// ovo pitanje nije o tome kako napraviti sharding, tj. hash based i range based i sl. to se **NE traži**

(a) Vremenom opterećenje (količina podataka) raste te rastu veličina odnosno broj fragmenata i potrebno je napraviti preraspodjelu fragmenata, pri čemu se očekuje:

- Baza mora biti operativna za vrijeme preraspodjele
- Pravedna raspodjela opterećenja nakon preraspodjele
- Minimizirati mrežni promet prilikom preraspodjele Za to je potrebno koristiit neku vrstu consistent hashinga

(b) Postoje sljedeće strategije (vidjeti NoSQL4 S22-):

1. Fiksni broj fragmenata koji je napravljeni unaprijed, (npr. Riak ring)
    - Novi čvor preuzima svoj udio cijelih fragmenata od postojećih čvorova
2. Dinamički broj fragmenata (npr. Mongo)
    - Svaki fragment pripada jednom čvoru
    - Ako fragment naraste preko definirane veličine
        - Razdijeliti na dva jednaka dijela
        - Potencijalno jedan fragment prebaciti na drugi čvor
3. Fiksni broj fragmenata po čvoru (npr. Cassandra)
    - Kada se novi čvor priključi, odabire slučajno postojeći čvor i podijeli s njim opterećenje
        - Prije: postojeći čvor s N fragmenata, ukupno X MB
        - Poslije:
            - postojeći čvor s N fragmenata, ukupno ~ X/2 MB
            - novi čvor s N fragmenata, ukupno ~ X/2 MB

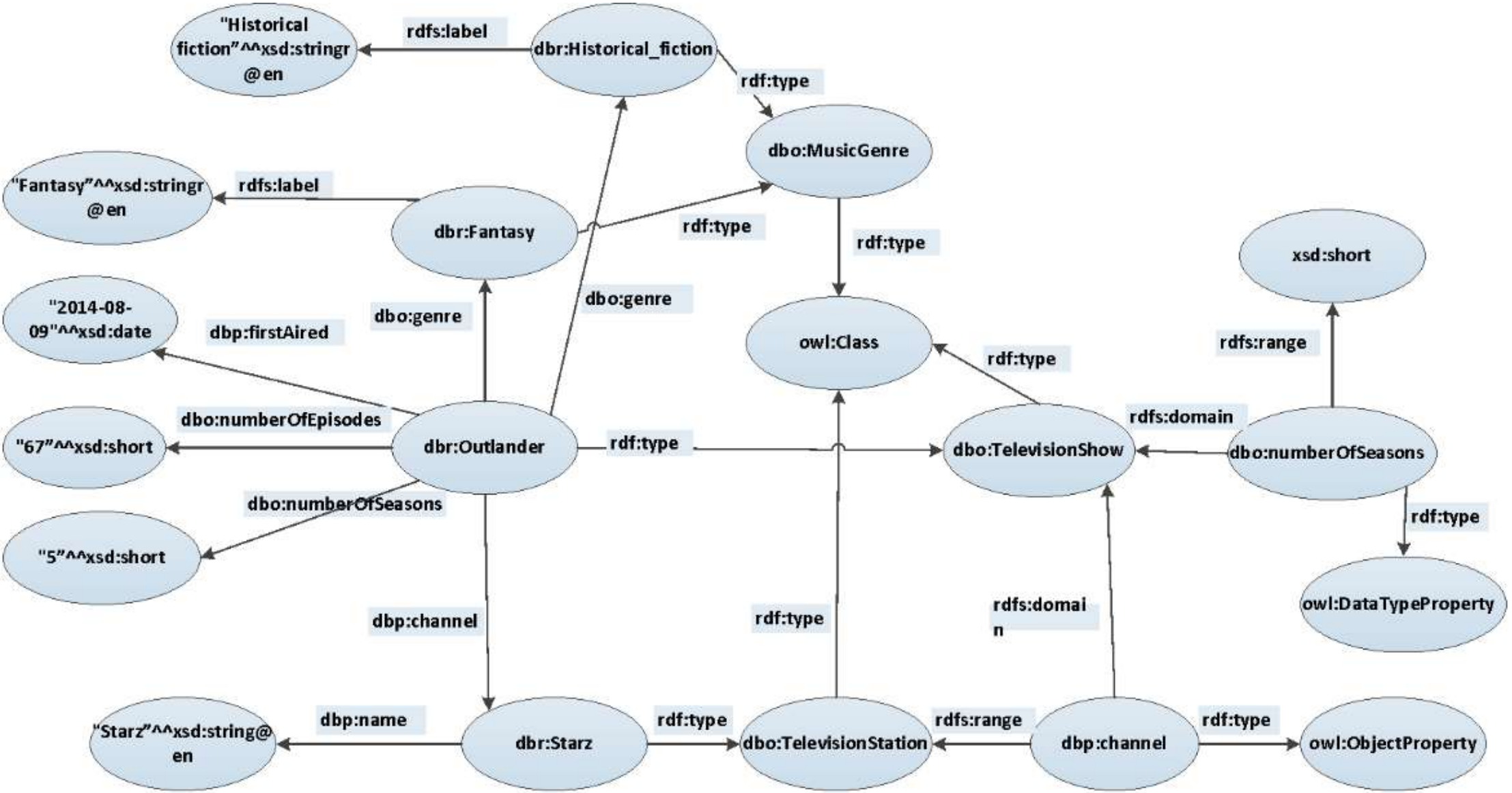
(5 bodova)

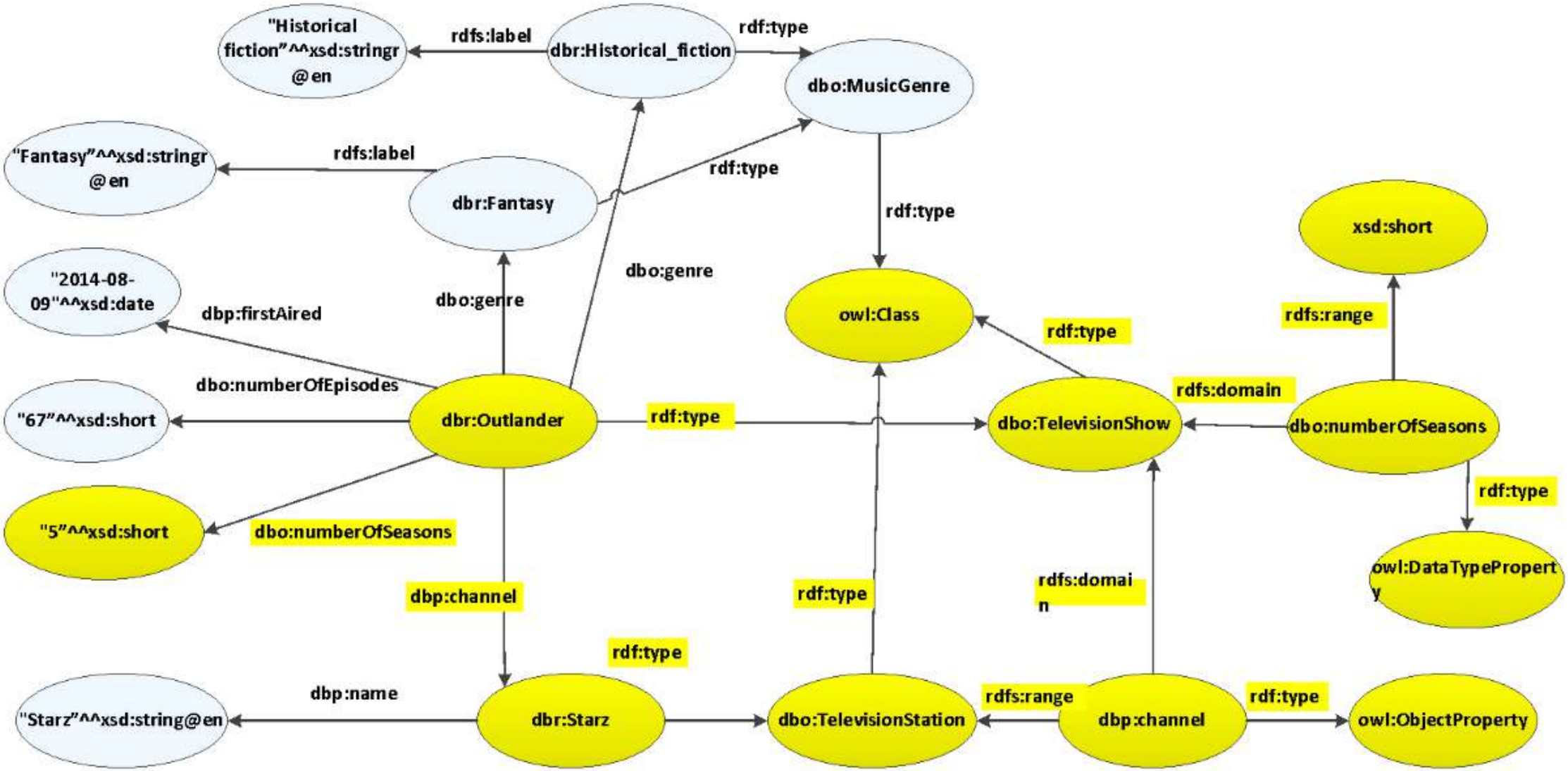
Pomoću kolekcije `nobelprizes` doznajte broj laureata po kategorijama za svaku dekadu u intervalu [1900, 2000). Dodatno, dohvatiti i prosječan broj laureata po dekadama. Za dekadu:

- [1900, 1909] koristiti oznaku "1900"
- [1910, 1919] koristiti oznaku "1910"
- ...
- [1990, 1999] koristiti oznaku "1990"

Primjer dijela rezultata (***eg. part of the result***):

```
[
  {
     "_id": "chemistry",
    "value": {
      "1900": 9,
      "1910": 8,
      "1920": 10,
      "1930": 13,
      "1940": 9,
      "1950": 14,
      "1960": 15,
      "1970": 15,
      "1980": 21,
      "1990": 18,
      "avg": 13.2
    }
  },
  ...
]
```


"Historical fiction"^^xsd:stringr @en
rdfs:label
dbr:Historical_fiction
rdf:type
dbo:MusicGenre
"Fantasy"^^xsd:stringr @en
rdfs:label
dbr:Fantasy
rdf:type
rdf:type
xsd:short
"2014-08-09"^^xsd:date
dbp:firstAired
dbo:genre
dbo:genre
owl:Class
rdf:type
rdfs:range
dbo:numberOfEpisodes
"67"^^xsd:short
dbr:Outlander
rdf:type
dbo:TelevisionShow
rdfs:domain
dbo:numberOfSeasons
dbo:numberOfSeasons
"5"^^xsd:short
rdf:type
owl:DataTypeProperty
dbp:channel
rdf:type
rdfs:domai n
"Starz"^^xsd:string@ en
dbp:name
dbr:Starz
rdf:type
dbo:TelevisionStation
rdfs:range
dbp:channel
rdf:type
owl:ObjectProperty


"Historical fiction"^^xsd:stringr @en
rdfs:label
dbr:Historical_fiction
rdf:type
dbo:MusicGenre
"Fantasy"^^xsd:stringr @en
rdfs:label
dbr:Fantasy
rdf:type
rdf:type
xsd:short
"2014-08-09"^^xsd:date
dbp:firstAired
dbo:genre
dbo:genre
owl:Class
rdf:type
rdfs:range
dbo:numberOfEpisodes
"67"^^xsd:short
dbr:Outlander
rdf:type
dbo:TelevisionShow
rdfs:domain
dbo:numberOfSeasons
rdf:type
"5"^^xsd:short
dbo:numberOfSeasons
owl:DataTypeProperty
dbp:channel
rdf:type
rdfs:domain
rdf:type
dbp:name
"Starz"^^xsd:string@en
dbr:Starz
dbo:TelevisionStation
rdfs:range
dbp:channel
rdf:type
owl:ObjectProperty

Correct answer:

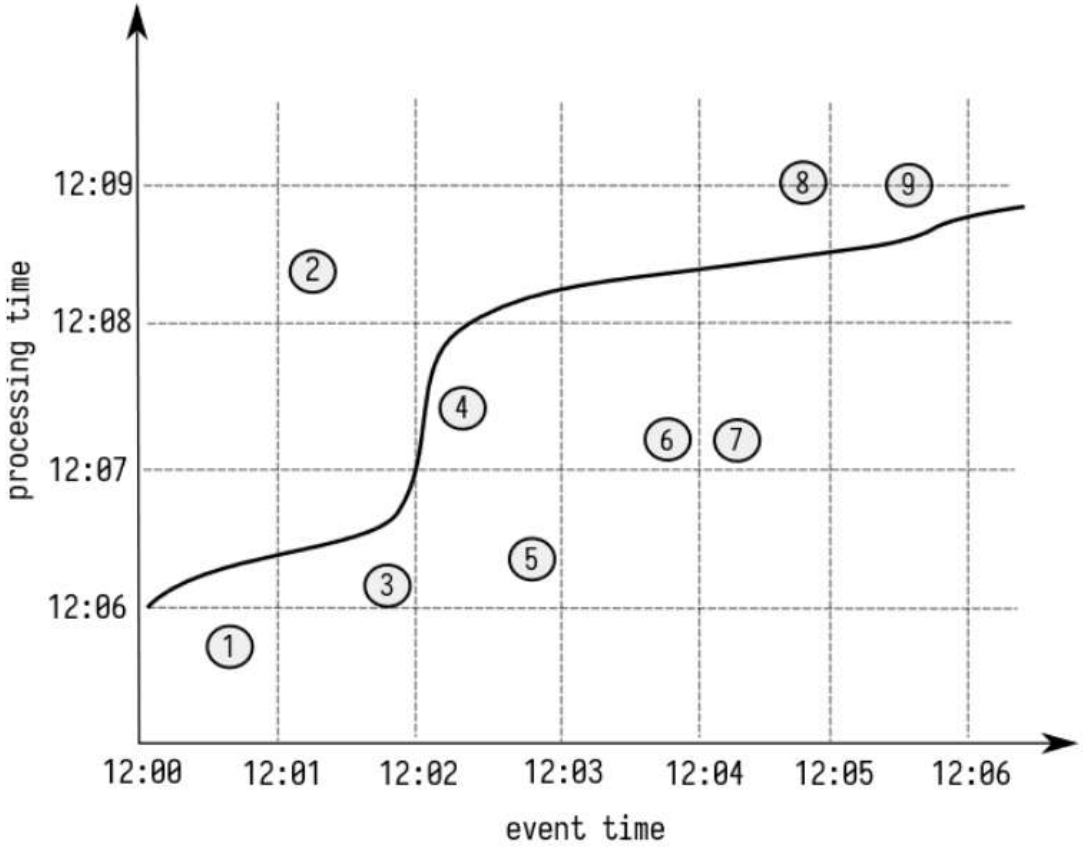
(a) Dva pristupa:

1. **Direktna komunikacija** od proizvođača do potrošača
    - moguć gubitak poruka, potrebno je napraviti vlastito aplikacijsko rješenje da se to spriječi ako se želi
    - npr. ZeroMQ
2. **Posrednički sustavi**
    - Posrednik je poslužitelj, a proizvođači i potrošači klijenti
    - Tipično asinkrona komunikacija, proizvođač samo preda poruku posredniku, koja naknadno biva dostavljena potrošaču
    - Ovdje imaju dva podpristupa:
        1. Message Broker/Queue
            - postoje i dva standarda: AMQP, JMS
            - npr. RabbitMQ
        2. Log-based
            - Proizvođači dodaju u dnevnik
            - Potrošači čitaju (ali ne brišu)
            - Radi performansi - dnevnik se razlaže u particije (i logički - teme)

(b) Moguće je:

- Odbaciti poruke
- Spremati poruke u red (dokad?)
- Kočiti proizvođače (backpressure, flow control), npr. unix pipes
- (kombinacija pristupa)

Na slici je prikazan slijed događaja u *event/processing time* domeni i heuristički *watermark*, pri čemu je vrijednost događaja koja nas zanima označena u krugu i radi jednostavnosti poprima vrijednosti 1-9:



Definiran je sljedeći *EOL (early/on time/later firings) trigger* (primijetite model akumulacije):

```
PCollection<KV<Team, Integer>> totals = input
  .apply(Window.into(FixedWindows.of(TWO_MINUTES))
     .triggering(
        AfterWatermark()
        .withEarlyFirings(AlignedDelay(ONE_MINUTE))
        .withLateFirings(AfterCount(2)))
     .discardingFiredPanes())
  .apply(Sum.integersPerKey());
```

Navedite što i kada se emitira kao n-torke koristeći sljedeću nomenklaturu:

`(processing_time, window, {E|O|L}, value)`

Npr. ako je u ponoć emitirana on-time vrijednost 7 za prozor 23:50-23:55 onda pišite:

`(00:00, 23:50-23:55, O, 7)`

Correct answer:

```
(12:06, 12:00-12:02, E, 1)
(12:07, 12:00-12:02, O, 3)
(12:07, 12:02-12:04, E, 5)
(12:08, 12:02-12:04, E, 10)
(12:08, 12:04-12:06, E, 7)
(12:09, 12:04-12:06, L, 17)
```

što odgovara sljedećoj slici (koju oni naravno nisu trebali predati):