Flink Streaming SQL 2018

公司：**dA**

职位：**Software Engineer**

演讲者：**Piotr Nowojsk**

FLINK FORWARD

Flink Streaming SQL 2018

FLINK FORWARD

目录
Agenda

为何选择 SQL？
Why SQL?
选择 Streaming SQL 将要面对的挑战
Challenges in Streaming SQL
在 Streaming SQL 中连接表的不同方式
Various ways to join tables in Streaming SQL
模式识别
Pattern recognition
其他近期成果
Other recent improvements.

FLINK FORWARD

为何选择 SQL

Why SQL?

众所周知的接口

Well known interface

无需编程——易于上手

No programming is required - easier to learn

申诉式语言表达你的商业逻辑

Declarative way to express your business logic

内建优化

Out-of-the box optimization

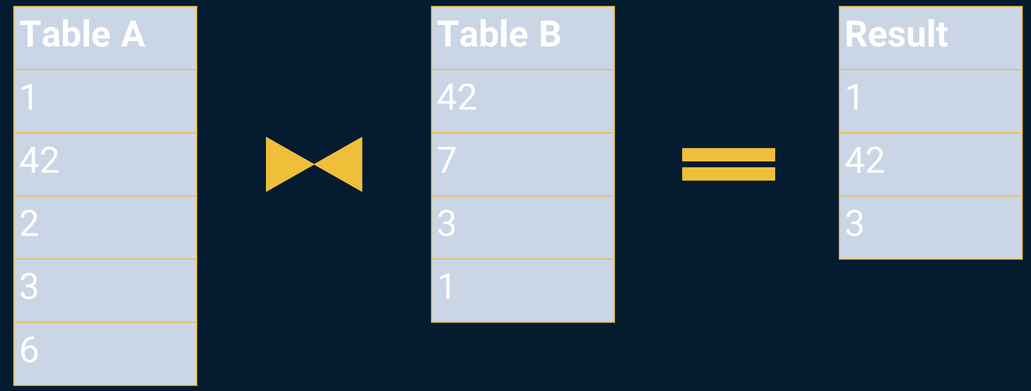
选择 Streaming SQL 将要面对的挑战
Challenges in Streaming SQL

批次处理实例
Batch example

```
SELECT
   a.id
FROM
   A a, B b
WHERE
   a.id = b.id
```

FLINK FORWARD

SELECT a.id FROM A a, B b WHERE a.id = b.id



FLINK FORWARD

归并连接算法

Sort-Merge Join

Table A
1
42
2
3
6

Table B
42
7
3
1

FLINK FORWARD

第一步 — 分类

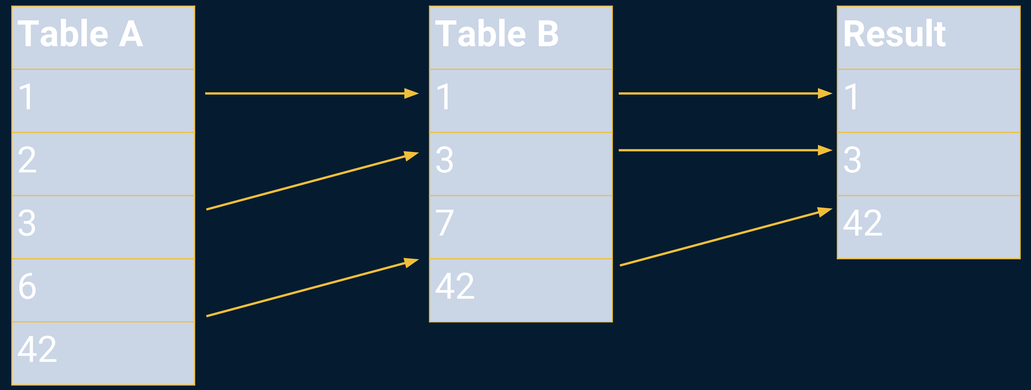
First step - Sort

Table A
1
2
3
6
42

Table B
1
3
7
42

第二步 合并及连接

Second step Merge and Join



多表连接算法

Hash Join

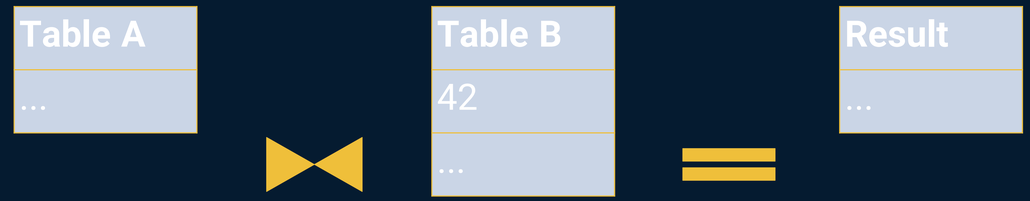
Table A
1
42
2
3
6

Table B
42
7
3
1

Result
1
42
3

连接连续查询

Join in continuous queries



连接连续查询

Join in continuous queries

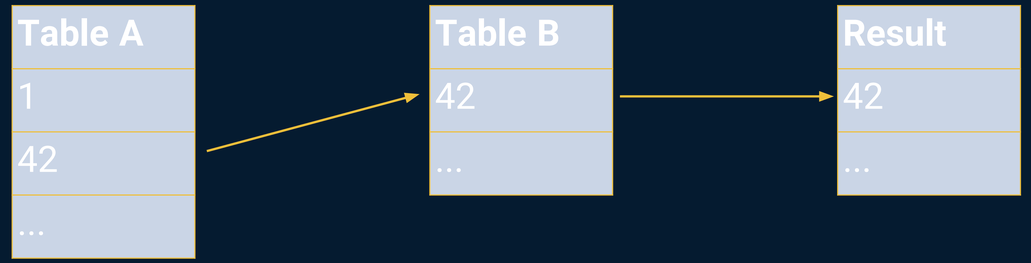
Table A
1
...

Table B
42
...

Result
...

连接连续查询

Join in continuous queries



连接连续查询

Join in continuous queries

Table A
1
42
...

Table B
42
7
...

Result
42
...

连接连续查询

Join in continuous queries

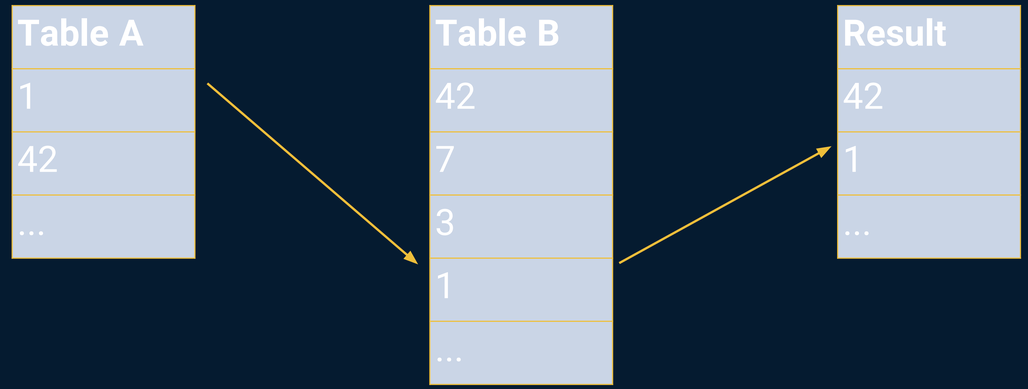
Table A
1
42
...

Table B
42
7
3
...

Result
42
...

连接连续查询

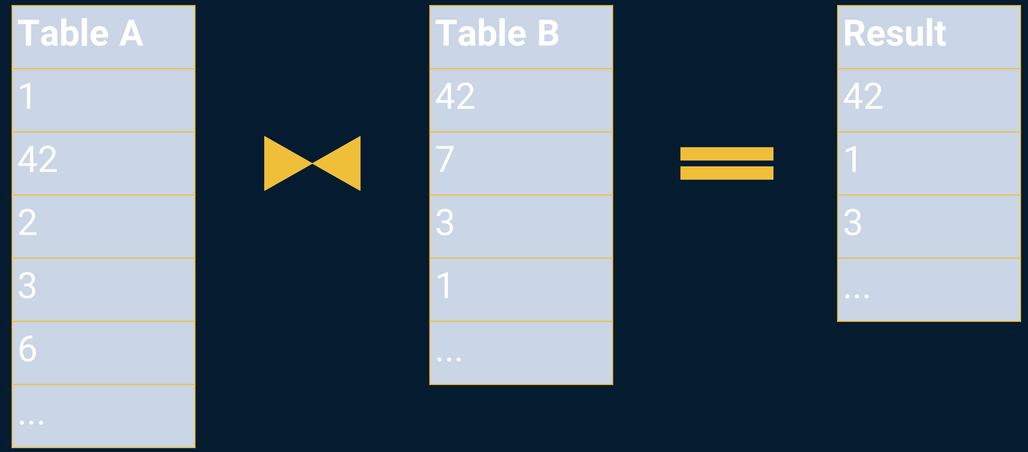
Join in continuous queries



FLINK FORWARD

连接连续查询

Join in continuous queries



时间窗口连接
Time-windowed Join

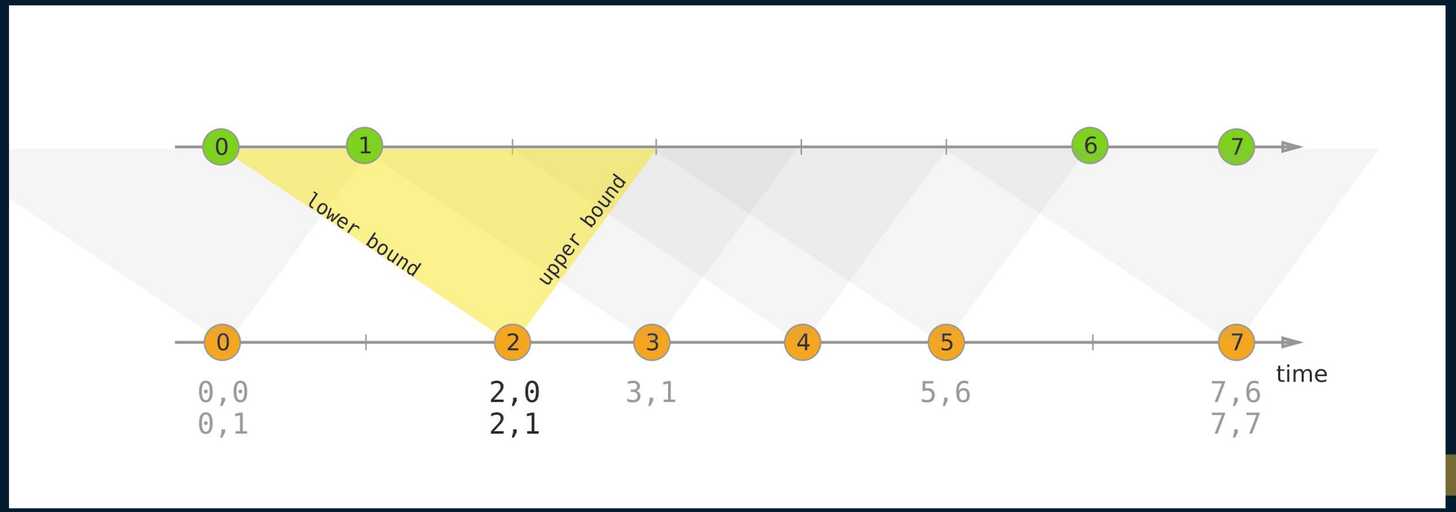
FLINK FORWARD

水位线
Watermarks

时间窗口连接
Time-windowed Join

```
SELECT
  *
FROM
  Orders o, Shipments s
WHERE
  o.id = s.orderId AND
  s.shiptime BETWEEN o.ordertime AND o.ordertime + INTERVAL '4' HOUR
```

时间窗口连接
Time-windowed Join



历史表
Temporal Tables

FLINK FORWARD

SELECT * FROM RatesHistory;

时间 **time**	货币 **currency**	汇率 **rate**
09:00	USD	102
09:00	Euro	114
09:00	Yen	1
10:45	Euro	116
11:15	Euro	119
11:49	USD	99
...	...	...

FLINK FORWARD

RatesHistory

时间 **time**	货币 **currency**	汇率 **rate**
09:00	USD	102
09:00	Euro	114
09:00	Yen	1
10:45	Euro	116
11:15	Euro	119
11:49	USD	99
...	...	...

```
TemporalTableFunction rates =
   ratesHistory
      .createTemporalTableFunction(
         "time",
         "currency");


tableEnv.registerFunction("Rates", rates);
```

FLINK FORWARD

RatesHistory

时间 **time**	货币 **currency**	汇率 **rate**
09:00	USD	**102**
09:00	Euro	**114**
09:00	Yen	**1**
10:45	Euro	116
11:15	Euro	119
11:49	USD	99
...	...	...

SELECT * FROM Rates('10:15');

时间 **time**	**货币** **currency**	**汇率** **rate**
09:00	USD	**102**
09:00	Euro	**114**
09:00	Yen	**1**

FLINK FORWARD

RatesHistory

时间 time	货币 currency	汇率 rate
09:00	USD	102
09:00	Euro	114
09:00	Yen	1
10:45	Euro	116
11:15	Euro	119
11:49	USD	99
...	...	...

SELECT * FROM Rates('11:50');

时间 time	货币 currency	汇率 rate
11:49	USD	99
11:15	Euro	119
09:00	Yen	1

FLINK FORWARD

RatesHistory

时间 **time**	货币 **currency**	汇率 **rate**
09:00	USD	102
09:00	Euro	114
09:00	Yen	1
10:45	Euro	116
11:15	Euro	119
11:49	USD	99
...	...	...

Orders

时间 **time**	货币 **currency**	量 **amount**
10:15	Euro	2
10:00	Yen	50
11:35	Euro	2
...	...	...

历史表连接
Temporal Table Join

```
SELECT
  o.amount * r.rate
FROM
  Orders o,
  LATERAL TABLE (Rates(o.time)) r
WHERE
  o.currency = r.currency
```

```
SELECT o.amount * r.rate
FROM
   Orders o,
   LATERAL TABLE (Rates(o.time)) r
WHERE
   o.currency = r.currency
```

RatesHistory

时间 **time**	货币 **currency**	汇率 **rate**
09:00	USD	102
09:00	**Euro**	**114**
09:00	Yen	1
10:45	Euro	116
11:15	Euro	119
11:49	USD	99
...	...	...

Orders

时间 **time**	货币 **currency**	量 **amount**
10:15	**Euro**	**2**
...	...	...

Result

汇率 * 量 **rate * amount**
228
...

FLINK FORWARD

```
SELECT o.amount * r.rate
FROM
   Orders o,
   LATERAL TABLE (Rates(o.time)) r
WHERE
   o.currency = r.currency
```

RatesHistory

时间 **time**	货币 **currency**	汇率 **rate**
09:00	USD	102
09:00	Euro	114
09:00	Yen	1
10:45	Euro	116
11:15	**Euro**	**119**
11:49	USD	99
...	...	...

Orders

时间 **time**	货币 **currency**	量 **amount**
10:15	Euro	2
10:00	Yen	50
11:35	**Euro**	**2**
...	...	...

Result

汇率 * 量 **rate * amount**
228
50
238
...

常见连接
Regular joins

语法
Syntax

SELECT * FROM A a, B b WHERE a.id = b.id

范围
Scope

两个表格对彼此完全可见
All rows from both tables visible to each other

内存
Memory

所有记录必须无限期保存
All records must be persisted indefinitely

FLINK FORWARD

时间窗口连接
Time-windowed Joins

语法 Syntax	SELECT * FROM A a, B b WHERE a.id = b.id AND a.time BETWEEN b.time AND b.time + 1 `DAY`
范围 Scope	所有行在被定义的时间窗口中可见 Rows visible within a defined time window
内存 Memory	所有记录在现在和窗口长度之间(加上水位线延迟) All records between now and window length (plus watermark delay)

FLINK FORWARD

历史表连接
Temporal Table Joins

语法
Syntax

```
SELECT * FROM A a, LATERAL TABLE (B(a.time))
WHERE a.id = b.id
```

范围
Scope

只对给定的B的最新版本可见 a.time
Visible is only the latest version of B for given a.time

内存
Memory

表格B：所有B的版本在现在和水位线延迟之间
Table B: all versions of B between now and watermark delay
表格A：所有记录在现在和水位线延迟之间
Table A: all records between now and watermark delay

模式识别

Pattern recognition

模式识别
Pattern recognition

示例 :

Examples:

- (S M{2,} E)
- (A B+ C* D)
- (START_ROW PRICE_DOWN+ PRICE_UP)

```
SELECT * FROM Ticker
MATCH_RECOGNIZE (
    PARTITION BY symbol
    ORDER BY rowtime
    MEASURES
        B.price AS endPrice,
        COUNT(A.price) AS count
    ONE ROW PER MATCH
    AFTER MATCH SKIP TO FIRST B
    PATTERN (A+ B)
    DEFINE
        A AS AVG(A.price) < 15)
```

```
SELECT * FROM Ticker
MATCH_RECOGNIZE (
    PARTITION BY symbol
    ORDER BY rowtime
    MEASURES
        B.price AS endPrice,
        COUNT(A.price) AS count
    ONE ROW PER MATCH
    AFTER MATCH SKIP TO FIRST B
    PATTERN (A+ B)
    DEFINE
        A AS AVG(A.price) < 15)
```

```
SELECT * FROM Ticker
MATCH_RECOGNIZE (
    PARTITION BY symbol
    ORDER BY rowtime
    MEASURES
        B.price AS endPrice,
        COUNT(A.price) AS count
    ONE ROW PER MATCH
    AFTER MATCH SKIP TO FIRST B
    PATTERN (A+ B)
    DEFINE
        A AS AVG(A.price) < 15)
```

```
SELECT * FROM Ticker
MATCH_RECOGNIZE (
    PARTITION BY symbol
    ORDER BY rowtime
    MEASURES
        B.price AS endPrice,
        COUNT(A.price) AS count
    ONE ROW PER MATCH
    AFTER MATCH SKIP TO FIRST B
    PATTERN (A+ B)
    DEFINE
        A AS AVG(A.price) < 15)
```

```
SELECT * FROM Ticker
MATCH_RECOGNIZE (
    PARTITION BY symbol
    ORDER BY rowtime
    MEASURES
        B.price AS endPrice,
        COUNT(A.price) AS count
    ONE ROW PER MATCH
    AFTER MATCH SKIP TO FIRST B
    PATTERN (A+ B)
    DEFINE
        A AS AVG(A.price) < 15)
```

FLINK FORWARD

```
SELECT * FROM Ticker
MATCH_RECOGNIZE (
    PARTITION BY symbol
    ORDER BY rowtime
    MEASURES
        B.price AS endPrice,
        COUNT(A.price) AS count
    ONE ROW PER MATCH
    AFTER MATCH SKIP TO FIRST B
    PATTERN (A+ B)
    DEFINE
        A AS AVG(A.price) < 15)
```

FLINK FORWARD

模式识别
Pattern recognition

MATCH_RECOGNIZE 从句与 GROUP BY 有一些相似之处
MATCH_RECOGNIZE clause has some similarities with GROUP BY
是 SQL 2016 标准的一部分
It is a part of the SQL 2016 standard

其他成果
Other work

SQL Client

Flink SQL 的命令行接口
Command line interface for Flink SQL
无编程需要
No programming required

进行中的项目
Ongoing work

连接器和格式:
Connectors & formats

SQL Client

外部目录支持:
External catalog support

THANKS