Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Develop your own Database

Week 8

Agenda

- Review Sprint 3
- Group Meetings
  - Remember to find a replacement for next week's meeting

Review

```
// Is used to do the actual comparsion of values.
auto matches_scan_criterion =
  [](const auto& existing_value, const auto& search_value,
     const ScanType& scan_type) -> bool {
       switch (scan_type) {
         case ScanType::OpEquals:
           return existing_value == search_value;
         [...]
```

Hasso Plattner Institut

Review

```cpp
const auto chunk = referenced_table->get_chunk(row_id.chunk_id);
const auto segment = chunk.get_segment(_column_id);
if ((value_segment = std::dynamic_pointer_cast<ValueSegment<T>>(segment))) {
  values = value_segment->values();
  // [...]
```

Review

```cpp
ValueID find_in_dict(T value) const {
  auto upper_bound_ref = std::find(_dictionary->cbegin(),
                                    _dictionary->cend(),
                                    value);

  if (upper_bound_ref == _dictionary->cend()) {
    return INVALID_VALUE_ID;
  }
  auto x = upper_bound_ref - _dictionary->cbegin();

  return static_cast<ValueID>(x);
}
[...]
case ScanType::OpLessThan: {
  auto value = typed_dict_segment->find_in_dict(search_value());
  for (auto index = ValueID(0); index < size; index++) {
    auto attribute = typed_dict_segment->attribute_vector()->get(i);
    if (attribute < value) {
      posList.emplace_back(RowID({i, ChunkOffset(index)}));
    }
```

Review

```cpp
for (size_t pos_list_id = 0; pos_list_id < size_of_pos_list; ++pos_list_id) {
  RowID row_id = _pos_list->at(pos_list_id);

  AllTypeVariant value = _referenced_table->get_value(row_id,
                                                      _referenced_column_id);

  resolve_data_type(data_type, [&](auto type) {
    using Type = typename decltype(type)::type;
    in_scope = scan_compare(scan_type,
                            type_cast<Type>(value),
                            type_cast<Type>(search_value));
```

Review

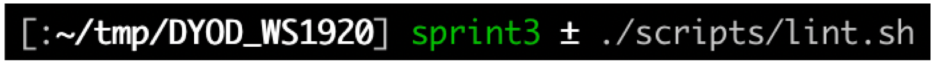
```
if (_add_all_chunk_offsets) {
  // If we need to add all values of the source pos_list, we can just reference the same list.
  result_segment = std::make_shared<ReferenceSegment>(reference_source_segment->referenced_table(),
                                                      reference_source_segment->referenced_column_id(),
                                                      reference_source_segment->pos_list());
} else {
  // We need to filter out some or all values. But, maybe for another segment that shared the
  // same PosList we already computed the resulting PosList for the new Segment.
```

Review

```
for (ChunkID i = ChunkID(0); i < in_table->chunk_count(); i++) {
// [...]
  case ScanType::OpEquals: {
    auto value = typed_dict_segment->find_in_dict(search_value());

    for (auto index = ValueID(0); index < size; index++) {
      auto attribute = typed_dict_segment->attribute_vector()->get(i);
```

Review



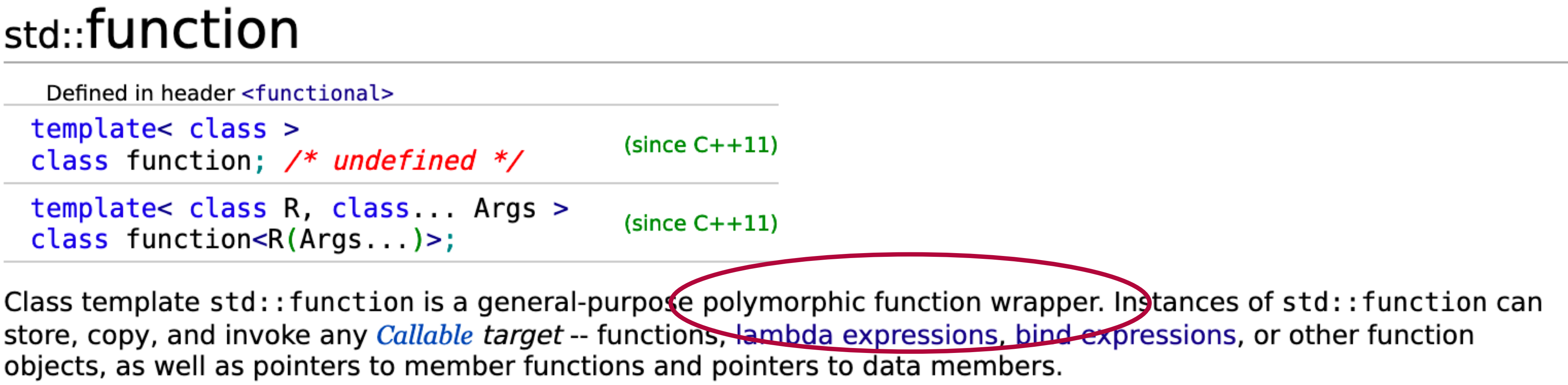
```
[:~/tmp/DYOD_WS1920] sprint3 ± ./scripts/lint.sh
```

std::function vs. lambda

- Seen many times:

```cpp
std::function<bool(const T&)> _create_compare_function() {
  switch (_scan_type) {
    case ScanType::OpEquals:
      return [=](const T& value) -> bool {
        return value == _search_value;
      };
    // [...]
```

std::function vs. lambda



std::function

Defined in header `<functional>`

```
template< class >
class function; /* undefined */
```
(since C++11)

```
template< class R, class... Args >
class function<R(Args...)>;
```
(since C++11)

Class template `std::function` is a general-purpose polymorphic function wrapper. Instances of `std::function` can store, copy, and invoke any *Callable target* -- functions, lambda expressions, bind expressions, or other function objects, as well as pointers to member functions and pointers to data members.

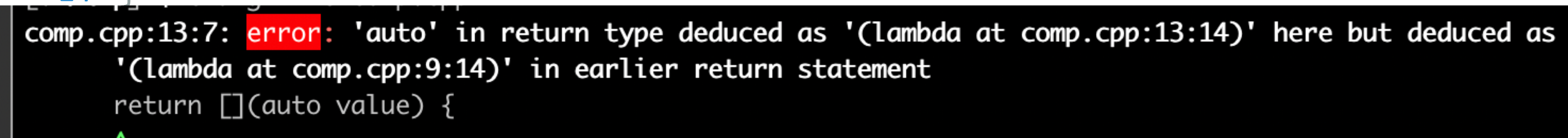
- http://quick-bench.com/AP81foBSotuVef9vlkDRYwvCn9Y

Hasso Plattner Institut

std::function vs. lambda

„How to create lambdas depending on the comparator?“

```cpp
 1 enum class Condition {
 2   Equals,
 3   NotEquals
 4 };
 5
 6 auto get_comparator(Condition condition) {
 7   switch(condition) {
 8     case Condition::Equals:
 9       return [](auto value) {
10         return value == 17;
11       };
12     case Condition::NotEquals:
13       return [](auto value) {
14         return value != 17;
15       };
16   }
17 }
```



```
comp.cpp:13:7: error: 'auto' in return type deduced as '(lambda at comp.cpp:13:14)' here but deduced as
      '(lambda at comp.cpp:9:14)' in earlier return statement
      return [](auto value) {
      ^
```

std::function vs. lambda

„How to create lambdas depending on the comparator?“

- You don‵t. Instead, pass whatever you want to do to the lambda

- Hollywood Principle / Inversion of Control

std::function vs. lambda

```
template <typename Functor>
void compare(Condition condition, Functor functor) {
  switch(condition) {
    case Condition::Equals:
      functor(std::equal_to<void>{});
      break;
    case Condition::NotEquals:
      functor(std::not_equal_to<void>{});
      break;
  }
}

void scan(const std::vector<int>& vector) {
  const auto search_value = 17;
  compare(Condition::Equals, [&](const auto comparator) {
    for (const auto value : vector) {
      if (comparator(value, search_value)) {
        std::cout << "match: " << value << std::endl;
      }
    }
  });
}
```

This avoids the performance issues of std::function.
It was not expected for Sprint 3.

std::function vs. lambda

http://quick-bench.com/tqljWKibQwNium5iZmmjLAFmVMQ

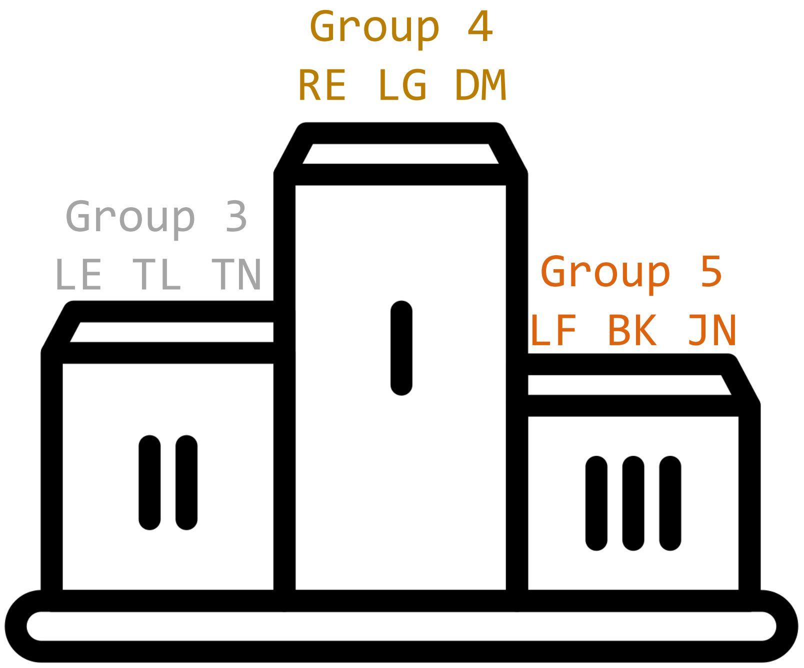
std::function vs. lambda

- Virtual method calls do not come for free
- In hot loops, we want to avoid them
- Trade-Off between performance and architecture

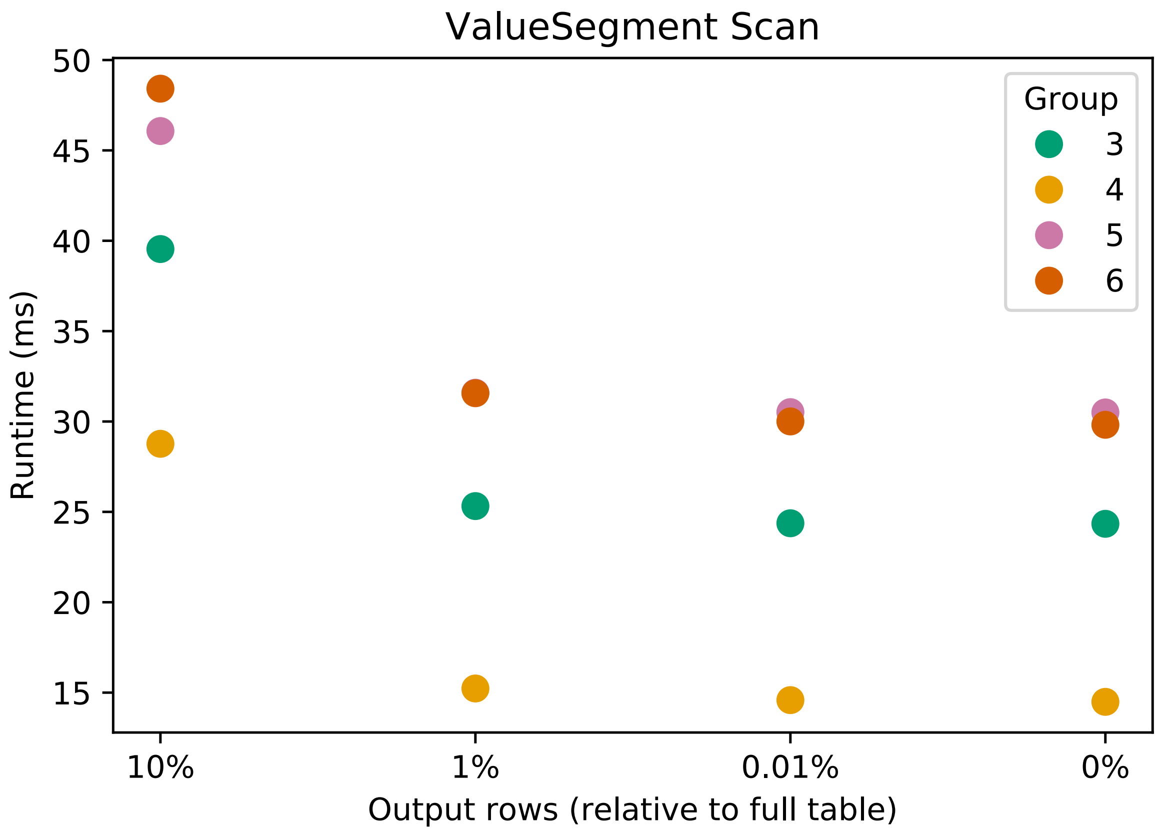
Performance Challenge

- 9 micro benchmarks to determine the performance of your table scans
  - Table size: 10M entries
  - Chunk sizes: **100K**, 1M, 10M
  - On Value-, Dictionary-, and ReferenceSegments
  - Predicate selectivities of 10%, 1%, 0.01%, 0%
  - We cleared the CPU cache between the experiments

Performance Challenge

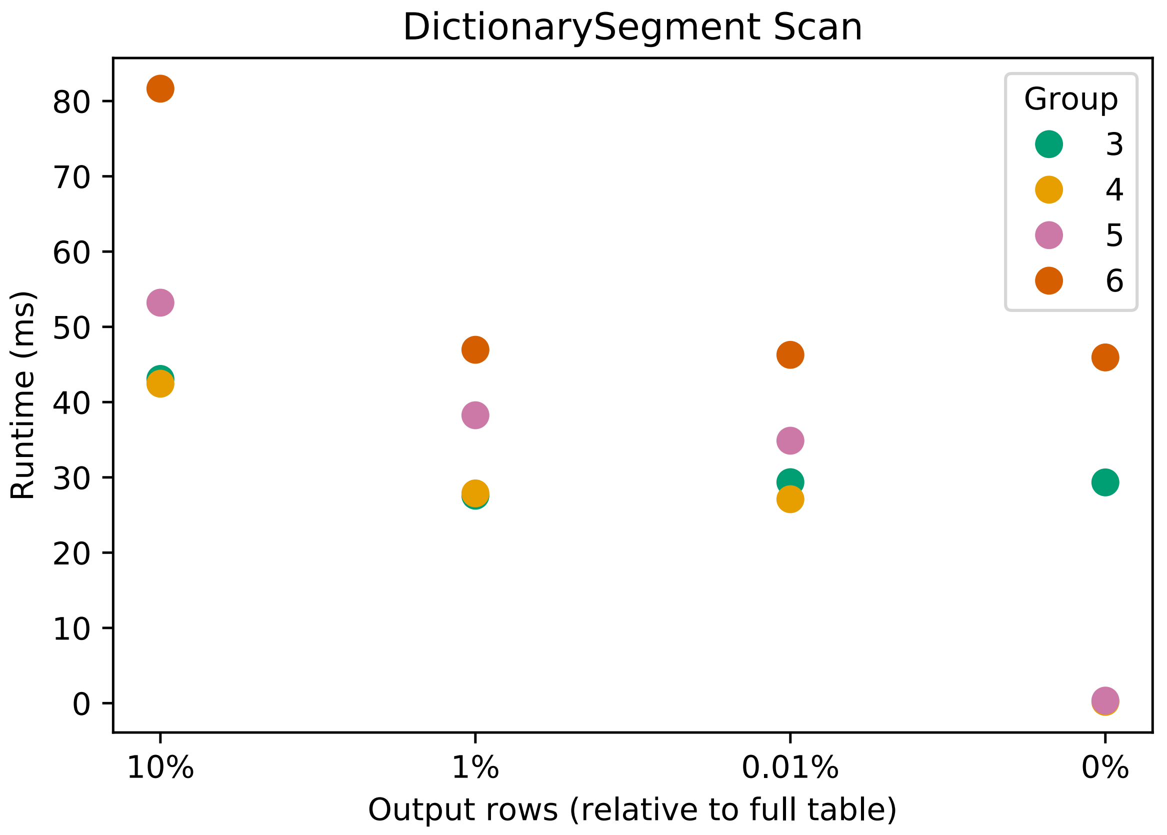


Performance Challenge



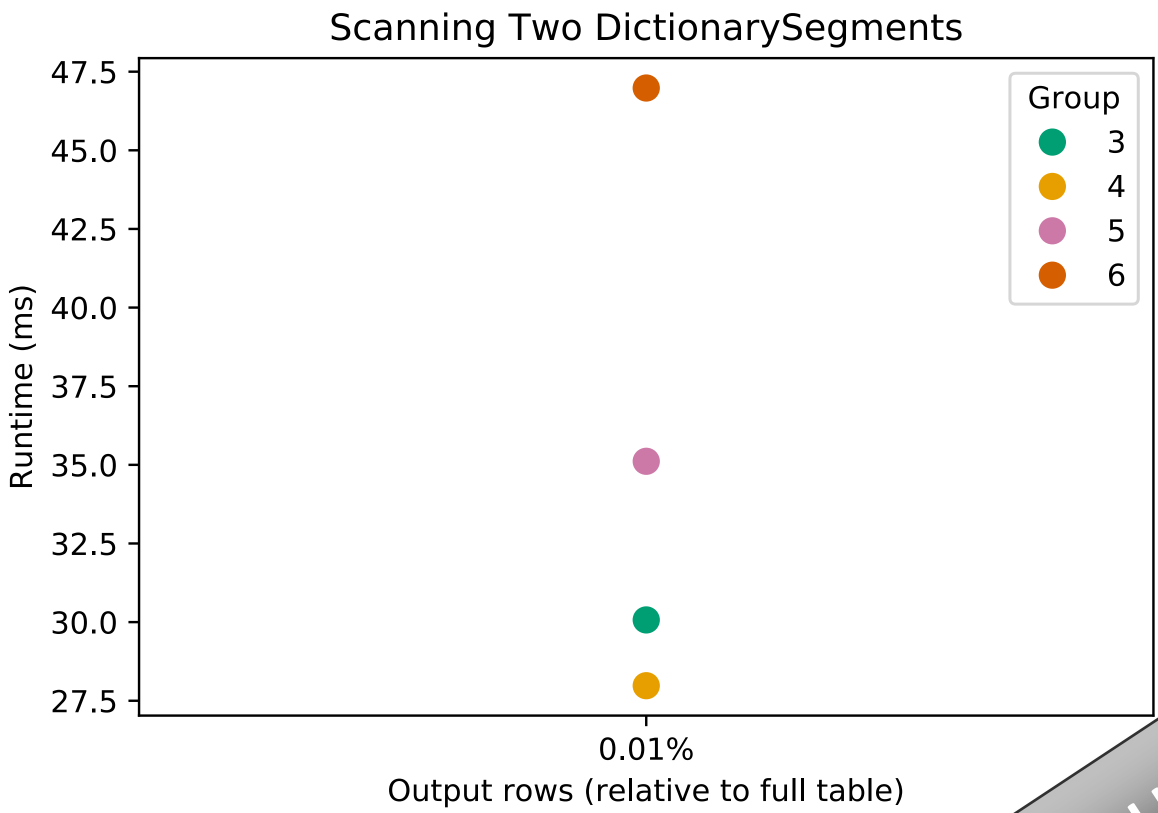
LE, TL, TN
RE, LG, DM
LF, BK, JN
II, YK, YL

Performance Challenge



```
LE, TL, TN
RE, LG, DM
LF, BK, JN
II, YK, YL
```

Performance Challenge



```
LE, TL, TN
RE, LG, DM
LF, BK, JN
II, YK, YL
```

Next: Group Meetings