DejaVu: Declarative Pattern Matching over Live and Archived Streams of Events

Nihal Dindar, Banş Güç, Patrick Lau, Aslı Özal, Merve Soner, Nesime Tatbul @ ETH Zurich

**Motivation**
- The need for finding contigous patterns on both live and stored data sequences in Complex Event Processing (CEP) applications such as financial data analysis, supply chain management, and system monitoring.
- Current solution proposals: Pattern matching over live data streams (e.g., SASE, Cayuga) OR Pattern matching over sequences of rows in relational tables (e.g., SQL-TS).

**Goal**
- To design and implement a scalable complex event processing system that can seamlessly perform pattern detection over both live AND historical streams of events, behind a uniform declarative query interface.

**SQL-based Query Language**
```sql
SELECT notify Thief (timestamp, book_tag)
FROM Books MATCH_RECOGNIZE( PARTITION BY TagId)
MEASURES B.Timestamp AS timestamp,
B.TagId AS book_tag
ONE ROW PER MATCH
AFTER MATCH SKIP PAST LAST ROW
INCREMENTAL MATCH
PATTERN (A B)
DEFINE A AS (A.ReaderId = 'Shelf')
B AS (B.ReaderId = 'Exit')

```

**Hybrid Queries: Live Streams vs Archived Streams**
- **Example:**
  ```sql
  SELECT notify Thief (timestamp, book_tag)
  FROM Books MATCH_RECOGNIZE( PARTITION BY TagId)
  MEASURES B.Timestamp AS timestamp,
  B.TagId AS book_tag
  ONE ROW PER MATCH
  AFTER MATCH SKIP PAST LAST ROW
  INCREMENTAL MATCH
  PATTERN (A B)
  DEFINE A AS (A.ReaderId = 'Shelf')
  B AS (B.ReaderId = 'Exit')
  )
  ```

- **Implementation**
  - When a pattern is detected on the Live Stream ending at time t, look for patterns in the Archived Streams which end before time t.

- **Optimization**
  - When a pattern is detected on the Live Stream, we only need to process the Archived Stream starting from the last processed position.

**Live Stream Store**
- An in-memory storage engine that accepts push-based inputs.
- Acts like a tuple queue, providing live events into the query processing engine as they arrive from their sources.
- Used to answer continuous pattern matching queries.
- Can be accessed either in "push mode" or "pull mode".

**Archived Stream Store**
- A persistent storage engine where live events can be fully or selectively materialized for historical access.
- Only allows updates in the form of append and preserves the data order.
- Used to answer one-time and hybrid pattern matching queries.
- Can also support the live stream store in dealing with bursts and failures.

**Push or Pull?**
- The Query Processing Engine can access the live streams in two alternative modes:
  - **Push:** By default, each new input event is pushed directly into the corresponding Input Holders via the Router, in order to feed the Query Processing Engine.
  - **Pull:** The Query Processing Engine asks for new input events whenever it is ready to process them via the Router.
- DejaVu adaptively switches from Push to Pull when:
  ```sql
  \frac{QPE \text{ input consumption rate}}{\text{store input push rate}} < \tau
  ```

**DejaVu Query Processing Engine**
- DejaVu has been built on the MySQL relational database engine; as such, it extends MySQL with a number of key capabilities including:
  - The ability to process continuous queries over streaming data.
  - The ability to process pattern matching queries.
- DejaVu represents each pattern with a Finite State Machine (FSM), which runs as an integral part of the MySQL query plan.
- Each FSM instance can be at multiple active states at a given time, due to the inherent non-determinism and/or overlapping semantic windows. In this case, multiple FSM instances can share input tuples through Input Holders.
- A Router component forwards the relevant tuples to each of the Input Holders in an efficient manner.