Complex event processing with Esper: theoretical background and applications

Malik Khalfallah

Brief bio of the presenter

- CS engineer diploma from the university of Constantine
- PhD from the university of Lyon & Airbus Group research center
- Maitre de Conférences associé at the university of Créteil in Paris
- Industry and research interests: data management, avionics systems

Presentation outline

- Complex event processing: conceptual presentation
  - Processing stream of events
  - Aggregating data to process them windows mechanism
  - Queries to detect patterns of events
- Application of complex event processing with Esper
- Use case: Trivia example
- Limitations of Esper and workarounds from the industry

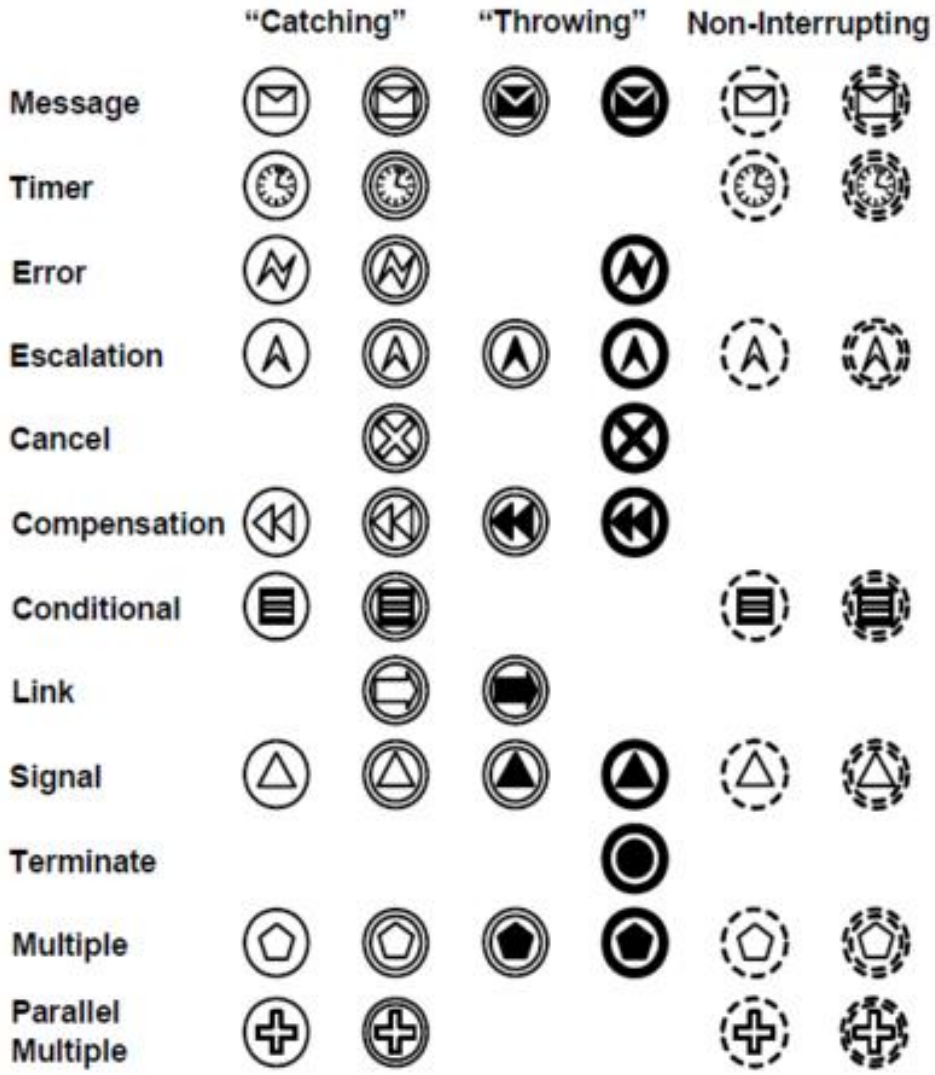
Complex Event Processing: Definition of terms

- In organizations, events are generated and handled by business processes

Complex Event Processing: Definition of terms

- In organizations, events are generated and handled by business processes

- Languages to model business processes, such as BPMN, capture different types of events that could be of interest in business processes



Complex Event Processing: Definition of terms

Event 1: professors appear in suit

Event 2: A student appears in suit

Event 3: laughs

Successful thesis defense

Complex Event Processing: Definition of terms

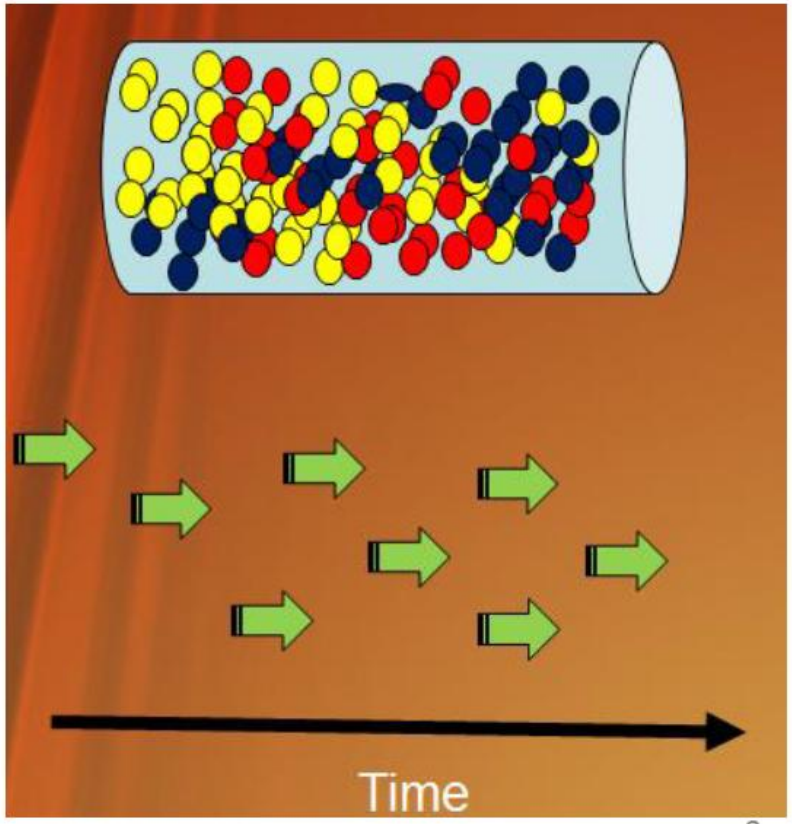
Event Models

<u>Streaming</u>

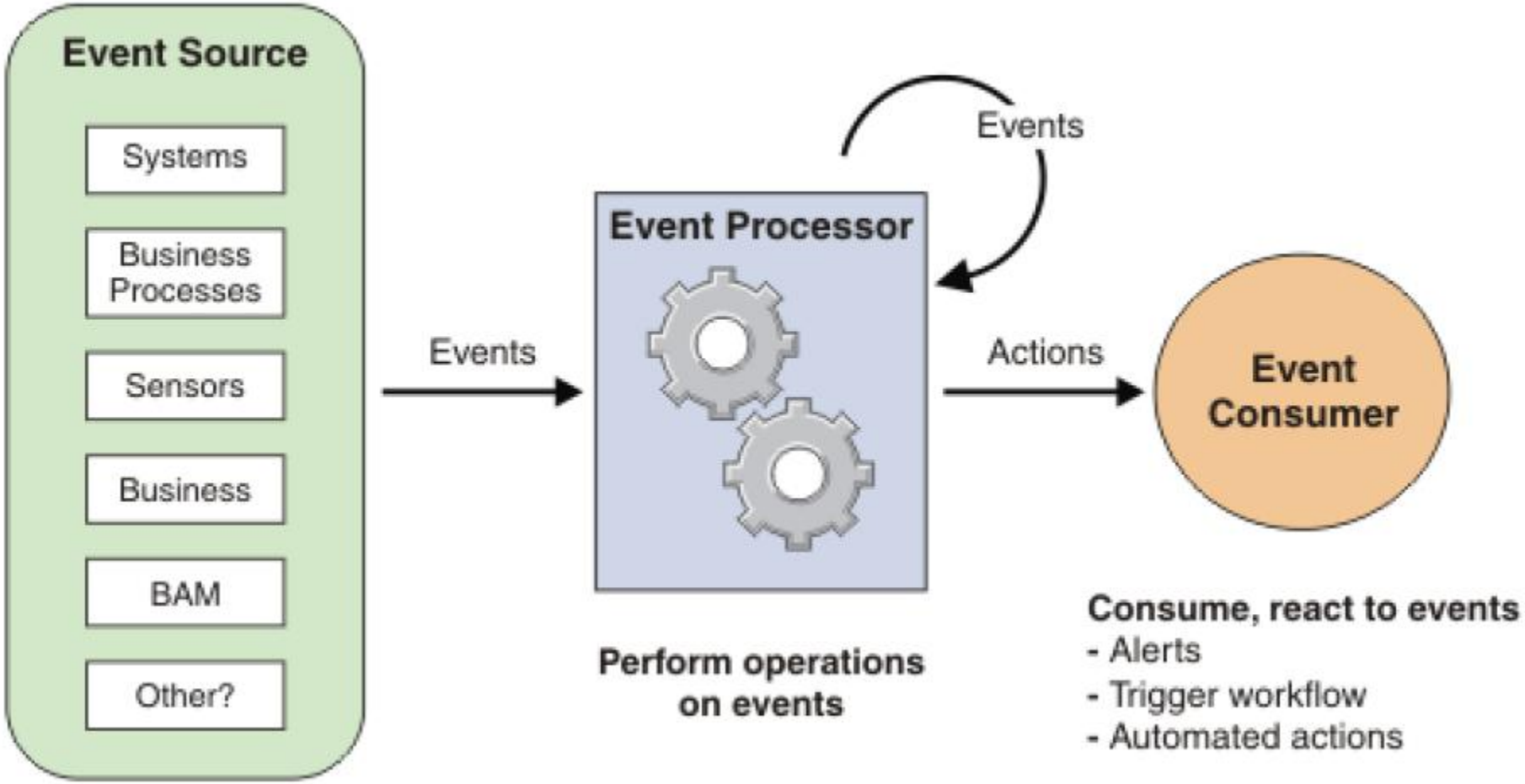
- Large, dense data streams
- Eg. Financial trading information
- 000's of events / second

<u>Non-Streaming</u>

- Business events
- Eg. New Order, BAM



Complex Event Processing: Global Architecture



Complex event processing vs Stream processing

	Complex event processing	Stream processing
Also known as	CEP, event stream analysis and event series analysis	Real-time computation, stream computing
Example providers	**Esper**	**Spark**, **Flink, drooles**
Type	Business Intelligence and decision making	Container technologies e.g. J2EE
Pattern matching and detection, filtering, transformation, aggregation, event hierarchies, detecting	Central to CEP	Not central to stream processing
Transporting events between processes and hosts	Bot central to CEP in general	Central to stream processing
Continuous Queries (Statements)	Express stream analysis in event processing language (EPL)no need to restart the server or container	Code your own operators
Target	Analysis	Extract-Transform-Load (ETL)

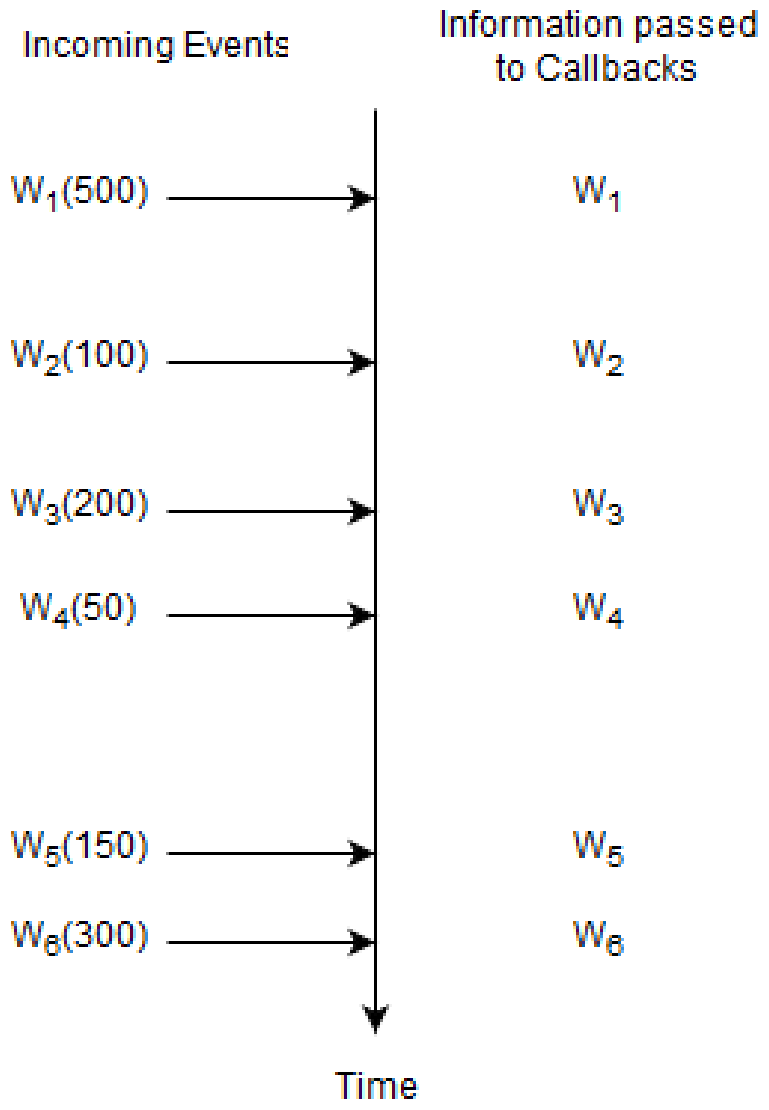
Event processing language (EPL)

- The Esper event processing language (EPL) converges event stream processing (filtering, joins, aggregation) and complex event processing (causality) into one single language
- The core language is SQL conforming ensuring rapid learning
- The language, of course, includes event windows and causality patterns as first citizens

EPL: Operators

- Operators on events let a business situation be inferred or identified
- This might involve combining multiple methods to identify a specific pattern:
  - Correlation
  - Filtering
  - Causality
- Event operators can function on a single stream as well as across streams

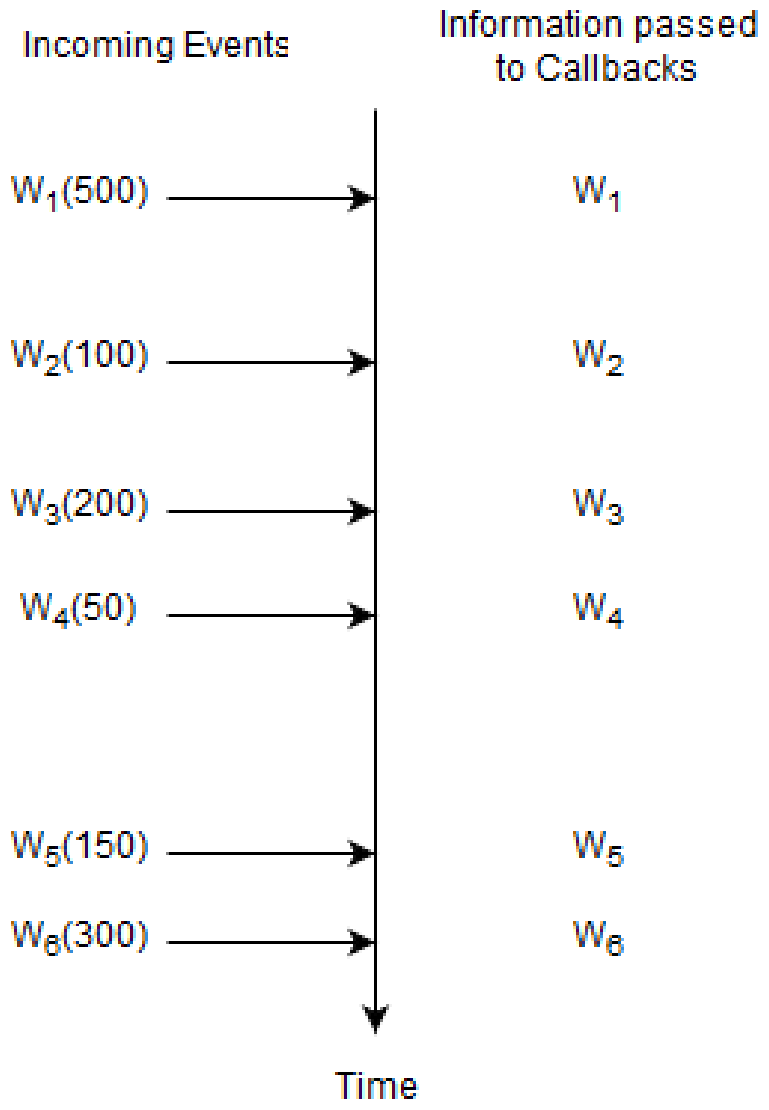
EPL: simple Select statements



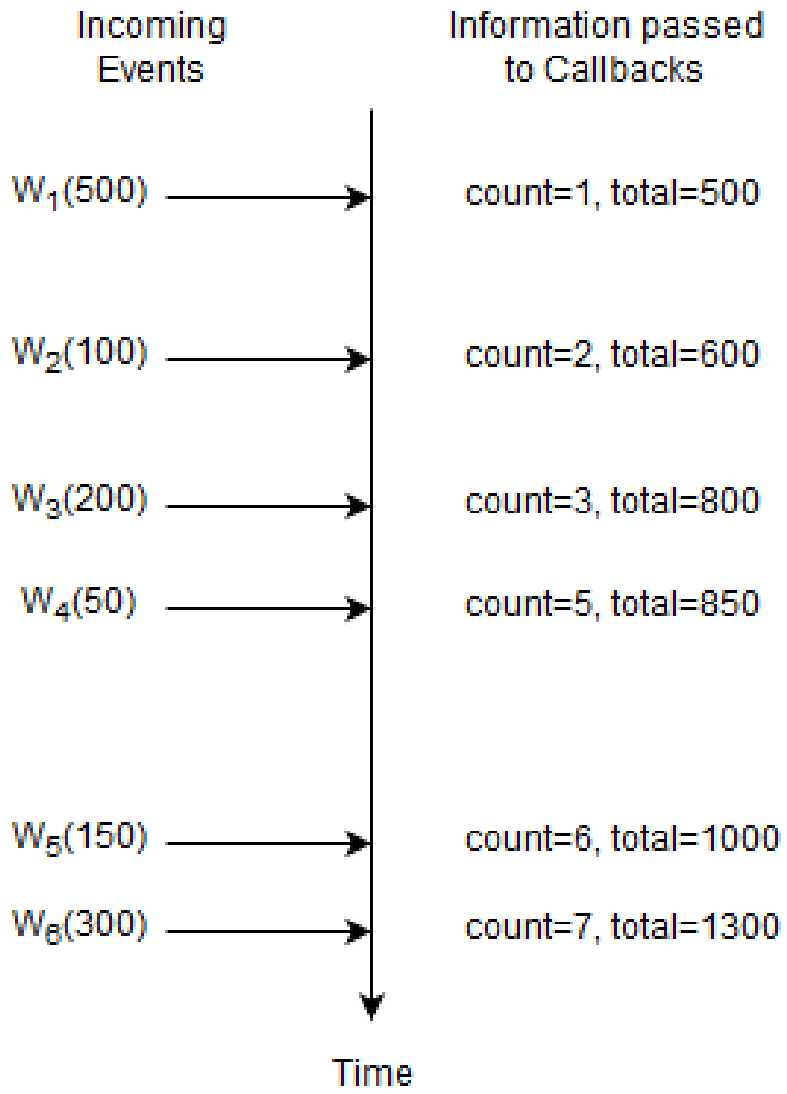
EPL: simple Select statements

```
select * from Withdrawal
```

For this statement, the runtime remembers no information and does not remember any events. A statement where the runtime does not need to remember any information at all is a statement without state (a *stateless* statement).



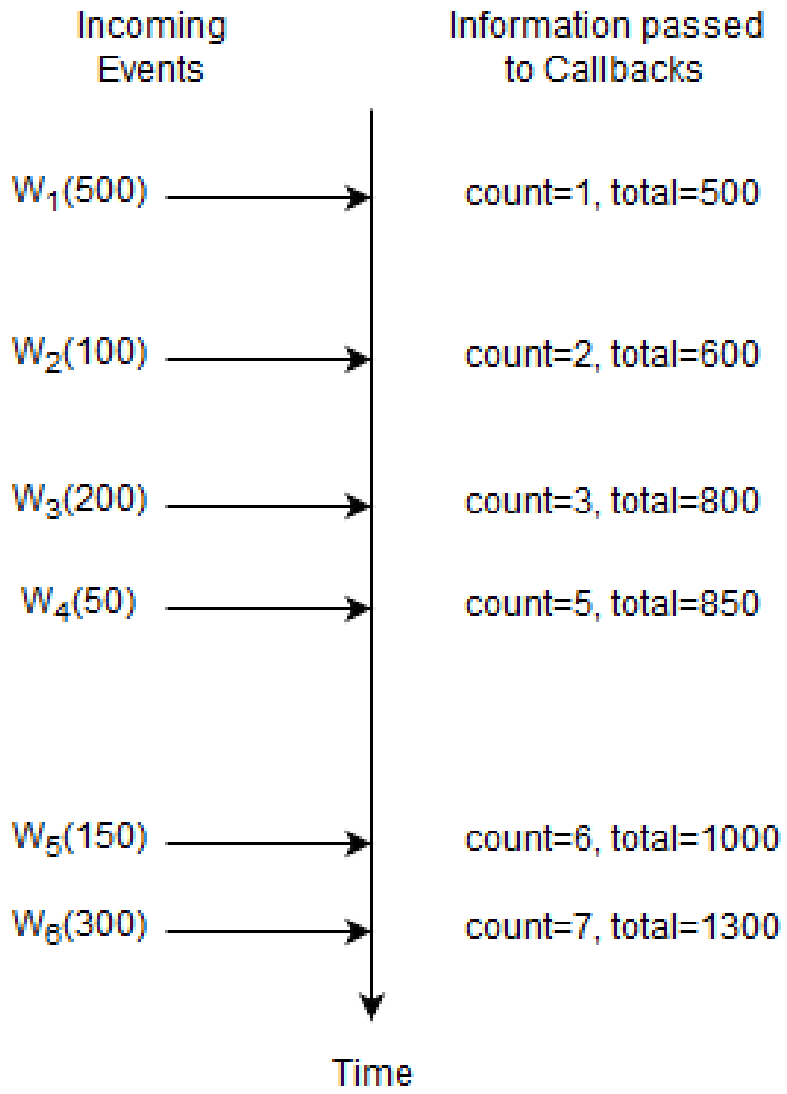
EPL: Aggregation



EPL: Aggregation

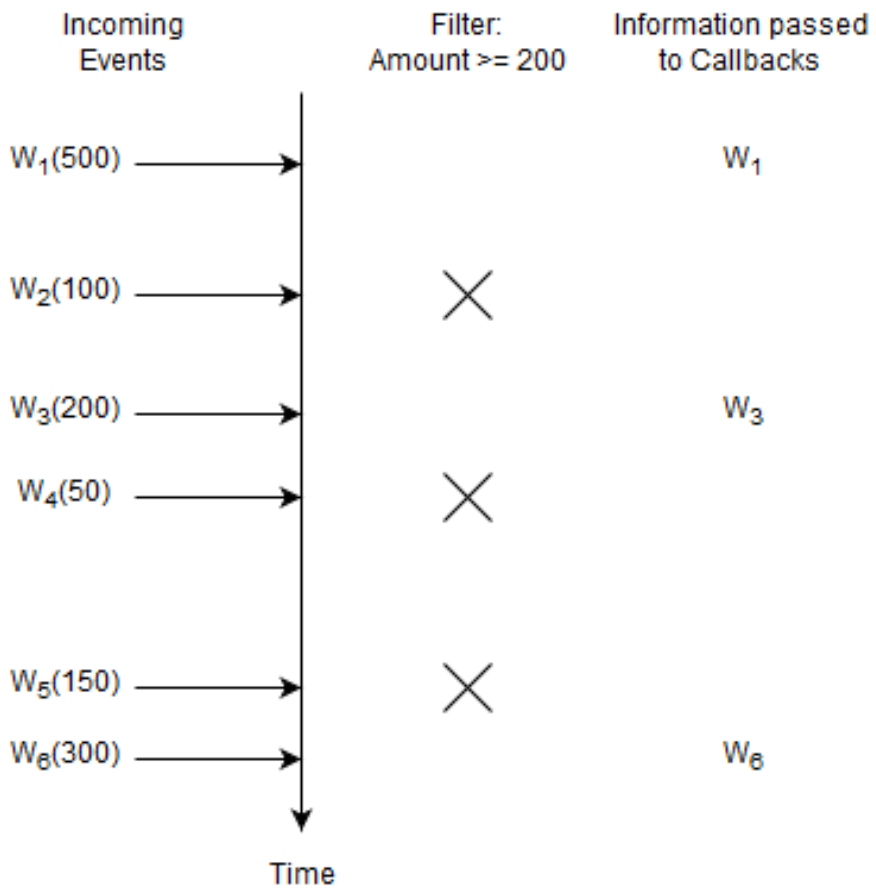
```
select count(*), sum(amount)
from Withdrawal
```

Here, the runtime only remembers the current number of events and the total amount. The count is a single long-type value and the total is a single double-type value (assuming amount is a double-value, the total can be BigDecimal as applicable). This statement is not stateless and the state consists of a long-typed value and a double-typed value.



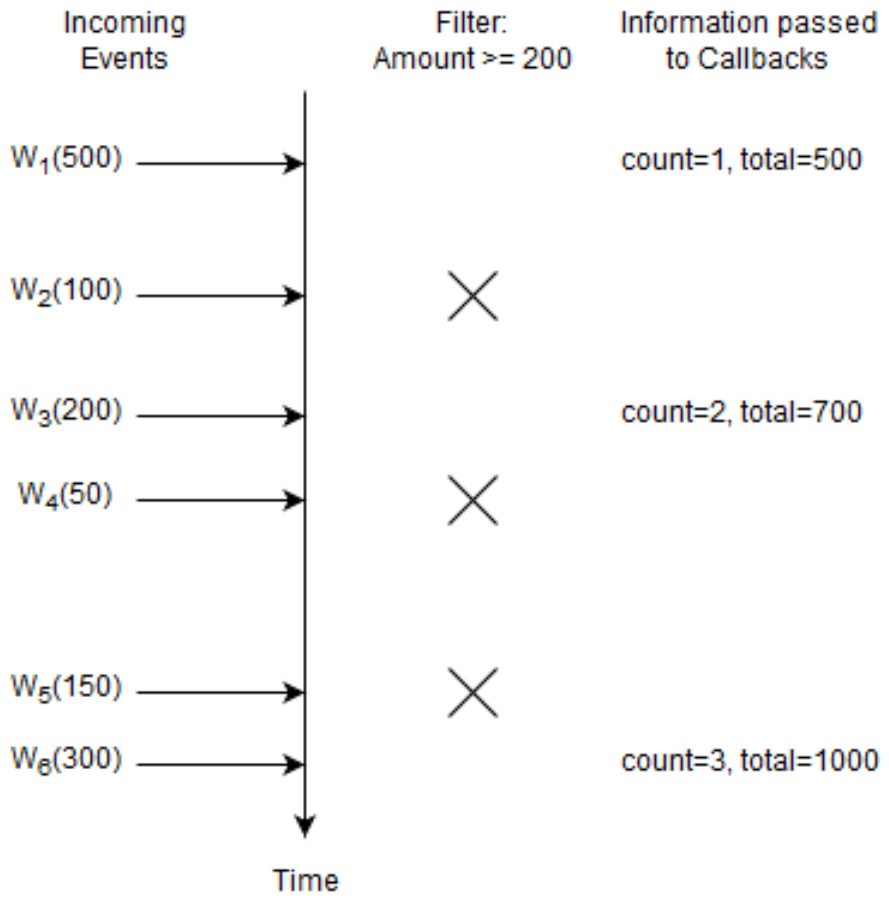
EPL: Basic Filter

```
select * from

Withdrawal(amount >= 200)
```



EPL: Basic Filter and Aggregation

```
select count(*), sum(amount)

from Withdrawal(amount >= 200)
```



Hands-On

- Tool to test EPL queries
- Example of a simple EPL query

EPL: Windows

- Certain operators need that events be grouped before being applied
- Grouping relevant events is done via "**Windows**"
- Windows could have different policies:

Window type	Description
Temporal or time Windows	Events are retained from t_0 to t_1
Data windows	Events are retained as long as size n has not been reached

EPL: Windows

- Following the mechanism of evicting events from windows, we can identify two types of windows

Window type	Description
Tumbling windows	Events in the window are evicted
Sliding windows	Only the oldest event or events are evicted

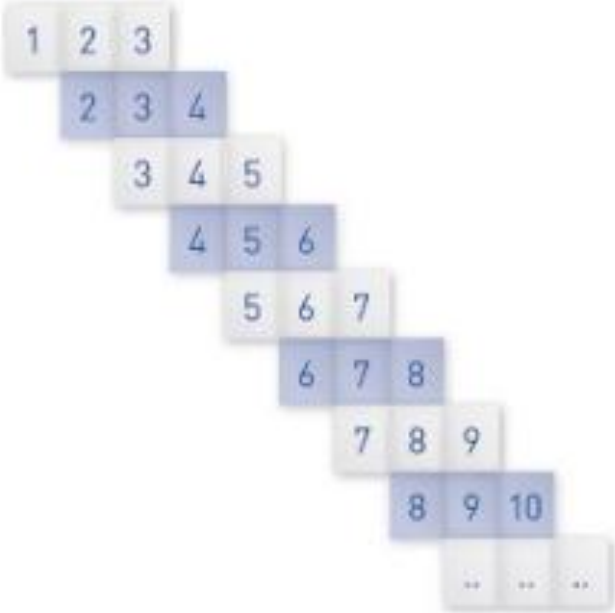
EPL: Windows Illustration

- Each box represents a new event
- Applying a dimension of 3 and specifying a **tumbling** window will result in the events being grouped as follows:

All events have been evicted => no events are repeated

EPL: Windows Illustration

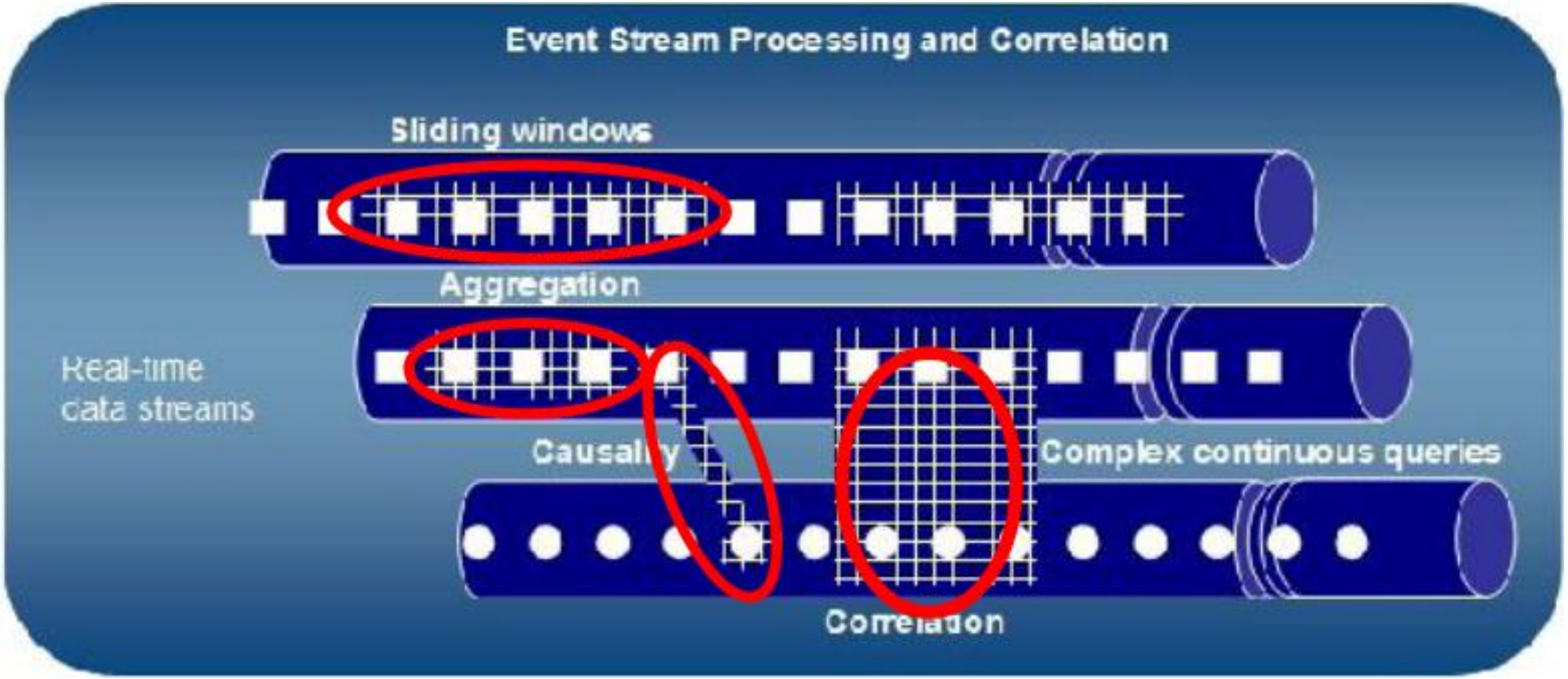
- Applying the same dimension of 3, if it is a sliding window the events will be grouped as follows:



Hands-On

- Time Window Example
- Data Window Example

EPL: detecting patterns



EPL: Event Patterns

- **Sequence Operator (;):** *(E1;E2)*

EPL: Event Patterns

- **Sequence Operator (;):** *(E1;E2)*
- **Disjunction Operator (∨):** *(E1 ∨ E2)* , at least one
- **Conjunction Operator (∧):** (E1 ∧ E2)
- **Simultaneous Operator (=):** *(E1 = E2)*
- **Negation Operator (¬):** *(E1 ∧ ¬ E2)*

EPL: Event Patterns

- **Sequence Operator (;):** *(E1;E2)*
- **Disjunction Operator (∨):** *(E1 ∨ E2)* , at least one
- **Conjunction Operator (∧):** (E1 ∧ E2)
- **Simultaneous Operator (=):** *(E1 = E2)*
- **Negation Operator (¬):** *(E1 ∧ ¬ E2)*
- **Quantification (Any):** Any(n) E1, when n events of type E1 occurs
- **Aperiodic Operator (Ap):** *Ap(E2, E1, E3)* , E2 Within E1 & E3
- **Periodic Operator (Per):** Per(t, E1, E2) , every t time-steps in between E1 and E2

EPL: Event patterns with ‘Every’ operator

- Example of event stream:
  - $A_1, B_1, C_1, B_2, A_2, D_1, A_3, B_3, E_1, A_4, F_1, B_4$
  - The result of $every(A -> B)$

EPL: Event patterns with ‘Every’ operator

- Example of event stream:
  - $\boxed{A_1, B_1}\ C_1, B_2, A_2, D_1, A_3, B_3, E_1, A_4, F_1, B_4$
  - The result of $every(A -> B)$
    - Matches on B_1

EPL: Event patterns with ‘Every’ operator

- Example of event stream:
  - $\boxed{A_1, B_1}\ C_1, B_2, A_2, D_1, \boxed{A_3, B_3}\ E_1, A_4, F_1, B_4$
  - The result of $every(A -> B)$
    - Matches on B_1
    - Matches on B_3

EPL: Event patterns with ‘Every’ operator

- Example of event stream:
  - $A_1, B_1, C_1, B_2, A_2, D_1, A_3, B_3, E_1, A_4, F_1, B_4$
  - The result of $every(A -> B)$
    - Matches on B_1
    - Matches on B_3
    - Matches on B_4

EPL: Event patterns with ‘Every’ operator

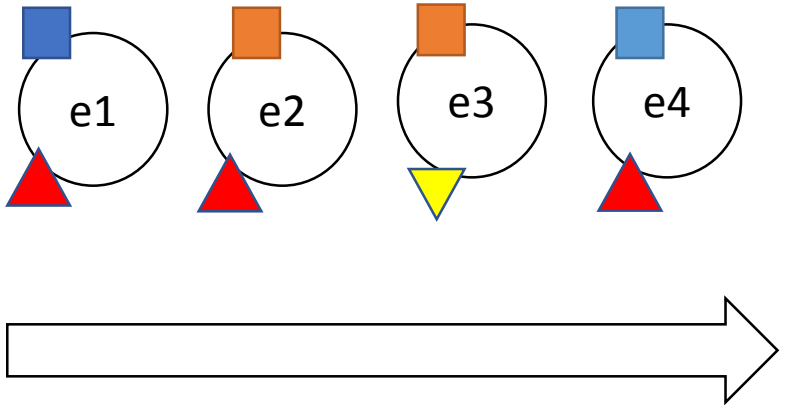
- Example of event stream:
  - $A_1, B_1, C_1, B_2, A_2, D_1, A_3, B_3, E_1, A_4, F_1, B_4$
  - The result of $every(A -> B)$
    - Matches on B_1
    - Matches on B_3
    - Matches on B_4
  - The result of $everyA -> B$
    - Matches on $B1$ for combination $\{A1, B1\}$
    - Matches on $B3$ for combination $\{A2, B3\}$ and $\{A3, B3\}$
    - Matches on $B4$ for combination $\{A4, B4\}$
  - The result of $\mathrm{A} \rightarrow every\ B$
  - The result of every $\mathrm{A} \rightarrow every\ B$

Hands-On

- Example of a simple pattern of events
- Example of a complex pattern of events

EPL: Match-Recognize Patterns

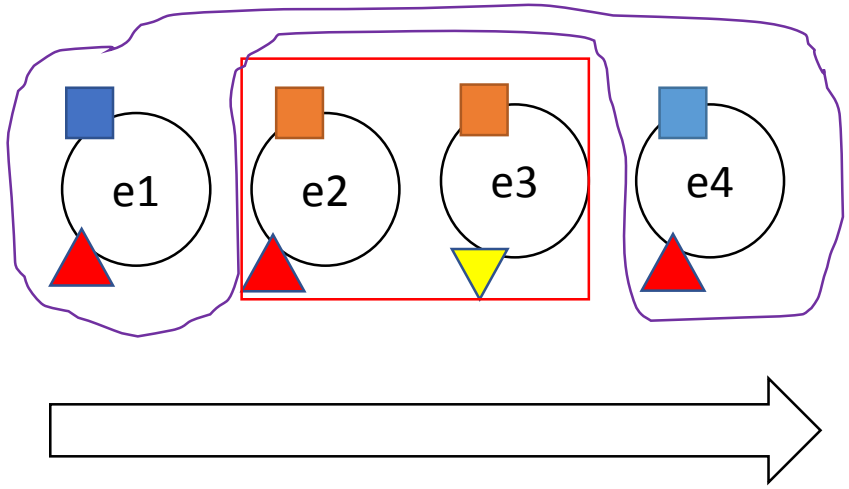
- Match-Recognize allows us to find patterns among events. Let's consider the following match-recognize expression




```
select * from AlertNamedWindow
  match_recognize (
    partition by origin
    measures a1.origin as origin,
    a1.alarmNumber as alarmNumber1,
    a2.alarmNumber as alarmNumber2
    pattern (a1 a2)
    define
      a1 as a1.priority = 'high',
      a2 as a2.priority = 'medium'
)
```

EPL: Match-Recognize Patterns

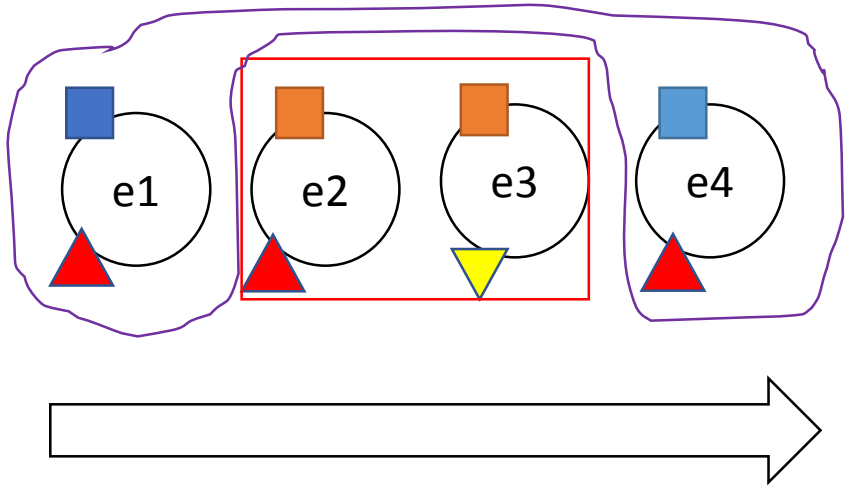
- Match-Recognize allows us to find patterns among events. Let's consider the following match-recognize expression




```
select * from AlertNamedWindow
  match_recognize (
    partition by origin
    measures a1.origin as origin,
    a1.alarmNumber as alarmNumber1,
    a2.alarmNumber as alarmNumber2
    pattern (a1 a2)
    define
      a1 as a1.priority = 'high',
      a2 as a2.priority = 'medium'
)
```

EPL: Match-Recognize Patterns

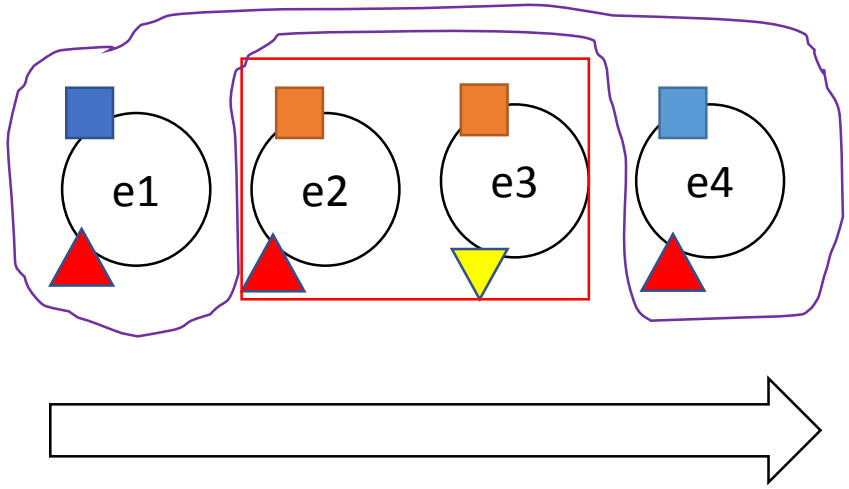
- Match-Recognize allows us to find patterns among events. Let's consider the following match-recognize expression




```
select * from AlertNamedWindow
  match_recognize (
    partition by origin
    measures a1.origin as origin,
    a1.alarmNumber as alarmNumber1,
    a2.alarmNumber as alarmNumber2
    pattern (a1 a2)
    define
      a1 as a1.priority = 'high',
      a2 as a2.priority = 'medium'
)
```

EPL: Match-Recognize Patterns

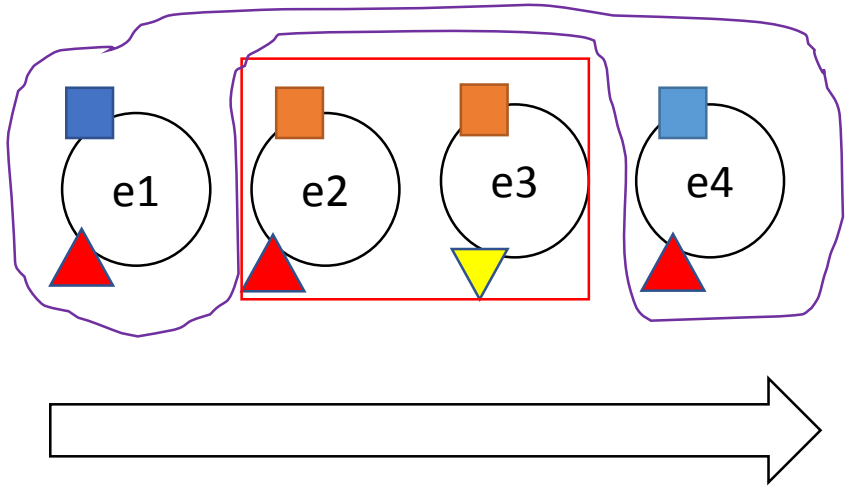
- Match-Recognize allows us to find patterns among events. Let's consider the following match-recognize expression




```
select * from AlertNamedWindow
  match_recognize (
    partition by origin
    measures a1.origin as origin,
    a1.alarmNumber as alarmNumber1,
    a2.alarmNumber as alarmNumber2
    pattern (a1 a2)
    define
      a1 as a1.priority = 'high',
      a2 as a2.priority = 'medium'
)
```

EPL: Match-Recognize Patterns

- Match-Recognize allows us to find patterns among events. Let's consider the following match-recognize expression

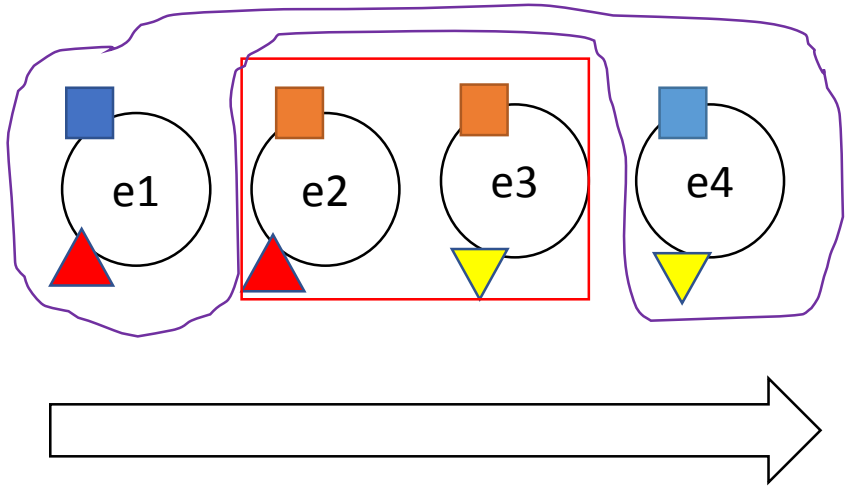



```
select * from AlertNamedWindow
  match_recognize (
    partition by origin
    measures a1.origin as origin,
    a1.alarmNumber as alarmNumber1,
    a2.alarmNumber as alarmNumber2
    pattern (a1 a2)
    define
      a1 as a1.priority = 'high',
      a2 as a2.priority = 'medium'
)
```

Example: here

EPL: Match-Recognize Patterns

- Match-Recognize allows us to find patterns among events. Let's consider the following match-recognize expression




```
select * from AlertNamedWindow
  match_recognize (
    partition by origin
    measures a1.origin as origin,
    a1.alarmNumber as alarmNumber1,
    a2.alarmNumber as alarmNumber2
    pattern (a1 a2)
    define
      a1 as a1.priority = 'high',
      a2 as a2.priority = 'medium'
)
```

Example: here

Hands-On

- Example 1 of match-recognize
- Example 2 of match-recognize

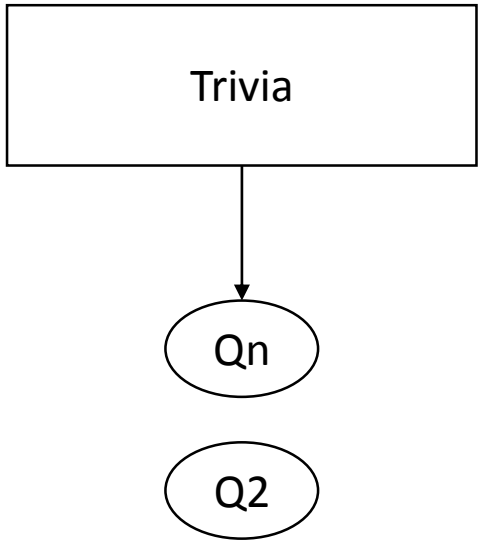
Solution Patterns

- https://www.espertech.com/esper/solution-patterns/

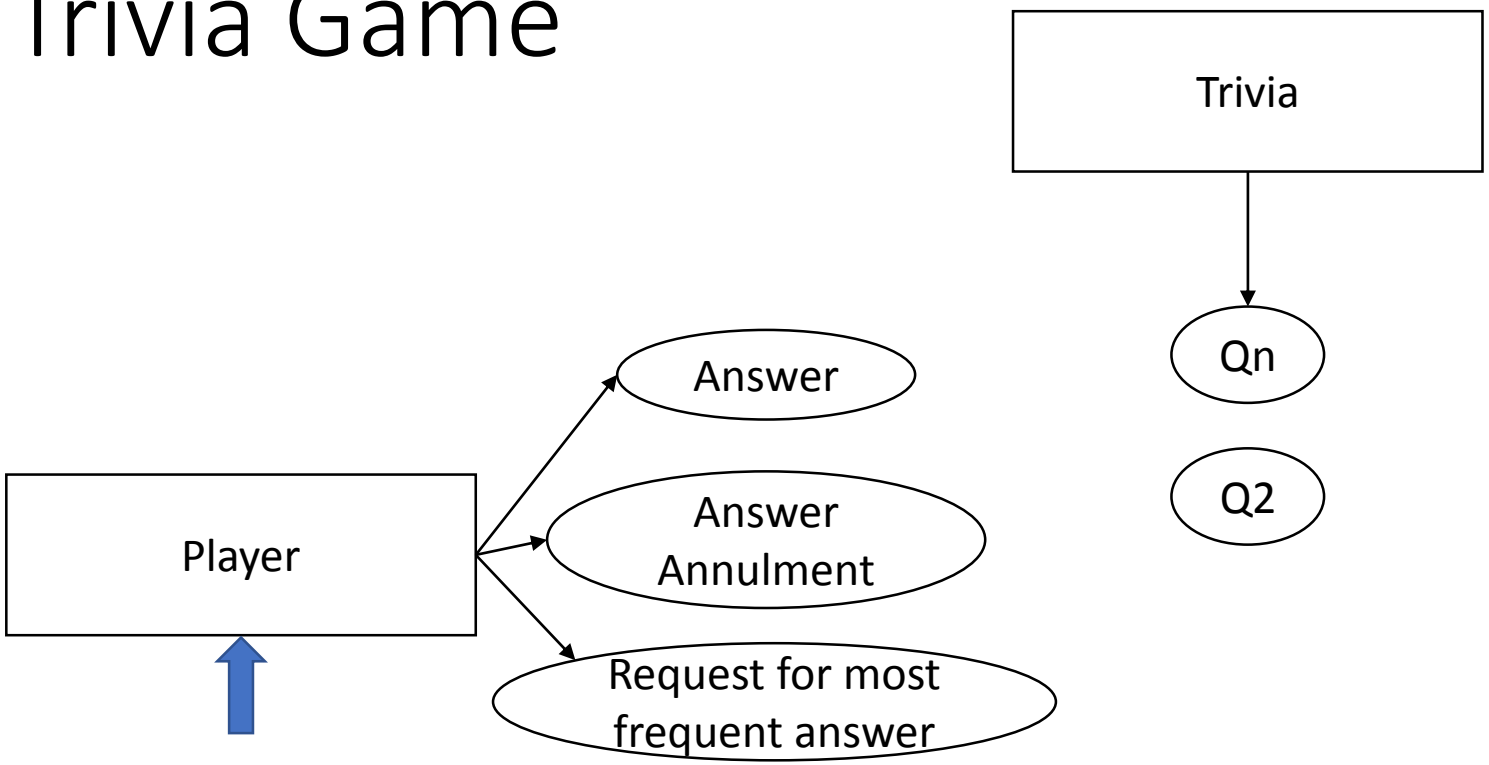
Use case: implementing Trivia game using CEP

- For this use case, we will use EPL notebook
- It is a Jupyter-like notebook
- Two main clauses are used:
  - %esperepl
  - %esperscenario
- We re-run our simple EPL example here

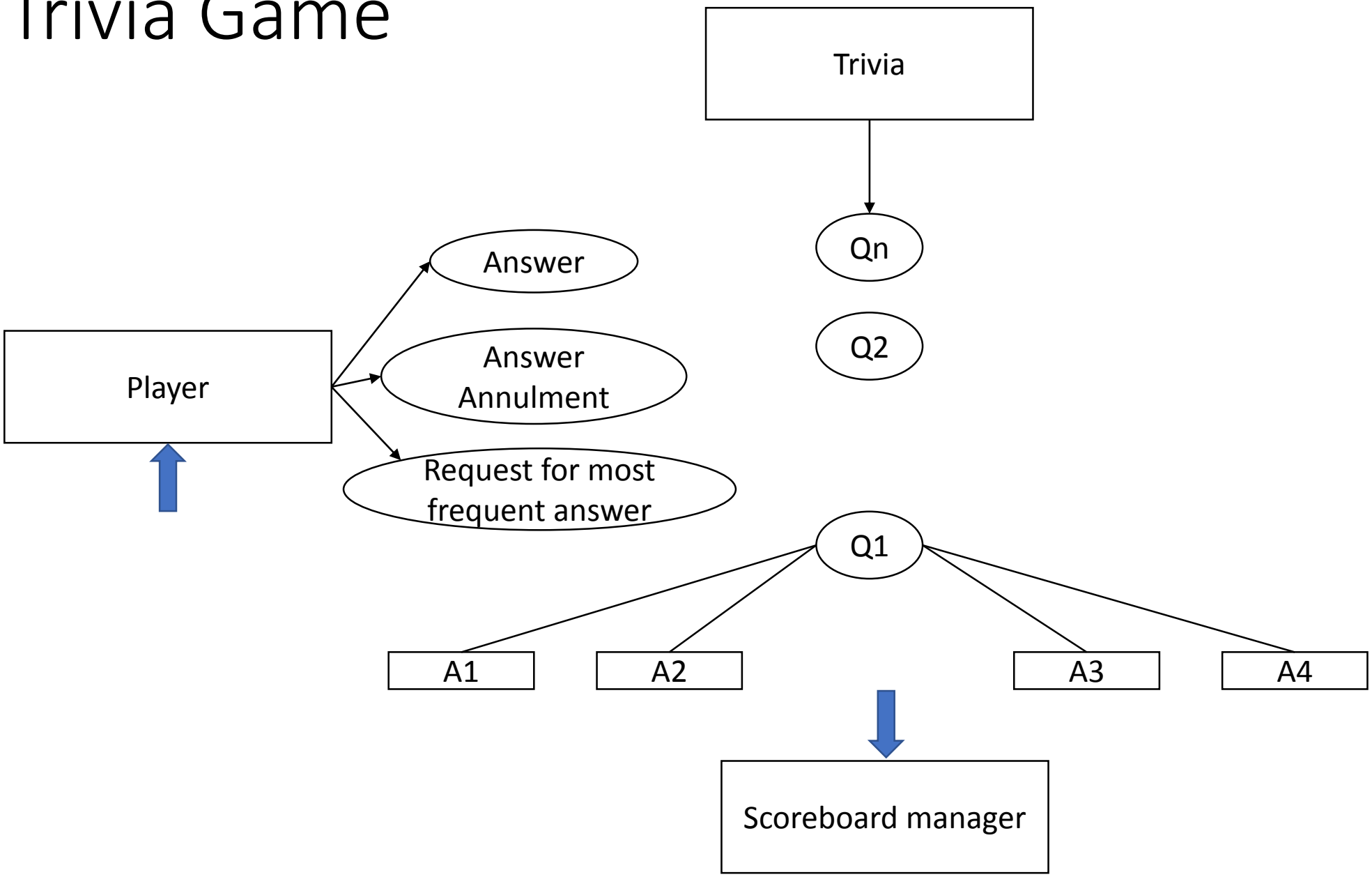
Trivia Game



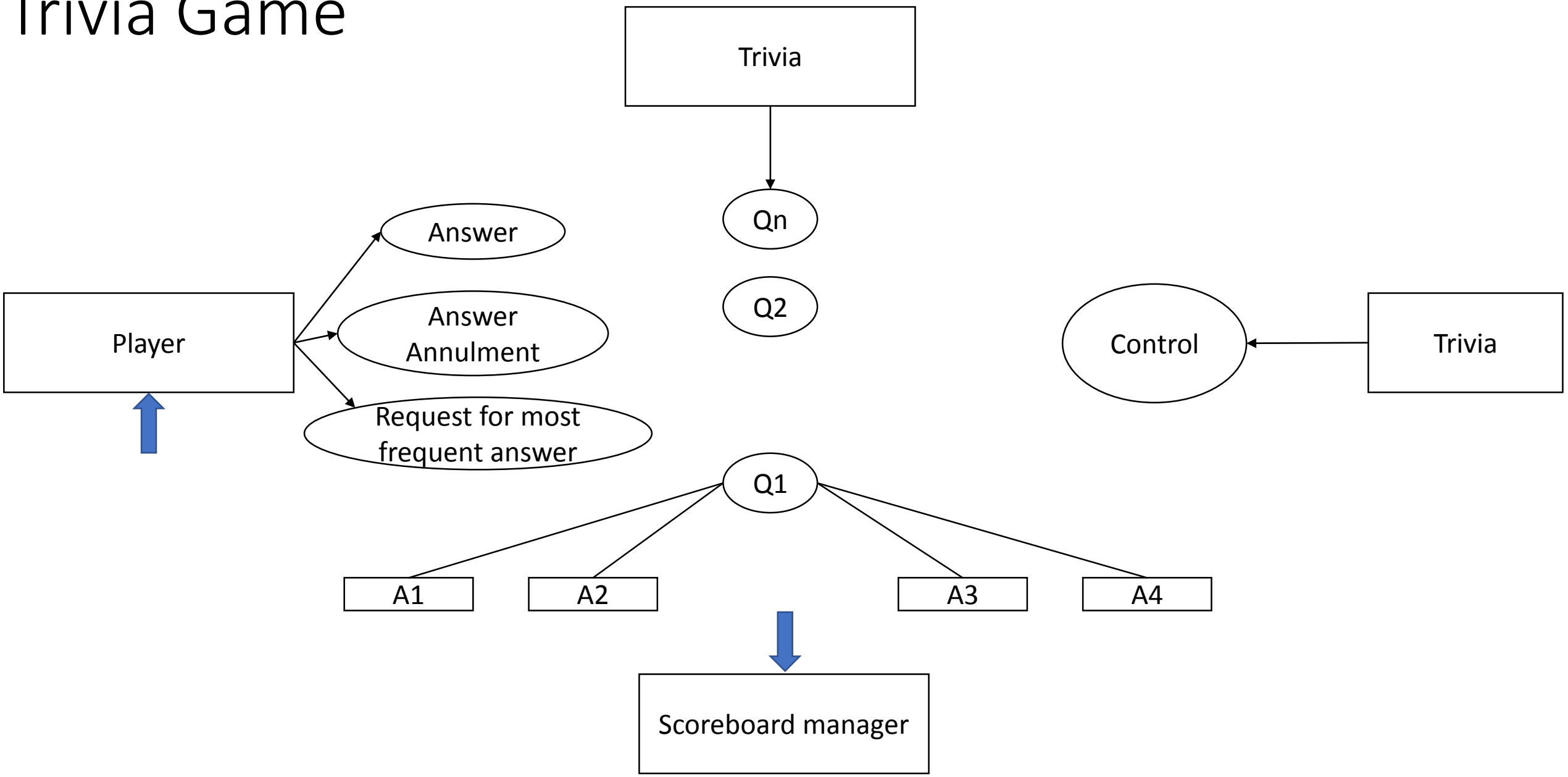
Trivia Game


Trivia
Qn
Q2
Player
Answer
Answer Annulment
Request for most frequent answer

Trivia Game


Trivia
Qn
Q2
Answer
Answer Annulment
Request for most frequent answer
Player
Q1
A1
A2
A3
A4
Scoreboard manager

Trivia Game


Trivia
Qn
Q2
Answer
Answer Annulment
Request for most frequent answer
Player
Control
Trivia
Q1
A1
A2
A3
A4
Scoreboard manager

Trivia Game: Illustration

Q1

Wed Mar 06 2024 09:00:00
Answer = 2

Trivia Game: Illustration

Q1

Wed Mar 06 2024 09:00:00
Answer = 2

Player 1

Wed Mar 06 2024 09:00:01

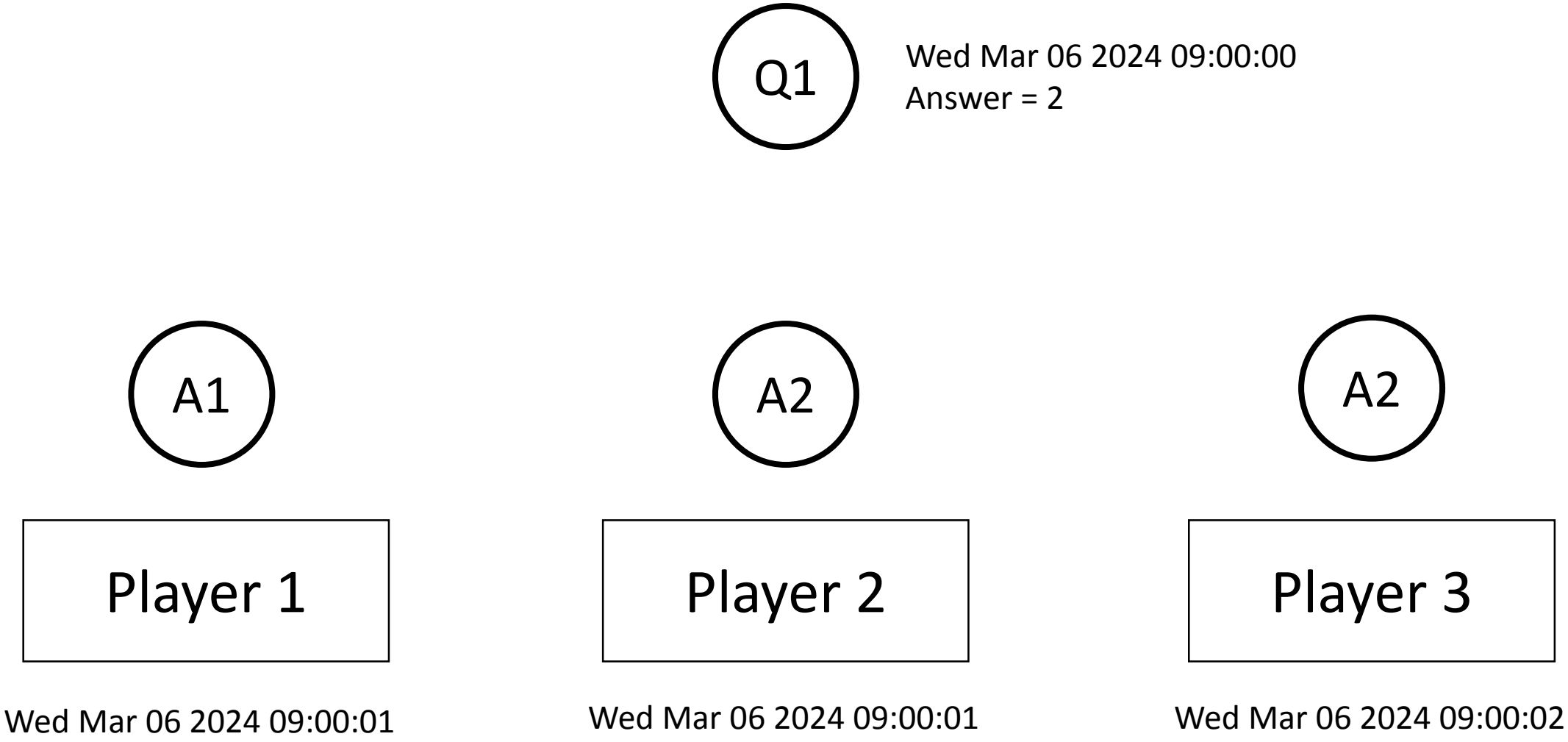
Player 2

Wed Mar 06 2024 09:00:01

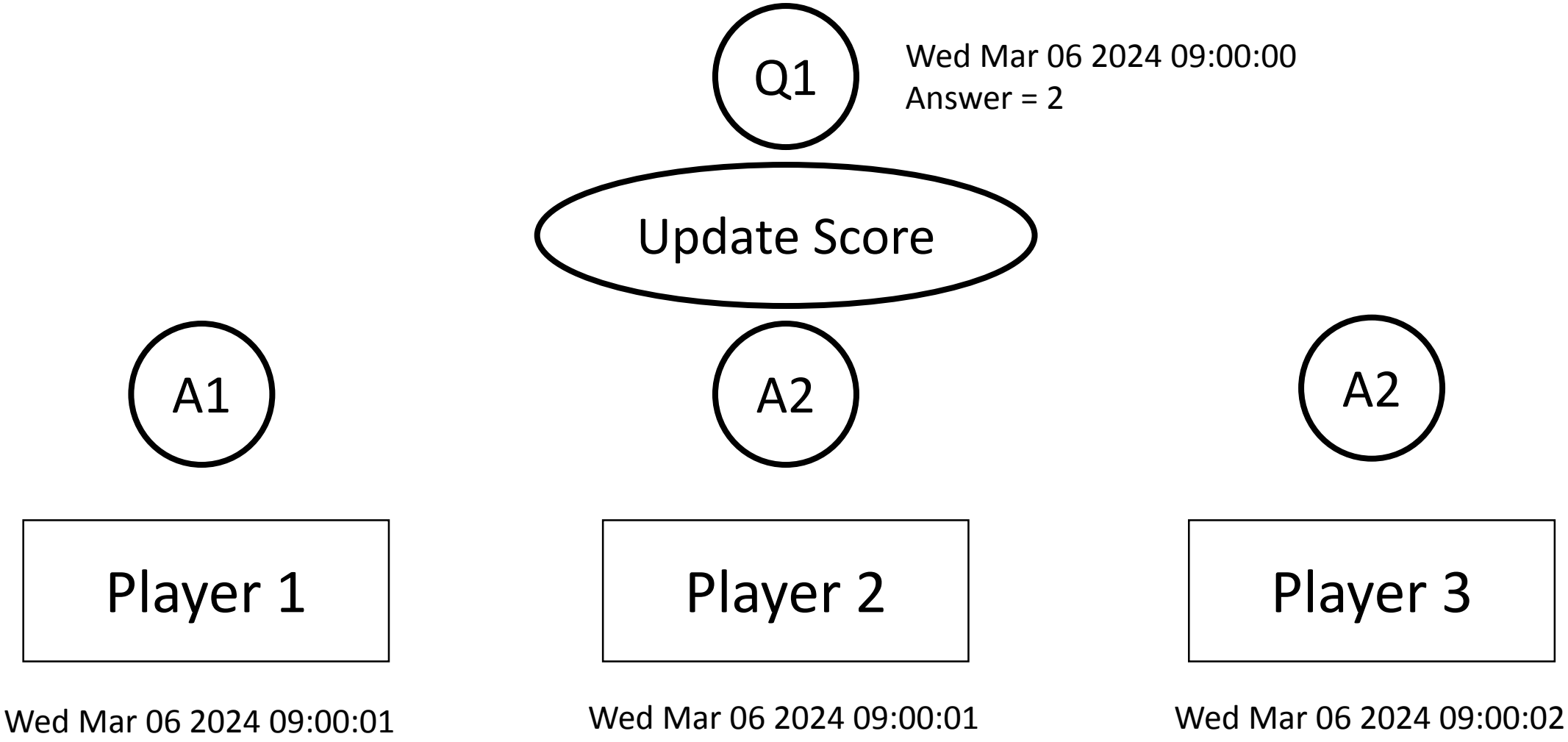
Player 3

Wed Mar 06 2024 09:00:02

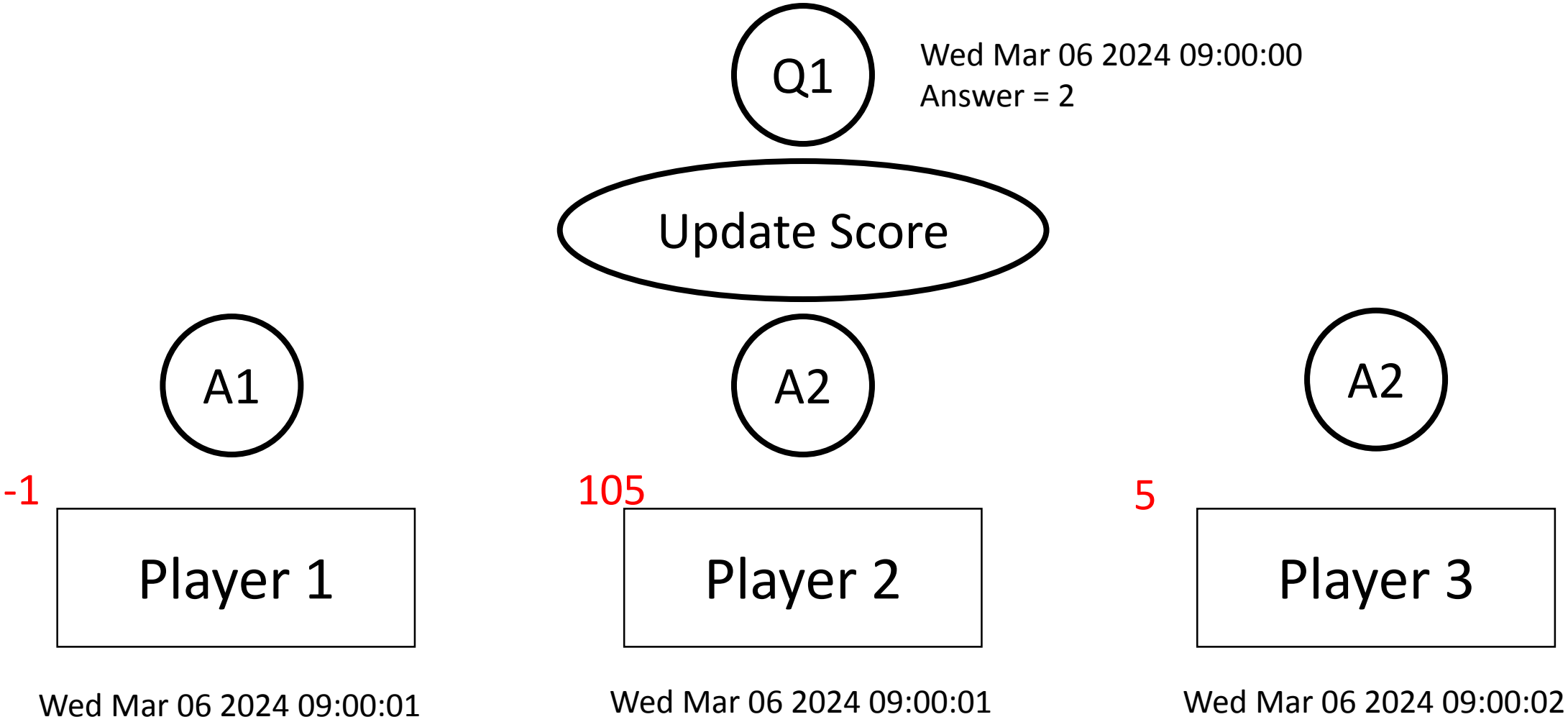
Trivia Game: Illustration



Trivia Game: Illustration



Trivia Game: Illustration



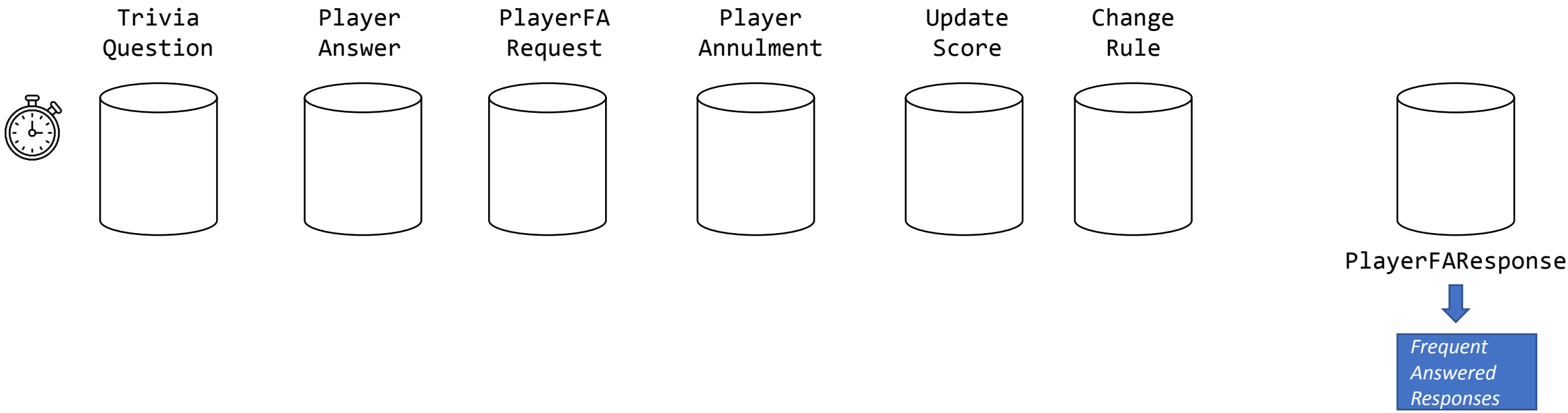
Trivia rules

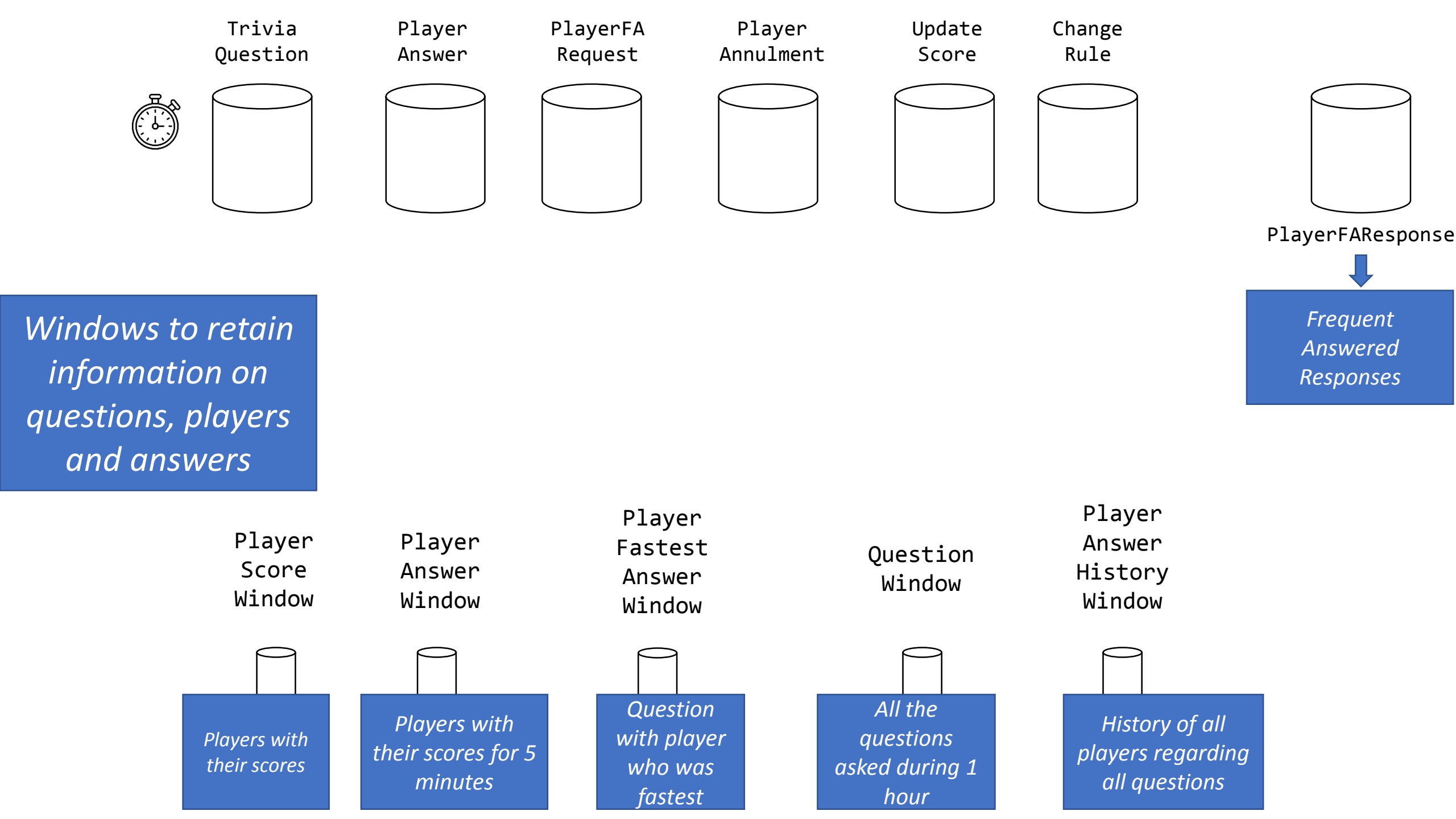
The scoring system creates score event with points for player according to the following scoring table:

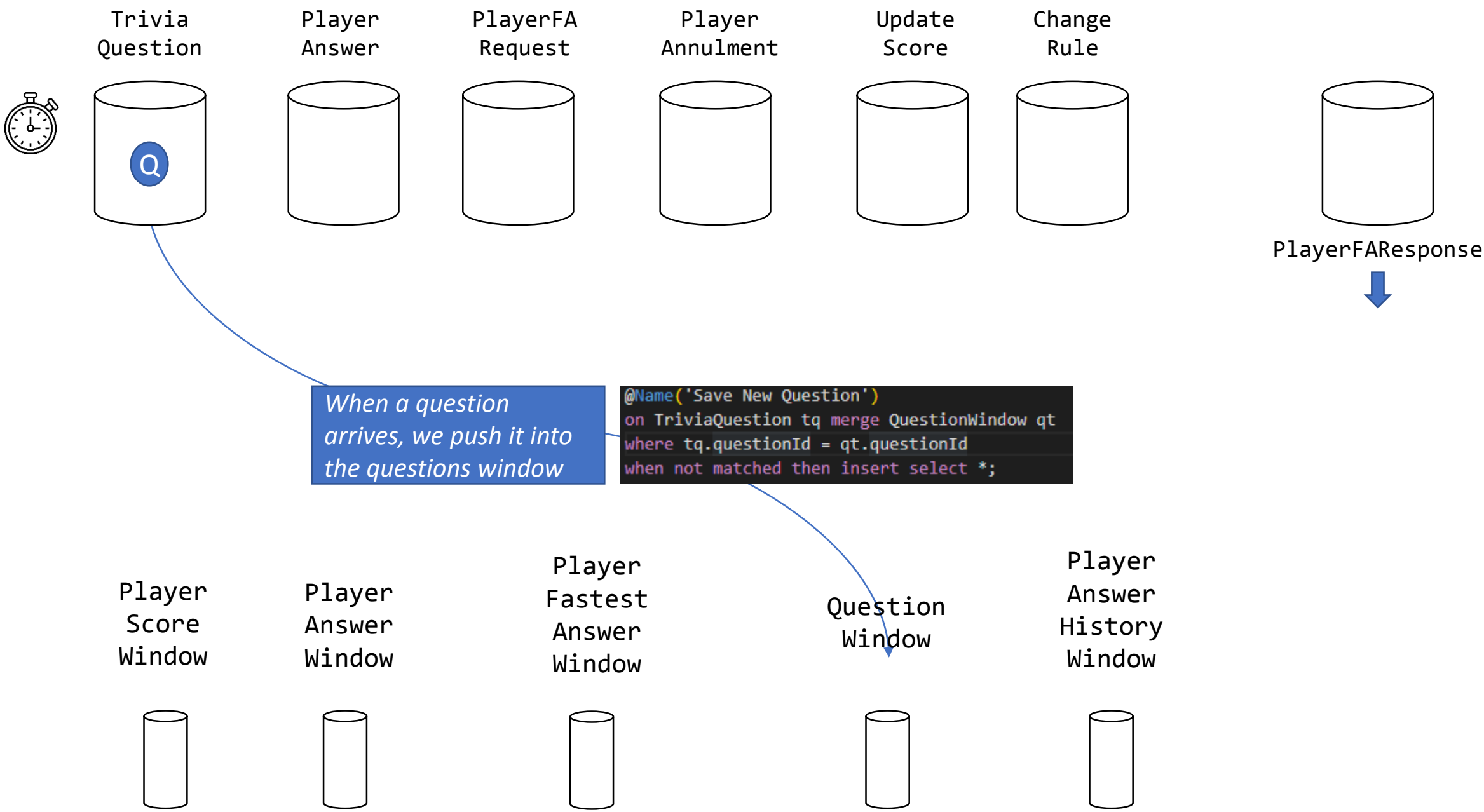
Correct answer	5
Correct answer after asking for the most frequent answer	1
First who answered	100
Incorrect answer	-1
Three answers incorrect without a correct answer in the middle	-50
Correct answers to 10 consecutive questions*	500
Correct answers to 10 questions within 30 minutes* during late night hours (1:00 – 5:00)	500

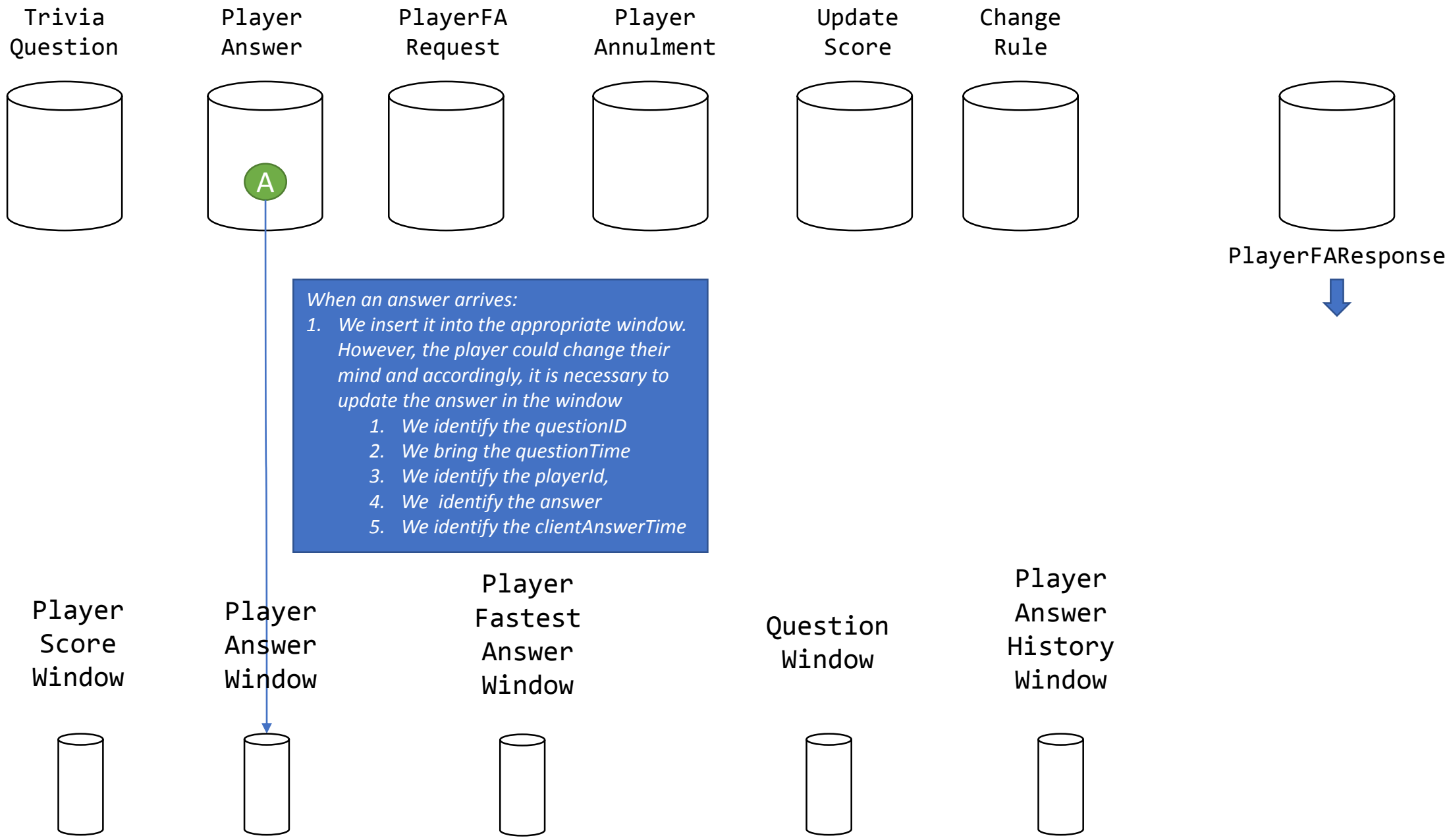
each correct answer is counted towards a single bonus of the same type and cannot be counted twice.

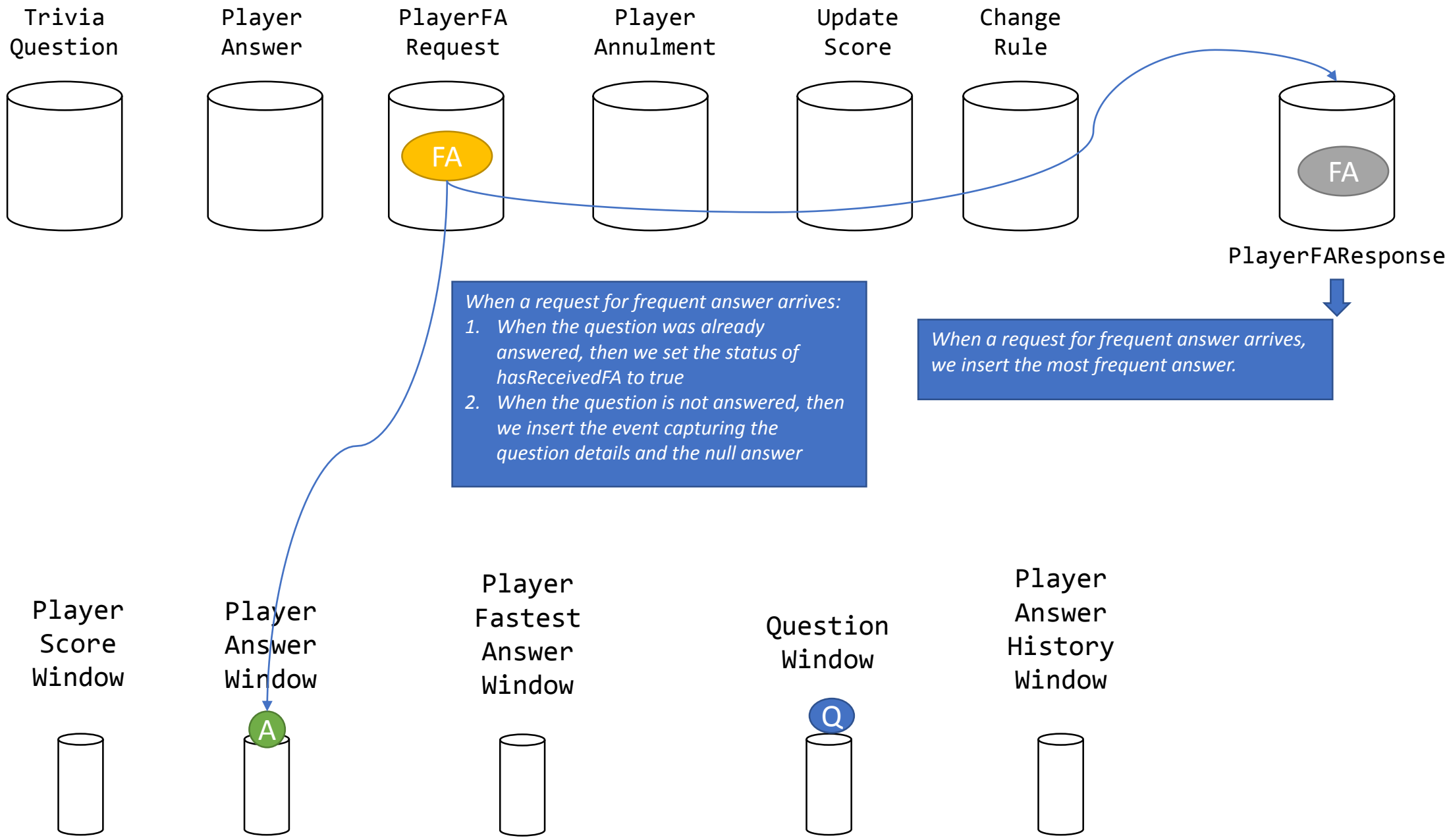
If there are several players that are tied in one of the "best" categories, each of them receives the bonus of 1000 points.

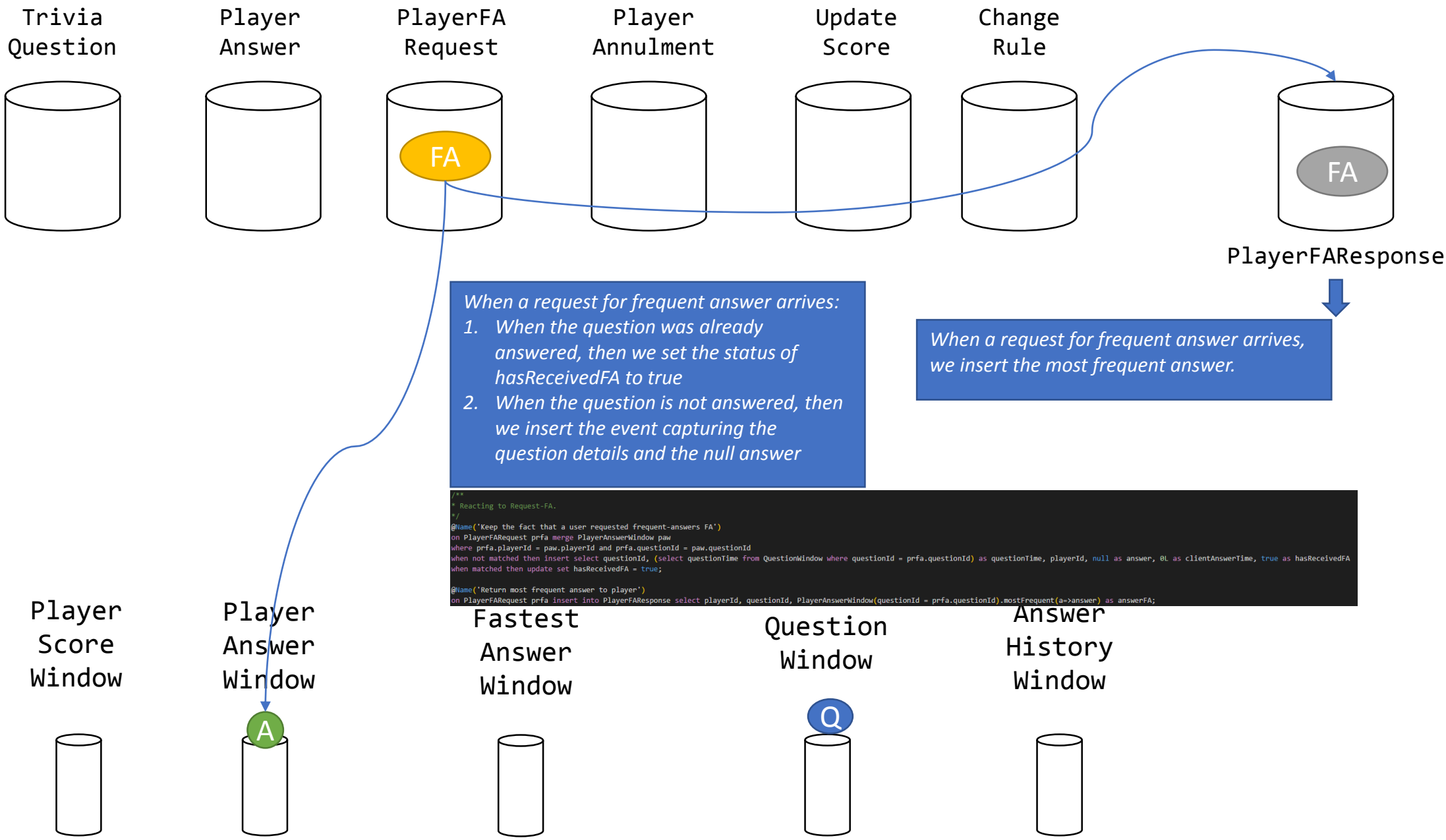

Trivia Question
Player Answer
PlayerFA Request
Player Annulment
Update Score
Change Rule
PlayerFAResponse
Frequent Answered Responses


Trivia Question
Player Answer
PlayerFA Request
Player Annulment
Update Score
Change Rule
PlayerFAResponse
Frequent Answered Responses
Windows to retain information on questions, players and answers
Player Score Window
Player Answer Window
Player Fastest Answer Window
Question Window
Player Answer History Window
Players with their scores
Players with their scores for 5 minutes
Question with player who was fastest
All the questions asked during 1 hour
History of all players regarding all questions

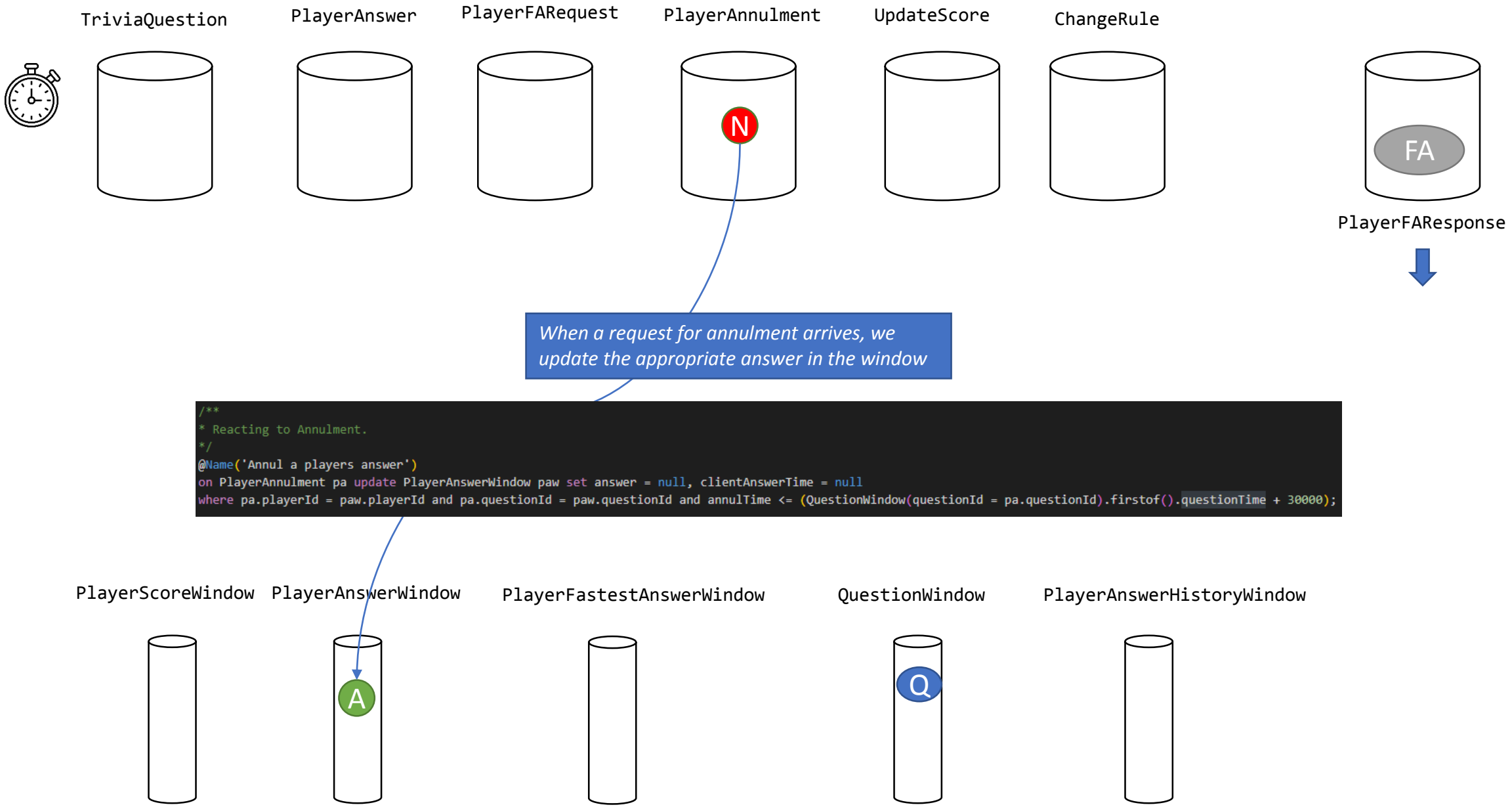

Trivia Question
Q
Player Answer
PlayerFA Request
Player Annulment
Update Score
Change Rule
PlayerFAResponse
When a question arrives, we push it into the questions window
@Name('Save New Question')
on TriviaQuestion tq merge QuestionWindow qt
where tq.questionId = qt.questionId
when not matched then insert select *;
Player Score Window
Player Answer Window
Player Fastest Answer Window
Question Window
Player Answer History Window


Trivia Question
Player Answer
PlayerFA Request
Player Annulment
Update Score
Change Rule
A
PlayerFAResponse
When an answer arrives:
1. We insert it into the appropriate window. However, the player could change their mind and accordingly, it is necessary to update the answer in the window
1. We identify the questionID
2. We bring the questionTime
3. We identify the playerId,
4. We identify the answer
5. We identify the clientAnswerTime
Player Score Window
Player Answer Window
Player Fastest Answer Window
Question Window
Player Answer History Window


Trivia Question
Player Answer
PlayerFA Request
Player Annulment
Update Score
Change Rule
FA
FA
PlayerFAResponse
When a request for frequent answer arrives:
1. When the question was already answered, then we set the status of hasReceivedFA to true
2. When the question is not answered, then we insert the event capturing the question details and the null answer
When a request for frequent answer arrives, we insert the most frequent answer.
Player Score Window
Player Answer Window
A
Player Fastest Answer Window
Question Window
Q
Player Answer History Window




```
/**
* Reacting to Request-FA.
*/
@Name('Keep the fact that a user requested frequent-answers FA')
on PlayerFARequest prfa merge PlayerAnswerWindow paw
where prfa.playerId = paw.playerId and prfa.questionId = paw.questionId
when not matched then insert select questionId, (select questionTime from QuestionWindow where questionId = prfa.questionId) as questionTime, playerId, null as answer, 0L as clientAnswerTime, true as hasReceivedFA
when matched then update set hasReceivedFA = true;

@Name('Return most frequent answer to player')
on PlayerFARequest prfa insert into PlayerFAResponse select playerId, questionId, PlayerAnswerWindow(questionId = prfa.questionId).mostFrequent(a->answer) as answerFA;
```


TriviaQuestion
PlayerAnswer
PlayerFARequest
PlayerAnnulment
UpdateScore
ChangeRule
N
FA
PlayerFAResponse
When a request for annulment arrives, we update the appropriate answer in the window
/**
* Reacting to Annulment.
*/
@Name('Annul a players answer')
on PlayerAnnulment pa update PlayerAnswerWindow paw set answer = null, clientAnswerTime = null
where pa.playerId = paw.playerId and pa.questionId = paw.questionId and annulTime <= (QuestionWindow(questionId = pa.questionId).firstof().questionTime + 30000);
PlayerScoreWindow
PlayerAnswerWindow
PlayerFastestAnswerWindow
QuestionWindow
PlayerAnswerHistoryWindow
A
Q

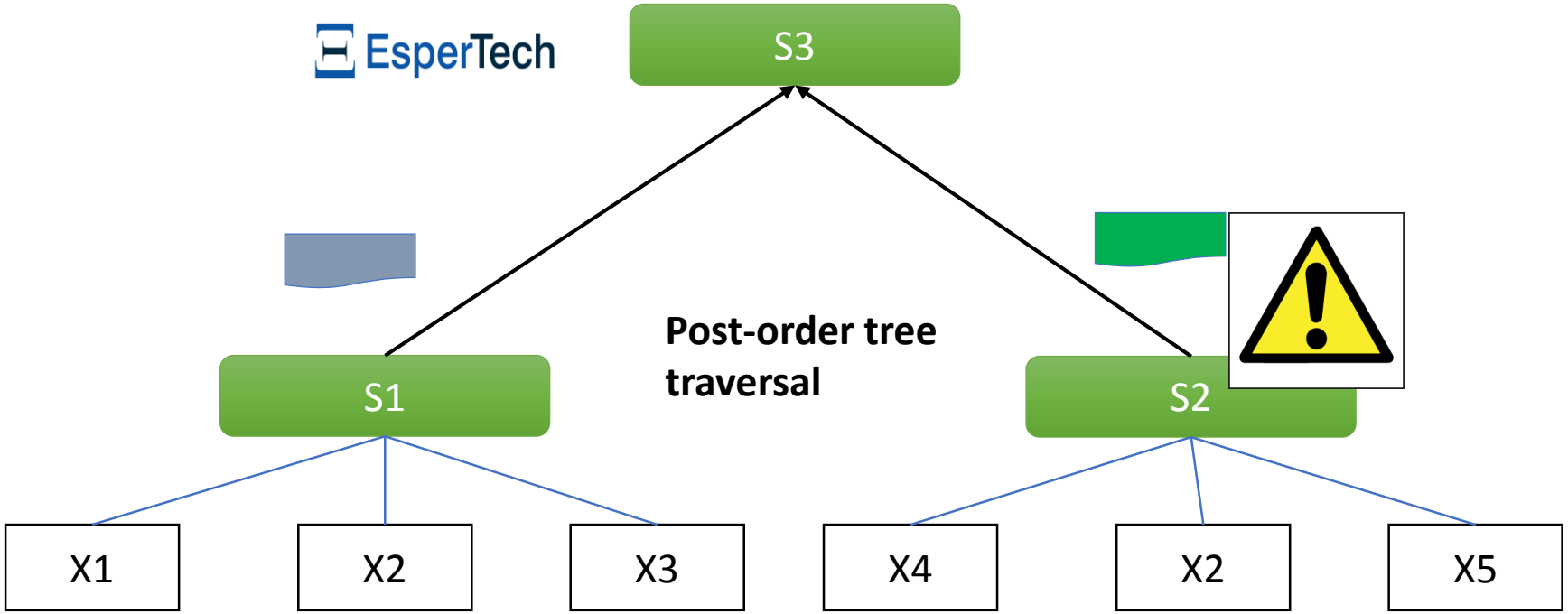
Hands-On

- Trivia Example in EPL notebook

Limitations of Esper

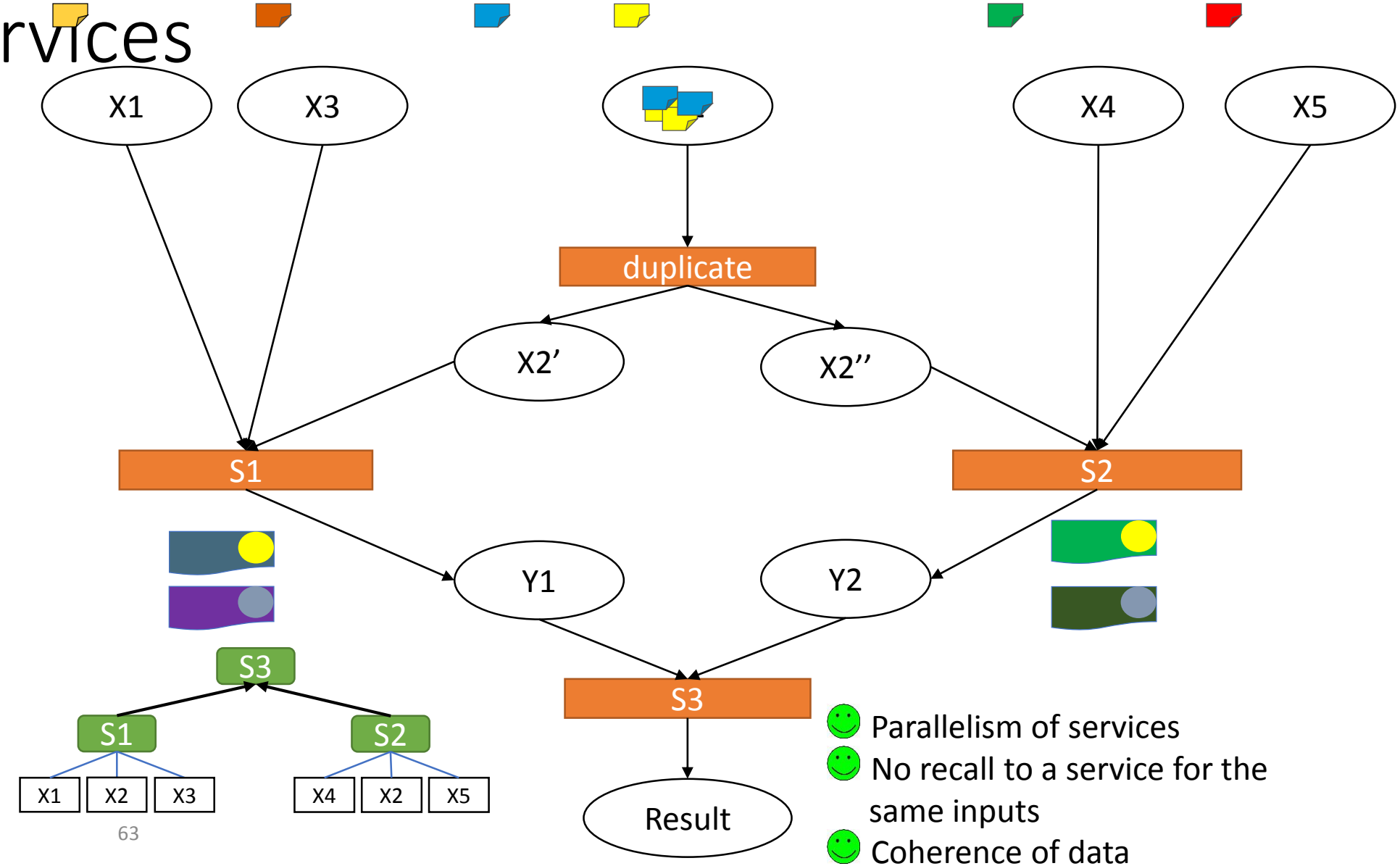
- In certain circumstances Esper's implementation is not efficient and thus was not adequate for industrial deployment
- This occurred with Esper's feature of User Defined Functions (UDF)
- UDFs are services that can be called when running EPL queries to further process incoming events

Limitation of Esper: Industrial Use Case



Need for a more dynamic structure

Algorithm: Optimization of parallelism of services



- Parallelism of services
- No recall to a service for the same inputs
- Coherence of data

Feedback

- Ensuring parallel execution of independent services
- Preventing the execution of the same service two times for the same inputs
- Upholding the preconditions of the services prior to their invocation
- Maintaining the history of all inputs and intermediate results
- Published paper [IEEE SCC conferences]

Conclusion

- CEP constitutes an important asset in data analytics
- It is widely used in the industry and simplifies the implementation of numerous use cases
- Multiple contributions are being made in this area
- Multiple challenges still exist and more particularly the added value of AI in this field
- Esper constitutes one of the well documented open source frameworks that could be used as a benchmark for future improvements