Functional Dependencies in a Flight Delay dataset

Dataset Pre-processing

- Filter out rows with Null and "NA" string values.
- Tranform "NA" strings in specific Delay columns to 0's.
- Filter out diverted and cancelled flights.
- Sample 15 million rows.
- Transform column values that include expected departure and arrival times from badly formatted hours and minutes to full timestamps.
- Use the newly transformed timestamp columns together with delay columns to correctly calculate actual departed and arrival times.
- Drop columns that had their information merged with other columns or columns that are assumed to not be part of dependencies.
- Reduced amount of columns 29 -> 19 without loosing much information.

Expected dependencies

Functional dependencies

- [CRSDepTime (Planned DepTime), DepTime] → [DepDelay]
- [Origin, destination] → [Distance]

δ dependencies

- [DepTime] → [CSRDepTime (Planned DepTime)]
- [AirTime] → [CSRAirTime (Planned Airtime)]

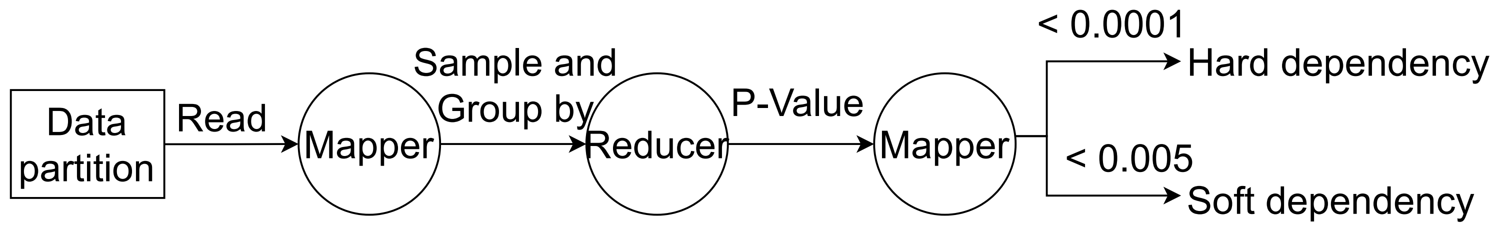
Soft functional dependencies

- [TailNum] → [UniqueCarrier]
- [ActualElapsedTime, TaxiIn, TaxiOut] → [AirTime]
- [AirTime, TaxiIn, TaxiOut] → [ActualelapsedTime]

Computation

Finding (soft) dependencies with CORDS

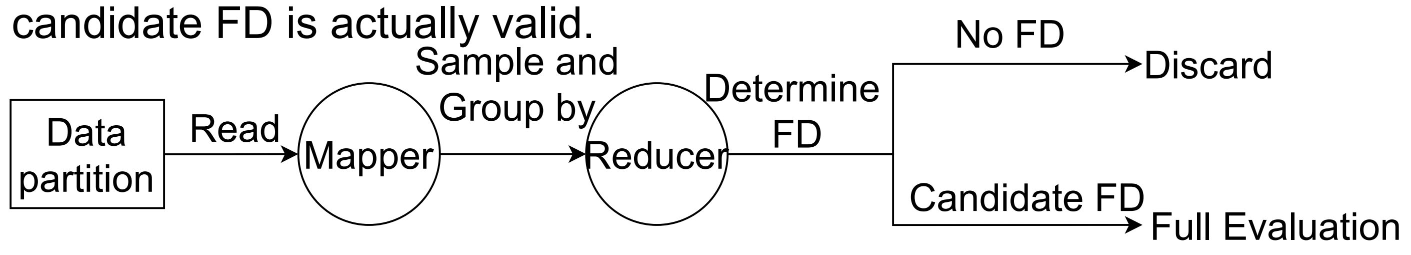
With this method, mappers sample the partitioned data, and send rows with equal column values to the same reducer. The reducers perform a χ2-test. Mappers filter the resulting p-values to check for hard and soft dependencies, with error/certainty depending on the used sample size and p-values.



(Alternative) finding functional dependencies with SMARTFD

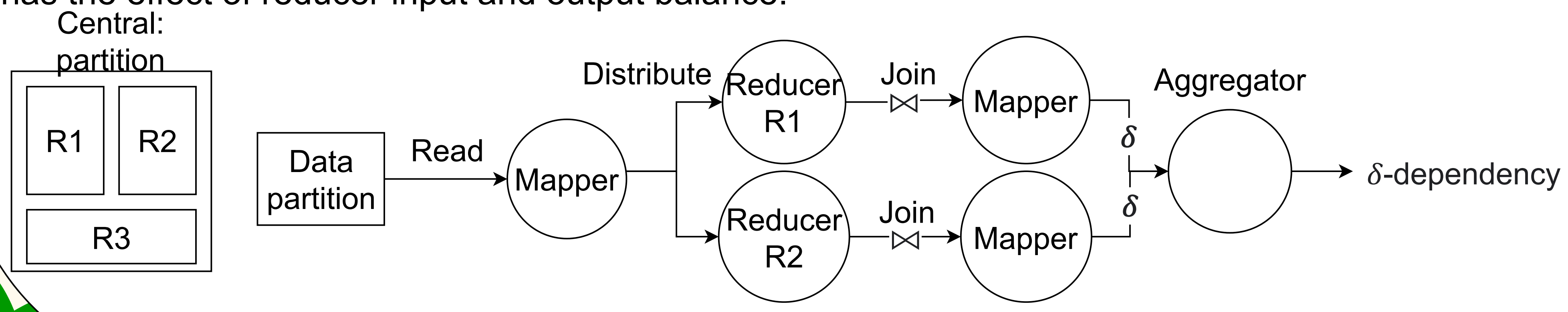
With SMARTFD, mappers sample the partitioned data, and send rows equal column values to the same reducer. Each reducer checks the sample for its subset of candidate FDs, and determines which ones are eligible for full evaluation.

During the full evaluation each candidate FD is validated by each mapper sending the data of it's partition to the reducer based on the left hand side columns. The reducers can then determine if the candidate FD is actually valid.



Finding δ-dependencies with 1-Bucket-Theta

1-Bucket-Theta [ref] first partitions the join matrix. Mappers hash row numbers to a row and column value. The rows are sent to the reducers of to the join matrix regions these rows and columns overlap with. Each reducer performs a join on equal column values, upon which mappers evaluate the δ-value of the row pairs, with which an aggregator determines the δ-dependency bound. The random hashing has the effect of reducer input and output balance.



Video Presentation:
https://youtu.be/U2TM0xyHrh8