chunks), and inserting the re-clustered data in new chunks (SdmsIcebergEngine::insert)

- The re-clustering is done as part of the benchmark, but **timed is only the query execution after the re-clustering**
- There is a basic version of the benchmark provided in `benches/basic_bench.rs`, as well as more info on the full benchmark in `benches/query_bench.rs`

Data & Queries

- In both basic and advanced tests, there are only two tables:
  - A customer table with a primary key, as well as an orders table that has a foreign key relation to the customer table.
- For simplicity, all queries (and workloads) are of the following structure:



Data & Queries (ii)

- Have a look at the (local-only) basic benchmark in `benches/basic_bench.rs`: They include basic data generation, workload definition, as well as query execution helper functions, to help you in testing your optimizer.
  - run `cargo bench --bench "basic_bench"`
- The data and workloads of the benchmark executed in the pipeline are not disclosed
  - The query/workload structure is the same as above
  - The schema of data is shown in `benches/query_bench.rs`

Task Files

Task	File
Optimizer/Re-Clustering	`src/engine/optimizer.rs`

Rust concepts

No new concepts required.

Benchmark (Bonus) - Grading

- After the programming project deadline, up to 2 Bonus points will be awarded based on the benchmark performance:
  - Better than a baseline performance (shown in leaderboard as “Baseline 1”) → 1 Bonus point
  - Better than a stronger baseline performance (shown in leaderboard as “Baseline 2”) → 1 Bonus point
- The two baselines will be published 2 weeks before the project deadline.

- The benchmark for Lab 3 is testing the query engine after your re-clustering, as decribed in task 5.

Benchmark (Bonus) - Output

```
68     Compiling rand-utf8 v0.0.1
69      Finished `bench` profile [optimized] target(s) in 5.66s
70       Running benches/disk_manager_bench.rs (target/release/deps/disk_manager_bench-e781c7525be8fd76)
71  $ echo "Delivering Results"
72  Delivering Results
73  $ PROJECT_NAME="SDMS Lab 0 (Bonus)" python3 deliver.py benchmark test_benchmark.json
74  Noted score 3335731 for "NB " with leaderboard name: Brave Broccoli for SDMS Lab 0 (Bonus) at localtime 2024-10-24 13:41:16.798413
75  Cleaning up project directory and file based variables
76  Job succeeded
```

Figure 13: Example benchmark output.
Shows your run time and personal nickname.

Benchmark (Bonus) - Leaderboard

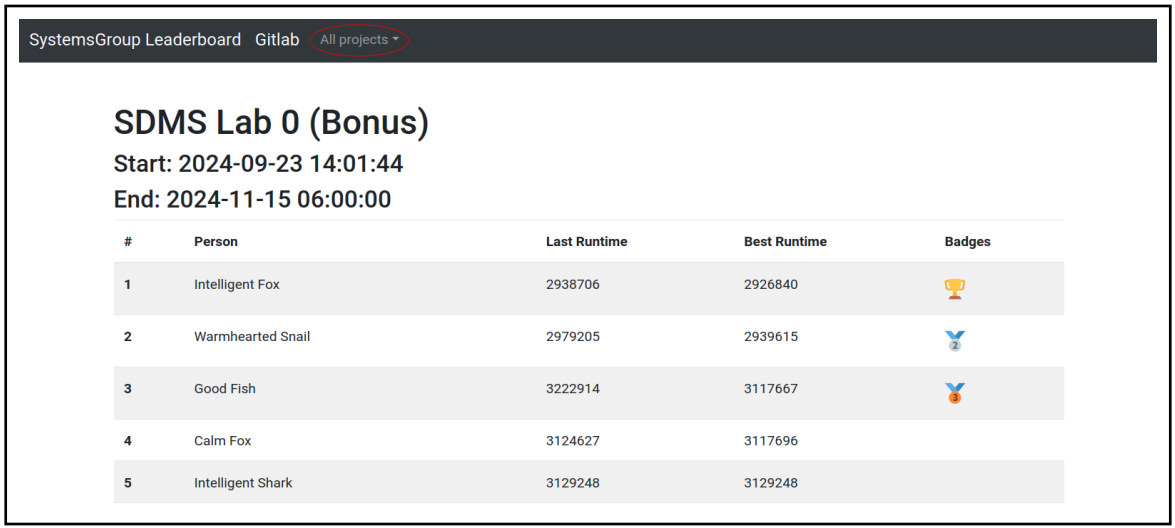


Figure 14: Leaderboard of Lab 0

Click on “All projects” to select the different leaderboards.

General Hints

- Some of the tasks are independent. You can start working on both task 1 and task 2
- Some tests write **files to disk** (in the `target/sdms` folder). Be careful not to fill up you disk, and do `cargo clean` regularly.
- Write your own tests (edge cases, mimic advanced tests description, ...)
  - Keep in mind: All code lines containing “#[test]”, “print”, “assert”, “dbg!” are commented out before evaluation
- Check the header of each file to see if it is being replaced by the evaluation pipeline runner
  - If it says it is, you either do not need to / should not change anything, or you need to add your implementation in other files.
- The evaluation pipelines shows information on which tests failed on which assertion
- There are ~75% basic and ~25% advanced tests for Lab 3
- The project must be solved individually. Copying code from other students counts as plagiarism.
- **Start early!** and use the time you have.

Reminder: How to test locally (basic tests):

```
$ cargo test --no-fail-fast # builds the projects and runs the tests. you can add "-- --
nocapture" to include print output
# ...
test result: FAILED. 0 passed; TODO failed; 0 ignored; 0 measured; 0 filtered out; finished
in 0.00s
```

Reminder: How to Ask Questions

1. First check the FAQ
2. Then check if your issue was already discussed in the forum and you can find a solution there
3. Then consider asking a question in the forum or via e-mail if it is a personal concern.

Please do not write us direct messages in Moodle. Also: Ask in the exercise :)

Deadline 07.03.2025 - 15:59:59

Have Fun!