Семинар по обработке потоковых данных при помощи IBM InfoSphere Streams

Андрей Выходцев (andrey.vykhodtsev@ru.ibm.com)
Консультант по направлениям Netezza и BigData
Андрей Орлов (andrey.orlov@ru.ibm.com),
Консультант по IBM InfoSphere Streams

Платформа IBM Big Data

InfoSphere BigInsights
Hadoop-based low latency analytics for variety and volume

Hadoop

Information Integration

InfoSphere Information Server
High volume data integration and transformation

Stream Computing

InfoSphere Streams
Low Latency Analytics for streaming data

MPP Data Warehouse

IBM InfoSphere Warehouse
Large volume structured data analytics

IBM Netezza High Capacity Appliance
Queryable Archive Structured Data

IBM Netezza 1000
BI+Ad Hoc Analytics on Structured Data

IBM Smart Analytics System
Operational Analytics on Structured Data

IBM Informix Timeseries
Time-structured analytics

IBM InfoSphere Streams v2.0

Это платформа для аналитики в режиме реального времени на BIG data

- Объем
  - Гигабайт в секунду
  - Петабайт в день
- Многообразие
  - Любой тип данных
  - Любой вид аналитики
- Скорость
  - Задержки в микросекундах
- Гибкость
  - Динамическое реагирование
  - Быстрая разработка приложений

Зачем использовать InfoSphere Streams?

- Приложения, которые требуют обработки, фильтрации и анализа потоковых данных на лету
  - Сенсоры: экология, промышленность, видео наблюдение, GPS, …
  - “Выхлоп данных”: сетевые/системные/веб-серверные лог-файлы
  - Высокоскоростные транзакции данных: финансовые транзакции, детальные записи информации звонков

- Критерии
  - Сообщения обрабатываются изолированно или данные фиксируются определенным окном
  - Источники - нетрадиционные данные (изображения, текст …)
  - Скорость и объем передачи данных требуют несколько узлов обработки
  - Анализ и реагирование с миллисекундной задержкой
  - Скорость и объем слишком велик для временного хранения данных

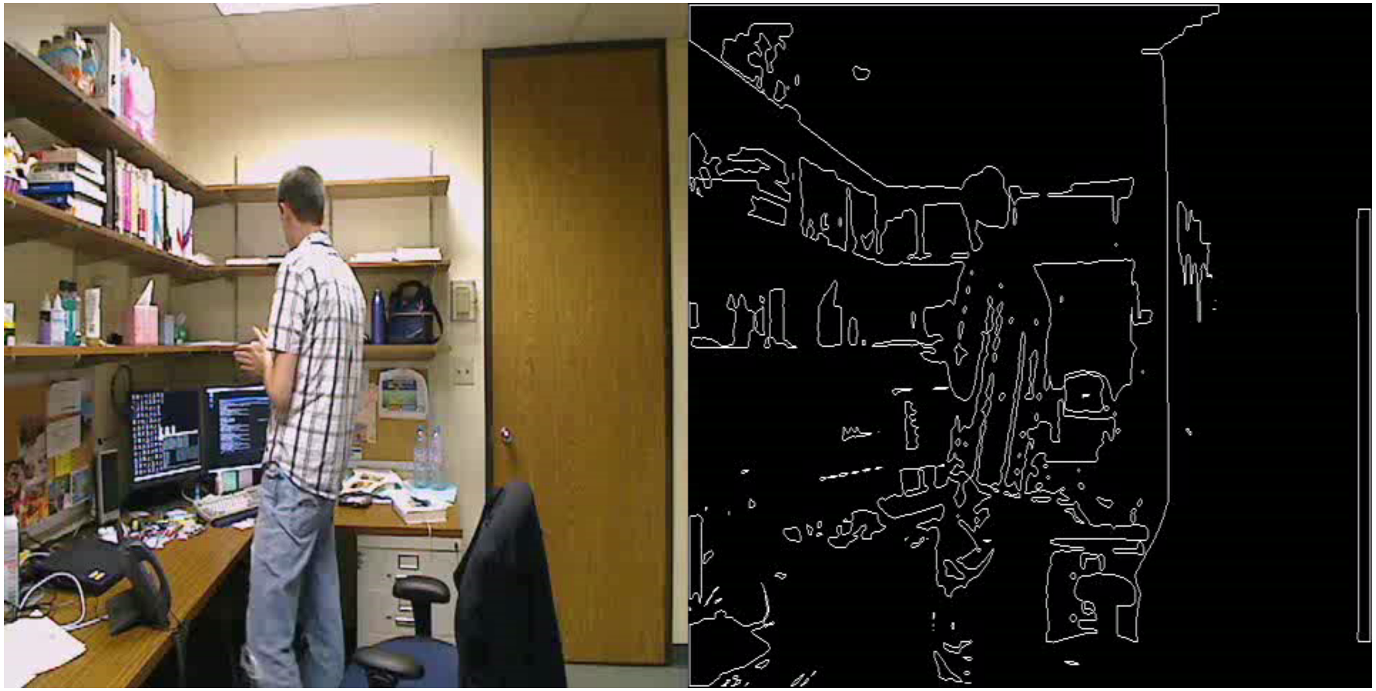
Правоохранительная деятельность и безопасность

- Видеонаблюдение, телефонные записи

- Миллионы сообщений в секунду с низкой плотностью критически важных сообщений

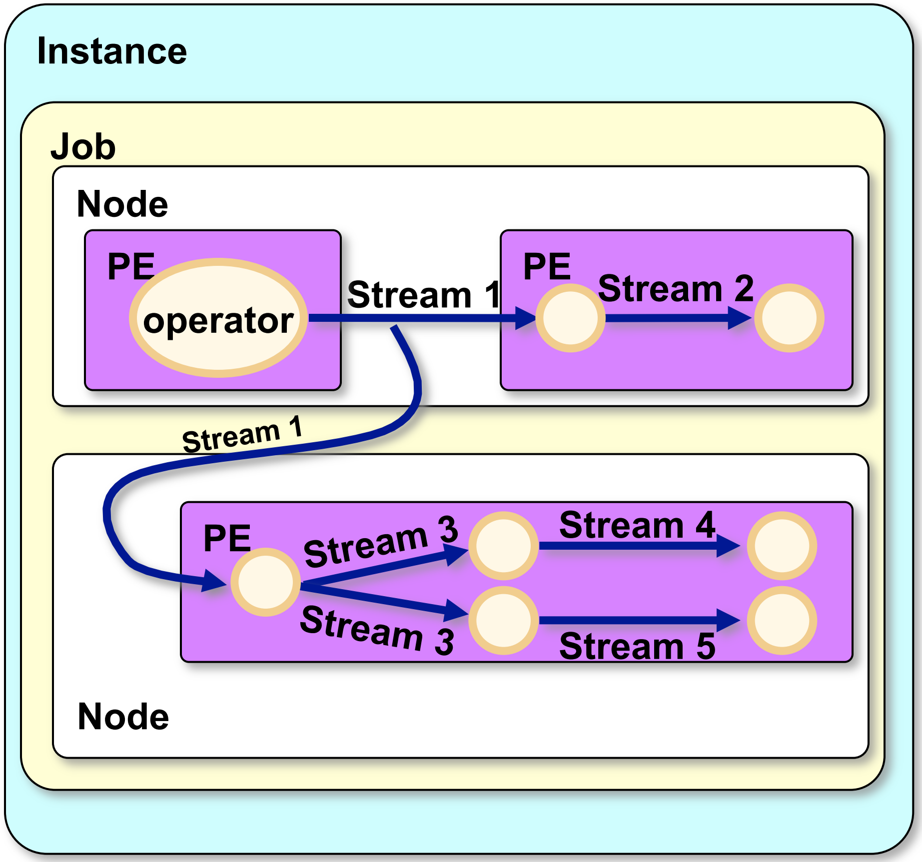
- Определить закономерность и взаимосвязи огромных источников информации

Выделение контуров на видео



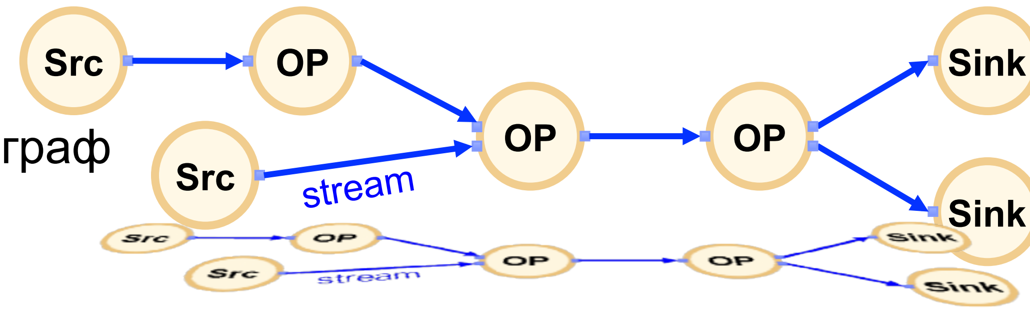
Объекты InfoSphere Streams: Исполнение

- Instance
  - Исполняемый экземпляр InfoSphere Streams на одном или нескольких хостах
  - Набор компонентов и сервисов
- Processing Element (PE)
  - Основная единица исполнения, запускаемая в инстансе Streams
- Job
  - Развернутое приложение Streams исполняемое в инстансе
  - Состоит из одного или нескольких PEs

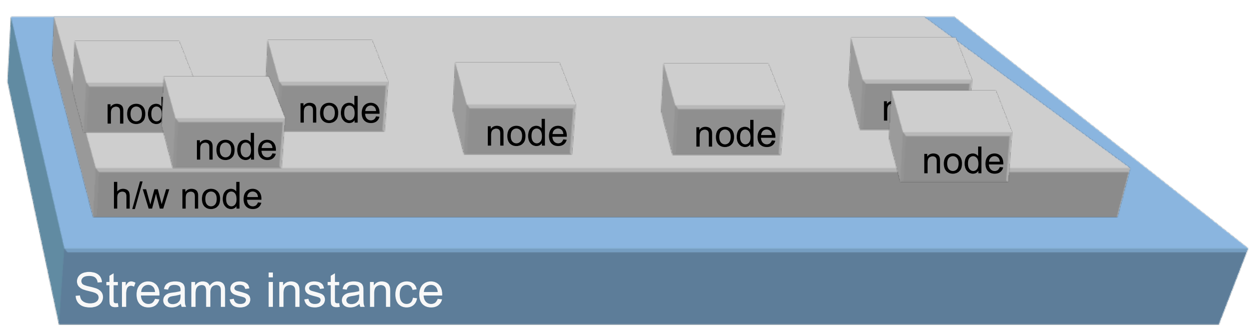


От проектирования к развертыванию

- Граф приложения Streams:
  - Направленный (циклический) граф
  - Набор операторов
  - Подключение к потокам

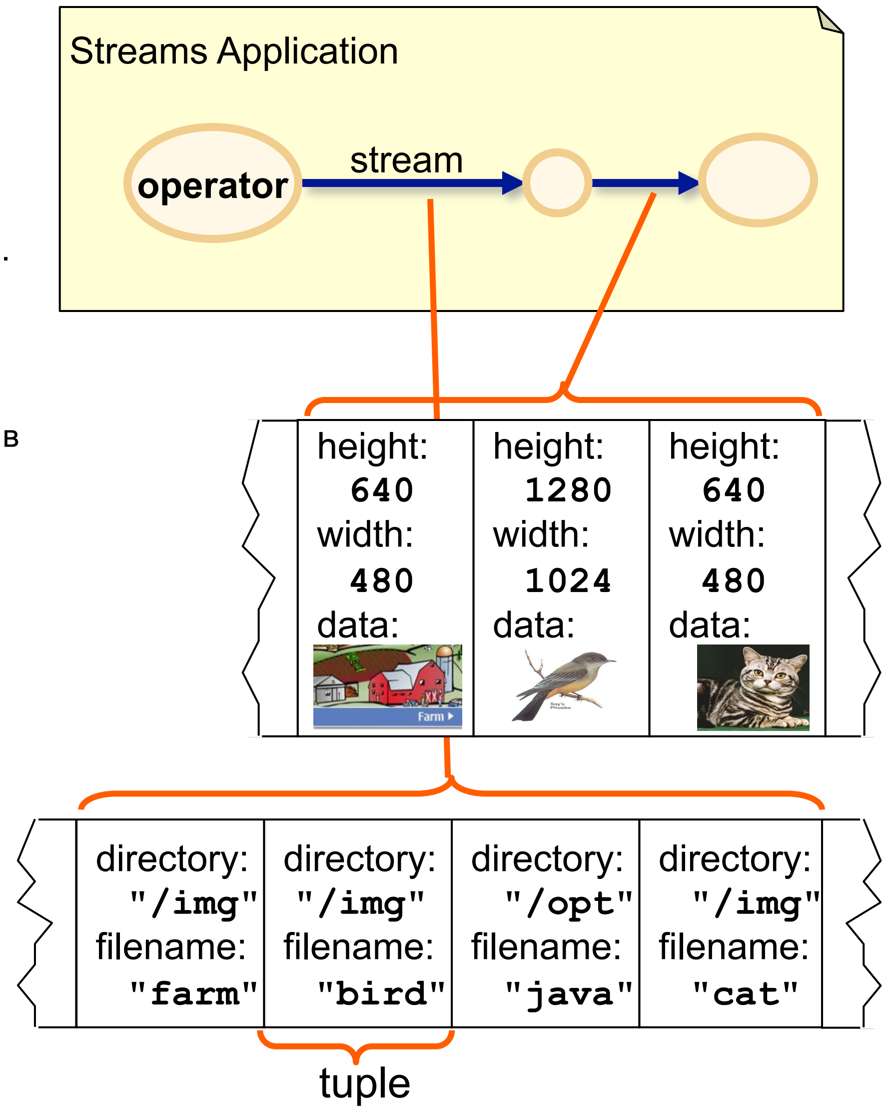


- Каждое законченное приложение может быть переложено в job
- Jobs развертываются в instance
- Instance включает в себя одни node на уровне железа
- Или множество процессных node



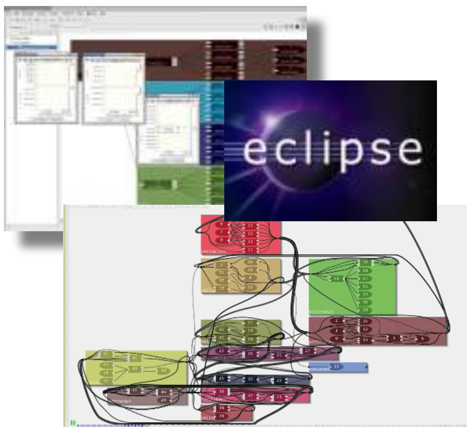
Объекты InfoSphere Streams: Разработка

- Оператор
  - Основная «строительная» единица Streams Processing Language.
- Поток
  - Бесконечная последовательность *кортежей*. Поочередности или сразу множеством с помощью окна.
- Кортеж
  - Структурированный список атрибутов и их типы.
- Тип потока
  - Спецификация имени и типа данных каждого атрибута в кортеже.
- Окно
  - Конечная последовательность кортежей в потоке
  - Базируется на *количестве, времени, значении* или *флага*.



IBM InfoSphere Streams v2.0

Development Environment



- **Eclipse IDE**
- **Streams Live Graph**
- **Streams Debugger**

Runtime Environment

- RHEL v5.3 и выше
- Многоядерные процессоры x86
- Поддержка InfiniBand
- Кластерное выполнение

Toolkits, Adapters & Samples

- Standard Toolkit
- Internet Toolkit
- Database Toolkit
- Financial Toolkit
- Mining Toolkit
- User-defined toolkits
- Over 50 samples

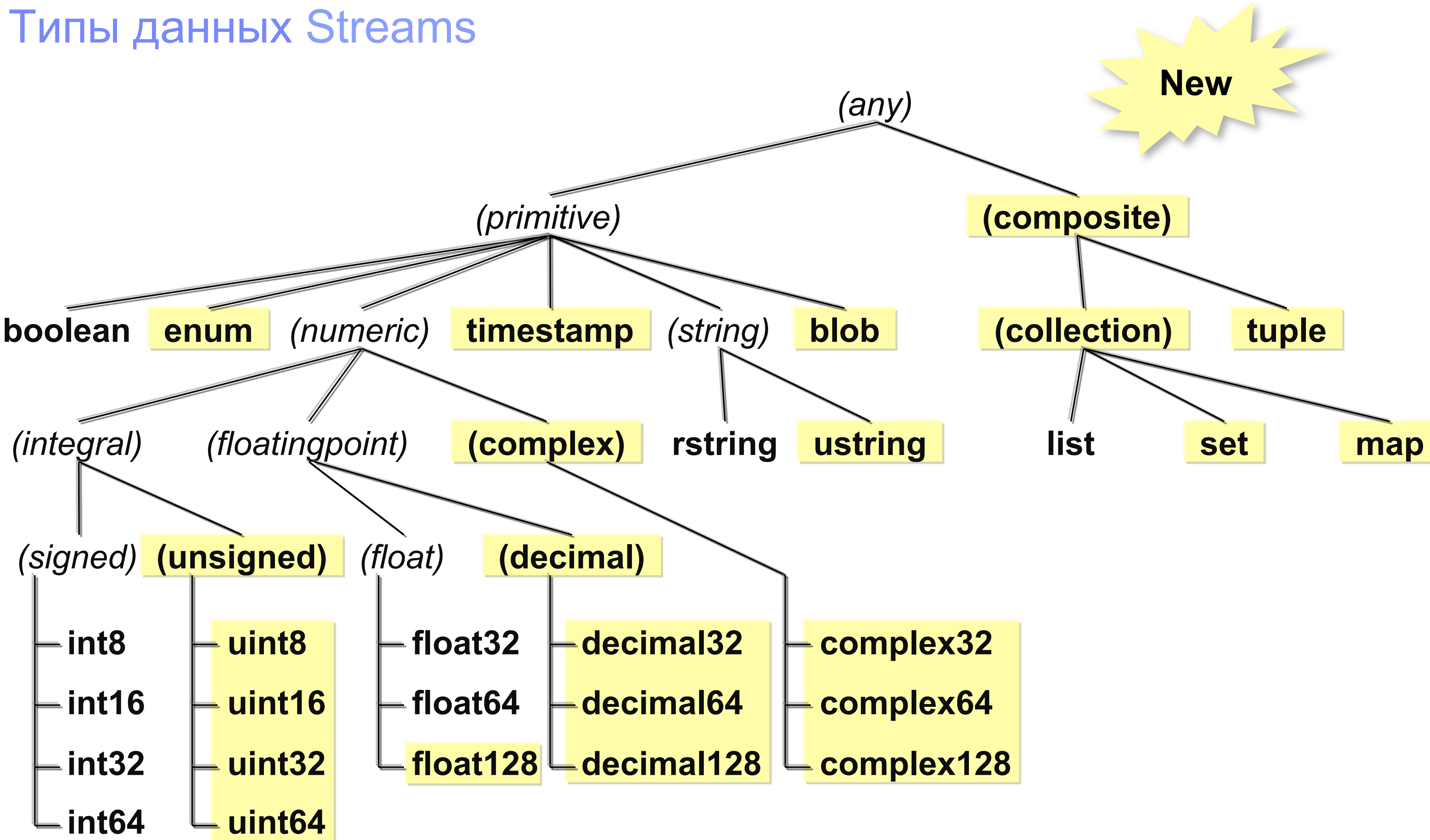
Что такое Streams Processing Language?

- Разработан для потоковых вычислений
  - Определяет граф потока
  - Большое количество типов для создание кортежей данных

- Поддержка процедурных языков
  - Полная функциональность C++ и Java
  - Собственная логика в операторах

- Расширение
  - Пользовательские типы данных
  - Пользовательские функции написанные на SPL или других языках (C++ or Java)
  - Пользовательские операторы написанные на SPL
  - Пользовательские операторы написанные на C++ илиJava

Типы данных Streams

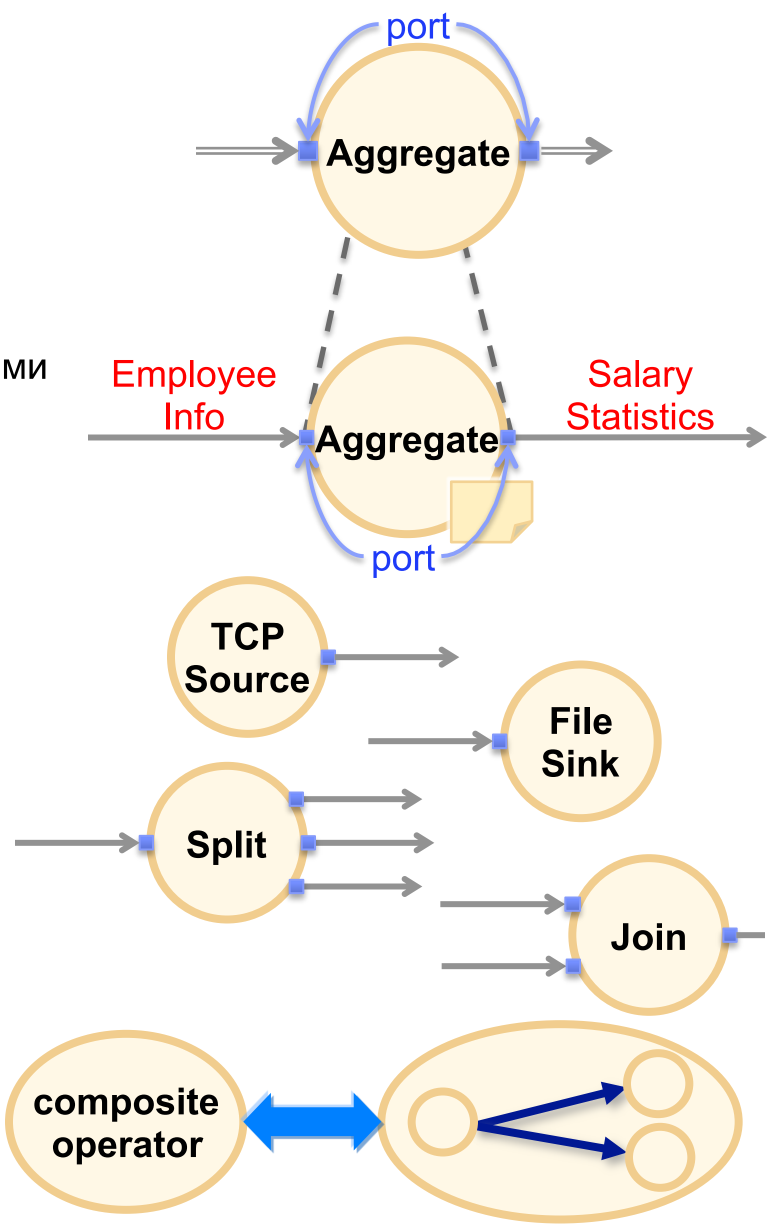


Collections

- list: array with bounds-checking [0, 17, age-1, 99]
  - Свободный доступ: любой элемент в любое время
  - Упорядочен: первый элемент с 0 - someList[0]
- set: unordered collection {"cats", "yeasts", "plankton"}
  - Нет повторяющихся элементов
- map: key-to-value mappings {"Mon":0, "Sat":99, "Sun":-1}
  - Не имеет порядка
- Конструктор типа для определения типа коллекции
  - list<*type*>, set<*type*> list<uint16>, set<rstring>
  - map<*key-type*,*value-type*> map<rstring[3],int8>
- Вложенность на любое число уровней
  - map<int32, list<tuple<ustring name, int64 value>>>
  - {1 : [{"Joe",117885}, {"Fred",923416}], 2 : [{"Max",117885}], -1 : []}

Некоторые определения термов SPL

- Оператор представляет из себя манипулятор
  - **Кортежи** с одного или многих входных **потоков**
  - Выход кортежей на один или несколько потоков
- Оператор подключается к потоку через порт
  - Оператор определяется входными и выходными портами
- Вызов оператора
  - Со назначением входных и выходных потоков
  - С собственными параметрами, логикой, …
- Входные и выходные порты
  - Ни одного входного порта: Источники - TCPSource,…
  - Ни одного выходного порта: Приемники – FileSink,…
  - Множество выходные портов: Split,…
  - Множество входных портов: Join,…
- Составные операторы – набор операторов
  - Инкапсуляция подграфа
    - Примитивные операторы
    - Составные операторы (вложенные)
  - Как макросы в процедурных языках

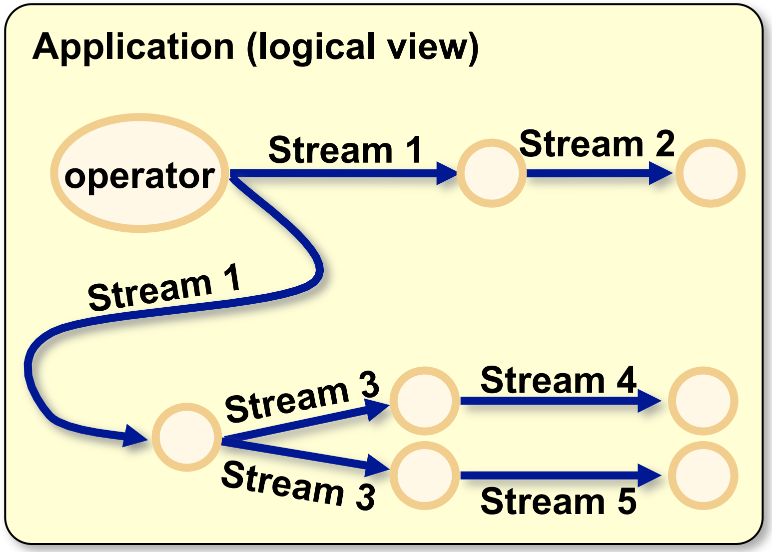


Составной оператор

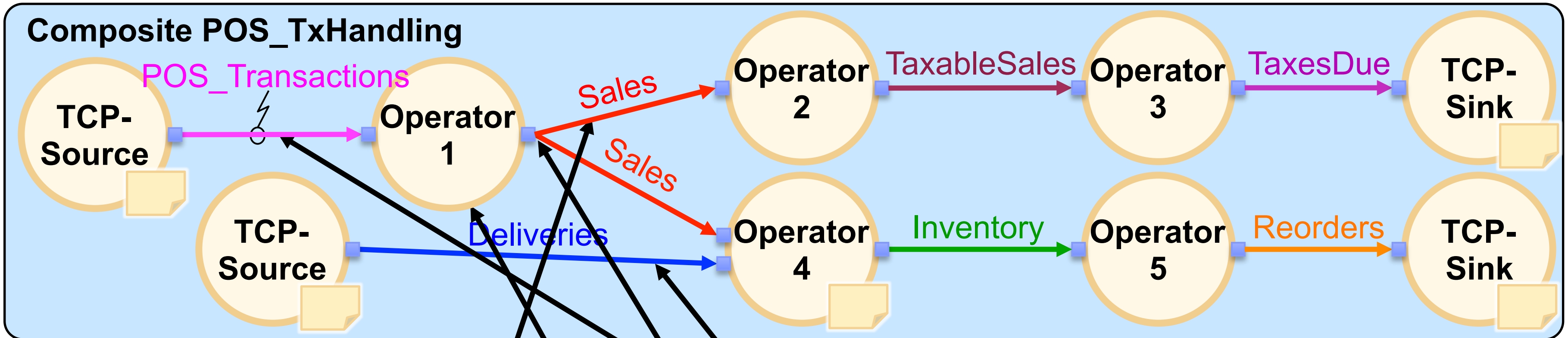
- Каждый граф является составным
  - Составной граф – один или несколько операторов
  - Может иметь входные и выходные порты
  - Только исходный код
    - Ничего общего с процессными элементами (PEs)

```
composite Main {
  graph
    stream … {
    }
    stream … {
    }
  . . .
}
```

- Приложение – это main composite
  - Нет входных и выходных портов
  - Потоки могут быть экспортированы и импортированы из других приложений, работающих в одном инстансе



Composing a Flow Graph with Stream Definitions




```
composite POS_TxHandling
{
    graph
        stream<…> POS_Transactions = TCPSource() {…}
        stream<…> Sales = Operator1(POS_Transactions) {…}
        stream<…> TaxableSales = Operator2(Sales) {…}
        stream<…> TaxesDue = Operator3(TaxableSales) {…}
        () as Sink1 = TCPSink(TaxesDue) {…}
        stream<…> Deliveries = TCPSource() {…}
        stream<…> Inventory = Operator4(Sales; Deliveries) {…}
        stream<…> Reorders = Operator5(Inventory) {…}
        () as Sink2 = TCPSink(Reorders) {…}
}
```


16