

DSE 203

Data Integration and Analytics for Demand Prediction

Query Capability & Learning

Milestone 3

Salah Ahmad

Ehab Abdelmaguid

Disha Singla

Nolan Thomas

Sanjay Kenchareddy

Key accomplishments for Milestone-3

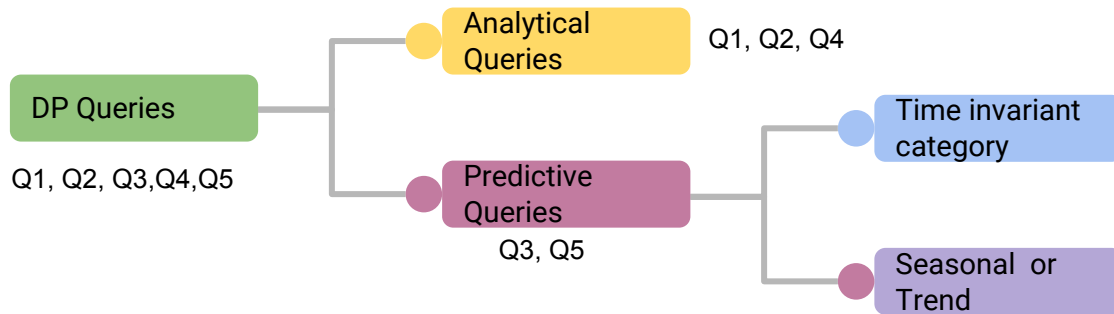
- 1. Worked with other DP teams and finalized five queries**
 - a. Two queries use ML model for prediction of demand.
 - b. Three queries use data analytics to predict demand
- 2. Captured end-to-end process flow for each of the queries**
- 3. Enhanced wrapper design**
 - a. Work with databases hosted on remote server
 - b. Added capability to translate datalog rules into source specific query
- 4. Enhanced Query processing engine design**
 - a. Refined QPE tasks (ordering of queries, joins outside source db etc)
 - b. Refined QPE interface with Query optimizer and source wrappers
- 5. Explored a set of Performance optimizations**
- 6. Developed GUI application tool to convert Datalog syntax to SQL**

Demand Prediction Queries

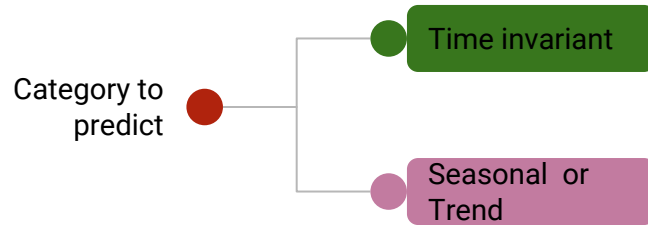
1. What are the top 3 categories of books that are most read around Christmas?
2. What time of the year are the sales of “Education” books the highest?
3. Given month m and category c , predict the amount of sales for the category.
4. Which book categories show a downward trend in demand in Winter and Spring?
5. Is there a category that we should discontinue stocking?

Queries are grouped as follows:

- Analytical Queries: we can answer by performing some analytics on historical data
- Predictive Queries: ML uses Prediction model to answer queries

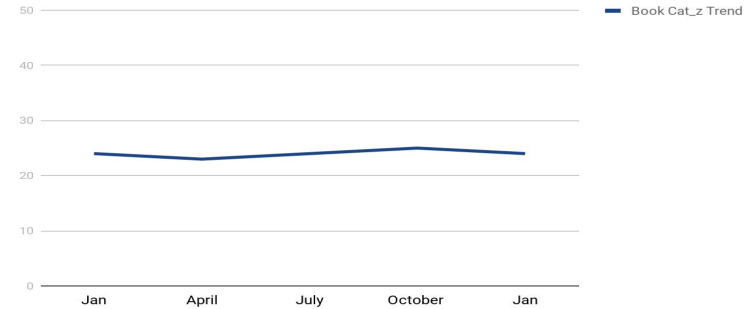


High Level Machine Learning Prediction approach

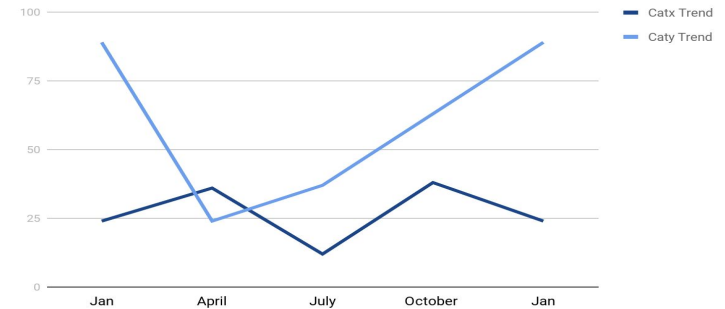


- Each category will be grouped as “Time invariant” or “Seasonal” by analyzing historical data
- ML uses historical data to predict sales for “seasonal Invariant” category (avg sale of last n months)
- ML builds regression model for “seasonal variant” category. It trains and saves the weights for each category
- ML applies model to predict sales for “seasonal variant” cat




Time invariant Book Category (almost flat)



Different trends per Seasonal Book Category



Data Sources for Queries

Query	 PostgreSQL	 Asterix DB	 Solr	Intra source joins	Inter source joins	Query type
1	Orderline, product	ClassificationInfo	None	Yes	Yes	Analytical
2	Orderline, product	ClassificationInfo	None	Yes	Yes	Analytical
3	ML features Postgres	ML features asterix DB	ML features Sentimental analysis of text review	Yes	Yes	Prediction
4	Orderline, product	ClassificationInfo	None	Yes	Yes	Analytical
5	ML features Postgres	ML features asterix DB	ML features Sentimental analysis of text review	Yes	Yes	Prediction

Performance Optimizations

Goals:

1. Reduce frequent operations on source tables
2. Improve query processing time
3. Minimize I/O operations for each query
4. Minimize processing of data outside of source DBs

Optimizations being considered/Evaluated

1. Find the order of queries that require less joins in processing engine (outside of source DB)
2. Use analytics/computation capabilities of source DB
 - a. Minimize the processing in mediator
3. Define views on PGDB to
 - a. Extract features for ML training and prediction
 - b. Perform data analytics
4. Use Solr DB for sentimental analysis
5. Refined QPE interface with Query optimizer and source wrappers

Machine learning Feature set per Category (Received from ML Team)

Postgres DB

Month
Dollar amount sold last month
Dollar amount sold last quarter
Dollar amount sold last year
Number sold last month
Number sold last quarter
Number sold last year
Is in Campaign

Asterix Db

Average rating for category
Difference between this category rating and all others

Solr DB

Sentiment Factor Value

Each feature set per book category need to be retrieved from all 3 databases

Assumptions :

1. Training Data will be aggregated and delivered to ML model on month basis
2. sentimental analysis will be calculated in solr db ,2 options we have either calculate sentimental every time or we materialize it in a new column in solr database

Views supporting training data

SQL VIEW Example

```
CREATE OR REPLACE VIEW cat_month_agg
AS SELECT p.nodeid, c.month,
        c.year, SUM(n.totalprice) AS totalprice, SUM(n.numunits) AS totalunits
FROM orders o, orderlines n, products p, calendar c
WHERE o.orderid = n.orderid
AND n.productid = p.productid
AND o.orderdate = c.date
GROUP BY p.nodeid, c.month, c.year;
```

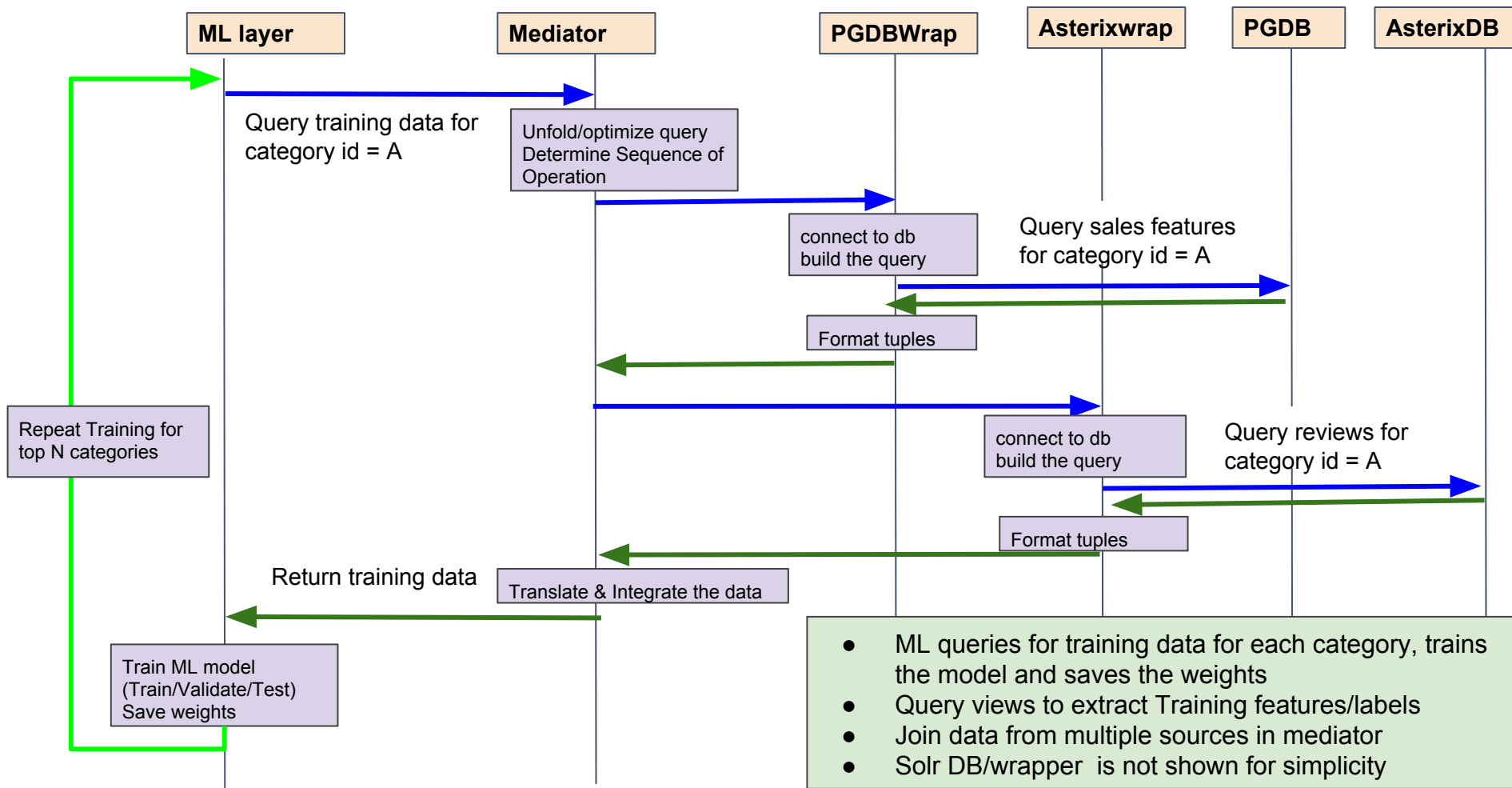
Corresponding Datalog Example :

```
sales(nodeid,month,year,totalprice,totalunits) :-
S1.order(orderid,_,_,orderdate),S1.orderlines(_,orderid,_,_,_,totalunits,totalprice),
S1.products(productid,_,_,_,_,_,nodeid),
S1.calendar(orderdate,year,month)
```

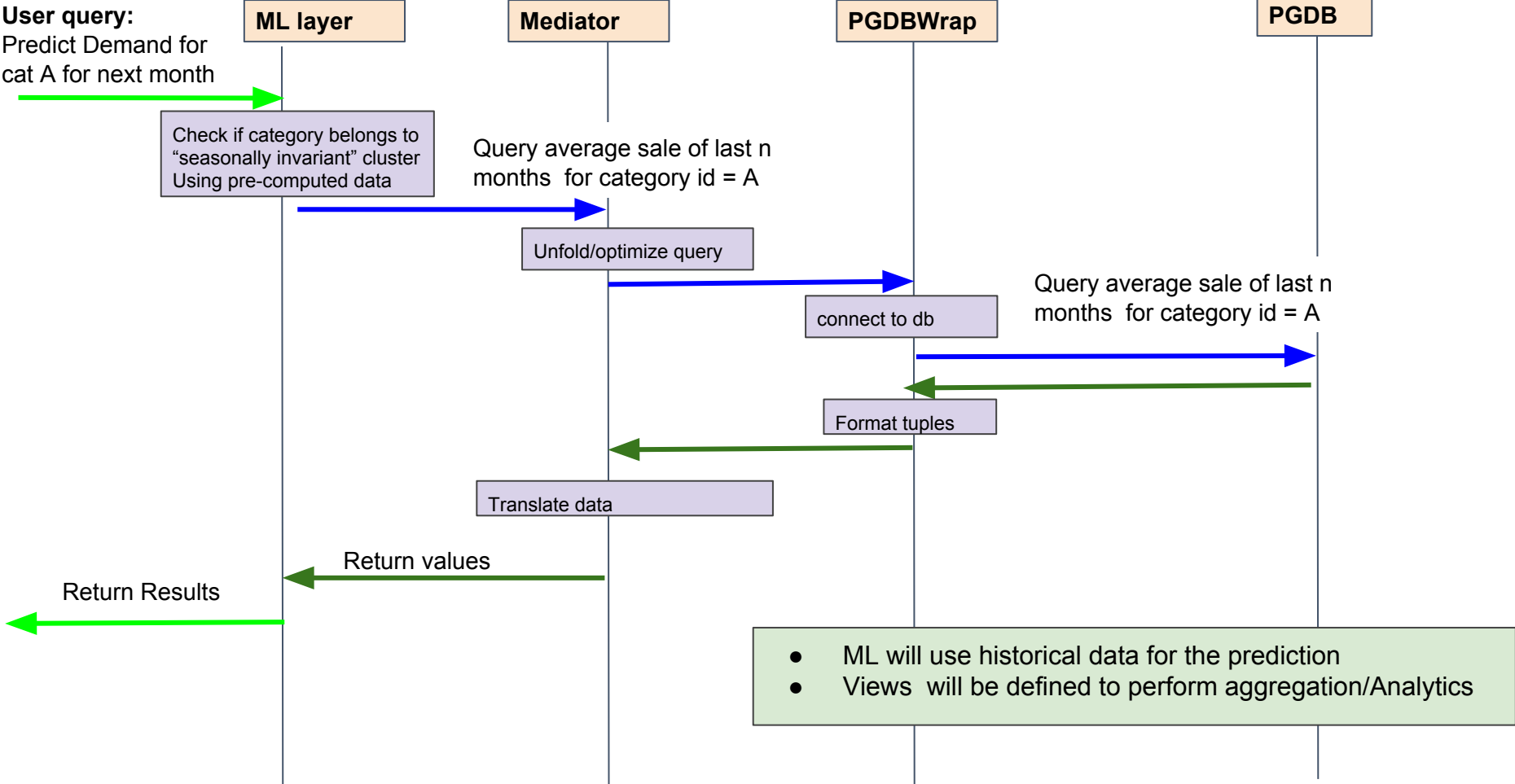
```
cat_month_agg(nodeid,month,year,totalprice,totalunits) :-
```

```
group_by(sales(nodeid,month,year,totalprice,totalunits), [nodeid, month, year],
totalprice=sum,totalunits=sum)
```

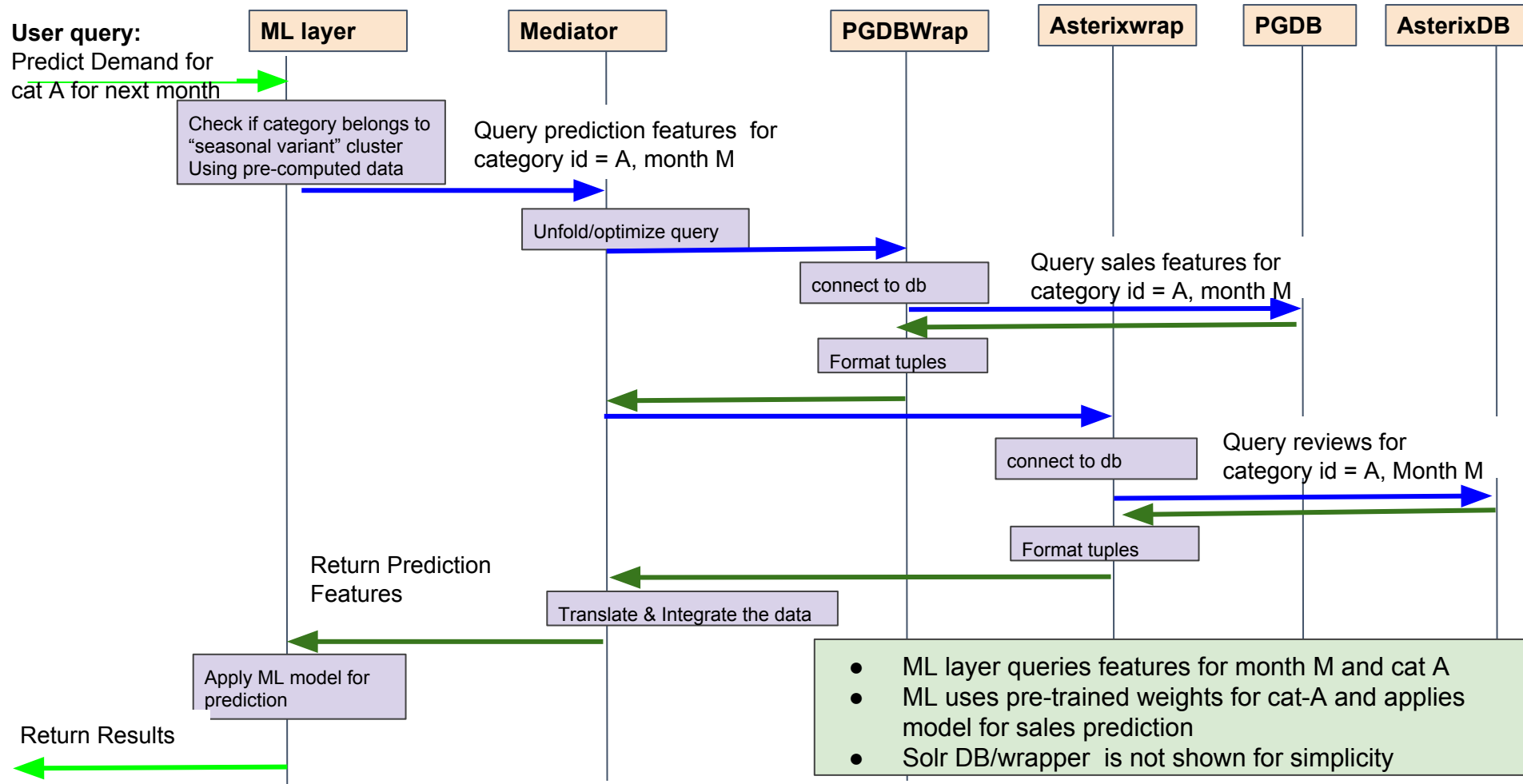

Training ML model for prediction task (one category at a time)



Demand Prediction Case-1: Category belongs to “Seasonal Invariant” cluster (use historical data)



Demand Prediction Case-2: Category belongs to “Seasonal variant” cluster (Use ML Model)



Query Flows

Query1: Top 3 categories around Christmas

`Ans(category, total) :-`

`Requires 2 steps`

`Step 1 :`

`by_total<total>(category,total) :-`

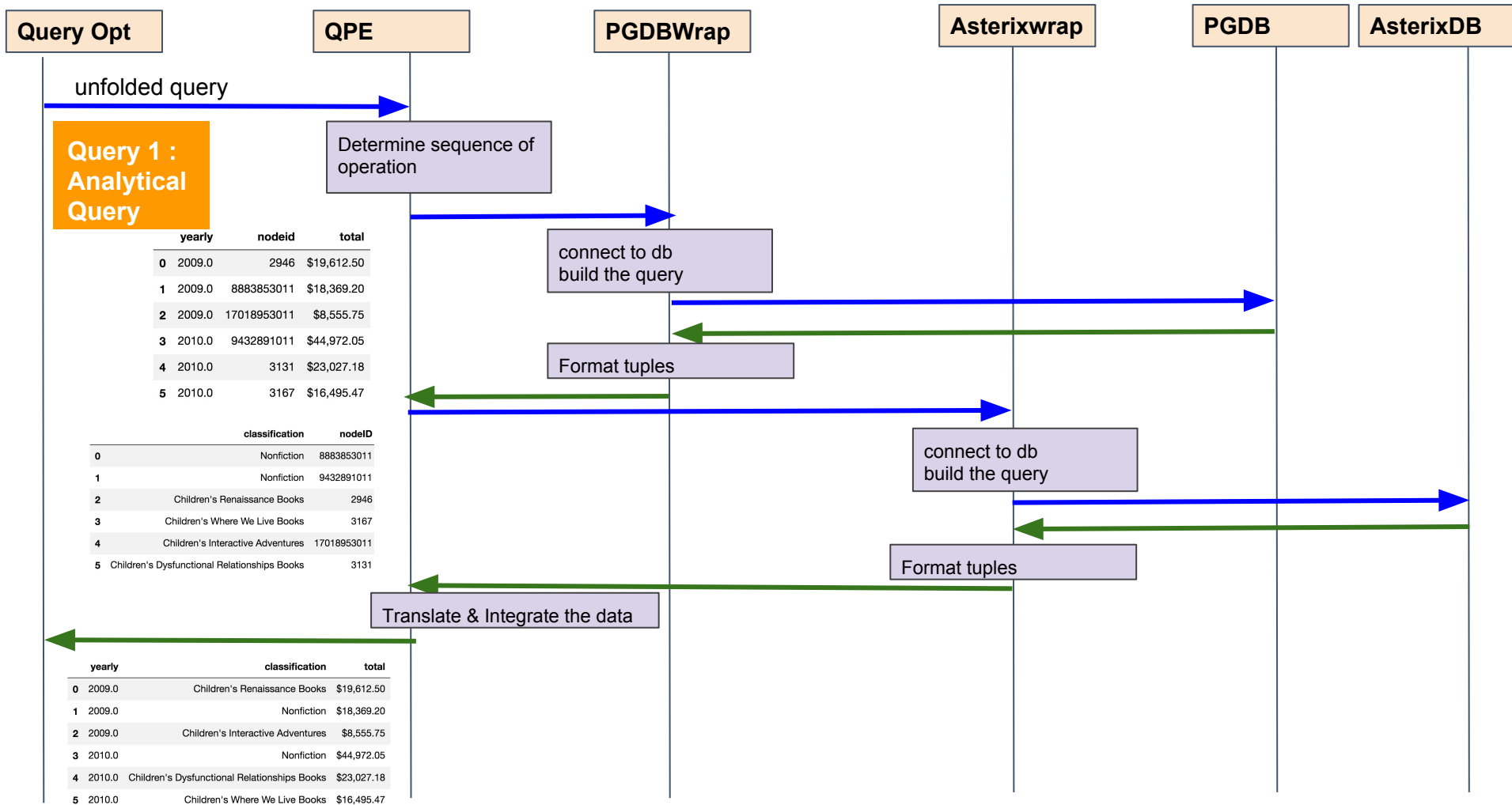
`group_by(cat_month_agg(_,category,total), [category], total=sum, month=12))`

`Step 2:`

`Ans(category, total) :- by_total[N](category,total), N <= 3`

Assumptions

- Creation of virtual view to represent seasonal sales by category/classification:
 - `cat_month_agg(nodeid,category,total,month,year)`
- Considering the December around Christmas



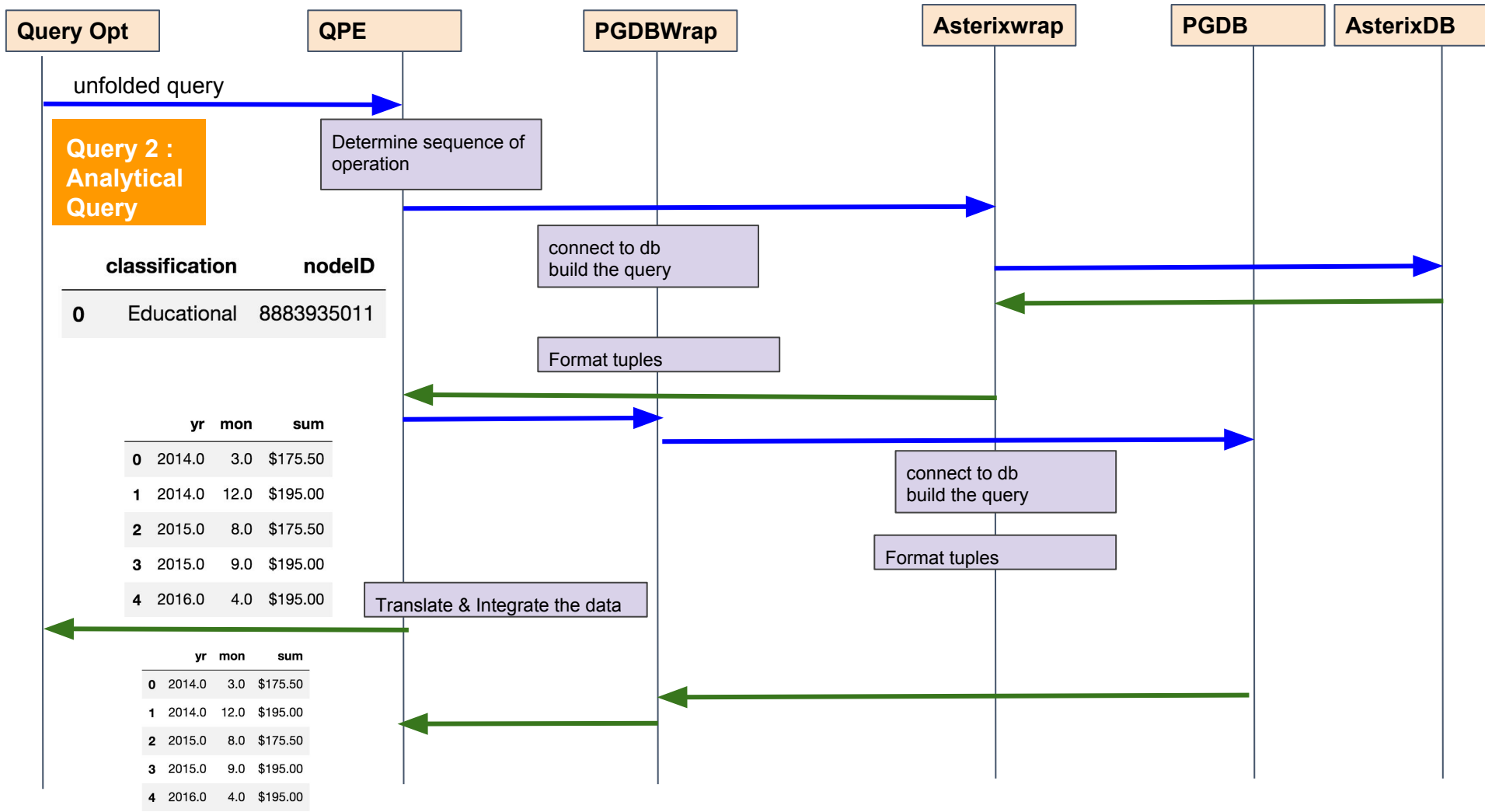
Query2: Time of the year sales of “Education” books highest

Ans(year, month, total) :-

`cat_month_agg(_,category, total,year, month), category='Education'`

Assumptions

- Creation of virtual aggregation view monthly aggregate sales total by category/classification:
 - `cat_month_agg(_,category, total,year, month)`
- NOTE: We only found data for 5 months in the entire data set



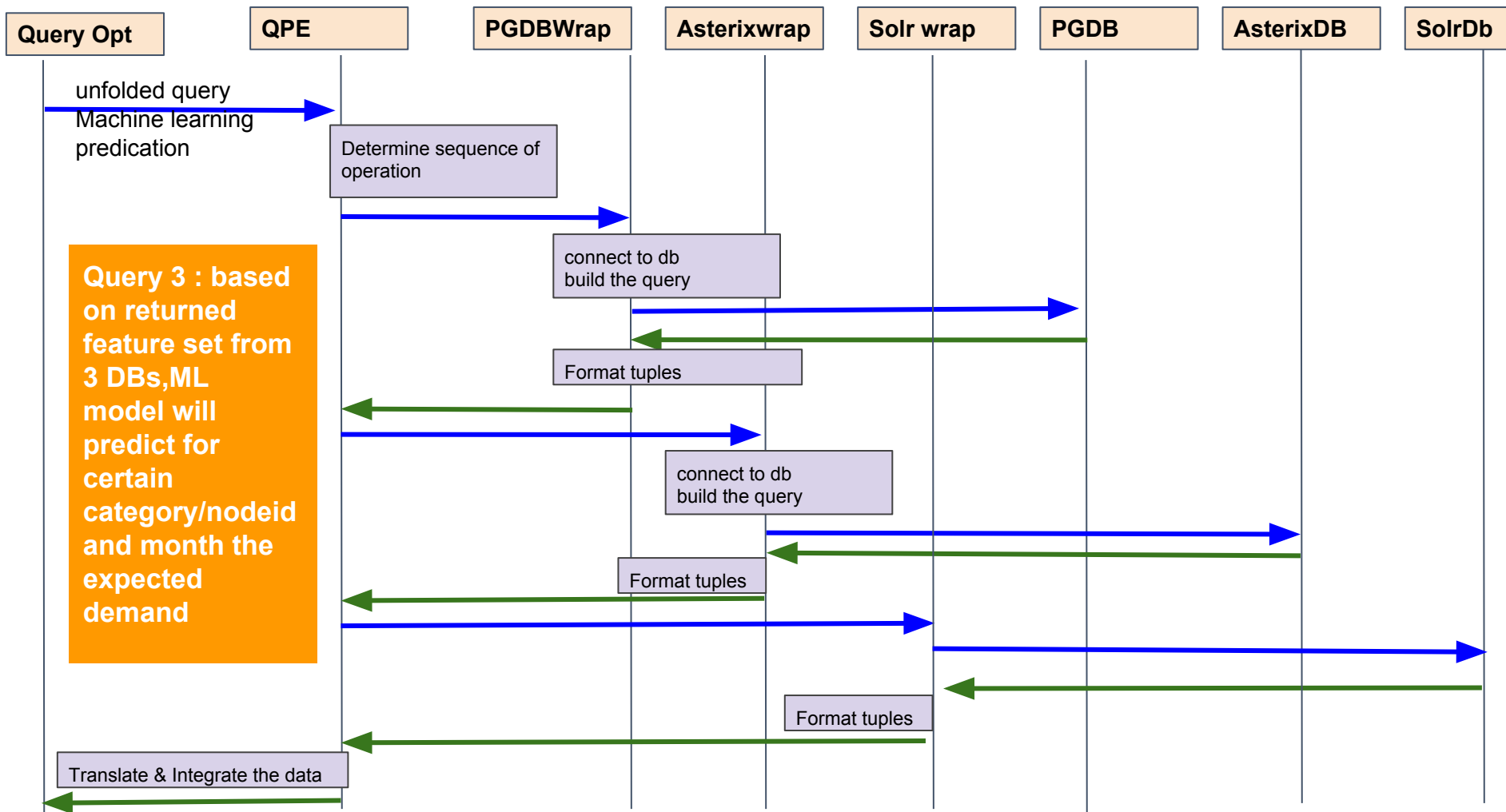
Query3 : Given month m and category c, predict the amount of sales for the category.

Request

```
Ans(month,category,predictedsales) :-  
demandpredication(_,nodeid,month,year,_,predictedsales),  
classification(node_id,category), year=2018, month=12
```

Assumptions

- Machine learning features are being captured from several databases (postgres ,Astrex DB , Solr). Those are captured in the `demandpredication` object



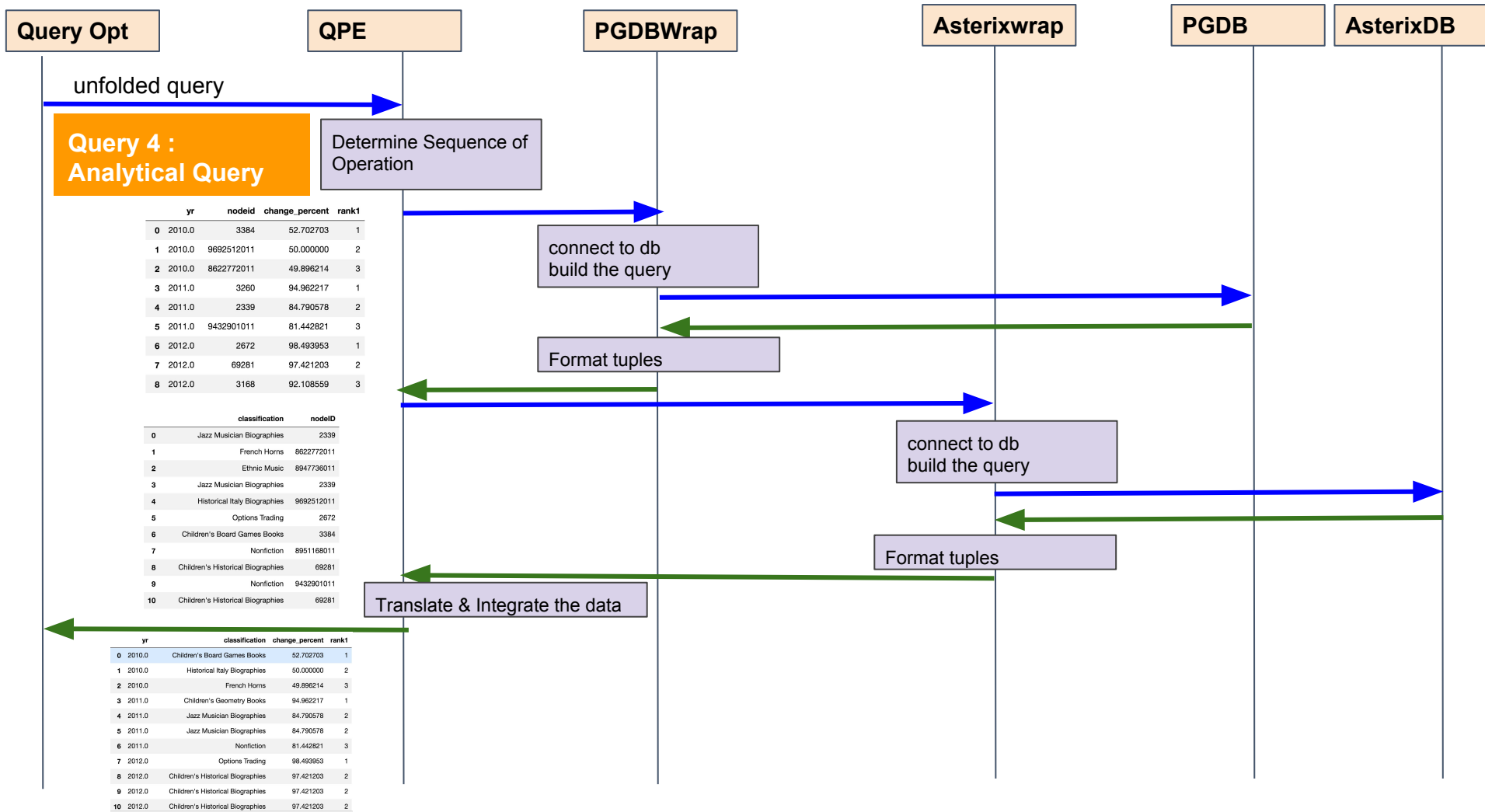
Query 4: categories showing downward trend in Winter & Spring

Ans(year, percentchange, category, rank):-

```
seasonal_trend_agg(season,nodeid,rank,percentchange,year),  
classification_info(nodeid,category), (season='Winter';season='Spring')
```

Assumptions

- Customer is wanting to determine yearly seasonal variation
- Creation of virtual view to represent seasonal sales by category/classification:
 - `seasonal_trend_agg(season,nodeid,rank,percentchange,year)`
- Totals for ranking the classifications was aggregated per classification, month and the year
- For this presentation we are returning top three categories each year with downward trend



Query 5: Is there a category that we should discontinue stocking ?

Request

`Ans(category,predictedsales) :-`

Requires 3 steps

Step 1 :

```
pred_next_year(category,predictedsales) :-  
demandpredication(_,nodeid,month,year,_,predictedsales),  
classification(nodeid,category), year=2018
```

Step 2 :

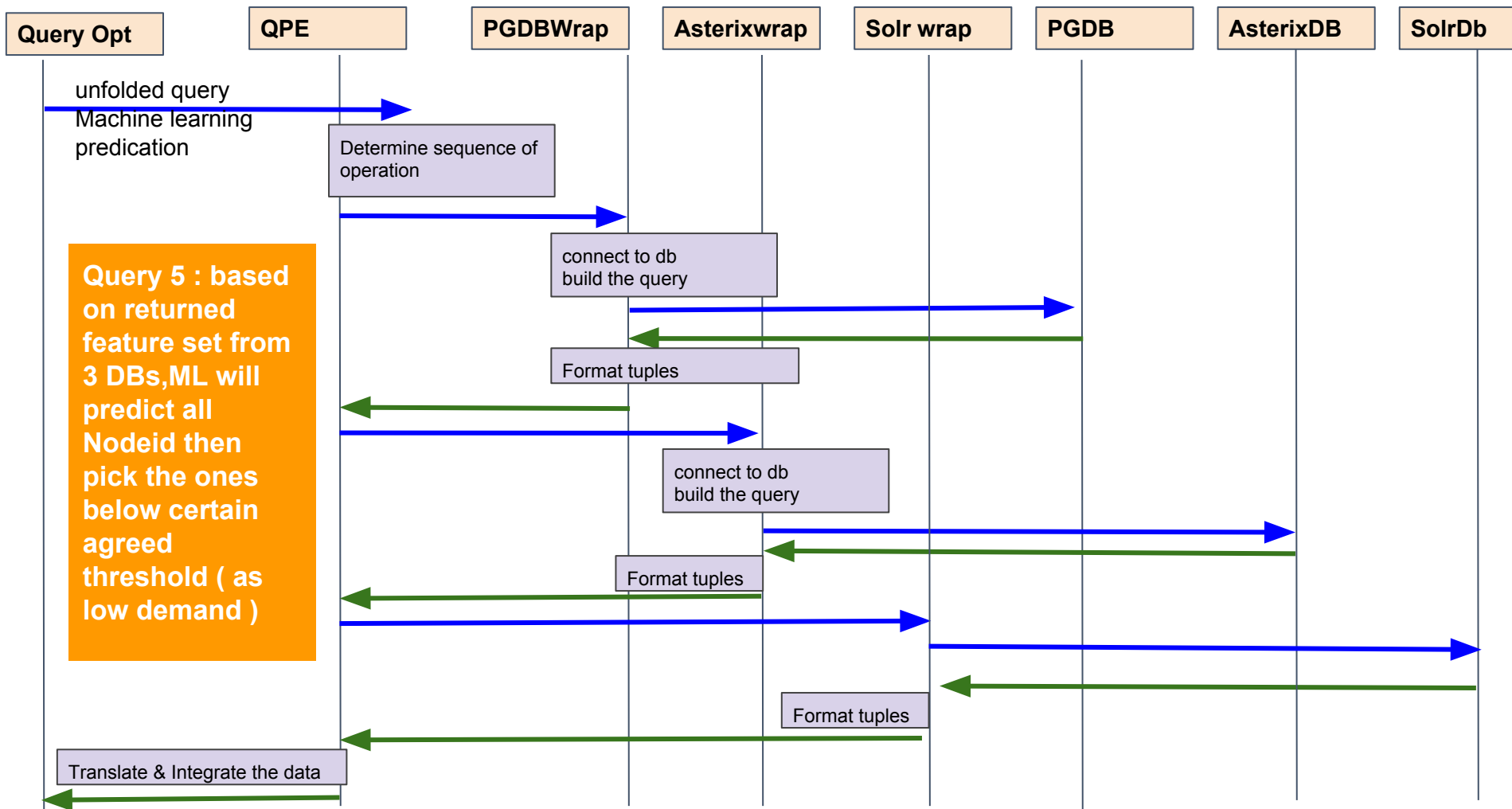
```
by_sales<predictedsales>(category,predictedsales) :-  
group_by(pred_next_year(predictedsales,category), [category], predictedsales=sum)
```

Step 3

```
Ans(category, predictedsales) :- by_sales[N](category,predictedsales), N <= 100
```

Assumption

- Choosing the 100 categories with the least predicted sales as the ones to discontinue for 2018 year



Datalog Query Tool

Ans(numunits, firstname, billdate) :-
orders(numunits, customerid, orderid),
customers(firstname, customerid),
orderlines(billdate, orderid),
orders.orderid > 1000, orders.numunits > 1,
limit 7

SELECT orders.numunits, customers.firstname,
orderlines.billdate FROM orders, customers, orderlines
WHERE orders.orderid=orderlines.orderid AND
orders.customerid=customers.customerid AND orders.orderid
> 1000 AND orders.numunits > 1 limit 7;

	numunits	firstname	billdate
0	2	LAWRENCE	2011-03-08
1	2	JAMES	2011-01-20
2	4	LINDA	2009-11-12
3	4	LINDA	2009-11-20
4	2	D.	2009-11-11
5	2	D.	2009-11-20
6	3	CHARLES	2009-11-10

This application prototype demonstrates a datalog query being processed thru our postgres wrapper

- Flask
- Jinja2
- Postgres Wrapper
- Rudimentary Datalog Parsing
- Webpy

Link:<https://github.com/kshannon/dse203-de-mand-pred/tree/master/query-code/flask>

Thank you