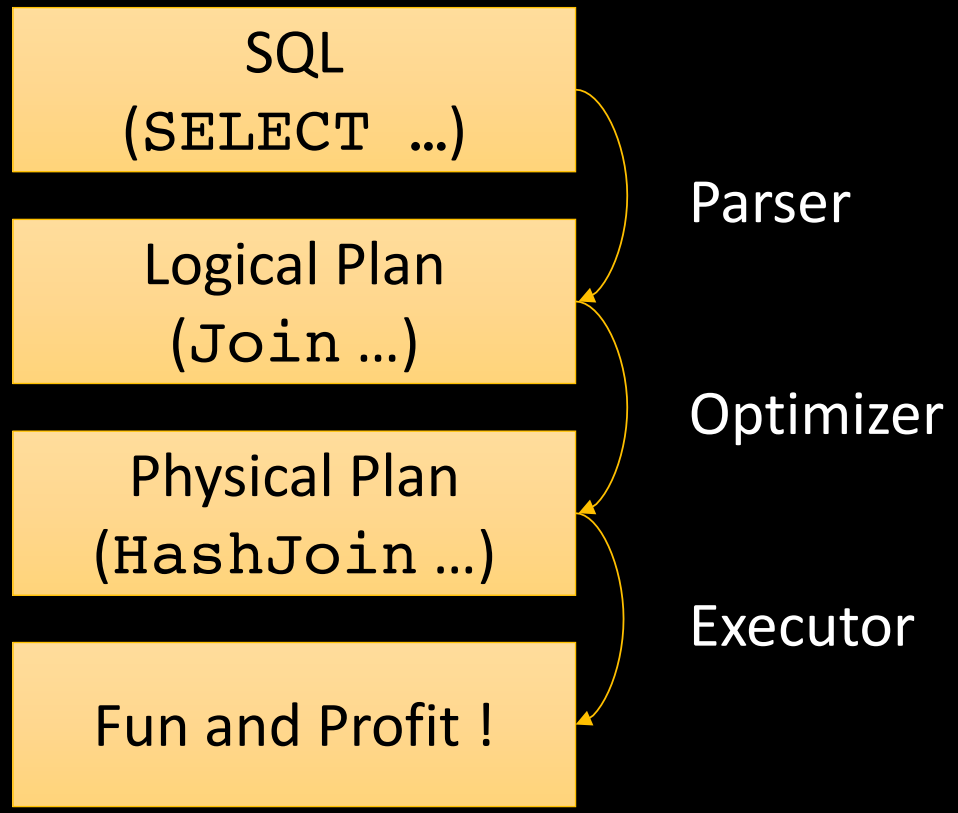
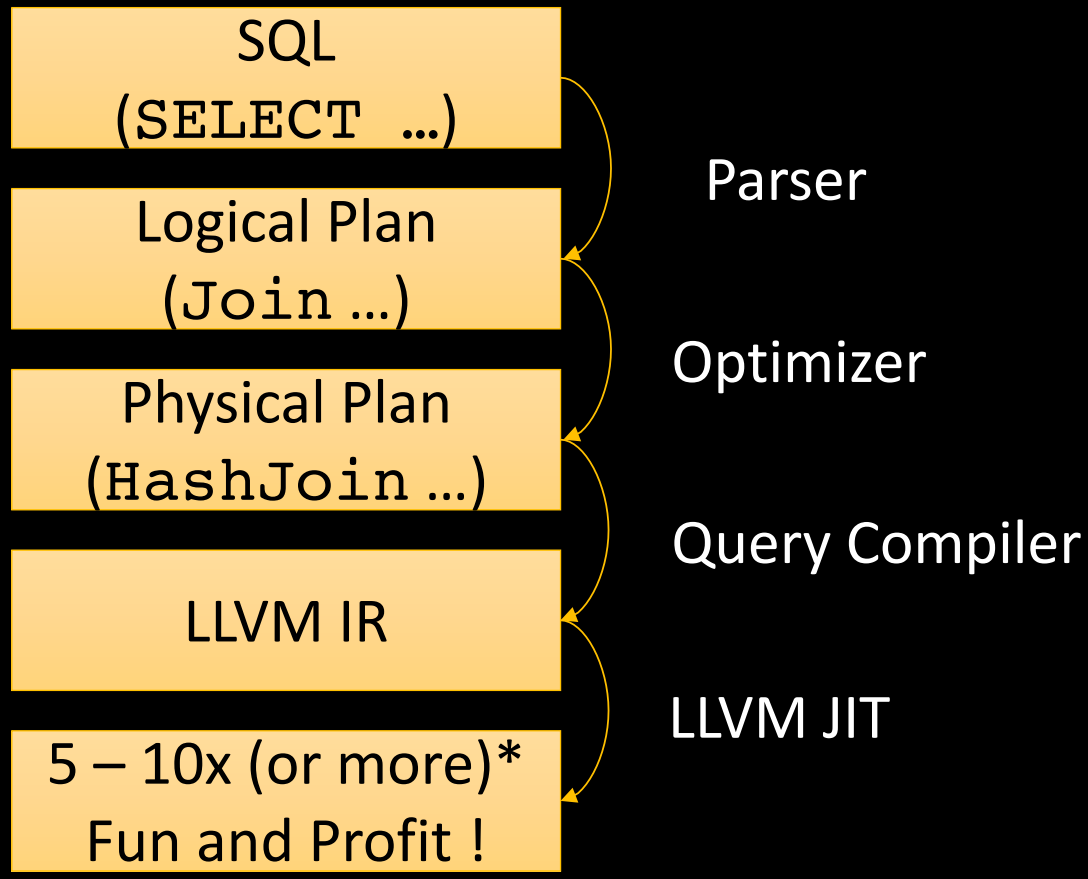
LLVM Query Compilation

Zhixun Tan
Shuyao Bi
Xinlyu Huang
Wei Cui

What are we doing

- Compile queries on the fly into native code with LLVM





What’s the plan?

- Parameterization
- Caching
- DELETE
- INSERT
- UPDATE
- INDEX (optional)

Tip: Call C++ functions from C/assembly

```
class DataTable {
  ...
  size_t DataTable::GetTileGroupCount() const;
};
```

```
struct DataTable {
  ...
};
size_t GetTileGroupCount(const DataTable* this);
```

```
size_t _ZNK7peloton7storage9DataTable17GetTileGroupCountEvy(DataTable* this);
```

namespace

class name

function name

Parameterization & Caching

```
SELECT *
FROM   table
WHERE  x < 10
```

```
void execute(output_t* output):
  for tuple in table:
    if tuple.x < 10:
      output->push(tuple)
```

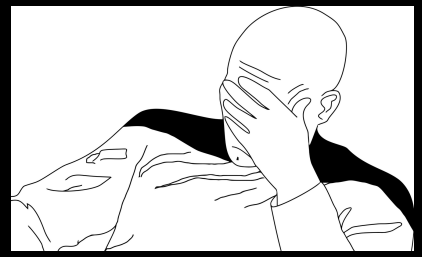
```
SELECT *
FROM   table
WHERE  x < 11
```

```
void execute(output_t* output):
  for tuple in table:
    if tuple.x < 11:
      output->push(tuple)
```

```
void execute(output_t* output, int param1):
  for tuple in table:
    if tuple.x < param1:
      output->push(tuple)
```

How’s the progress?

- Parameterization ❌ will consider newer design
- Caching comparison done
- DELETION implemented
- INSERTION ✔ simple case
- UPDATE
- INDEX (optional)



Any surprises ?

- SELECT * FROM User WHERE uid > 2;
- SELECT * FROM User WHERE 1;
- SELECT * FROM User WHERE 3 > 2;
- SELECT * FROM User WHERE uid < (SELECT max(uid)/2 FROM User);

- Parameters can be complicated.
- No type info in ParameterValueExpression .