

## 作业1:

- q38.sql

```
1 SELECT count(*)
2 FROM (
3     SELECT DISTINCT
4         c_last_name,
5         c_first_name,
6         d_date
7     FROM store_sales, date_dim, customer
8     WHERE store_sales.ss_sold_date_sk = date_dim.d_date_sk
9         AND store_sales.ss_customer_sk = customer.c_customer_sk
10        AND d_month_seq BETWEEN 1200 AND 1200 + 11
11     INTERSECT
12     SELECT DISTINCT
13         c_last_name,
14         c_first_name,
15         d_date
16     FROM catalog_sales, date_dim, customer
17     WHERE catalog_sales.cs_sold_date_sk = date_dim.d_date_sk
18        AND catalog_sales.cs_bill_customer_sk = customer.c_customer_s
19        AND d_month_seq BETWEEN 1200 AND 1200 + 11
20     INTERSECT
21     SELECT DISTINCT
22         c_last_name,
23         c_first_name,
24         d_date
25     FROM web_sales, date_dim, customer
26     WHERE web_sales.ws_sold_date_sk = date_dim.d_date_sk
27        AND web_sales.ws_bill_customer_sk = customer.c_customer_sk
28        AND d_month_seq BETWEEN 1200 AND 1200 + 11
29 ) hot_cust
30 LIMIT 100
```

- 执行结果

```
22/10/17 14:48:55 INFO BlockManager: Submitting ResultStage 101 (MapPartitionsRDD[342] at $anonfun$runPcdsQueries$5 at Benchmark.scala:77), which has no missing parents
22/10/17 14:48:55 INFO MemoryStore: Block broadcast_146_piece0 stored as values (estimated size 9.6 KiB, free 412.6 MiB)
22/10/17 14:48:55 INFO MemoryStore: Block broadcast_146_piece0 stored as bytes in memory (estimated size 4.9 KiB, free 412.6 MiB)
22/10/17 14:48:55 INFO BlockManagerInfo: Removed broadcast_146_piece0 on 10.72.8.77:64404 in memory (size: 28.1 KiB, free: 431.4 MiB)
22/10/17 14:48:55 INFO BlockManagerInfo: Added broadcast_146_piece0 in memory on 10.72.8.77:64404 (size: 4.9 KiB, free: 431.4 MiB)
22/10/17 14:48:55 INFO SparkContext: Created broadcast 146 from broadcast at DAGScheduler.scala:1383
22/10/17 14:48:55 INFO DAGScheduler: Submitting 1 missing tasks from ResultStage 101 (MapPartitionsRDD[342] at $anonfun$runPcdsQueries$5 at Benchmark.scala:77) (first 15 tasks are for partitions Vector(0))
22/10/17 14:48:55 INFO TaskSchedulerImpl: Adding task set 101.0 with 1 tasks resource profile 0
22/10/17 14:48:55 INFO TaskSetManager: Starting task 0.0 in stage 101.0 (TID 137) (10.72.8.77, executor driver, partition 0, MODE_LOCAL, 4453 bytes) taskResourceAssignments Map()
22/10/17 14:48:55 INFO Executor: Running task 0.0 in stage 101.0 (TID 137)
22/10/17 14:48:55 INFO ShuffleBlockFetcherIterator: Getting 0 (240.0 B) non-empty blocks including 4 (240.0 B) local and 0 (0.0 B) host-local and 0 (0.0 B) remote blocks
22/10/17 14:48:55 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms
22/10/17 14:48:55 INFO DataWritingSparkTask: Committed partition 0 (task 137, attempt 0, stage 101.0)
22/10/17 14:48:55 INFO Executor: Finished task 0.0 in stage 101.0 (TID 137), 2528 bytes result sent to driver
22/10/17 14:48:55 INFO TaskSetManager: Finished task 0.0 in stage 101.0 (TID 137) in 5 ms on 10.72.8.77 (executor driver) (1/1)
22/10/17 14:48:55 INFO TaskSchedulerImpl: Removed TaskSet 101.0, whose tasks have all completed, from pool
22/10/17 14:48:55 INFO DAGScheduler: ResultStage 101 ($anonfun$runPcdsQueries$5 at Benchmark.scala:77) finished in 0.020 s
22/10/17 14:48:55 INFO DAGScheduler: Job 56 is finished. Cancelling potential speculative or zombie tasks for this job
22/10/17 14:48:55 INFO TaskSchedulerImpl: Killing all running tasks in stage 101: Stage finished
22/10/17 14:48:55 INFO DAGScheduler: Job 56 finished: $anonfun$runPcdsQueries$5 at Benchmark.scala:77, took 1.701925 s
22/10/17 14:48:55 INFO OverwriteExpressionExec: Data source write support org.apache.spark.sql.execution.datasources.noop.NoopBatchWrite$Bd2b00 is committing.
22/10/17 14:48:55 INFO OverwriteExpressionExec: Data source write support org.apache.spark.sql.execution.datasources.noop.NoopBatchWrite$Bd2b00 committed.
  Stopped after 2 iterations, 5178 ms

Java HotSpot(TM) 64-Bit Server VM 11.0.124-LTS-237 on Mac OS X 11.5
Intel(R) Core(TM) i9-9880H CPU @ 2.30GHz
100% Memory
-----
Best Time(ms)  Avg Time(ms)  StdDev(ms)  Rate(M/s)  Per Row(ms)  Relative
-----
q18              2269          2589          453          2.3          435.1          1.0x

22/10/17 14:48:55 INFO SparkContext: Invoking stop() from shutdown hook
22/10/17 14:48:55 INFO SparkUI: Stopped Spark web UI on http://10.72.8.77:4040
22/10/17 14:48:55 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
22/10/17 14:48:55 INFO MemoryStore: MemoryStore cleared
22/10/17 14:48:55 INFO BlockManager: BlockManager stopped
22/10/17 14:48:55 INFO BlockManagerMaster: BlockManagerMaster stopped
22/10/17 14:48:55 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
22/10/17 14:48:55 INFO SparkContext: Successfully stopped SparkContext
22/10/17 14:48:55 INFO ShutdownHookManager: Shutdown hook called
22/10/17 14:48:55 INFO ShutdownHookManager: Deleting directory /private/var/folders/8s/xvq20w6j0mwfq30dpdq4gr000000gq/T/spark-e801333b-75ca-46ef-a252-37f38a484702
22/10/17 14:48:55 INFO ShutdownHookManager: Deleting directory /private/var/folders/8s/xvq20w6j0mwfq30dpdq4gr000000gq/T/spark-e11c0b4c-b207-4d01-b777-5f4c6d46c2c5
```

- 优化规则

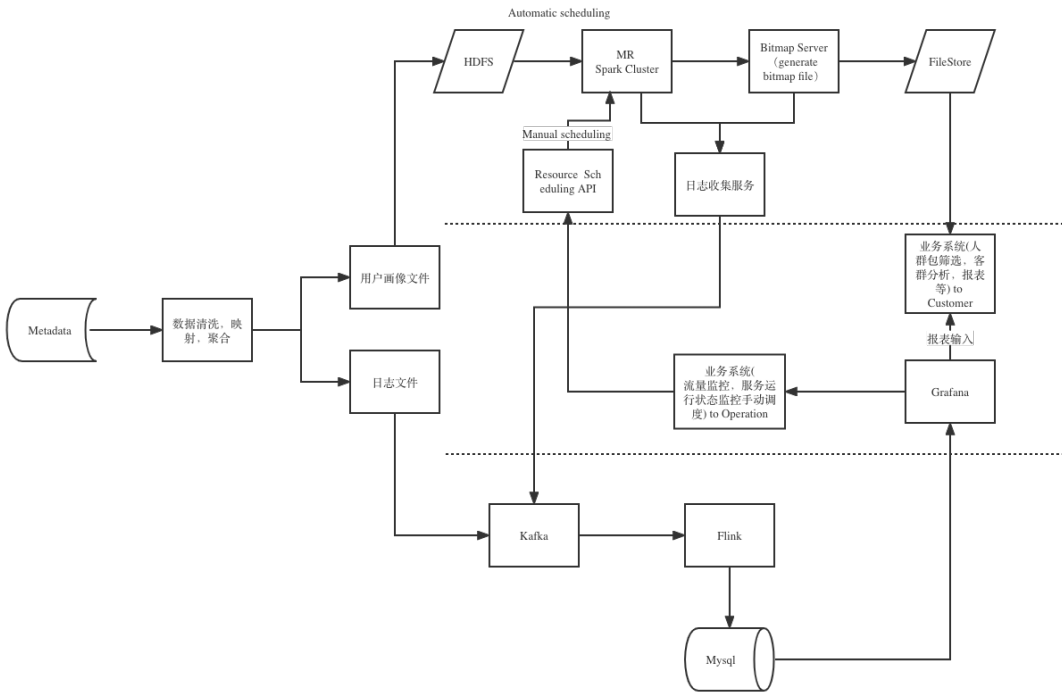
- 1 CollapseProject
- 2 PushDownPredicates
- 3 ConstantFolding
- 4 EliminateLimits
- 5 ColumnPruning
- 6 InferFiltersFromConstraints
- 7 PushDownLeftSemiAntiJoin
- 8 RemoveNoopOperators
- 9 ReorderJoin
- 10 ReplaceDistinctWithAggregate
- 11 ReplaceIntersectWithSemiJoin
- 12 RewritePredicateSubquery

- 优化规则描述

- PushDownPredicates：指谓词下推逻辑优化器，谓词可以下推的前提：不影响查询结果，即要保证下推前和下推后两个sql执行得到的效果相同，包含三条规则：  
CombineFilters，PushPredicateThroughJoin，PushPredicateThroughNonJoin。  
CombineFilters就是把SQL语句中的where过滤条件，尽可能地放在执行计划靠前的地方，尽早地将不必要的数据过滤掉从而提高了整个执行计划的效率；PushPredicateThroughJoin在完成join后，如果对join的结果进行过滤操作，会把过滤条件尽量下推到数据源读取时，减少数据传输和join时的数据量，从而提升性能；PushPredicateThroughNonJoin是指非join情况的谓词下推的优化器，主要适用于Filter节点子节点为Project、Aggregate等情况。
- ConstantFolding：即常量折叠，我们在select 语句中，如果掺杂了一些常量表达式，ConstantFolding 规则会自动地用等效文本值静态计算的结果来替换计算的表达式。例如上述语句中有三个纯常量运算表达式，即 d\_month\_seq BETWEEN 1200 AND 1200 + 11。一旦数据量很大的时候，每行都要计算一次该表达式的值，积少成多就会浪费了很多的时间。所以通过常量折叠可以将它预先这个表达式转化为d\_month\_seq BETWEEN 1200 AND 1211，这样就可以消除很多不必要的重复计算，从而提升sql语句的执行效率。

## 作业2:

## 1. 架构图



## 2.Lambda 的优缺点

Lambda 架构是一种数据处理架构，通过利用批处理和流处理方法来处理大量数据。这种架构方法通过使用离线批处理来提供批处理数据的全面和准确的视图，同时使用实时流处理来提供在线数据的视图来平衡延迟、吞吐量和容错性，两个视图输出可以在呈现之前合并。

Lambda 架构包含三层，分别是 Batch Processing Layer（批处理层）、Speed (Real-Time) Processing Layer（速度处理层）、Serving Layer（服务层）。

优点：

1. 架构比较简单，实时层计算量比较小，计算成本可控，整个计算系统稳定。
2. 在某些情况下，实时计算和离线计算可分开进行，避免计算高峰，能够缓解资源压力。
3. 对数据修正纠错也很友好，能够保证实时层处理出现问题之后整个计算数据不被破坏，可以通过重新运行离线任务，从而很快的将历史数据修正，保证最终一致性。

缺点：

1. 需要同时维护两套系统架构：批处理层和速度层，业务需要的不同需要分别编码维护，数据源的任何变化均涉及到两个部署的修改，任务量大，难以灵活应对。
2. 同时维护两个复杂的分布式系统，并且需要保证他们逻辑上产生相同的结果输出到服务层中，两系统可能同时需要占用资源，对资源需求大。
3. 部署复杂，需要部署离线及实时计算两套系统，给运维造成的负担比较重。
4. 随着数据量的急剧增加、批处理窗口时间内可能无法完成处理，并且对存储也会有巨大的挑战。