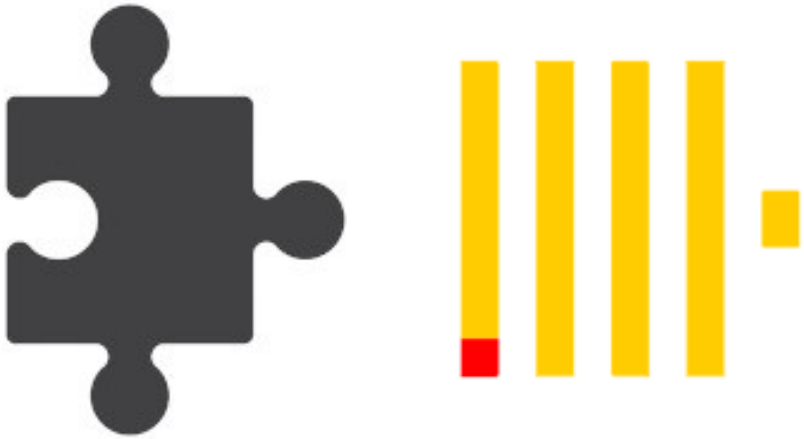
Паттерны ООП в ClickHouse



Косенко Дмитрий

разработчик софта

> 10 лет опыта

Образовательный доклад с вопросом к размышлению

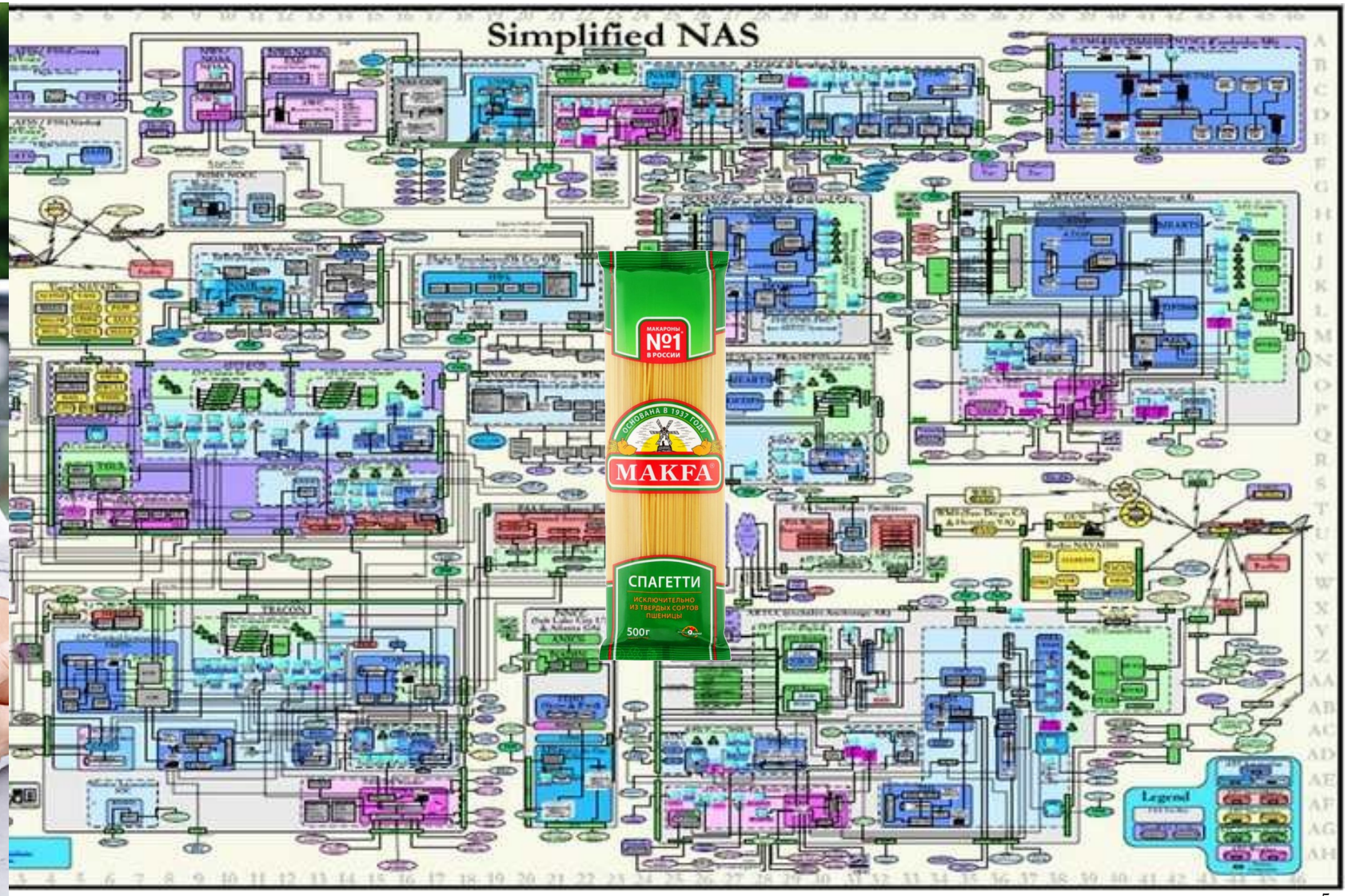
План

- Различие между разными уровнями архитектуры
- Какое место в архитектруре занимают паттерны?
- История паттернов + мемасики
- Паттерны и SOLID
- Немного устройства БД и особенностей ClickHouse
- Поищем паттерны в ClickHouse
- Итоги

Простая программа

```cpp
// Your First C++ Program

#include <iostream>

int main() {
    std::cout << "Hello World!";
    return 0;
}
```

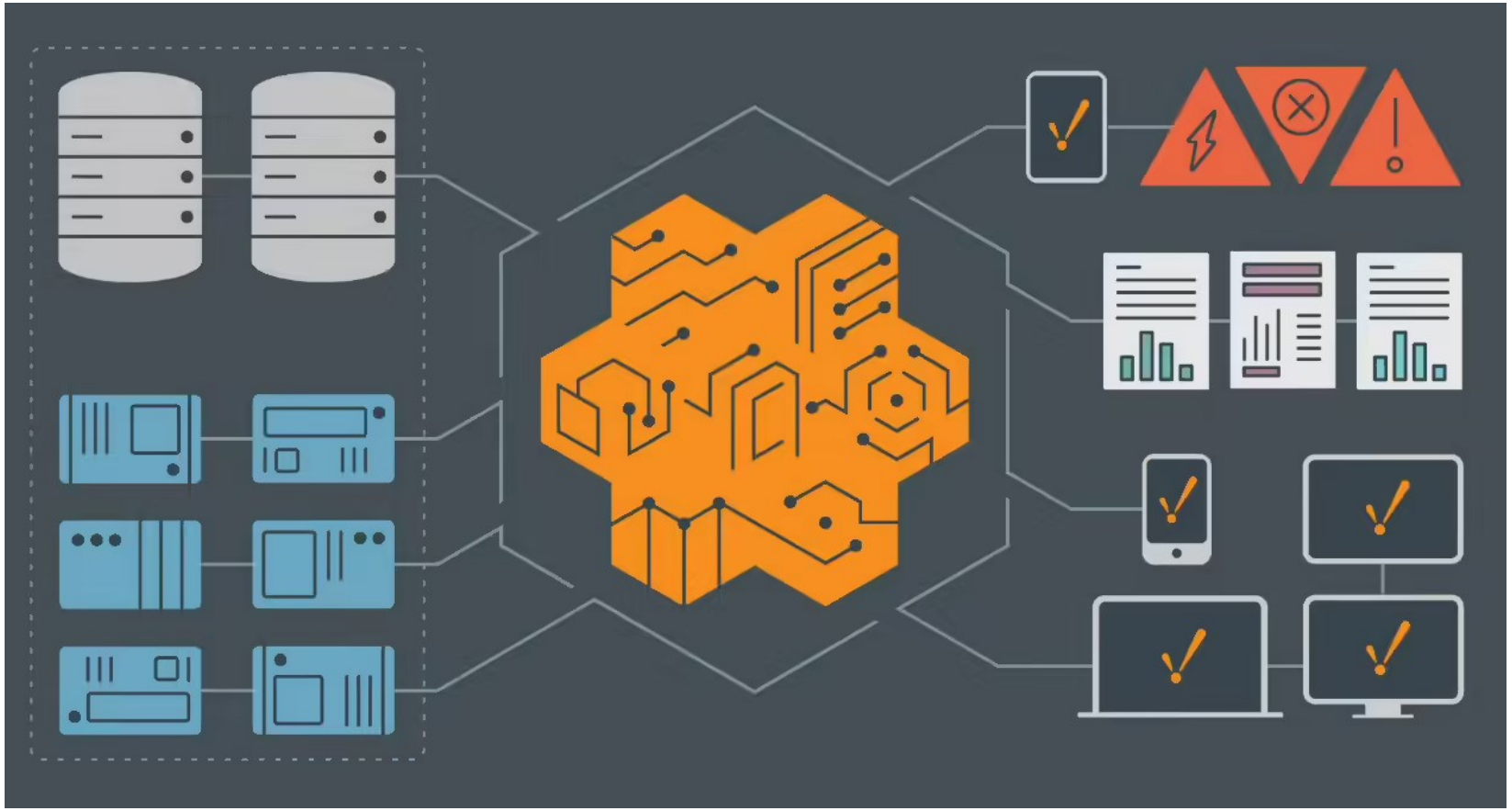
Чуть сложнее...



Архитектура

- упрощение взаимодействия между разработчиками
- снизить сложность восприятия системы
- изменения изолированы, не приводят к разрушению смежных фич
- совокупность важнейших решений об организации програмной системы
- накопить готовые решения для однотипных задач

Шаблоны



- быстрее
- рациональнее
- проще
- надёжнее

Такие разные паттерны...

- на наивысшем уровне - конструкция всей системы
- паттерны ООП - уровень кода
- идиомы - «низкоуровневые» шаблоны учитывающие специфику ЯП
- алгоритмы - шаблоны вычисления

Примеры

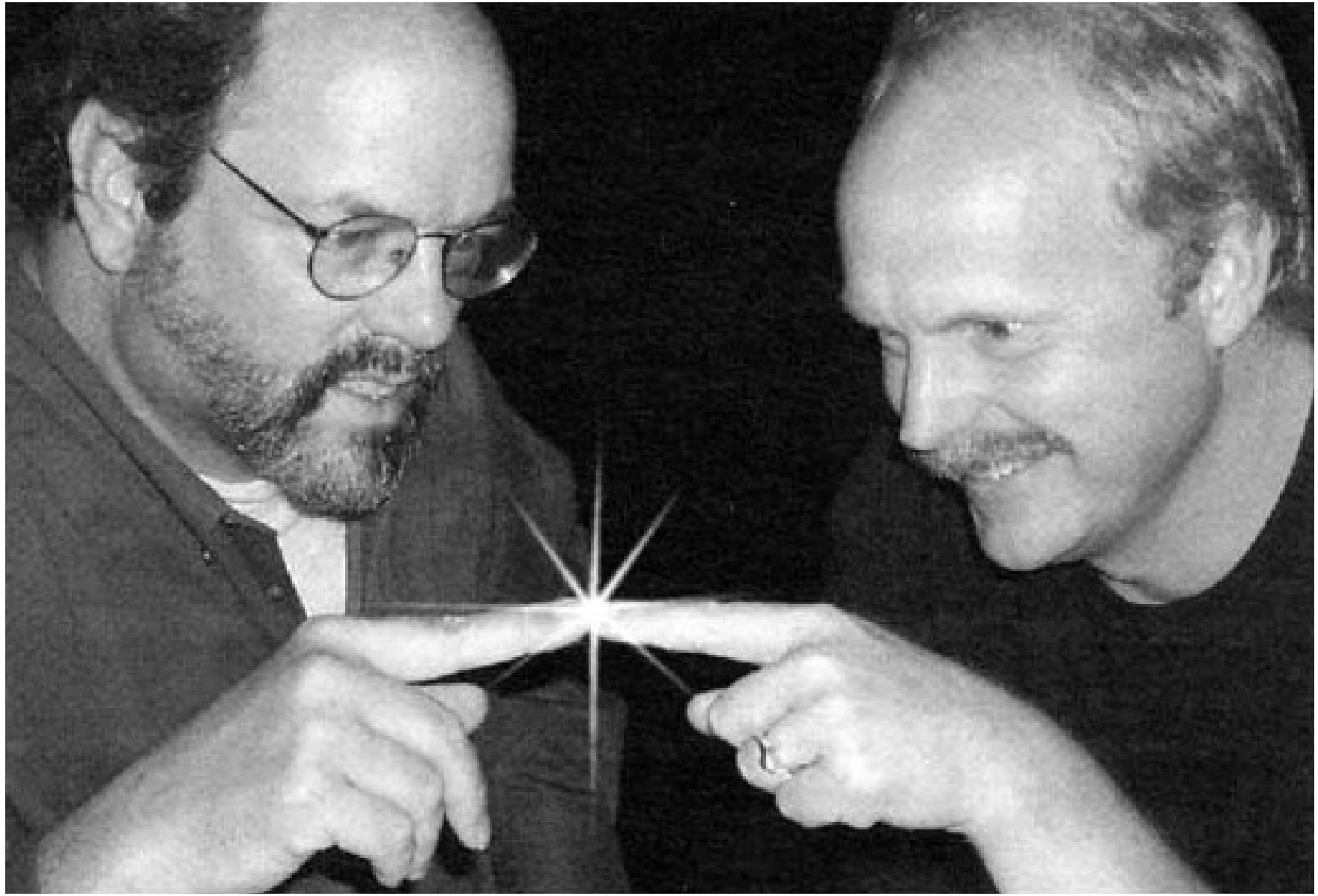
- event sourcing - это паттерн проектирования уровня системы
- singleton - паттерн ООП
- stl::iterator - идиома C++
- метод Монте-Карло - шаблон вычисления

Microservices - паттерн?

Как их классифицировать?

- The Essence of Architecture Composition
- Manifesto for Types of Architecture Levels
- Enterprise Architecture Layering

1970-е. Кристофер Александер

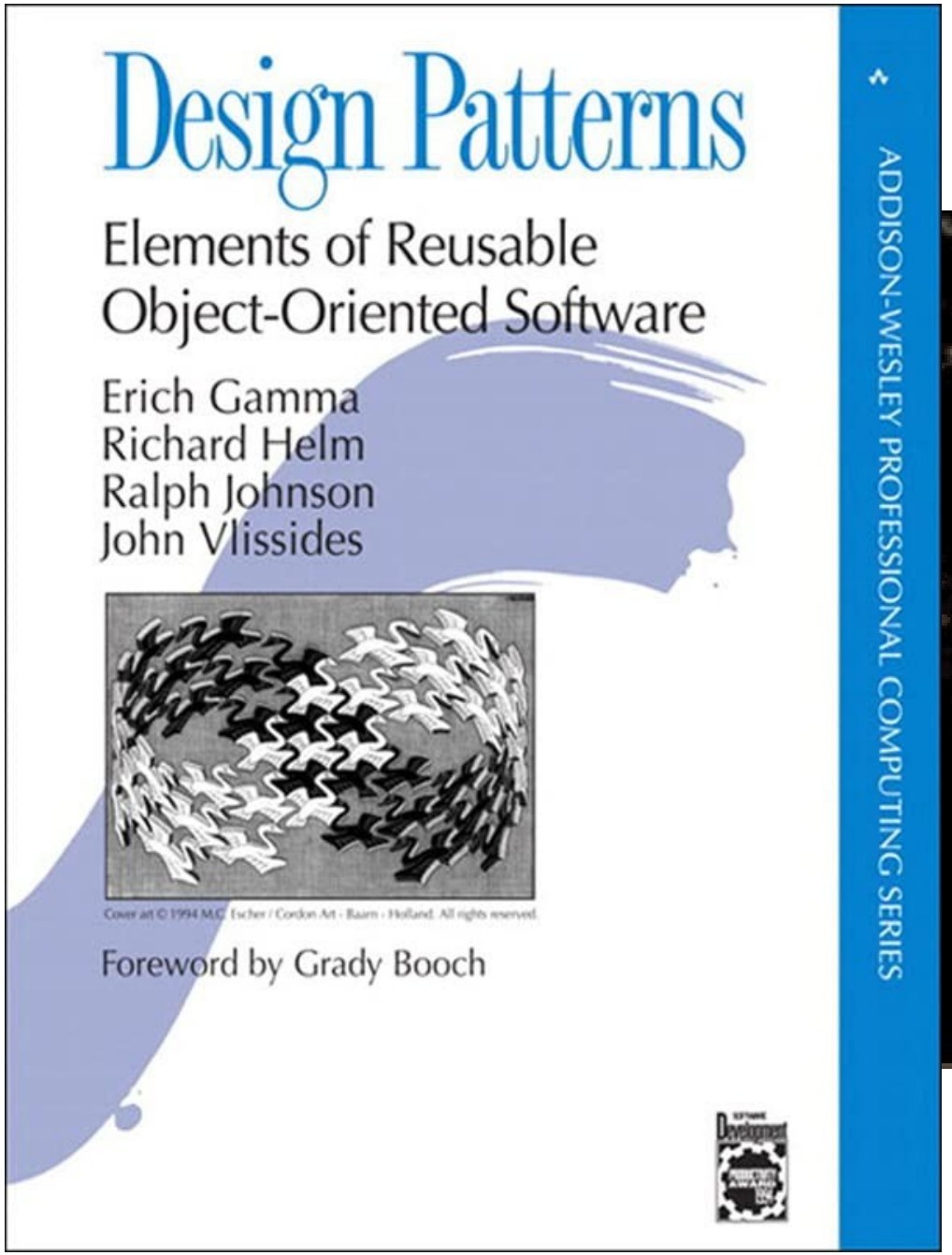


1987. Кент Бэк, Вард Каннингем

1988. Эрих Гамма

1991. Джеймс Коплин

“Банда”


Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES


Design Patterns

Паттерны

Behavioral

- Adapter
- Bridge
- Composite
- Decorator
- Flyweight
- Proxy
- Facade
- Chain of responsibility

Creational

- Singleton
- Factory Method
- Abstract Factory
- Builder
- Prototype
- Command
- Interpreter
- Iterator

Structural

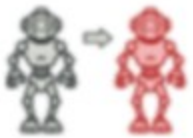
- Mediator
- Memento
- Observer
- State
- Strategy
- Template method
- Visitor

SOLID

- single responsibility
- open–closed
- Liskov substitution
- interface segregation
- dependency inversion

СУБ
Д

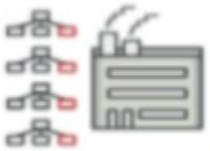
Interpreter



Prototype

Template method

Facade



Abstract Factory

Flyweight

СУБД

SELECT 1;

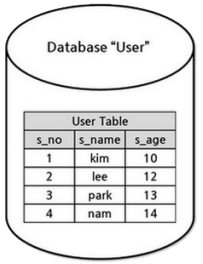
сервер

```
int main() {
    while (1) {
        handle();
    }
}
```

доступ к данным

анализатор запросов

чтение-запись данных



таблицы

файло-табличный преобразователь

файлы

хранение данных

2012(2016). Алексей Миловидов

ClickHouse

аналитическая СУБД для больших данных

https://github.com/ClickHouse/ClickHouse

31,5k

83.1%

LOC 1`138`607

5711

29

ClickHouse

- колоночная аналитическая СУБД
- обработка "сырых" данных в режиме реального времени
- собственный диалект SQL, в т.ч. вероятностные структуры данных
- нет точечных UPDATE / DELETE / SELECT
- нет транзакций
- отсутствие полноценного оптимизатора запросов
- оптимизирована для хранения данных на жёстких дисках

Джентельменский набор

- ☑ CTRL + F
- ☑ egrep -i
- ☑ 

ClickHouse

- Access
- AggregateFunctions
- Analyzer
- Backups
- Bridge
- BridgeHelper
- Client
- Columns
- Common
- Compression
- Coordination
- Core
- Daemon
- Databases
- DataTypes
- Dictionaries
- Disks
- Formats
- Functions
- Interpreters
- IO
- Loggers
- Parsers
- Planner
- Processors
- QueryPipeline
- Server
- Storages
- TableFunctions

ClickHouse

Bridge
BridgeHelper

?

Bridge

Interpreters

?

Interpreter

QueryPipeline

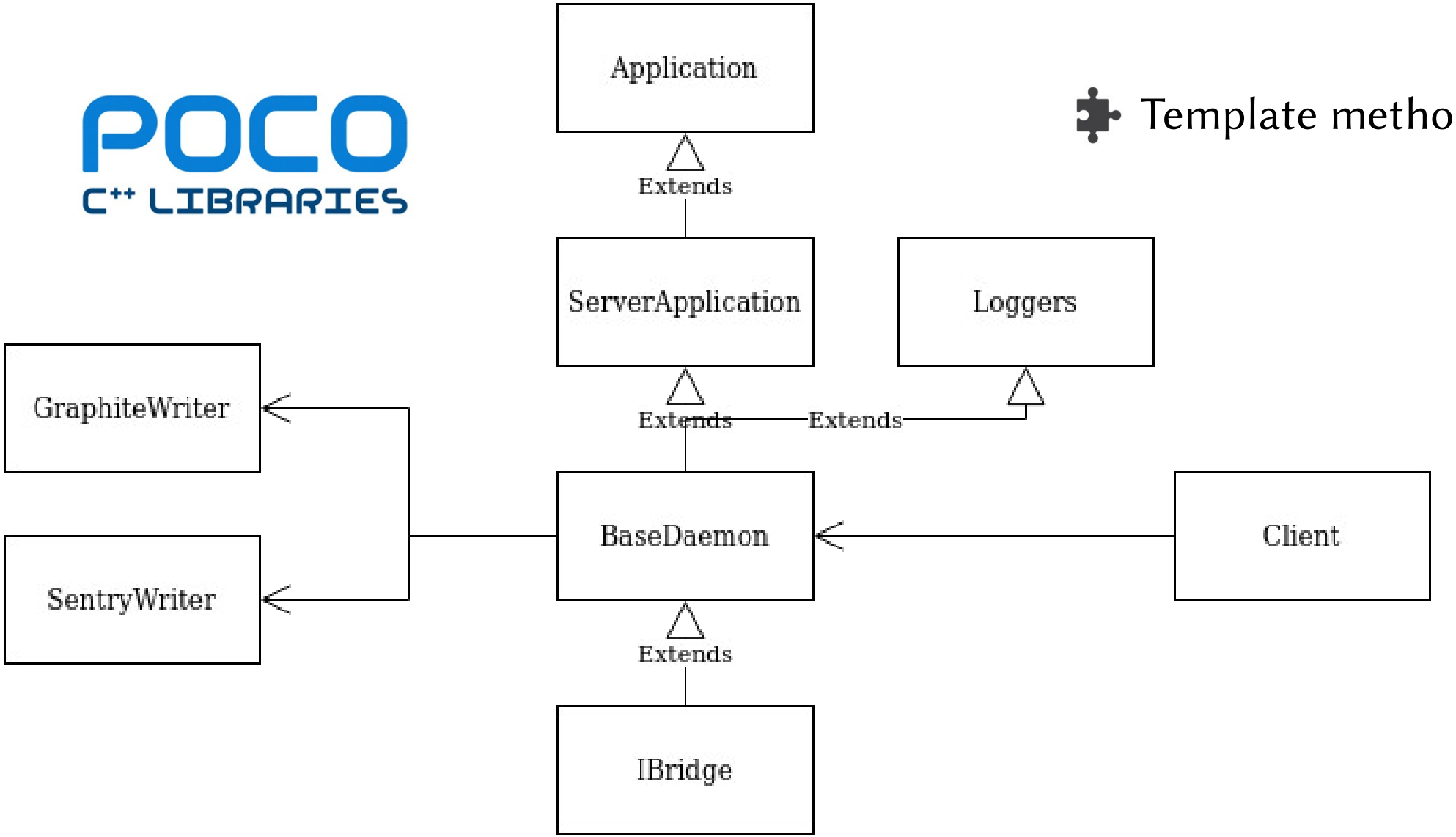
?

Chain of responsibility

Bridge Daemon

POCO C++ LIBRARIES

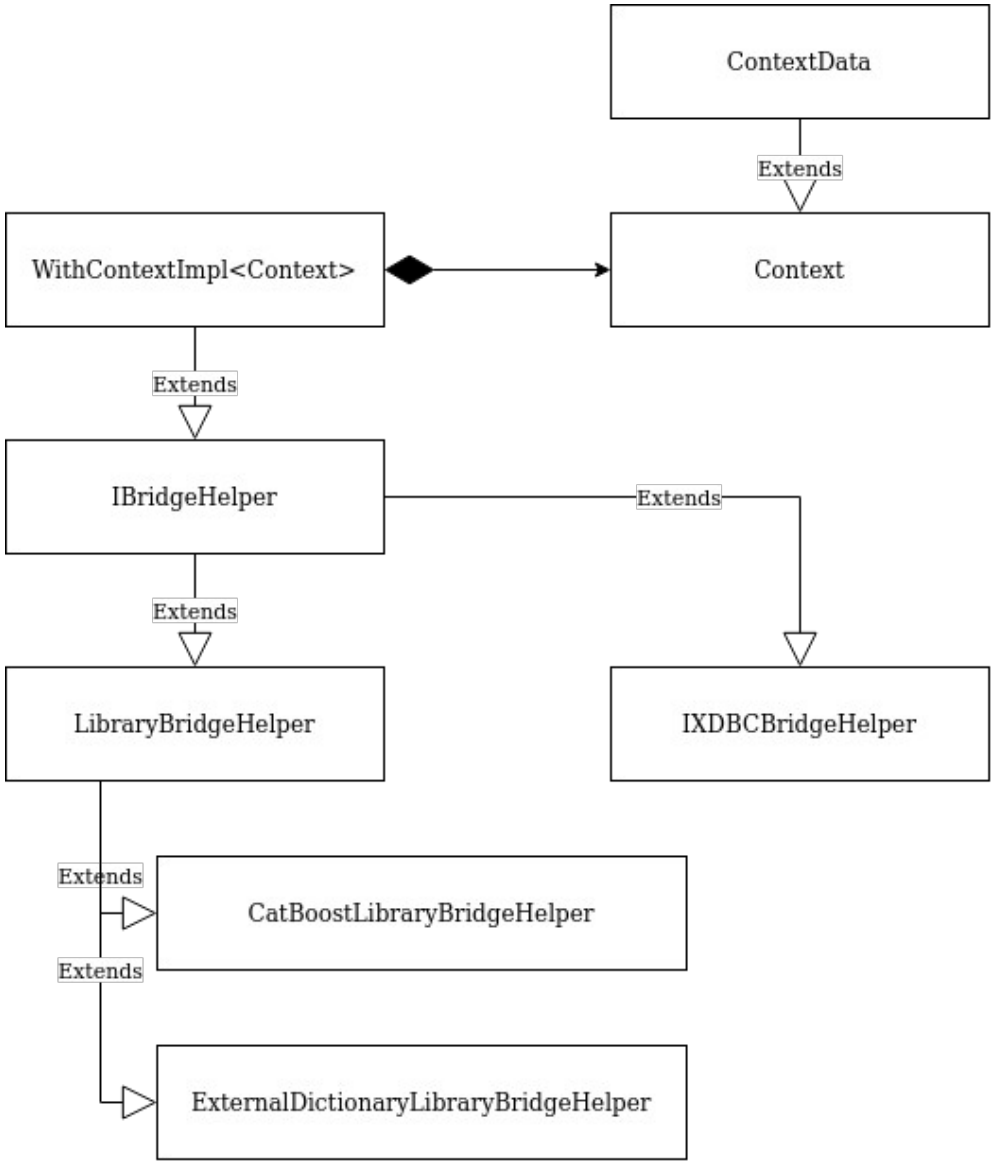
Template method



```
#include <Poco/Process.h>
#include <Poco/ThreadPool.h>
#include <Poco/Util/Application.h>
#include <Poco/Util/ServerApplication.h>
#include <Poco/Net/SocketAddress.h>
...
class BaseDaemon : public Poco::Util::ServerApplication, public Loggers
{
```

```
int IBridge::main(const std::vector<std::string> & /*args*/)
{
...
    auto server = HTTPServer(
        std::make_shared<HTTPContext>(context),
        getHandlerFactoryPtr(context),
        server_pool,
        socket,
        http_params);
...

    server.start();
    LOG_INFO(log, "Listening http://{}", address.toString());

    waitForTerminationRequest();
    return Application::EXIT_OK;
}
```

BridgeHelper




```
class Context;

/// Most used types have shorter names
using ContextPtr = std::shared_ptr<const Context>;
using ContextMutablePtr = std::shared_ptr<Context>;
using ContextWeakPtr = std::weak_ptr<const Context>;
using ContextWeakMutablePtr = std::weak_ptr<Context>;

template <class Shared = ContextPtr>
struct WithContextImpl
{
...
};

using WithContext = WithContextImpl<>;
using WithConstContext = WithContext; /// For compatib
using WithMutableContext = WithContextImpl<ContextMuta
```

Interpreters

- 510
- 83 = Visitor
- 116 = Interpreter

- ActionsDAG
- ActionsVisitor
- Aggregator
- Cluster
- ClusterDiscovery
- CompileDAG
- compileFunction
- Context
- DatabaseCatalog
- executeQuery

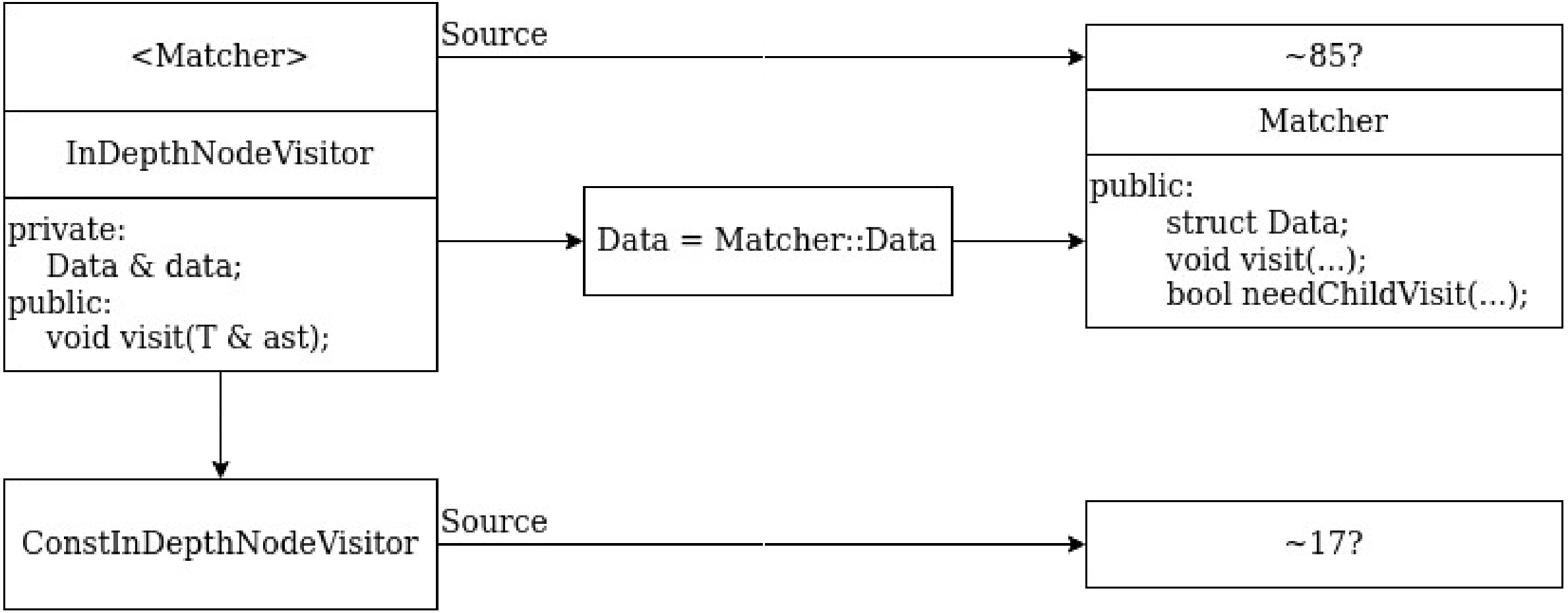
- ExpressionActions
- ExpressionAnalyzer
- FileCache
- IdentifierSemantic
- IInterpreter
- ITokenExtractor
- Lemmatizers
- Metadata
- TreeRewriter

Visitor

```
/// Visits AST tree in depth, call functions for nodes according to Matcher type data.
/// You need to define Data, visit() and needChildVisit() in Matcher class.
template <
    typename Matcher,
    bool _top_to_bottom,
    bool need_child_accept_data = false,
    typename T = ASTPtr
>
class InDepthNodeVisitor
{
public:
    using Data = typename Matcher::Data;
...

    void visit(T & ast)
    { ...
    }

private:
    Data & data;
...

    template <bool with_dump>
    void visitImplMain(T & ast)
    {
        if constexpr (!_top_to_bottom)
            visitChildren<with_dump>(ast);

        doVisit(ast);

        if constexpr (_top_to_bottom)
            visitChildren<with_dump>(ast);
    }
```

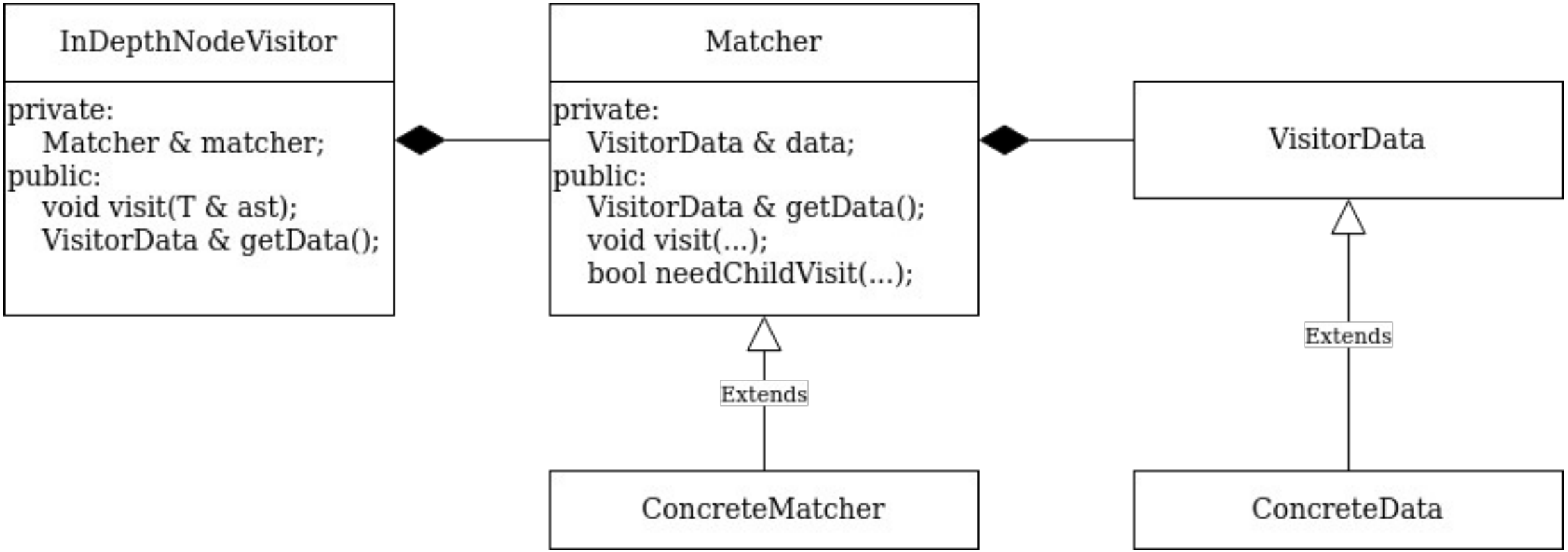
```
void doVisit(T & ast)
{
    try
    {
        Matcher::visit(ast, data);
    }
    catch (Exception & e)
    {
        e.addMessage("While processing {}",
        throw;
    }
}

template <bool with_dump>
void visitChildren(T & ast)
{ ...
}
```

InDepthNodeVisitor

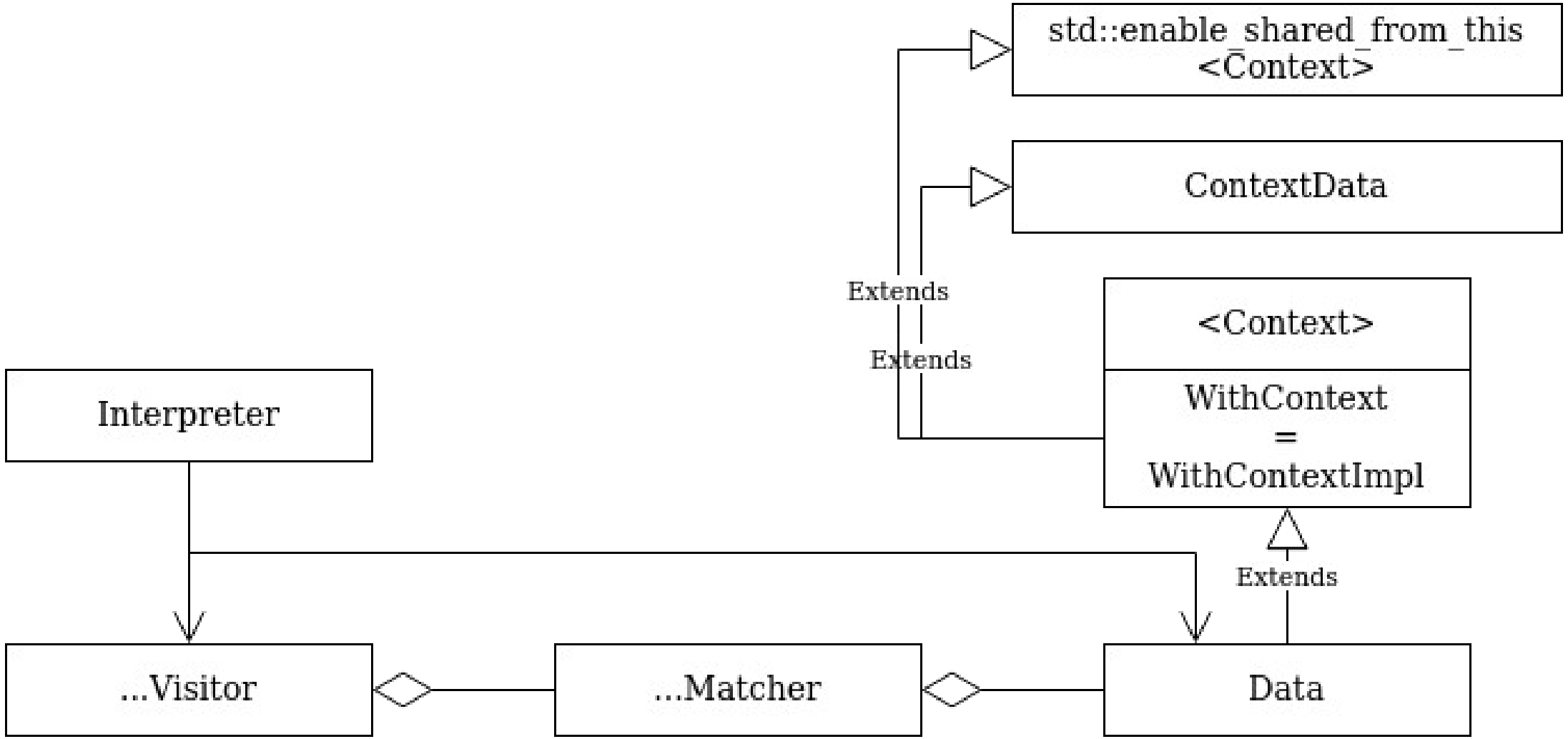


InDepthNodeVisitor

Strategy State Template method



Visitor

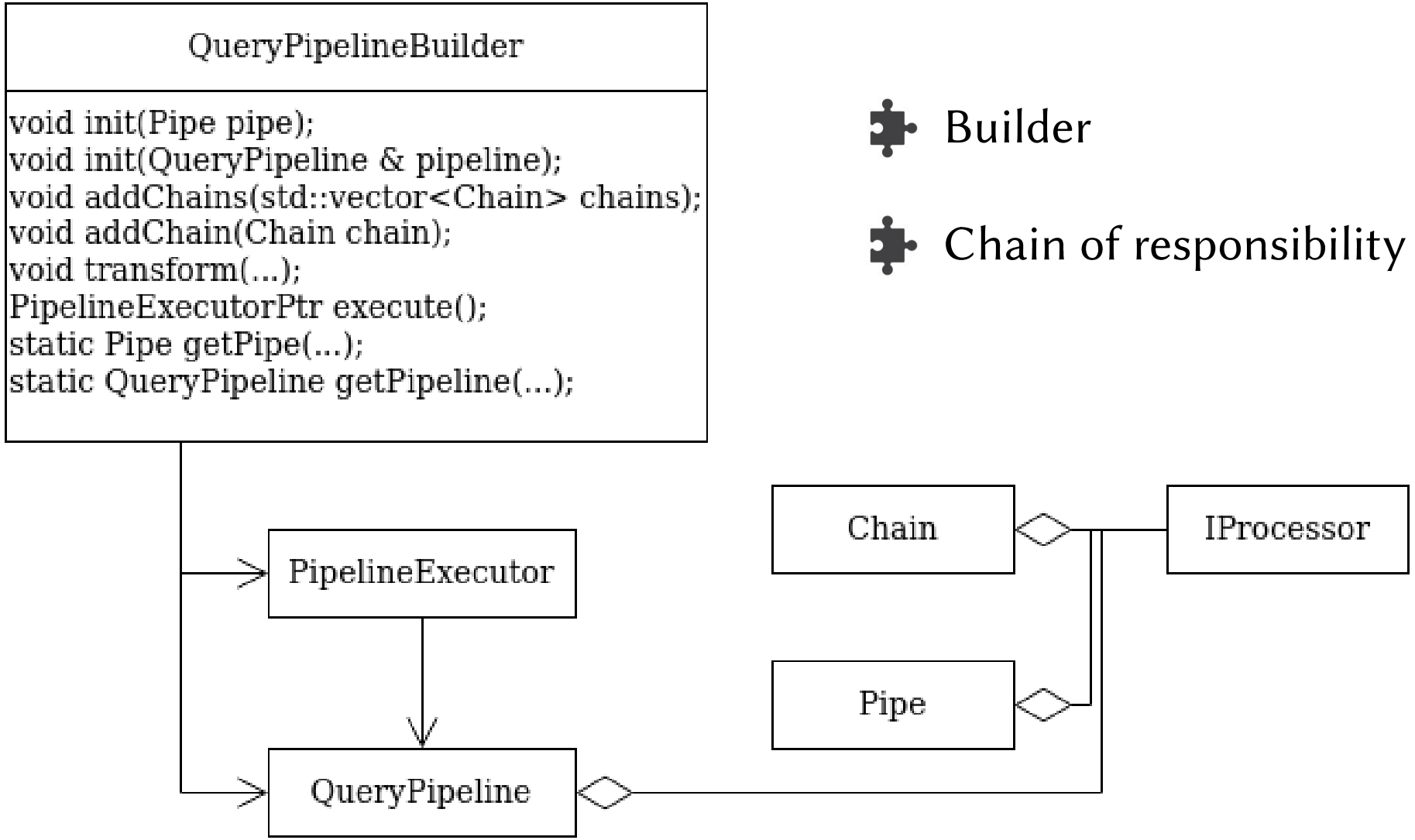

std::enable_shared_from_this
<Context>
ContextData
Extends
Extends
<Context>
WithContext
=
WithContextImpl
Interpreter
Extends
...Visitor
...Matcher
Data

Visitor

```
class RewriteOrderBy
{
public:
    struct Data {};
    static void visit(ASTPtr & ast, Data &);
    static bool needChildVisit(const ASTPtr &, const ASTPtr &) { return true; }
};

using RewriteOrderByVisitor = InDepthNodeVisitor<RewriteOrderBy, true>;
}
```

```
TreeRewriterResultPtr TreeRewriter::analyzeSelect(
    ASTPtr & query,
    TreeRewriterResult && result,
    const SelectQueryOptions & select_options,
    const TablesWithColumns & tables_with_columns,
    const Names & required_result_columns,
    std::shared_ptr<TableJoin> table_join) const
{
...
    // remove outer braces in order by
    RewriteOrderByVisitor::Data data;
    RewriteOrderByVisitor(data).visit(query);

    return std::make_shared<const TreeRewriterResult>(result);
}
```

QueryPipelineBuilder



- Builder
- Chain of responsibility

ClickHouse

- Access
- AggregateFunctions
- Analyzer
- Backups
- ☑ Bridge
- ☑ BridgeHelper
- Client
- Columns
- Common
- Compression
- Coordination
- Core
- ☑ Daemon
- Databases
- DataTypes
- Dictionaries
- Disks
- Formats
- Functions
- ☑ Interpreters
- IO
- Loggers
- ☑ Parsers
- Planner
- Processors
- ☑ QueryPipeline
- Server
- Storages
- TableFunctions

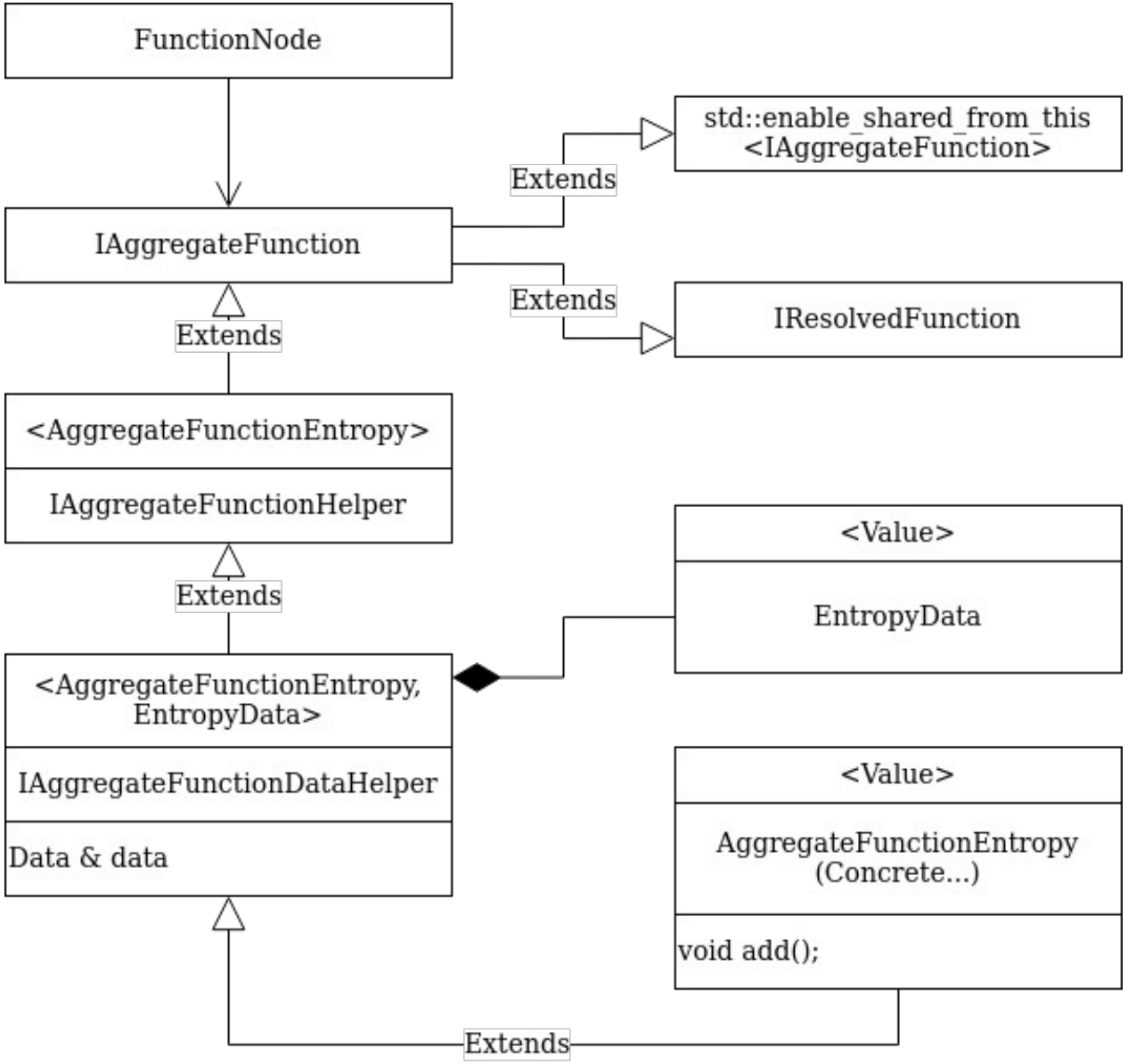
AggregateFunctions

- 180
- 144 = AggregateFunction*

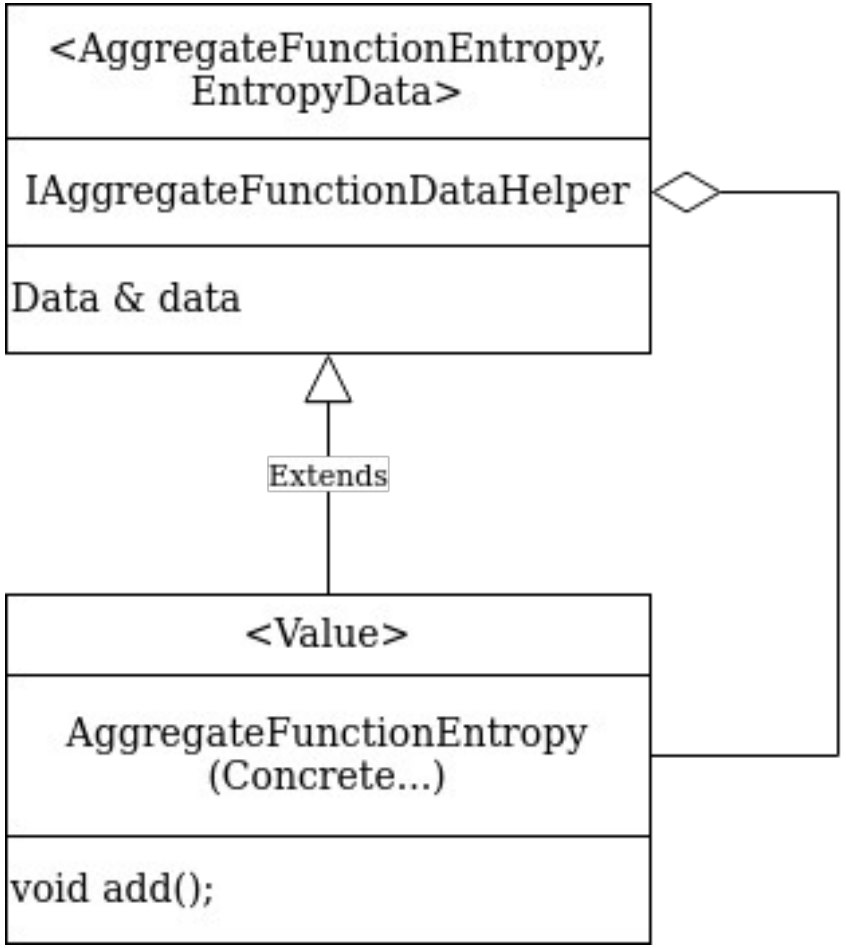
```
enum class FunctionKind
{
    UNKNOWN,
    ORDINARY,
    AGGREGATE,
    WINDOW,
};
```

- FactoryHelpers
- IAggregateFunction
- IAggregateFunctionCombinator
- parseAggregateFunctionParameters
- registerAggregateFunctions

IAggregateFunction

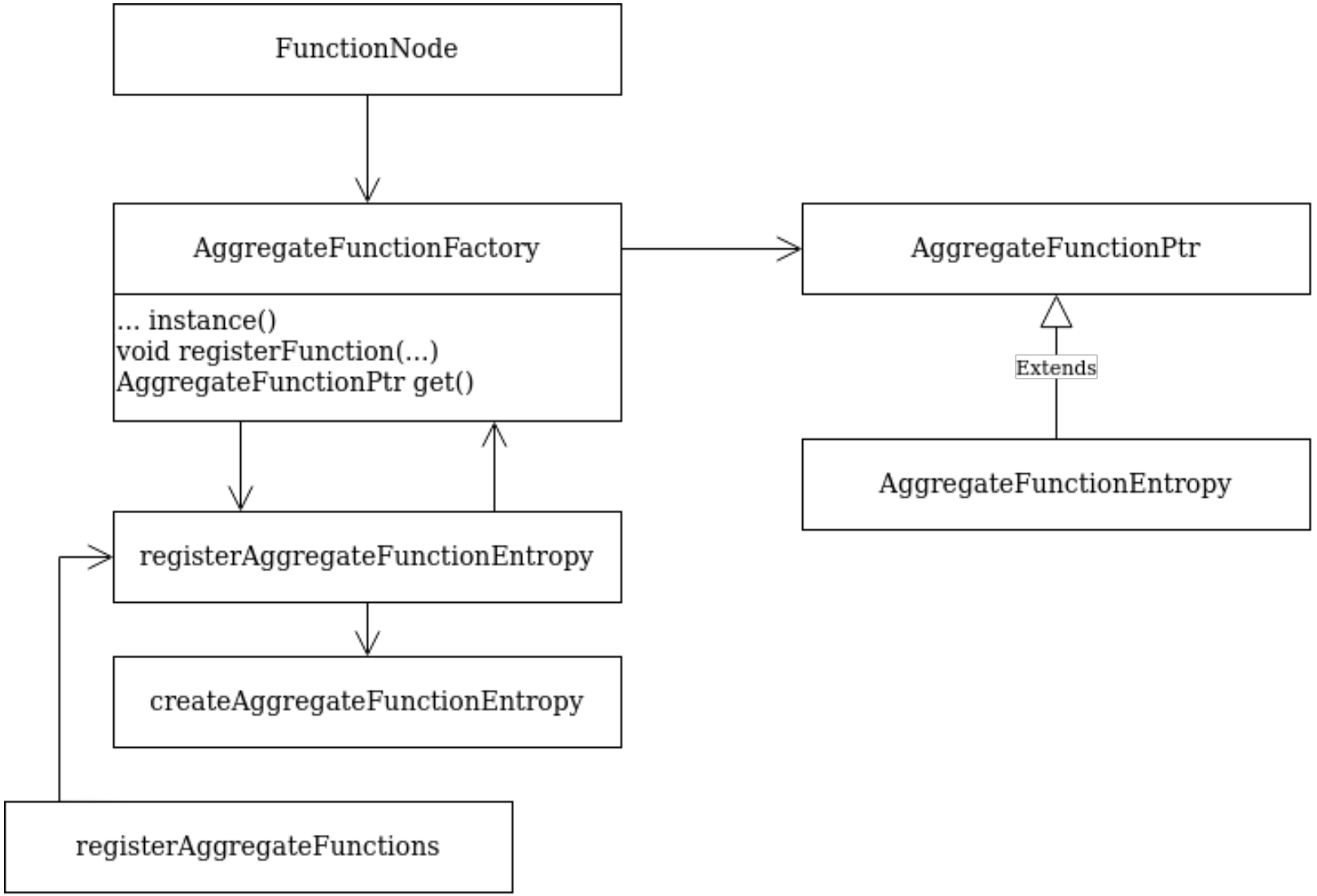


- Template method
- State
- Command
- Facade
- ...


<AggregateFunctionEntropy, EntropyData>
IAggregateFunctionDataHelper
Data & data
Extends
<Value>
AggregateFunctionEntropy (Concrete...)
void add();


A. Adapter
B. Bridge
C. Composite
D. Decorator

registerAggregateFunctions



- Factory method
- Singleton
- Abstract Factory

registerAggregateFunctions

```
void registerAggregateFunctions()
{
    {
        auto & factory = AggregateFunctionFactory::instance();
…
        registerAggregateFunctionEntropy(factory);
…
    }
```

```
AggregateFunctionPtr createAggregateFunctionEntropy(
    const std::string & name, const DataTypes & argument_types, const Array & para
{
…     /// Generic implementation for other types or for multiple arguments.
    return std::make_shared<AggregateFunctionEntropy<UInt128>>(argument_types);
}

}

void registerAggregateFunctionEntropy(AggregateFunctionFactory & factory)
{
    factory.registerFunction("entropy", createAggregateFunctionEntropy);
}
```

AggregateFunctionFactory

```
AggregateFunctionFactory & AggregateFunctionFactory::instance()
{
    static AggregateFunctionFactory ret;
    return ret;
}
```

```
class AnyFunctionVisitor : public InDepthQueryTreeVisitorWithContext<AnyFunctionVisitor>
{
    void enterImpl(QueryTreeNodePtr & node)
    {

        for (auto & inside_argument : inside_arguments)
        {

            auto aggregate_function = AggregateFunctionFactory::instance().get(function_name,

            auto any_function = std::make_shared<FunctionNode>(function_name);
            any_function->resolveAsAggregateFunction(std::move(aggregate_function));
```

ClickHouse

- Access
- ☑ AggregateFunctions
- ☑ Analyzer
- Backups
- ☑ Bridge
- ☑ BridgeHelper
- Client
- Columns
- Common
- Compression
- Coordination
- Core
- ☑ Daemon
- Databases
- DataTypes
- Dictionaries
- Disks
- Formats
- Functions
- ☑ Interpreters
- IO
- Loggers
- ☑ Parsers
- Planner
- Processors
- ☑ QueryPipeline
- ☑ Server
- Storages
- TableFunctions

“Холодный” поиск

5731 | grep -i “factory” | wc -l

Шаблоны поиска

singleton
factory
builder
prototype
chain
responsibility
command
interpreter
iterator
mediator

memento
observer
state
strategy
visitor
adapter
bridge
composit
decorator
flyweight

proxy
facade
kit
wrapper
handle
body
surrogate
action
transaction
cursor

token
dependen
publish
subscribe
policy

35

Особенности

- ☑ abstract factory
- ☑ template method
- ☑ virtual constructor ⟵ factory method
- ☑ handle/body ⟵ bridge
- ☑ action, transaction ⟵ command
- ☑ cursor ⟵ iterator
- ☑ token ⟵ memento
- ☑ policy ⟵ strategy
- ☑ prototype
- ☑ iterator

Поиск по словам

```
/src:~$ find . -type f | grep ".c$\|.cpp$\|.h$\|.hpp$" | xargs grep -o -i "factory" | wc -l
```

```
    };
    // src/Common/DateLUTImpl.cpp
    std::unique_ptr<cctz::ZoneInfoSource> custom_factory(
        const std::string & name,
        const std::function<std::unique_ptr<cctz::ZoneInfoSource>(const std:
    {
        std::string_view tz_file = getTimeZone(name.data());

        if (!tz_file.empty())
            return std::make_unique<Source>(tz_file.data(), tz_file.size());

        return fallback(name);
    }
}

ZoneInfoSourceFactory zone_info_source_factory = custom_factory;
```

Поиск по строкам

```
/src:~$ find . -type f | grep ".c$\|.cpp$\|.h$\|.hpp$" | xargs grep -i "factory" | wc -l
```

```
    };
    // src/Common/DateLUTImpl.cpp
    std::unique_ptr<cctz::ZoneInfoSource> custom_factory(
        const std::string & name,
        const std::function<std::unique_ptr<cctz::ZoneInfoSource>(const std::stri
    {
        std::string_view tz_file = getTimeZone(name.data());

        if (!tz_file.empty())
            return std::make_unique<Source>(tz_file.data(), tz_file.size());

        return fallback(name);
    }
}

ZoneInfoSourceFactory zone_info_source_factory = custom_factory;
```

Поиск по файлам

```
/src:~$ find . -type f | grep ".c$\|.cpp$\|.h$\|.hpp$" | xargs grep -l -i "factory" | wc -l
```

```
    };
    // src/Common/DateLUTImpl.cpp
    std::unique_ptr<cctz::ZoneInfoSource> custom_factory(
        const std::string & name,
        const std::function<std::unique_ptr<cctz::ZoneInfoSource>(const std:
    {
        std::string_view tz_file = getTimeZone(name.data());

        if (!tz_file.empty())
            return std::make_unique<Source>(tz_file.data(), tz_file.size());

        return fallback(name);
    }
}

ZoneInfoSourceFactory zone_info_source_factory = custom_factory;
```

Поиск по словам

0	facade	25	composit	153	adapter	1536	policy
0	flyweight	26	prototype	158	subscribe	1568	dependen
0	mediator	28	observer	239	body	2185	transaction
0	memento	33	singleton	387	strategy	2225	handle
4	surrogate	39	responsibility	393	bridge	2356	iterator
		65	publish	639	cursor	2742	visitor
		76	kit	672	wrapper	2842	builder
		79	dependents	709	proxy	3680	token
		84	decorator	976	chain	4352	command
						4503	interpreter
						7531	factory
						8404	action
						9589	state

58219

Поиск по строкам

0	facade
0	flyweight
0	mediator
0	memento
4	surrogate
22	composit
23	observer
26	prototype
31	singleton
39	responsibility
54	publish
62	kit
79	decorator
120	adapter
139	subscribe
231	body
285	strategy
309	bridge
466	cursor
507	proxy
578	wrapper
768	chain
1174	policy
1195	dependen
1805	handle
1811	iterator
1830	transaction
2033	visitor
2049	builder
2397	token
3066	command
3936	interpreter
6292	factory
6355	action
7106	state

44794

Поиск по файлам

0	facade	13	composit	104	chain	1380	factory
0	flyweight	15	singleton	147	wrapper	1382	interpreter
0	mediator	16	publish	161	policy		
0	memento	18	responsibility	205	transaction		
1	surrogate	21	kit	217	token		
3	prototype	31	bridge	221	dependen		
8	observer	34	decorator	222	command		
		35	adapter	307	builder		
		37	subscribe	344	handle		
		45	cursor	366	iterator		
		58	strategy	366	visitor		
		74	proxy	615	action		
		76	body	899	state		

7421

Что нашли?

LOC	1`138`607	строки	44794
	5711	файлы	7421

surrogate

prototype

surrogate

```
./IO/ReadHelpers.cpp:
    /// Surrogate pair.

./IO/ReadHelpers.cpp:
    return error("… second part of surrogate pair", …);

./IO/ReadHelpers.cpp:
    return error("… \\u of second part of surrogate pair",

./IO/ReadHelpers.cpp:
    return error("Incorrect surrogate pair of unicode escape sequences …);
```

prototype

```
./Formats/CapnProtoSchema.cpp:
    DataTypePtr getDataTypeFromCapnProtoType(const capnp::Type & capnp_type, bool

./Formats/CapnProtoSchema.cpp:
    auto nested_type = getDataTypeFromCapnProtoType(list_schema.getElementType(),
...
./Formats/StructureToCapnProtoSchema.cpp:
    String prepareAndGetCapnProtoTypeName(WriteBuffer & buf, const DataTypePtr &

./Formats/StructureToCapnProtoSchema.cpp:
    auto field_type_name = prepareAndGetCapnProtoTypeName(buf, data_type, column_
...
./Common/tests/gtest_wide_integer.cpp:
    /// (a prototype of a function that we may need)
```

Итоги

- Проследили историю развития паттернов проектирования, их роль в архитектуре софта
- Немного поговорили про СУБД и задачи, которые такие системы решают
- Проанализировали устройство ClickHouse
- Рассмотрели ряд паттернов проектирования в СУБД
- Поискали паттерны проектирования “холодным” поиском

Маркировать паттерны в коде

Спасибо за внимание

Косенко Дмитрий

@DmitriiKosenko

DmitriiKosenko

dmitrii.person@mail.ru