Modellazione ed Architetture Avanzate di Database

Simone Cullino & Roger Ferrod

Analisi della polarità di recensioni di prodotti Amazon

PostgreSQL mongoDB APACHE Spark amazon

1. INTRODUZIONE

INTRODUZIONE

amazon

Lo scopo del progetto è quello di:

- Utilizzare un DB NoSQL (MongoDB)
- Utilizzare un DB Relazionale SQL (Postgres)
- Analizzare la polarità di recensioni di prodotti di Amazon e confrontarle con il numero di stelle attribuite (Gold Value) usando un indice di correlazione (Pearson)
- Confrontare le performance delle soluzioni DBMS adottate

2. ANALISI DELLA POLARITA' – VADER

ANALISI DELLA POLARITA' - VADER

Abbiamo deciso di utilizzare, come algoritmo di analisi della polarità **VADER**, un metodo proposto da *C.J Hutto*.

L'algoritmo attribuisce ad ogni frase un valore numerico racchiuso in un intervallo <-1,1>

Media aritmetica tra score del titolo e score del corpo della recensione

ANALISI DELLA POLARITA' – VADER

Le risorse lessicali utilizzate, fornite sia dall'autore che dal corso, sono:

- **Lexicon.txt**: 7517 parole ed emoticons annotate manualmente con i relativi punteggi di polarità
- **Modifiers.txt**: 66 parole annotate, tra avverbi e aggettivi che modificano la polarità (ex little)
- **Negate.txt**: 58 forme di negazione
- **Slang.txt** : 61 slangs con descrizione
- **Specials.txt** : 5 modi di dire annotati (ex: yeah right)

ANALISI DELLA POLARITA' – VADER

L'algoritmo presenta 5 euristiche che modificano (con un fattore di scala empirico) il valore di polarità della frase:

- **Punteggiatura**: incrementa l'intensità senza modificare la semantica (ES: this is great!!!)
- **Lettere maiuscole**: una parola tutta maiuscola ha più importanza (ES: very USEFUL)
- **Modificatori**: modificatori presenti in modifiers.txt
- **Congiunzione "but"**: può invertire la polarità
- **Analisi 3-grammi**: permette di catturare le negazioni e invertire la polarità

3. DATASET

DATASET

Recensioni di Amazon da *Julian McAuley, University of California San Diego*.

Il Dataset consiste in due file:

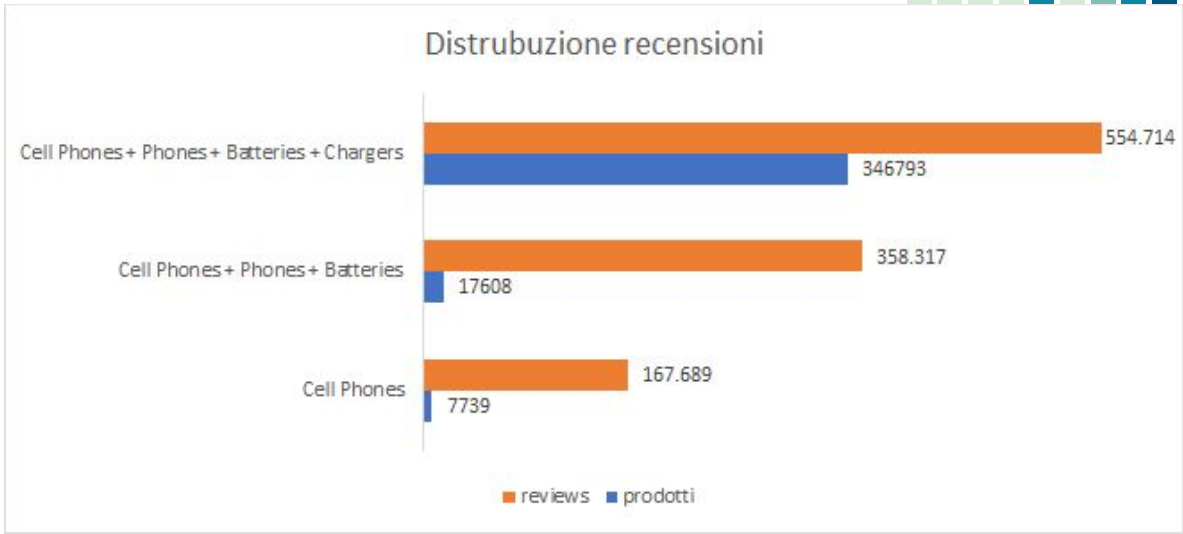
- **raw review data** (18gb) - 59.12 million **reviews**
- **metadata** (3.1gb) - metadata for 9.4 million **products**

L'analisi della polarità si concentra sulla categoria di prodotti: **Cell-Phones & Accessories** riducendo quindi il numero di prodotti a 350.312

DATASET

Abbiamo effettuato una serie di analisi per capire quale fosse il metodo migliore per selezionare un ulteriore sottoinsieme del dataset (per ovviare a problemi di risorse del DB centralizzato) evidenziando che :

- Non conviene selezionare per brand poiché l'81% dei prodotti non ha un brand associato.
- Conviene utilizzare le sotto categorie (nel nostro caso *CellPhones, Phones, Battery, Chargers*)



4. DATABASES

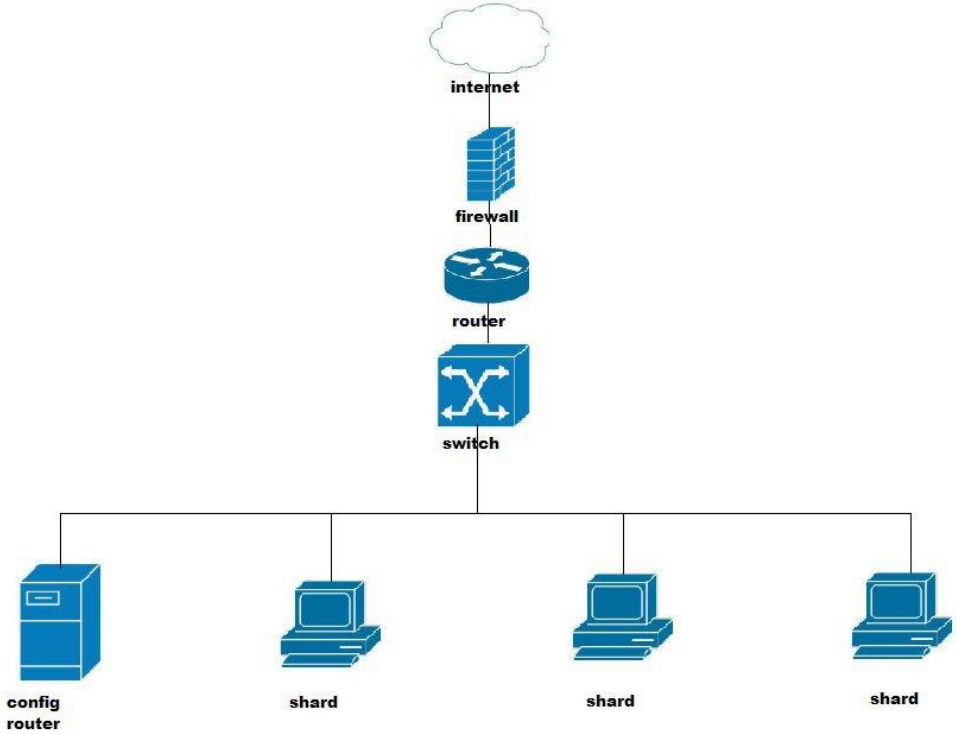
DATABASES - MONGODB

Cluster composto da 4 nodi:

- **Master** : ruolo di **server config** e **router**
- **Shard 1** : nodo sharding (contiene i dati)
- **Shard 2** : nodo sharding (contiene i dati)
- **Shard 3** : nodo sharding (contiene i dati)

Per semplicità e risorse di calcolo limitate abbiamo deciso di non replicare i dati, inoltre il *server config* è unico *(one-point of failure)* e risiede sulla stessa macchina del *router*.

DATABASES - MONGODB



DATABASES - POSTGRES

Database relazionale composto da 2 tabelle :

- **Products** : contiene i dati relativi ai prodotti presi in considerazione.
- **Reviews** : contiene i dati relativi alle recensioni dei prodotti della tabella products.

Inoltre abbiamo deciso di eseguire l'elaborazione sui DB Relazionali in 2 modi, usando :

- **Librerie JDBC** : implementando i metodi del DB con un approccio "classico" usando Java (server centralizzato)
- **Spark** : usando il framework spark per interrogare ed elaborare le tabelle del DB in modo partizionato e distribuito (su cluster)

5. SPARK

SPARK

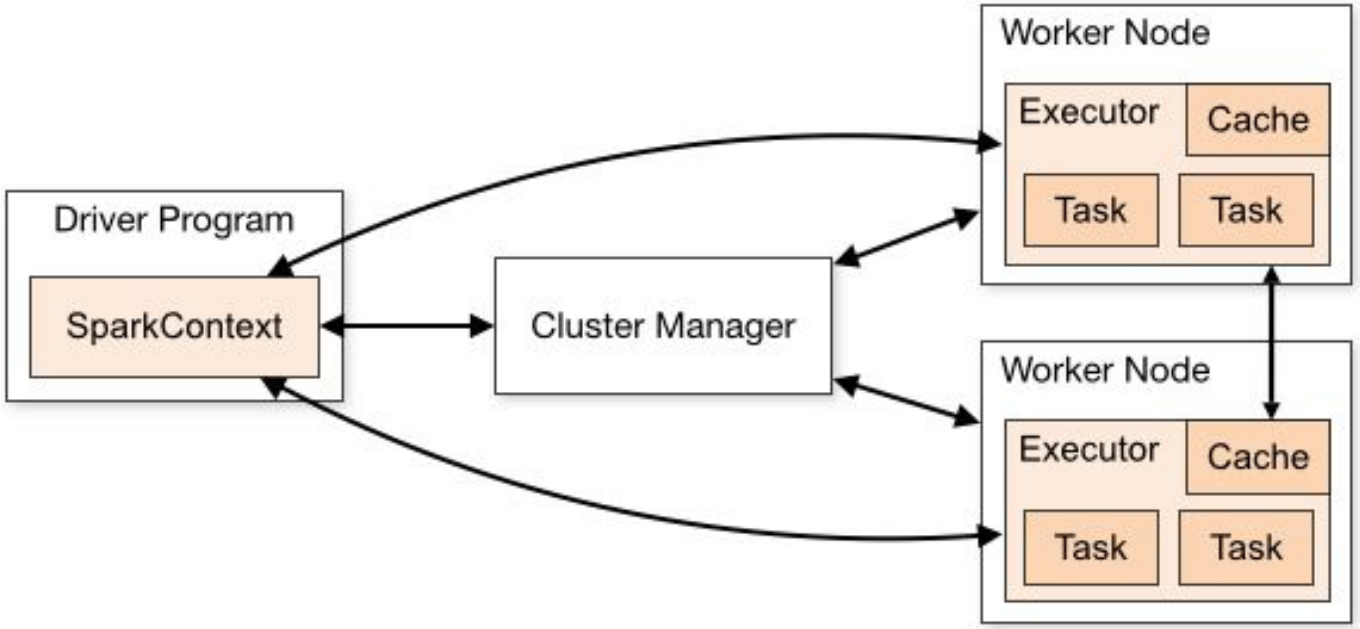
Si tratta di un framework pienamente compatibile con MongoDB e Postgres.

- Usato poiché le funzioni Map/Reduce di MongoDB sono implementate in Javascript, linguaggio non adatto ad applicazioni complesse.
- Permette di parallelizzare i calcoli anche con DB Relazionali.
- Fornisce numerose altri funzionalità (ES count, filter, groupBy)
- Gestisce il calcolo il più possibile in memoria centrale rivelandosi più efficiente di altri framework che utilizzano l'approccio Map/Reduce
  (prestazioni fino a 100 volte migliori per talune applicazioni)

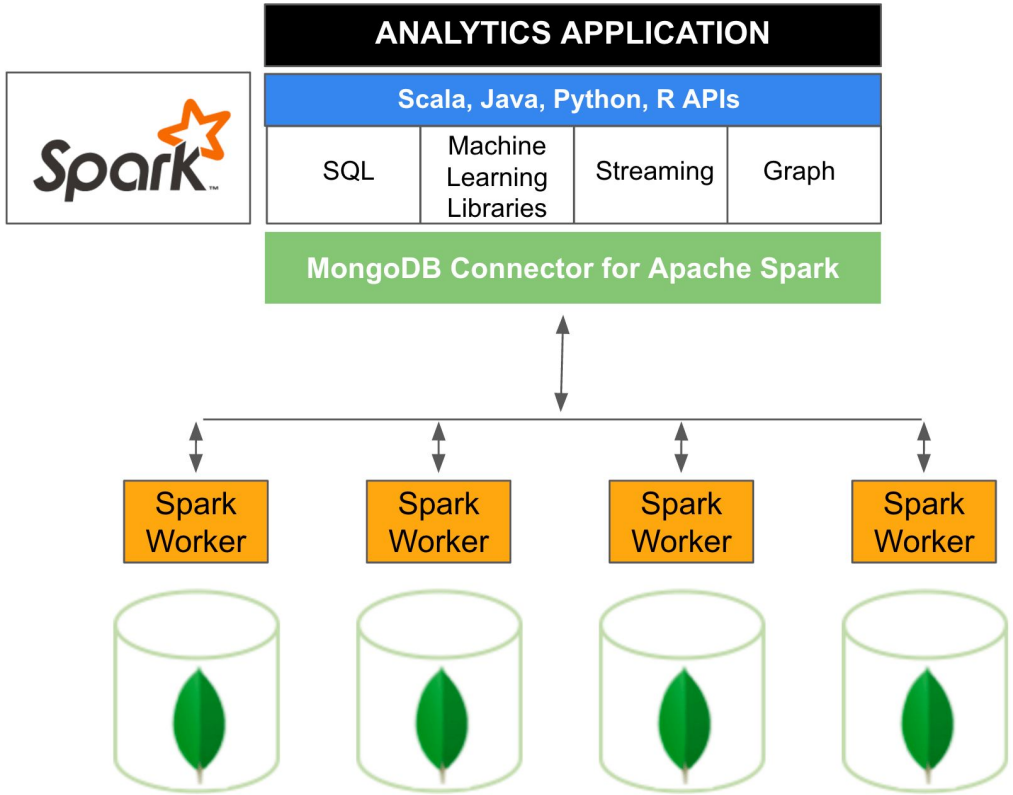
SPARK

- RDD: unità base di lavoro, è una variabile distribuita di dati omogenei
- Trasformazioni: operazioni che applicano una funzione ad un RDD creandone uno nuovo (ES map, filter)
- Azioni: operazione che riduce un RDD ad una variabile Java, ossia non distribuita (ES reduce, count)
- Dataset e Dataframe: strutture dati pseudo-relazionali, distribuite, contenenti rispettivamente oggetti arbitrari e oggetti Row
- Valutazione lazy
- Possibile richiedere caching su disco, RAM o entrambi

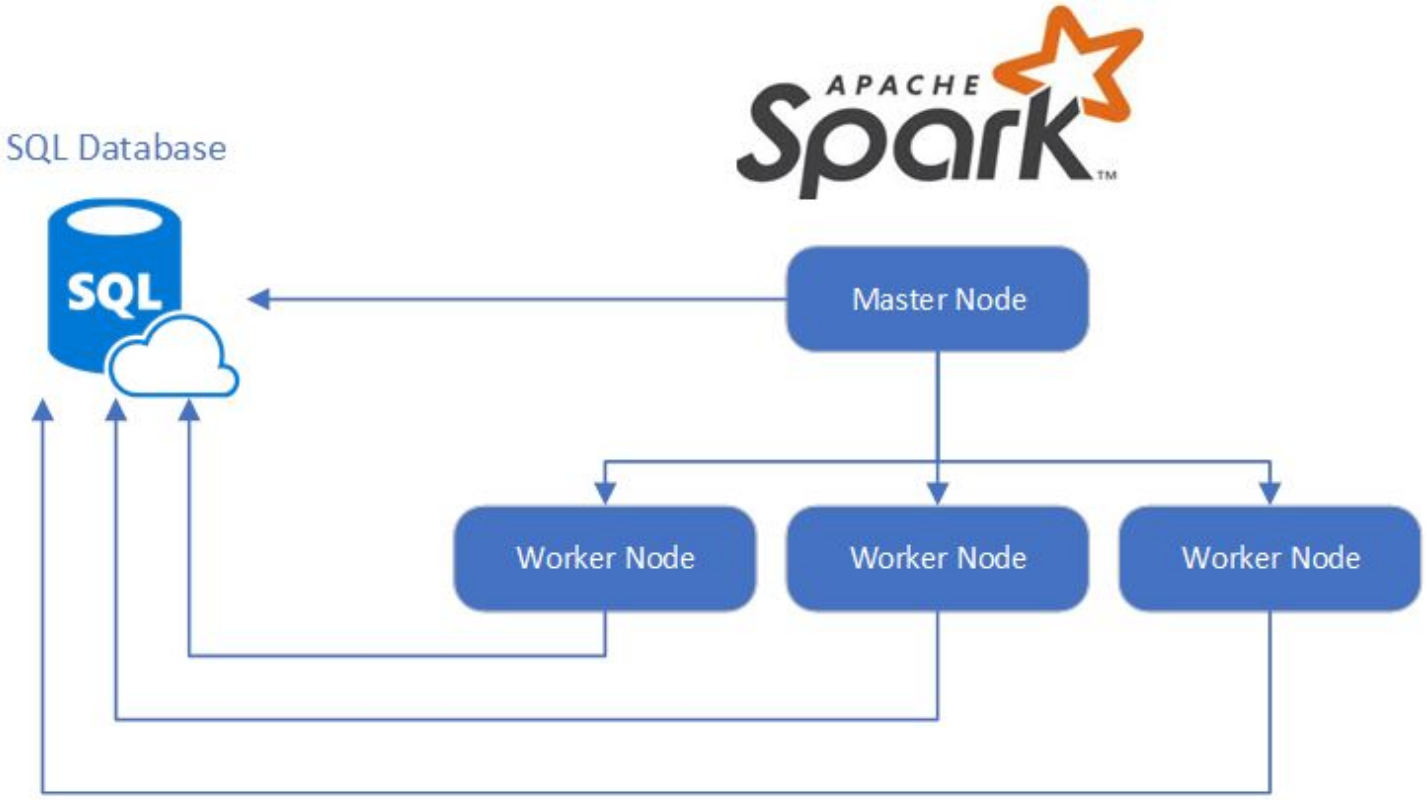
SPARK


Worker Node
Executor
Cache
Task
Task
Driver Program
SparkContext
Cluster Manager
Worker Node
Executor
Cache
Task
Task

INTEGRAZIONE SPARK-MONGODB



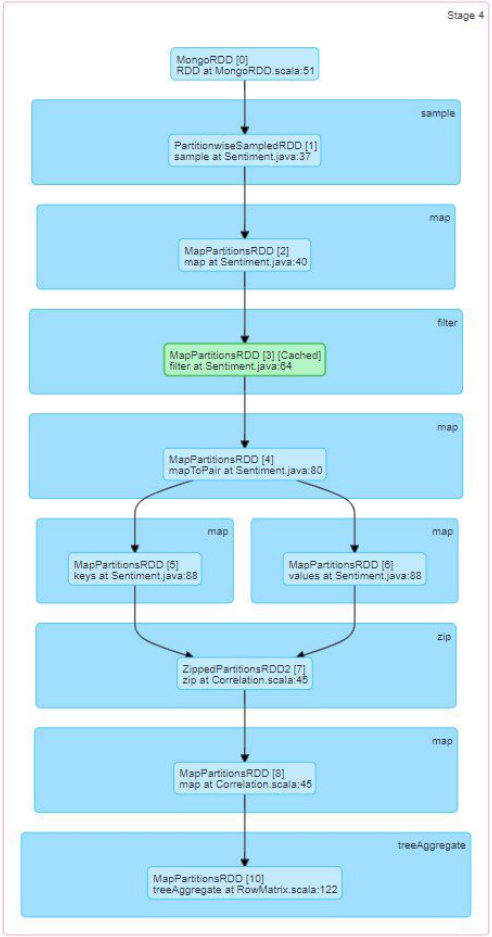
INTEGRAZIONE SPARK-POSTGRES

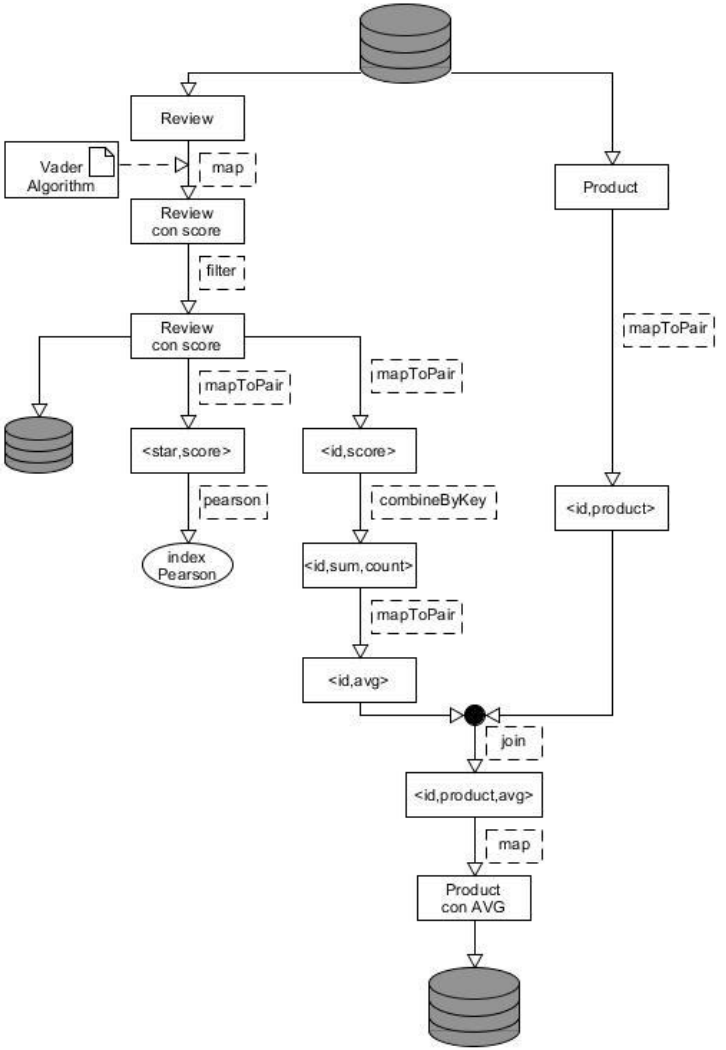


6. DIAGRAMMI DI FLUSSO

DIAGRAMMI DI FLUSSO NOSQL

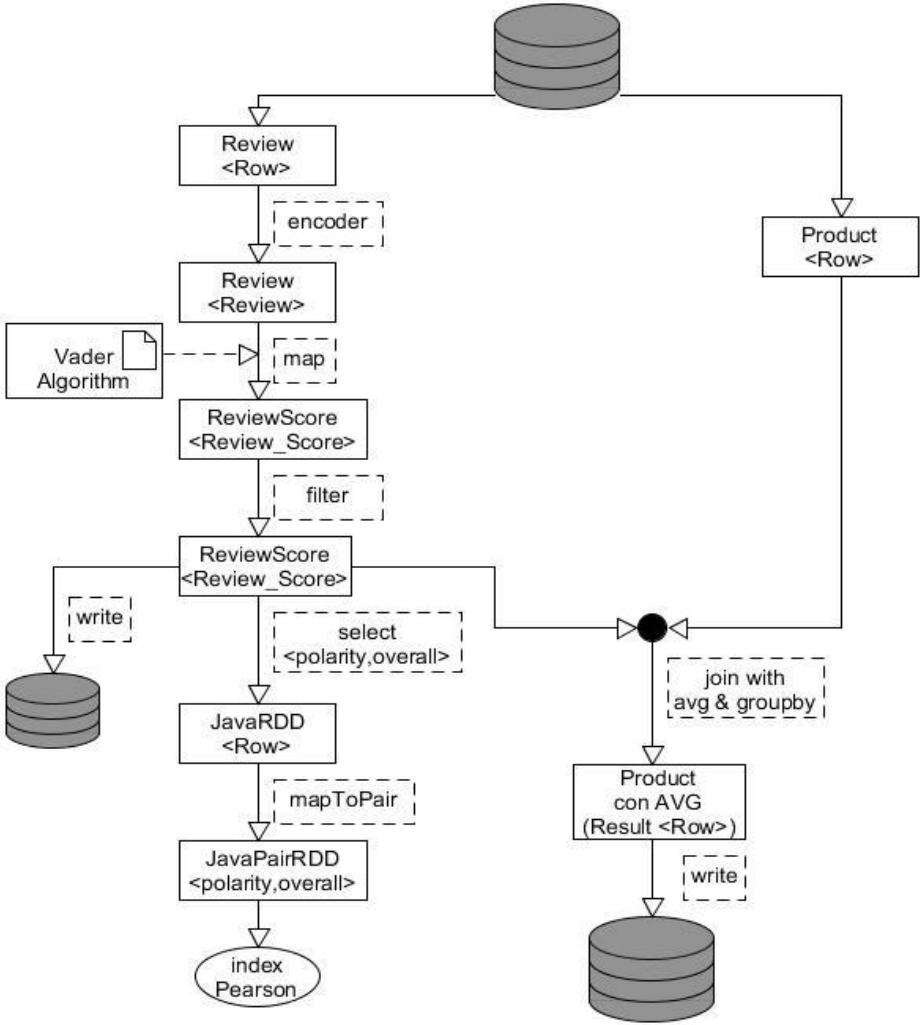
Nella figura a sinistra è riportato il diagramma di flusso generato da spark (ogni nodo corrisponde ad un task), nella figura a destra invece è riportato il diagramma di flusso del codice





DIAGRAMMI DI FLUSSO POSTGRES-SPARK

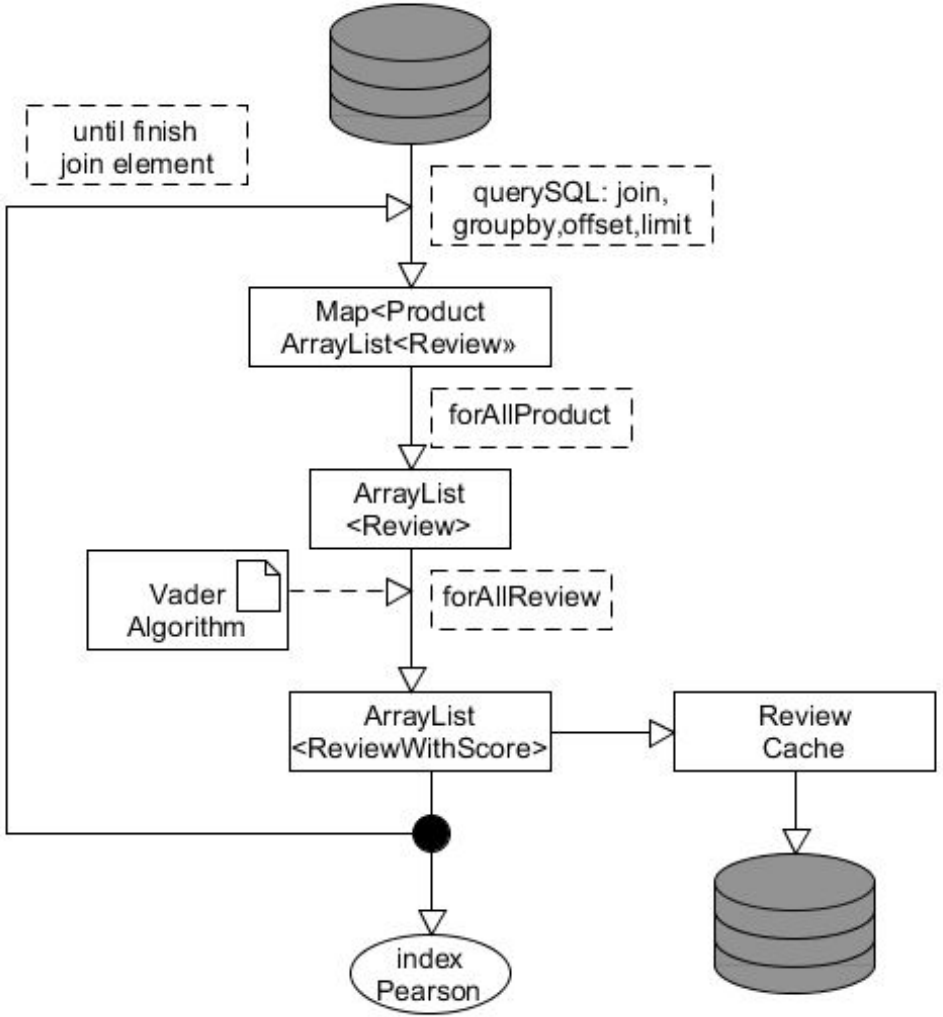
Diagramma di flusso del codice implementato su POSTGRES facendo uso della piattaforma SPARK



DIAGRAMMI DI FLUSSO POSTGRES-JDBC

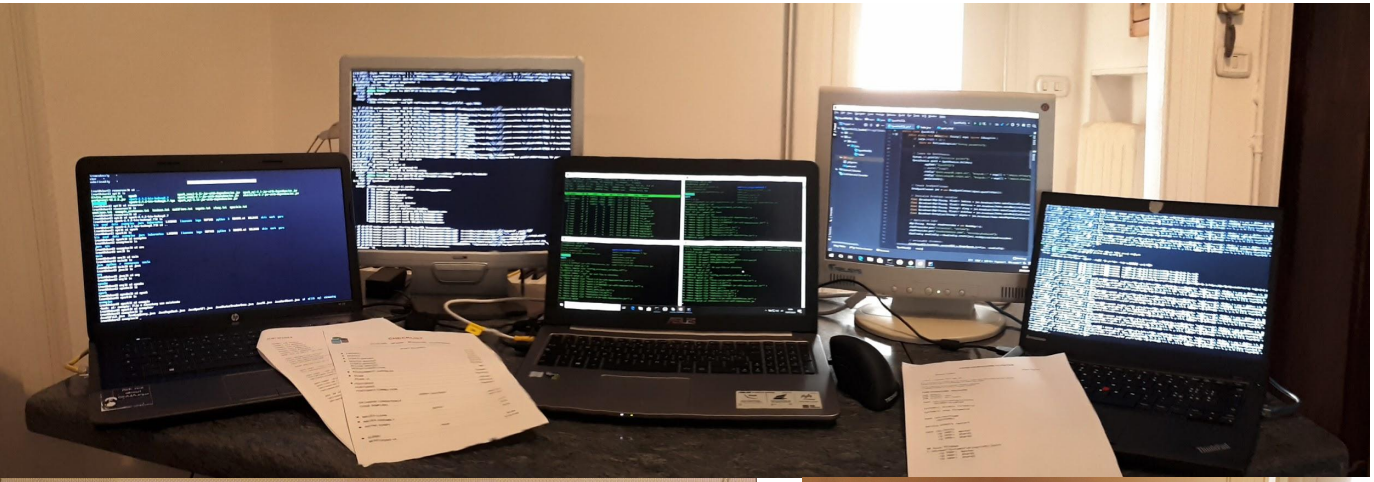
Diagramma di flusso del codice implementato su POSTGRES facendo uso solamente della libreria JDBC

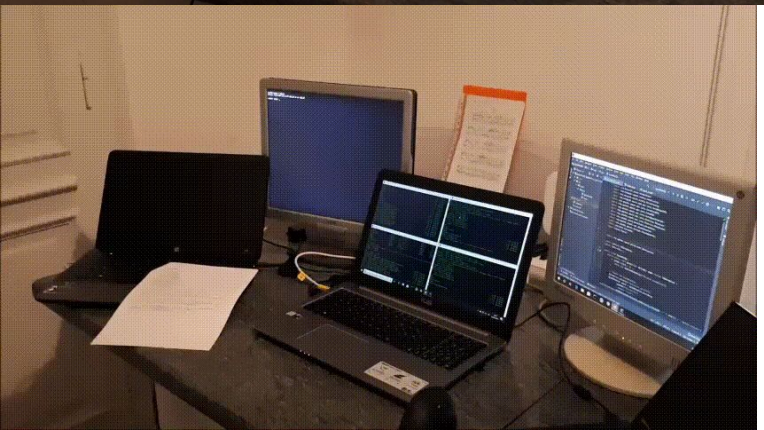
Grazie al sistema di caching implementato è stato possibile ridurre dell' 88% il tempo d'esecuzione

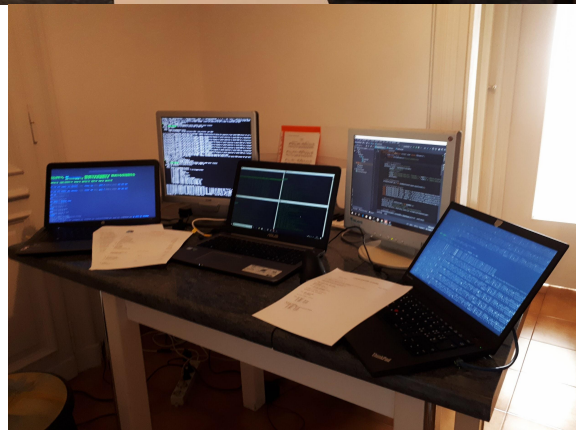


7. ESECUZIONE

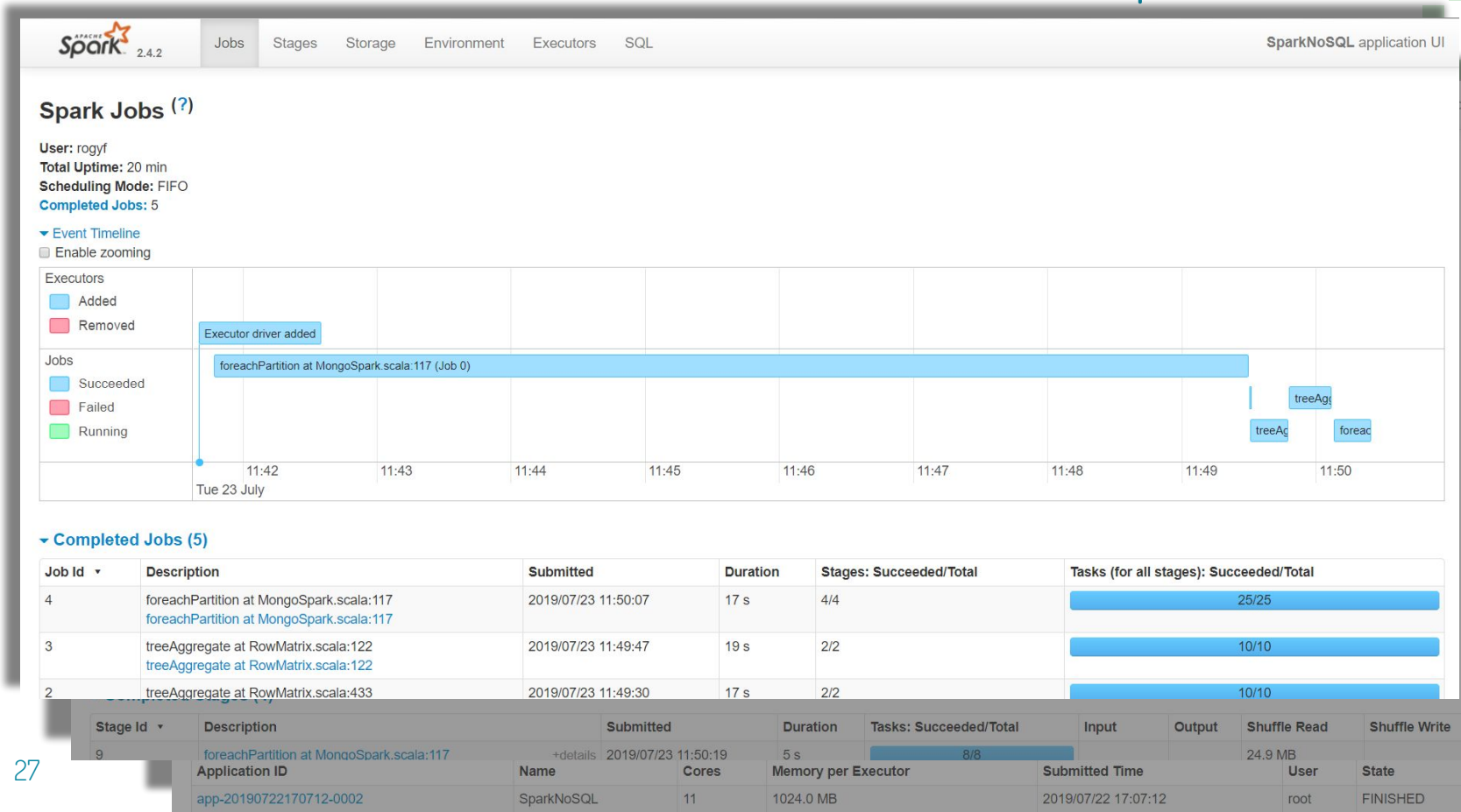
ESECUZIONE - IL CLUSTER







ESECUZIONE - SPARK NOSQL



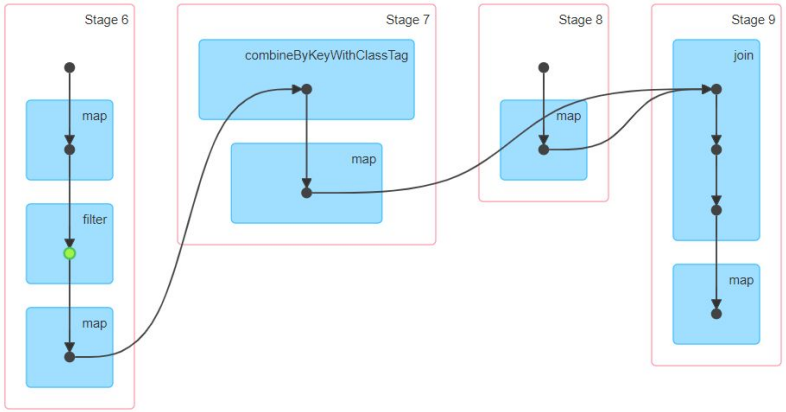
ESECUZIONE - SPARK NOSQL

Spark 2.4.2 | Jobs | Stages | Storage | Environment | Executors | SQL | SparkNoSQL application UI

Details for Job 4

Status: SUCCEEDED
Completed Stages: 4

▸ Event Timeline
▾ DAG Visualization

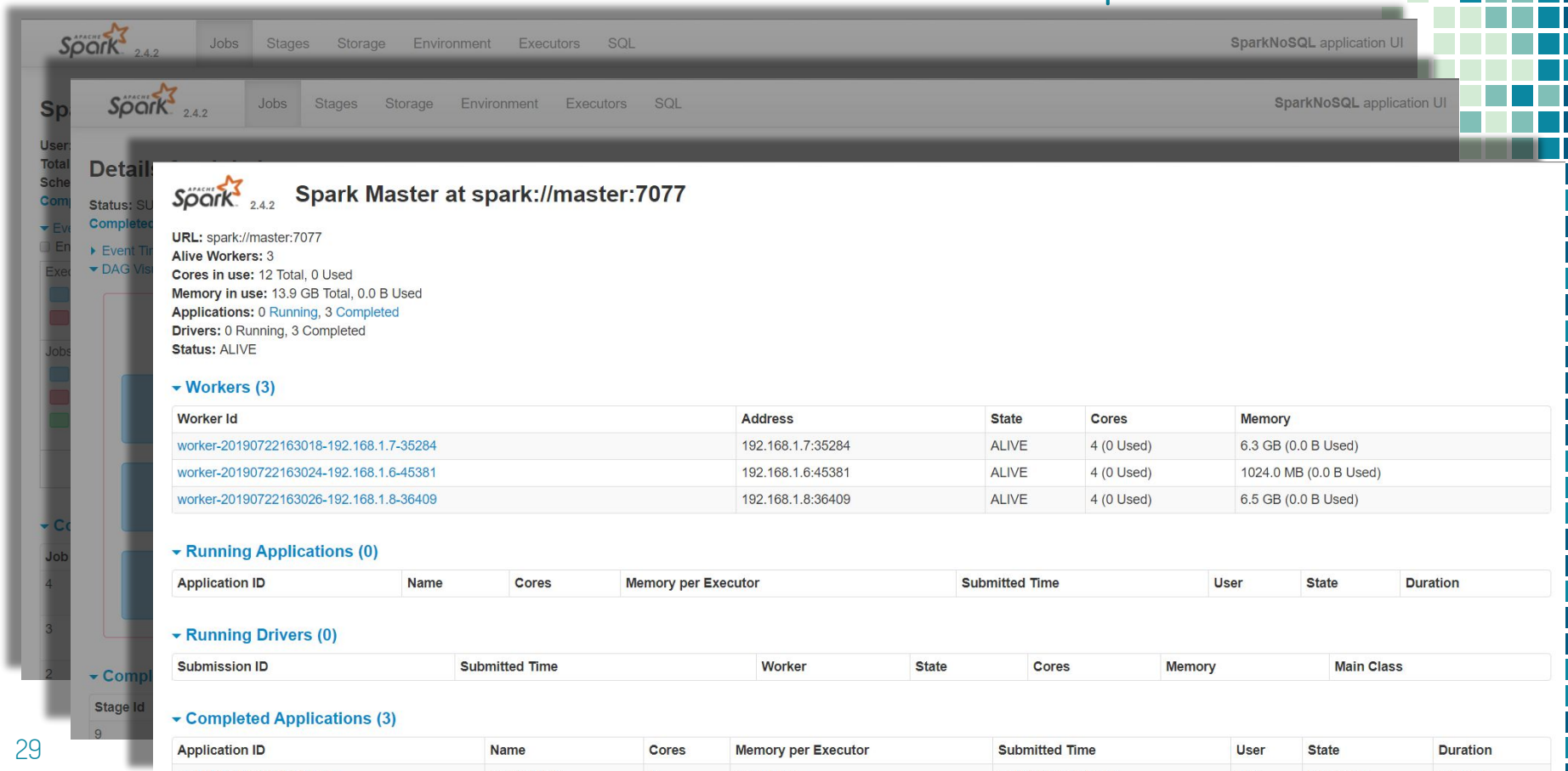


Completed Stages (4)

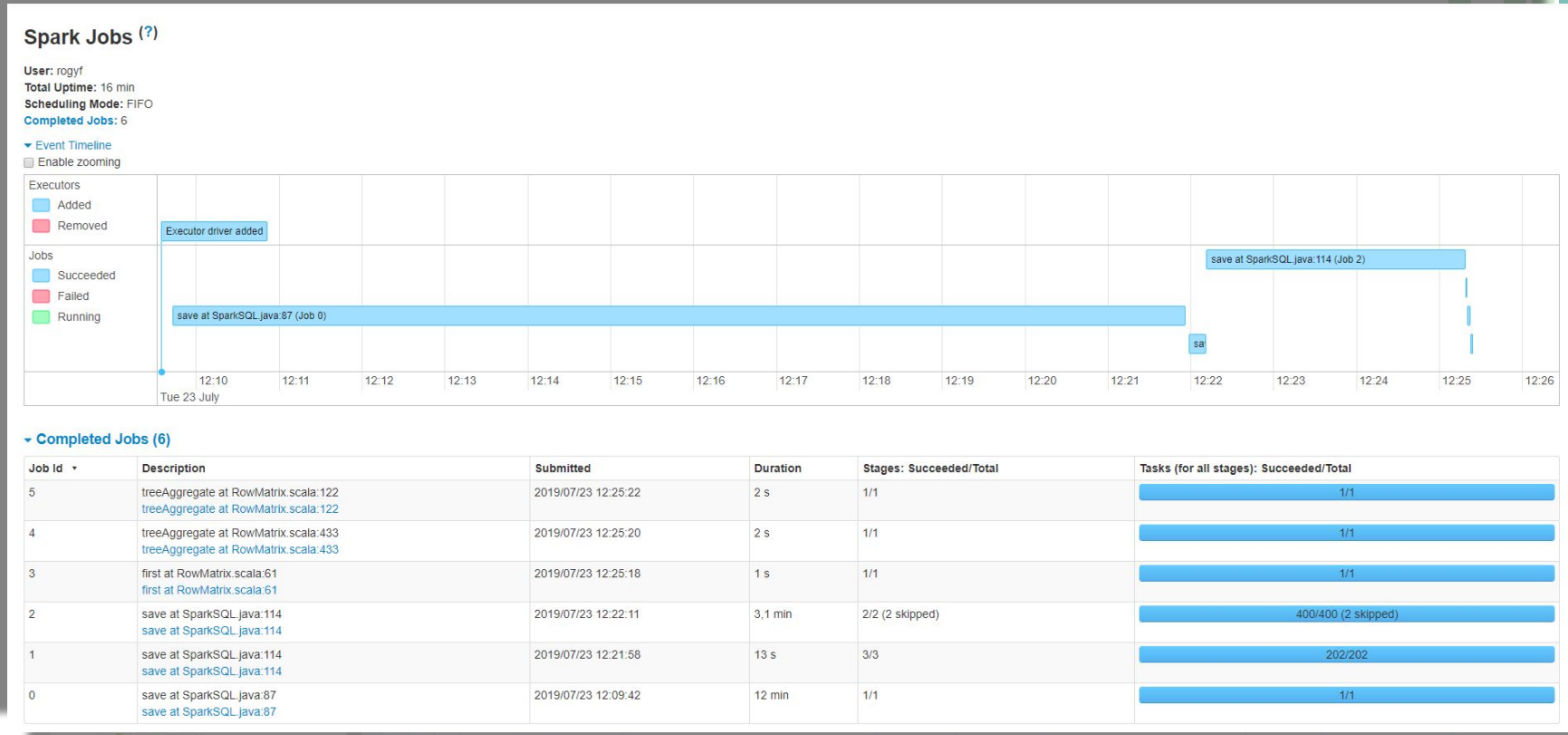
Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
9	foreachPartition at MongoSpark.scala:117 +details	2019/07/23 11:50:19	5 s	8/8			24.9 MB	

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
app-20190722170712-0002	SparkNoSQL	11	1024.0 MB	2019/07/22 17:07:12	root	FINISHED	4.5 min

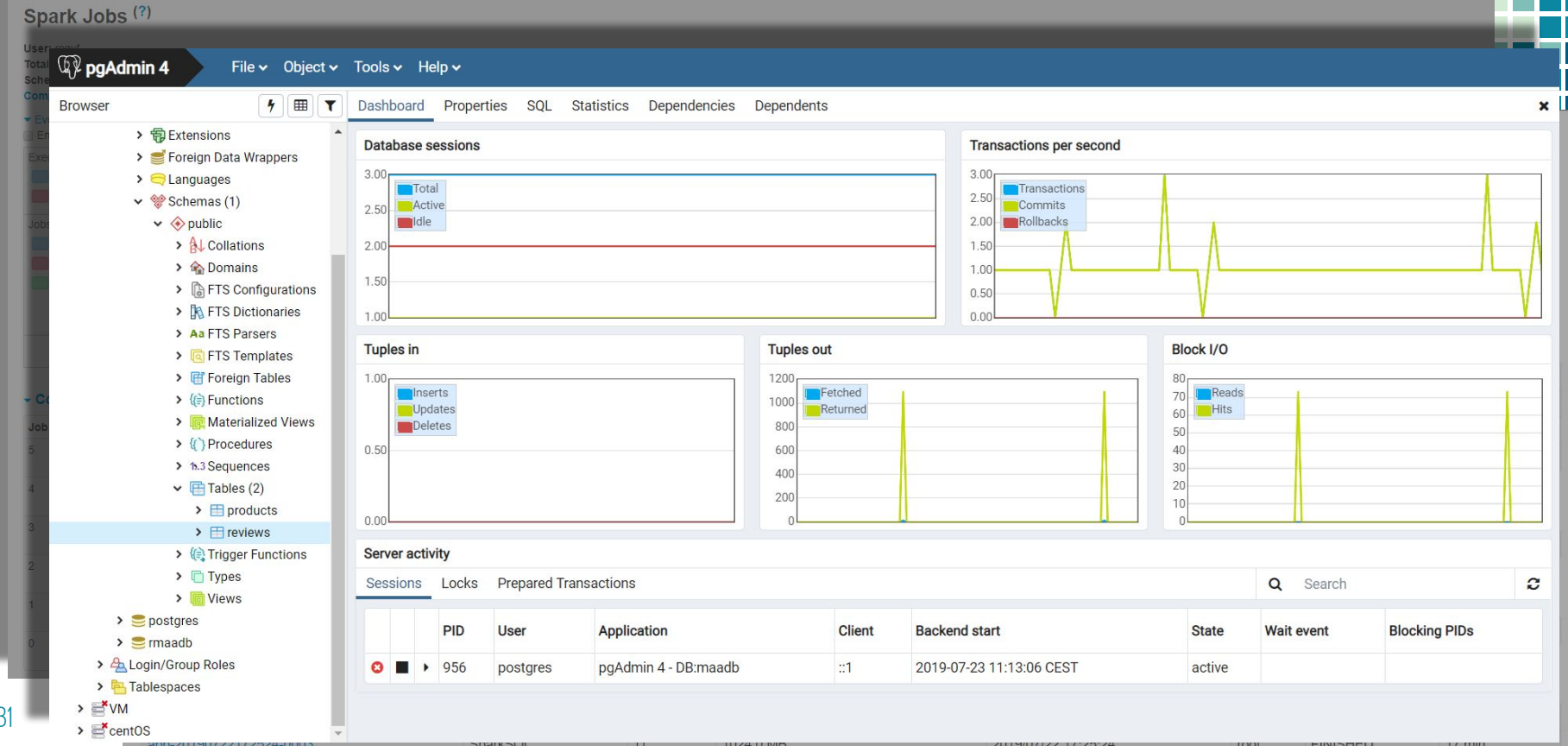
ESECUZIONE - SPARK NOSQL



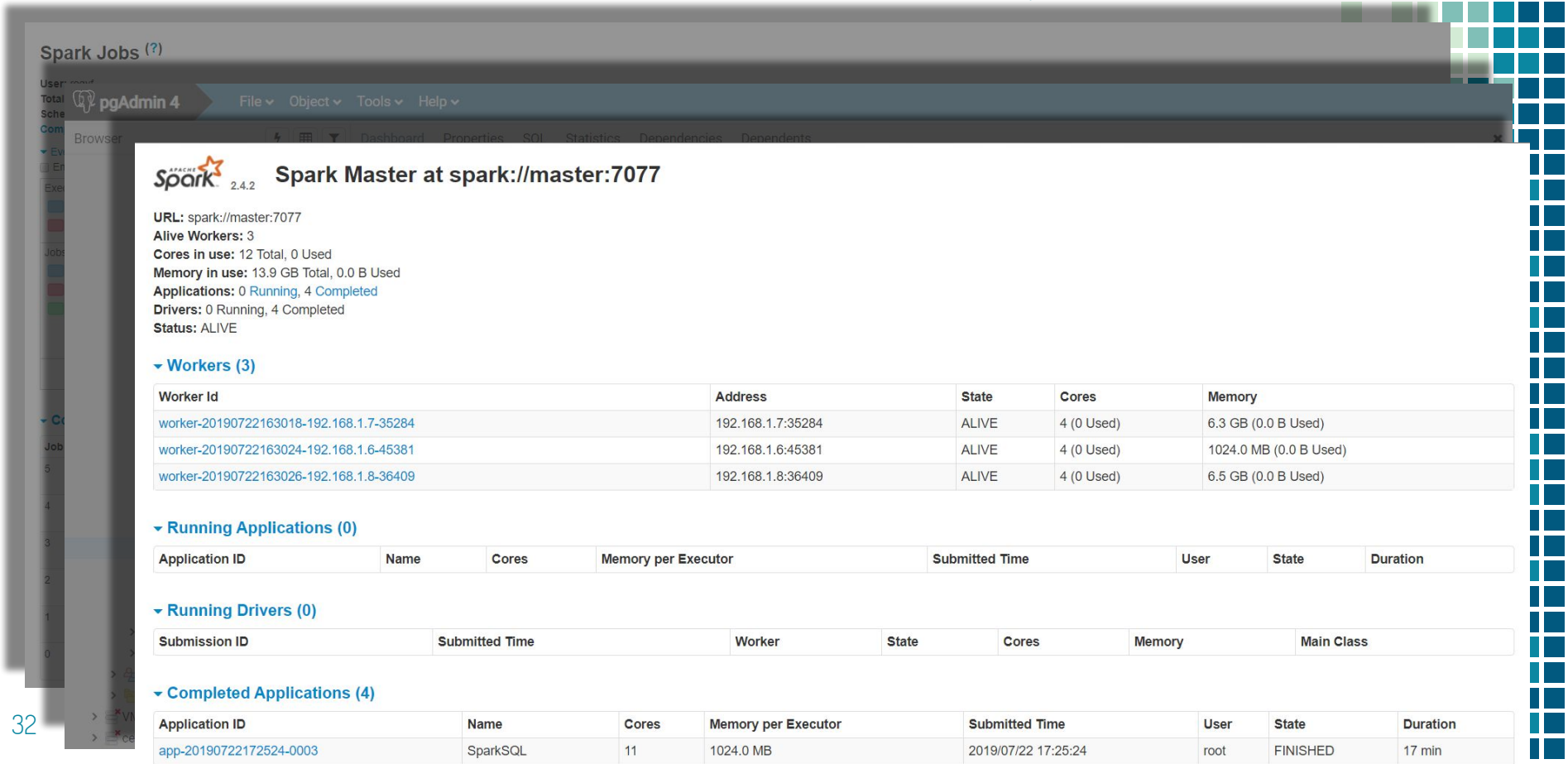
ESECUZIONE - SPARK SQL



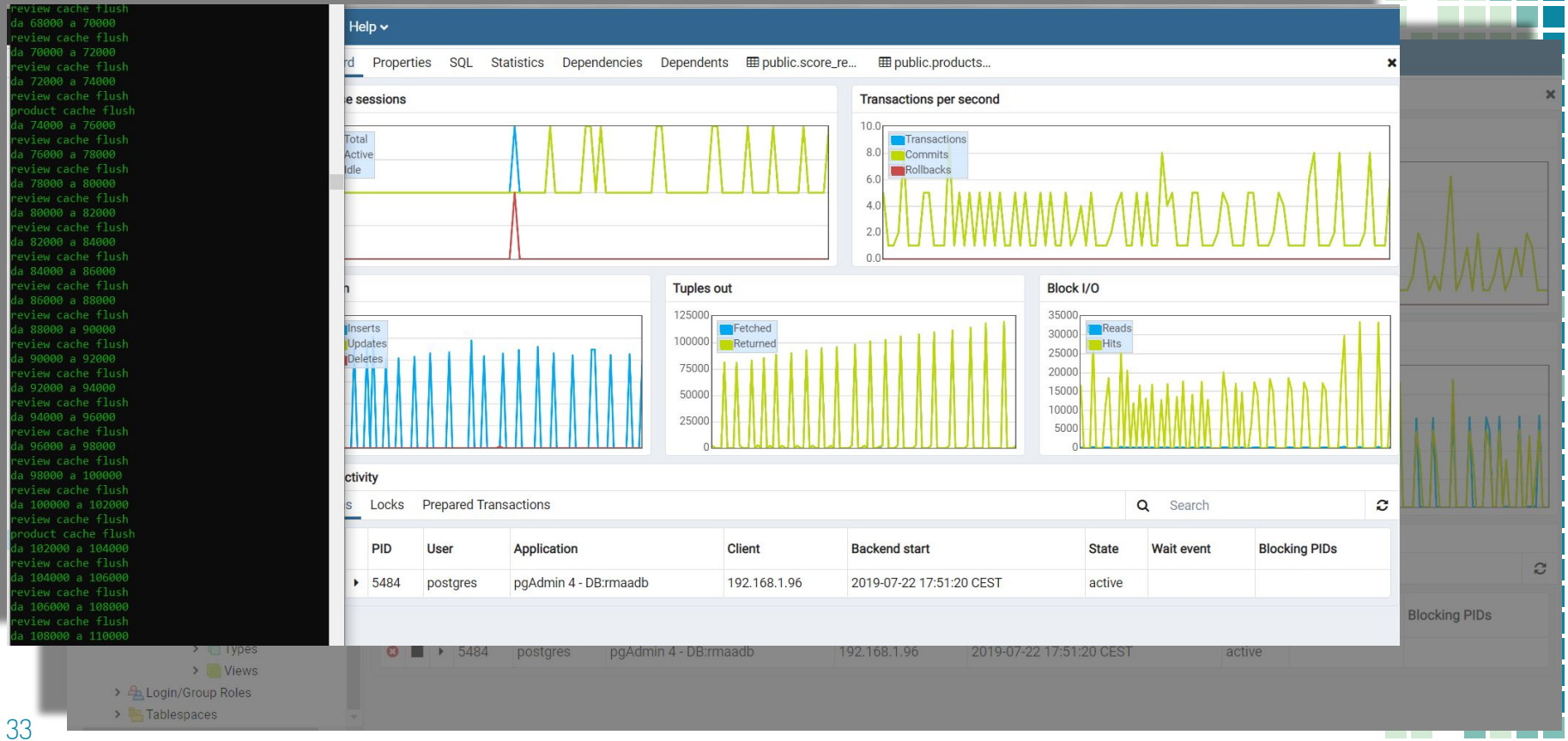
ESECUZIONE - SPARK SQL



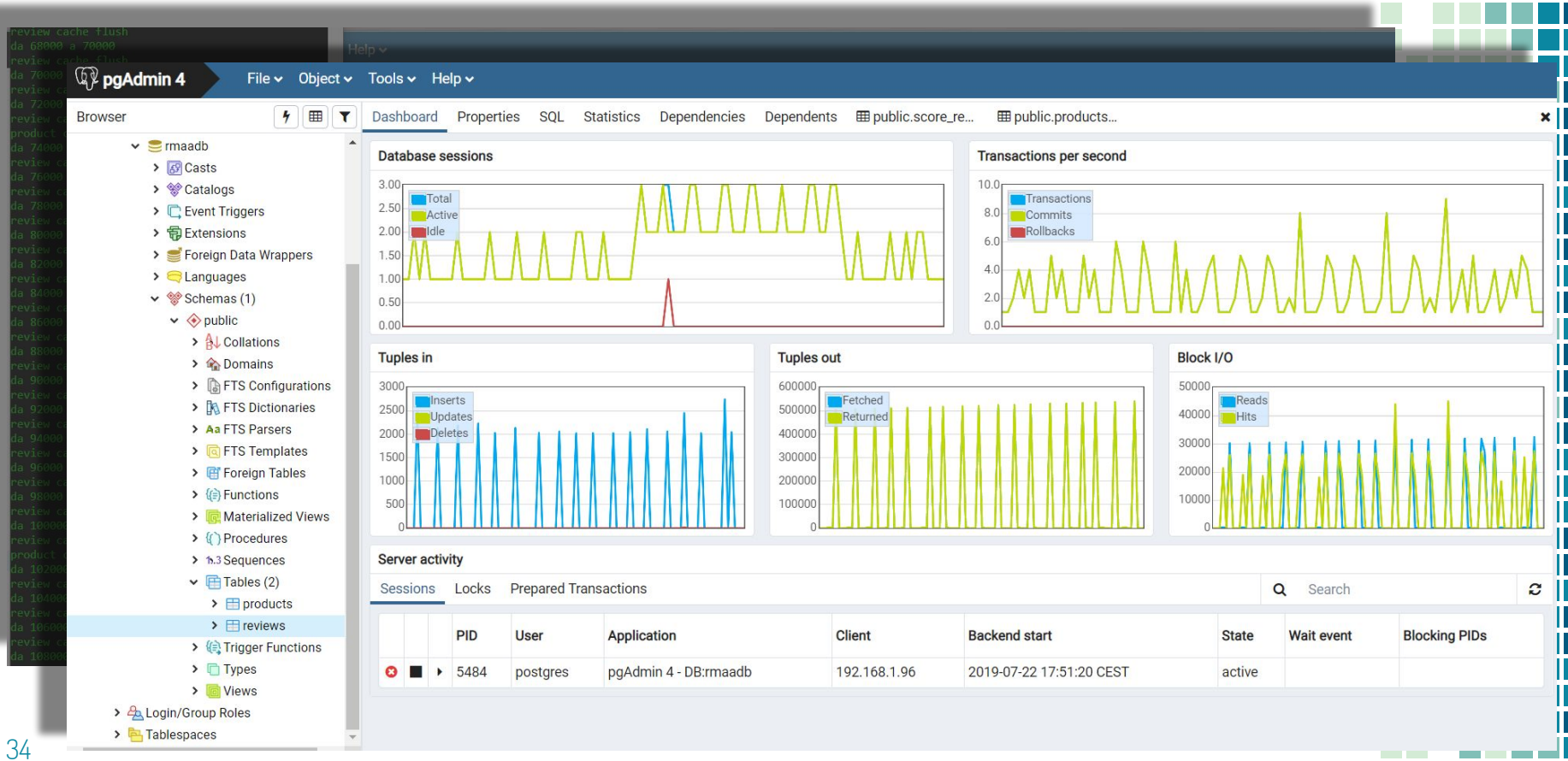
ESECUZIONE - SPARK SQL



ESECUZIONE - JDBC ONLY



ESECUZIONE - JDBC ONLY



7. RISULTATI

RISULTATI

I risultati che seguono si riferiscono alle seguenti configurazioni:

- Server centralizzato (Windows)
  - Intel Core i7 3.1 GHz RAM 12 GB

- Cluster di 4 nodi (CentOS)
  - Intel (VM) 3.1 GHz RAM 2 GB (worker - shard)
  - Intel Core i5 2.4 GHz RAM 8 GB (worker - shard)
  - Pentium Dual 1.6 GHz RAM 2 GB (master)
  - Intel Core i3 2.4 GHz RAM 8 GB (worker - shard)

Tot 12 core, 13.9 GB memoria, 3 worker, 3 shard

RISULTATI - LOCALE

L'esecuzione locale simula un cluster utilizzando diversi thread e processi, si tenga presente che il processore ha 2 core con 4 thread ciascuno (possibile simulare 8 core)

Soluzione	**Tempo esecuzione**
Spark-MongoDB	8 min
Spark-PostgreSQL	16 min
JDBC only	9 min

RISULTATI - CLUSTER

Esecuzione su cluster utilizza 11 core e 1024 MB per Executors (esecutori del codice nei nodi del cluster)

Soluzione	**Tempo esecuzione**
Spark-MongoDB	4 min
Spark-PostgreSQL	17 min
JDBC only	29 min

RISULTATI - PEARSON

I coefficiente di correlazione Pearson tra le stelle attribuite dall'utente (considerato come gold-value) nella recensione e la polarità calcolata mediante l'algoritmo VADER è pari a 0.65

Il risultato si potrebbe migliorare integrando risorse esterne (ad esempio *WordNet, BabelNet, ConceptNet*) per comprendere ironia e sarcasmo

CREDITS

Dataset:

R. He, J. McAuley. "Modeling the visual evolution of fashion trends with one-class collaborative filtering". WWW, 2016

J. McAuley, C. Targett, J. Shi, A. van den Hengel. "Image-based recommendations on styles and substitutes". SIGIR, 2015

Vader:

Hutto, Clayton J. and Eric Gilbert. "VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text." ICWSM, 2014

THANKS!