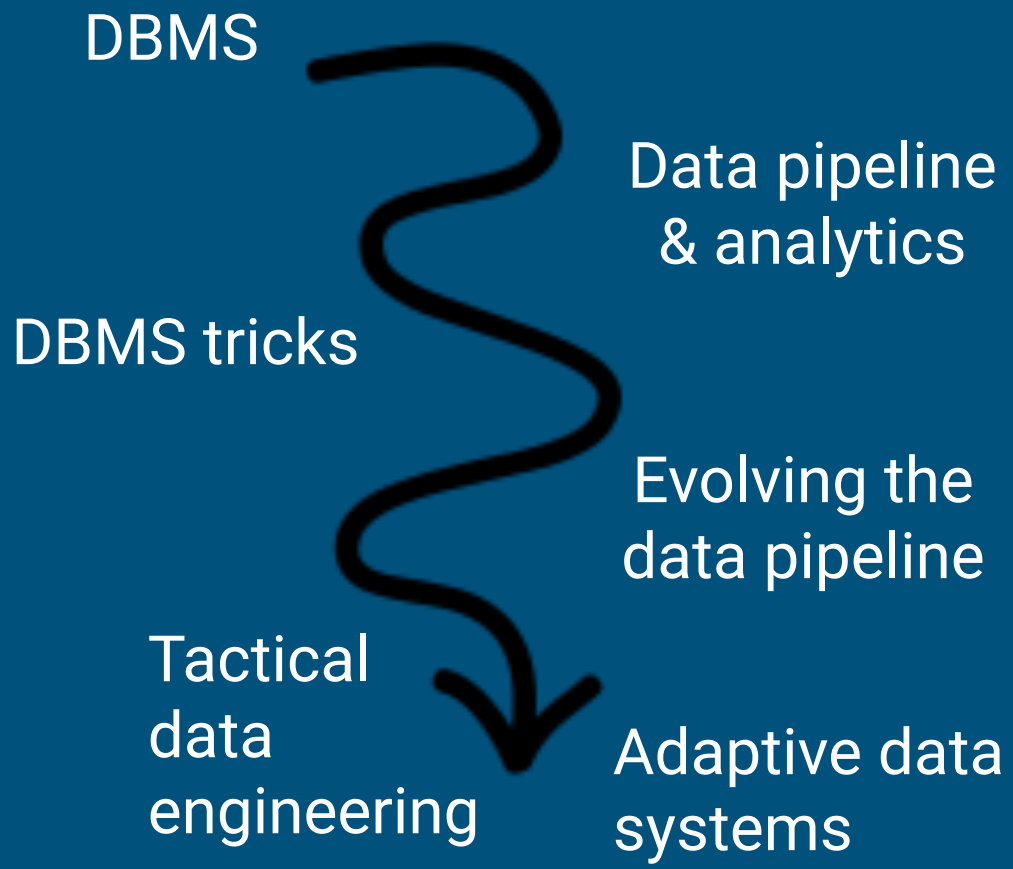
Tactical data engineering

Julian Hyde

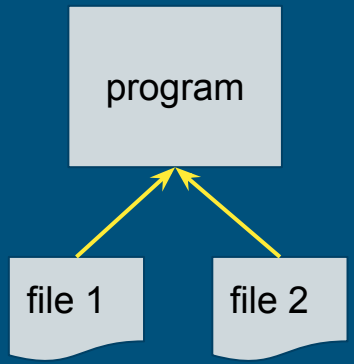
Data Council

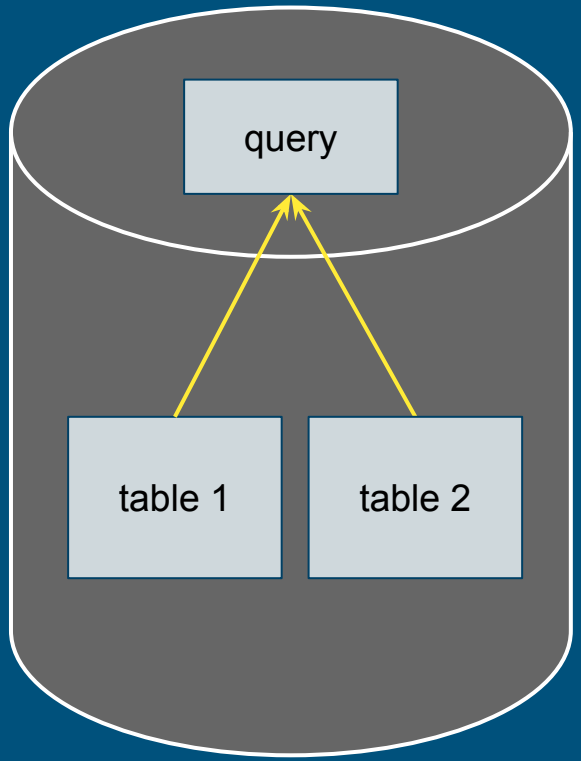
April 17–18, 2019
San Francisco

@julianhyde

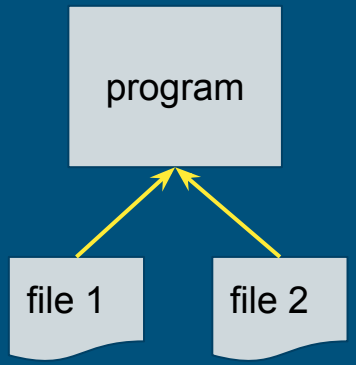

DBMS
Data pipeline & analytics
DBMS tricks
Evolving the data pipeline
Tactical data engineering
Adaptive data systems

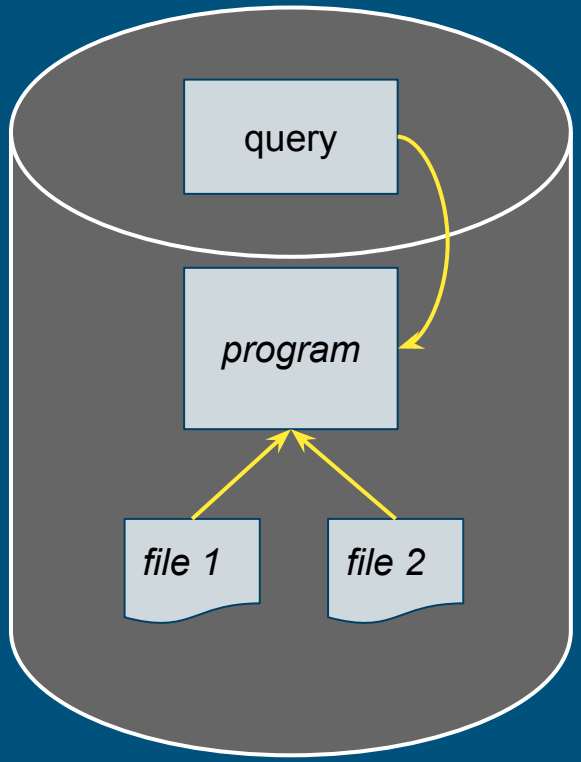
1. DBMS



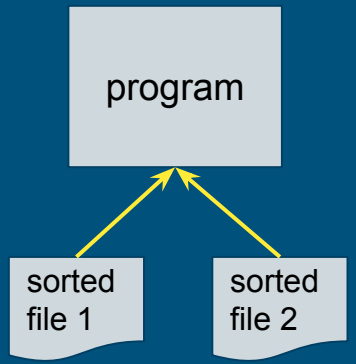


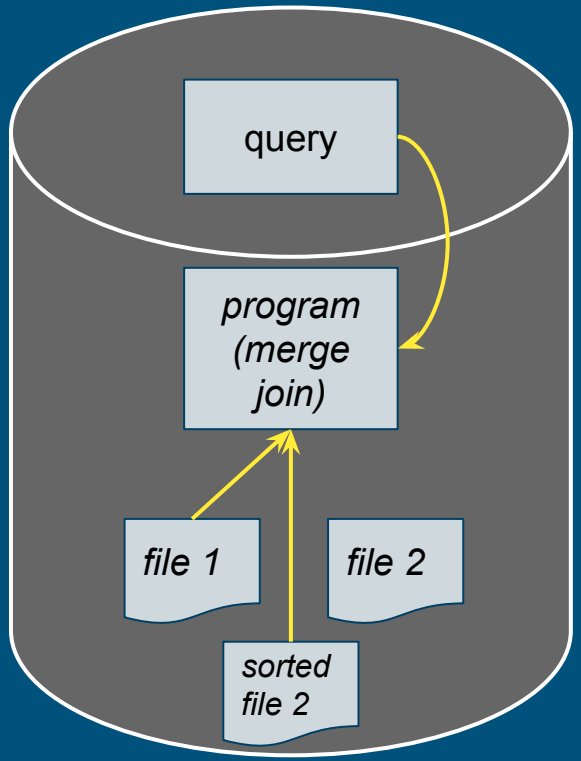
File system vs. DBMS





File system vs. DBMS





Efficient join: reorganize the data and rewrite the program

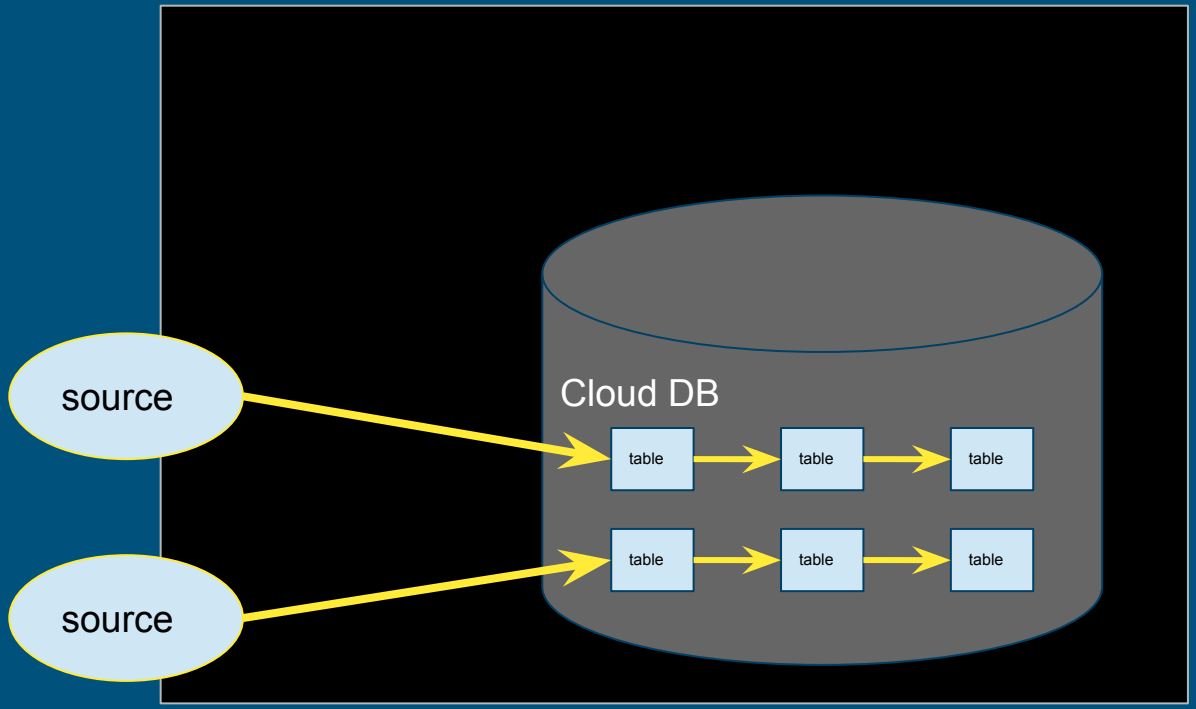
DBMS adds value

- Abstraction
- Declarative language
- Planning
- Easily reorganize data, add new algorithms
- Governance
- Metadata
- Security

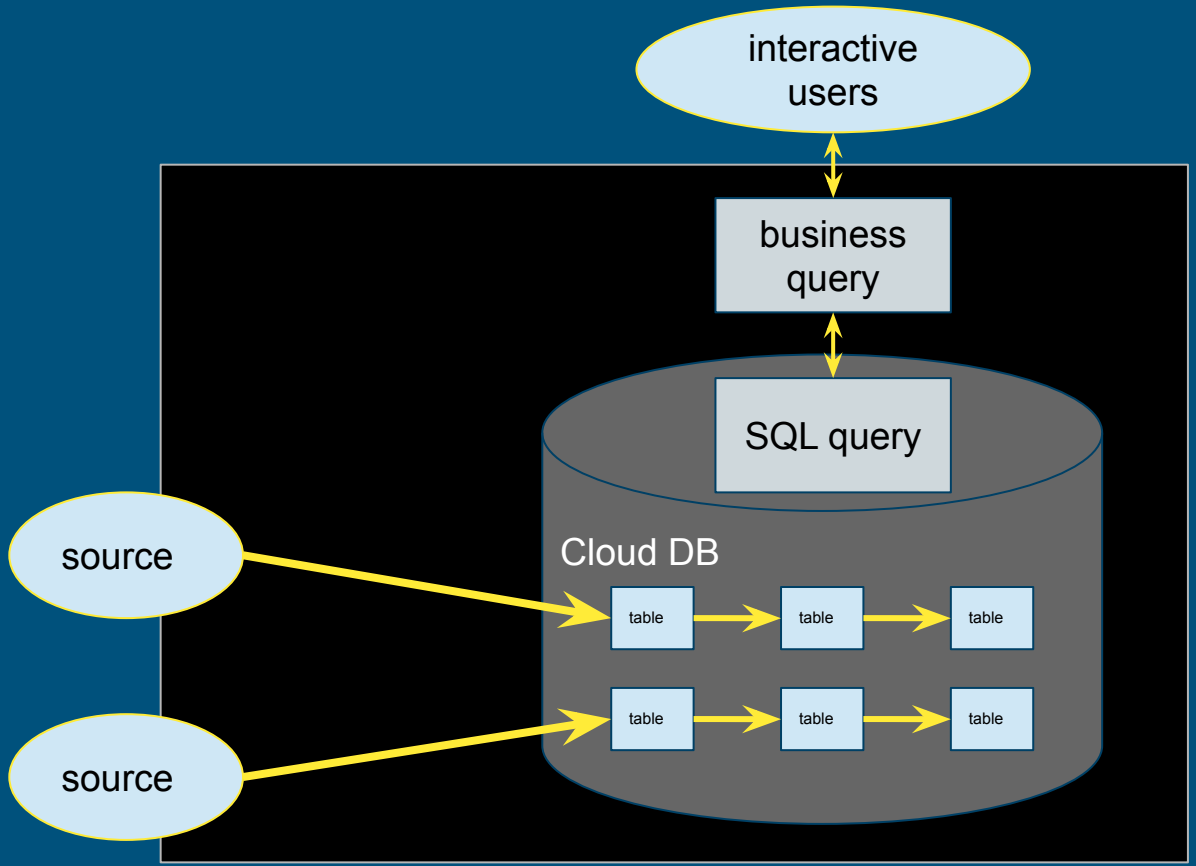
And, I propose:

- Adaptability

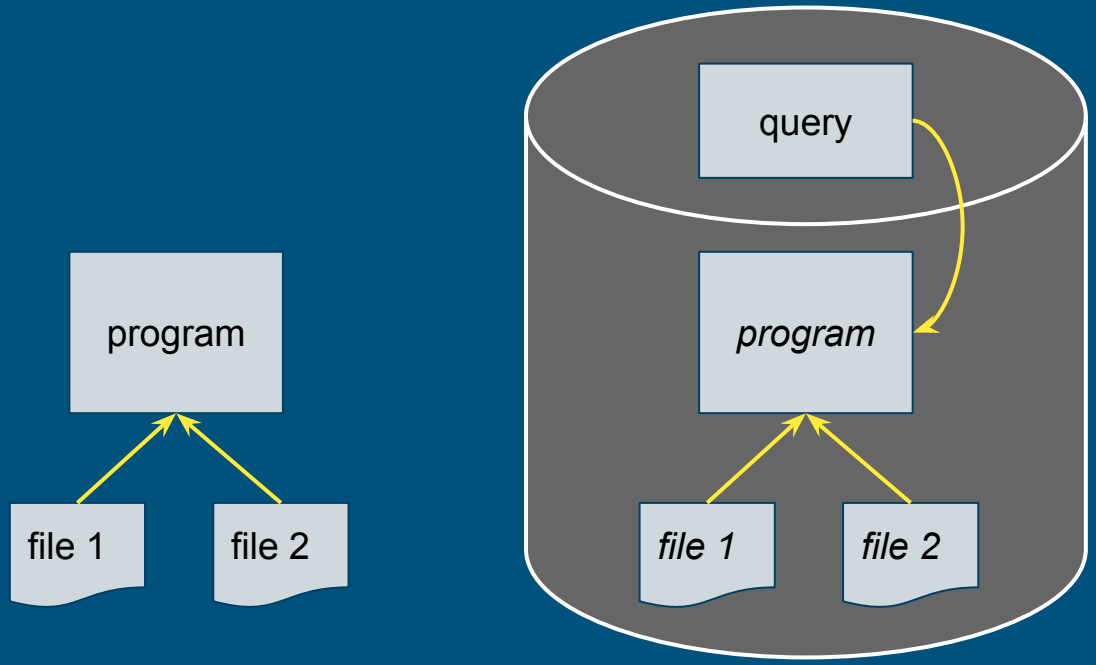
2. Data pipeline



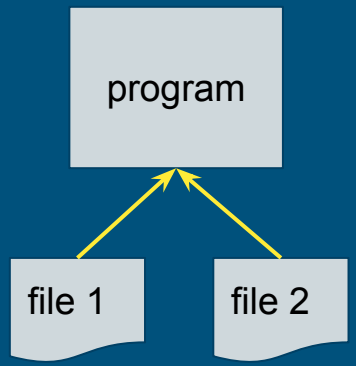
The data pipeline: Extract - Load - Transform

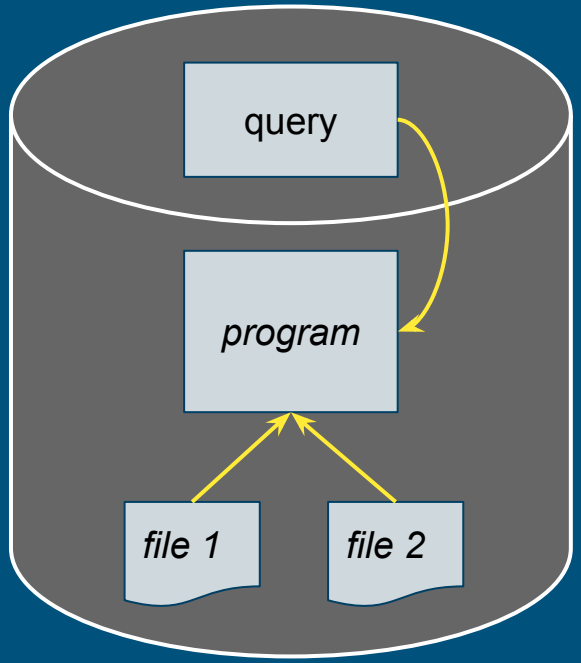


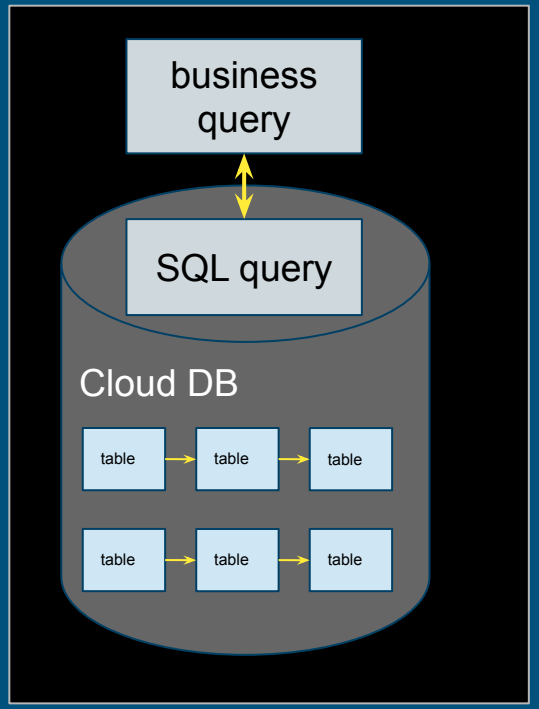
The data pipeline: Extract - Load - Transform



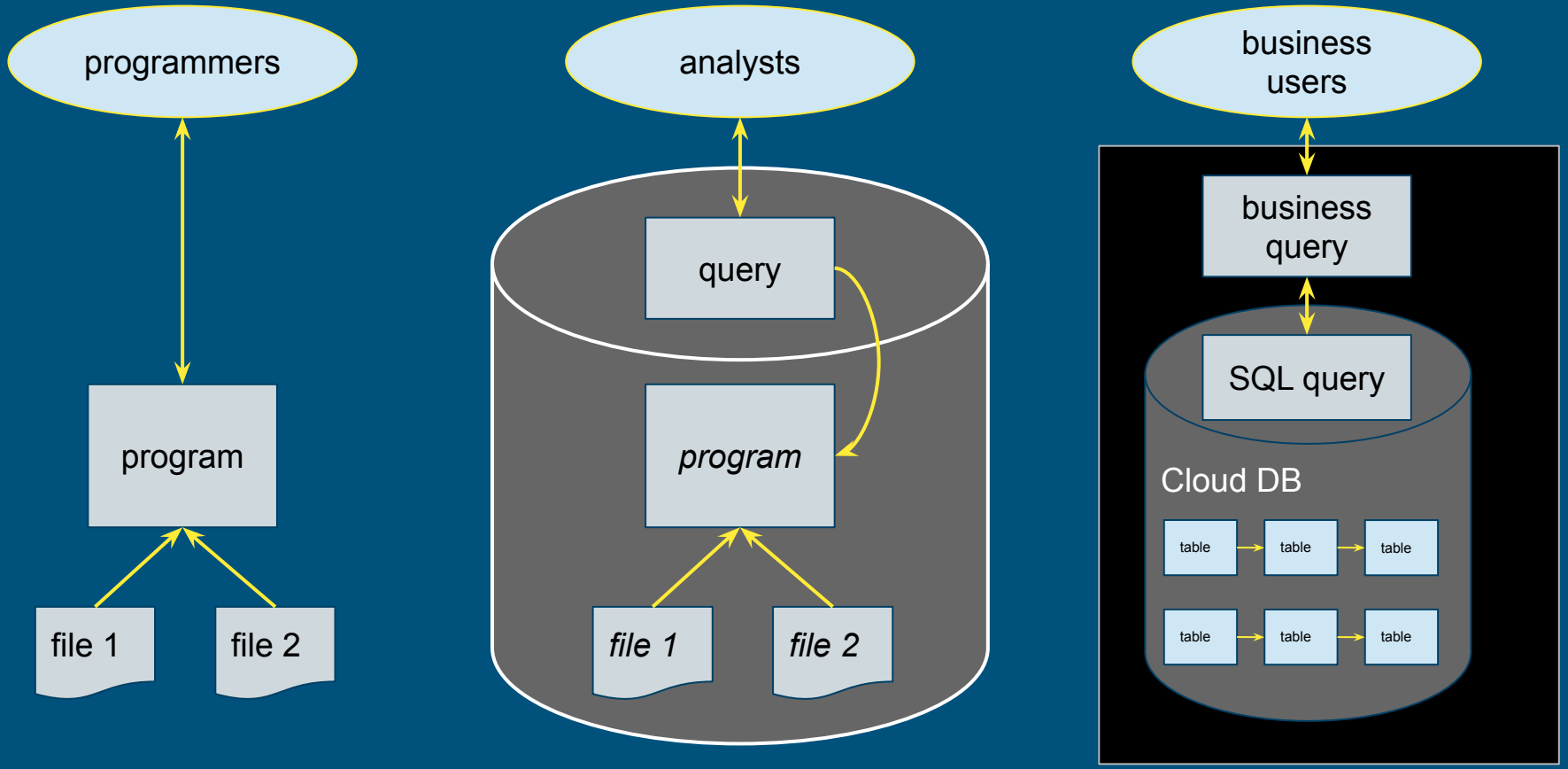
File system vs. DBMS







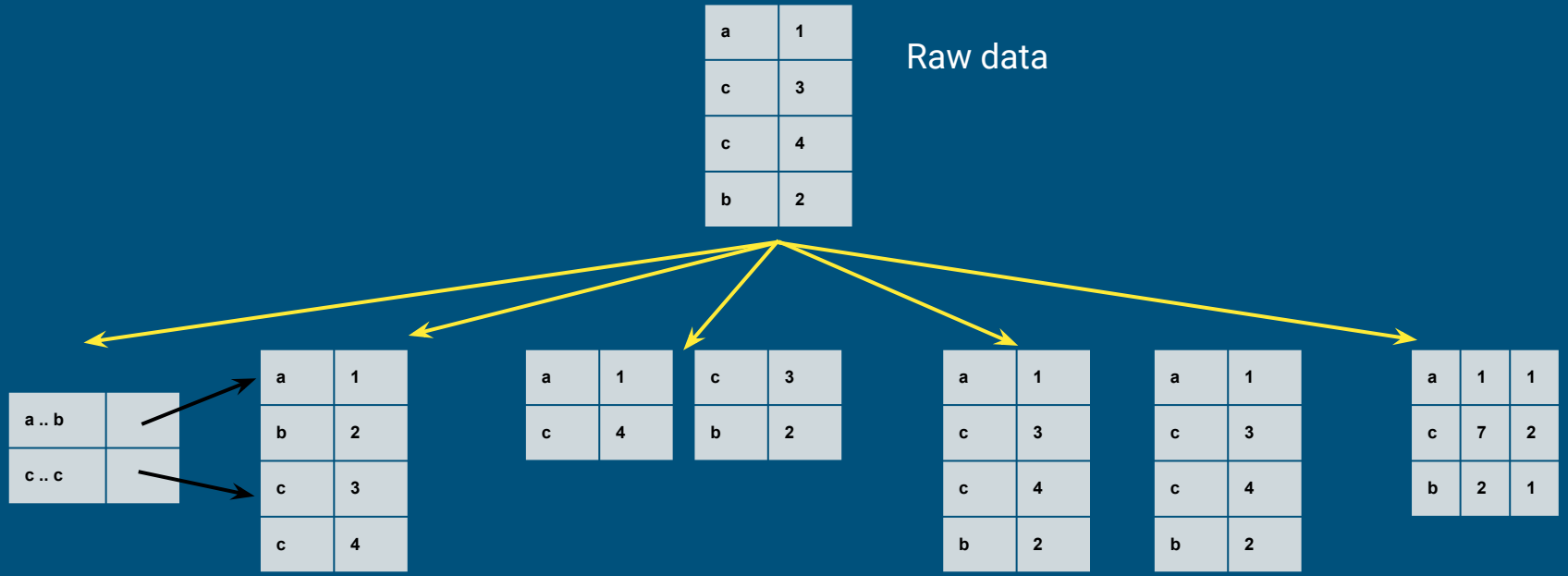
File system vs. DBMS vs. analytic data system



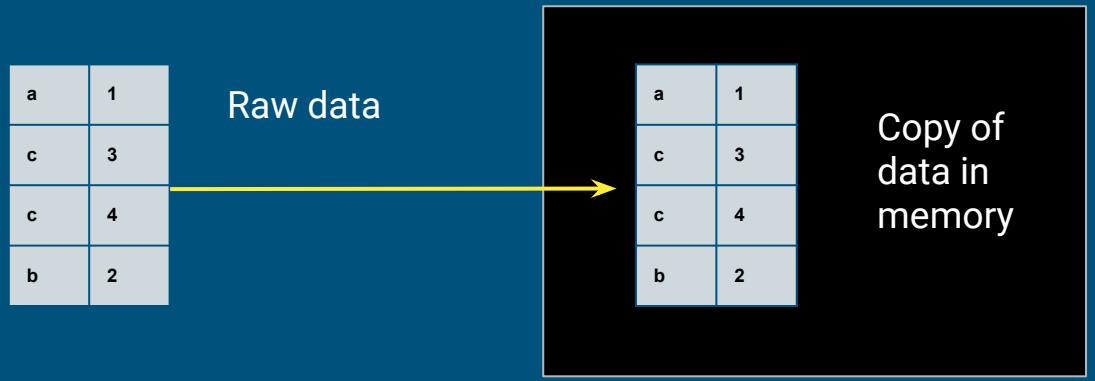
File system vs. DBMS vs. analytic data system

3. DBMS tricks

Re-organize data



Caching



Apache Calcite

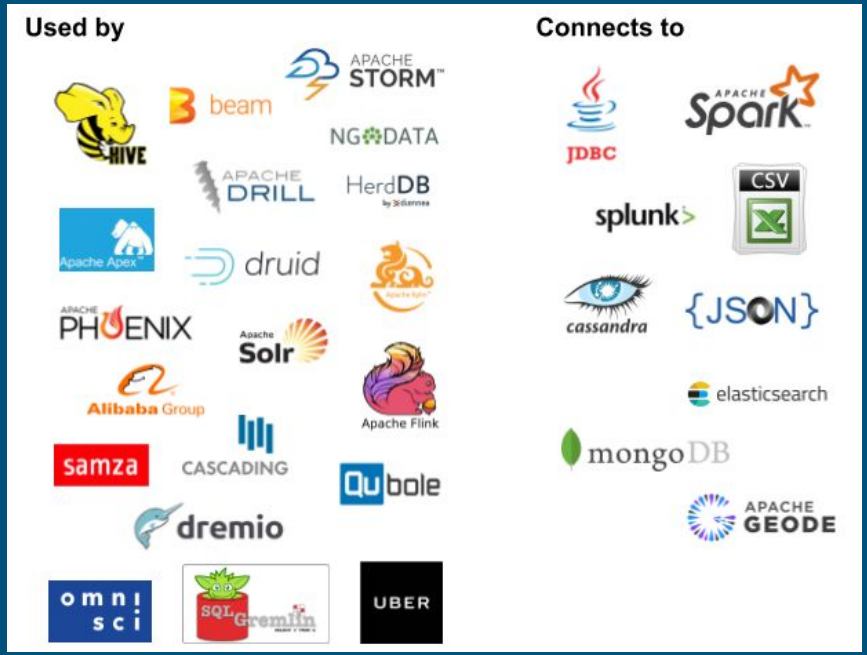
Apache top-level project

Query planning framework used in many projects and products

Also works standalone: federated query engine with SQL / JDBC front end

Apache community development model

calcite.apache.org
github.com/apache/calcite



Relational algebra

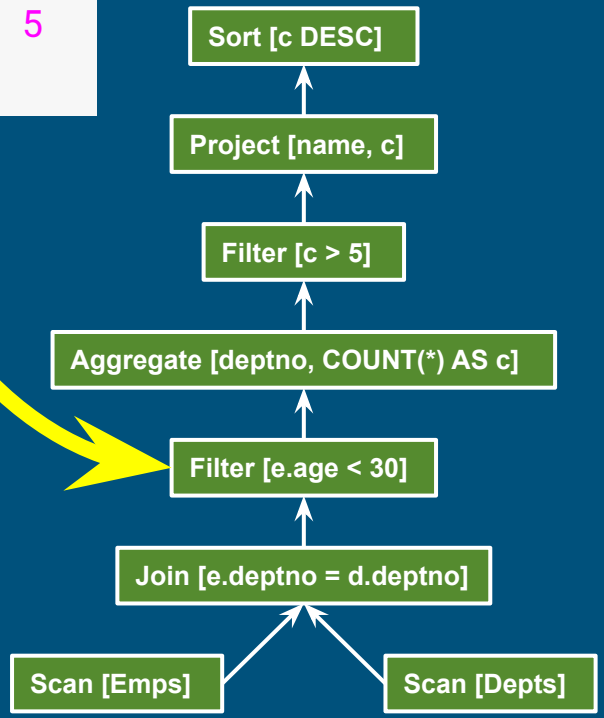
Based on set theory, plus operators: Project, Filter, Aggregate, Union, Join, Sort

Requires: declarative language (SQL), query planner

Original goal: data independence

Enables: query optimization, new algorithms and data structures

```
SELECT d.name, COUNT(*) AS c
FROM Emps AS e
JOIN Depts AS d USING (deptno)
WHERE e.age < 40
GROUP BY d.deptno
HAVING COUNT(*) > 5
ORDER BY c DESC
```



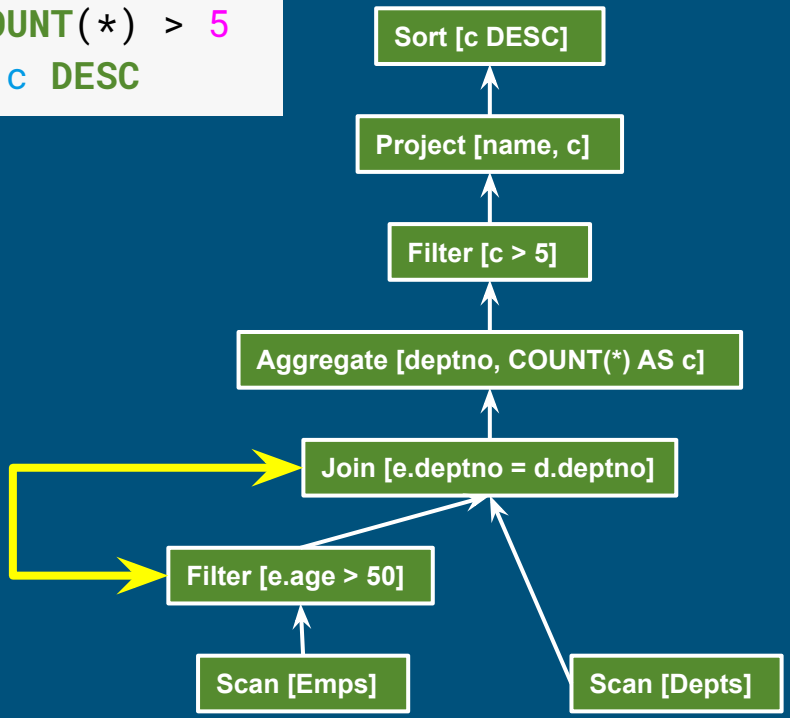
Algebraic rewrite

Optimize by applying rewrite rules that preserve semantics

Hopefully the result is less expensive; but it's OK if it's not (planner keeps "before" and "after")

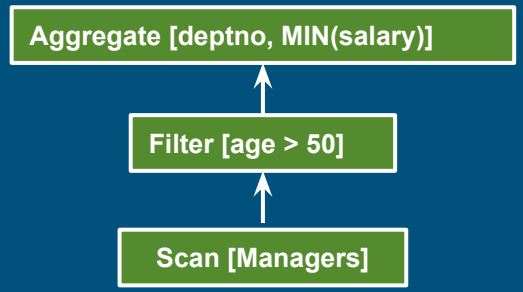
Planner uses dynamic programming, seeking the lowest total cost

```
SELECT d.name, COUNT(*) AS c
FROM (SELECT * FROM Emps
      WHERE e.age > 50) AS e
JOIN Depts AS d USING (deptno)
GROUP BY d.deptno
HAVING COUNT(*) > 5
ORDER BY c DESC
```

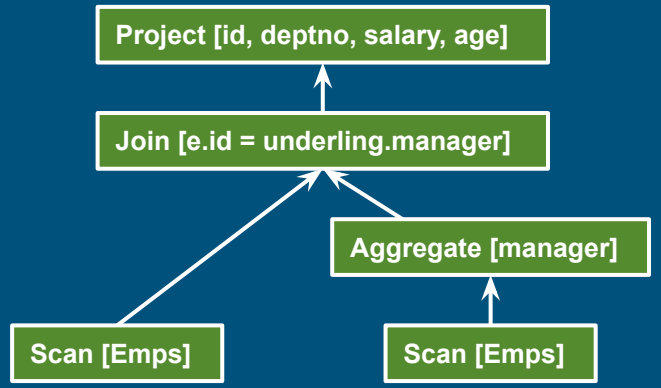


Views

```
SELECT deptno, MIN(salary)
FROM Managers
WHERE age > 50
GROUP BY deptno
```



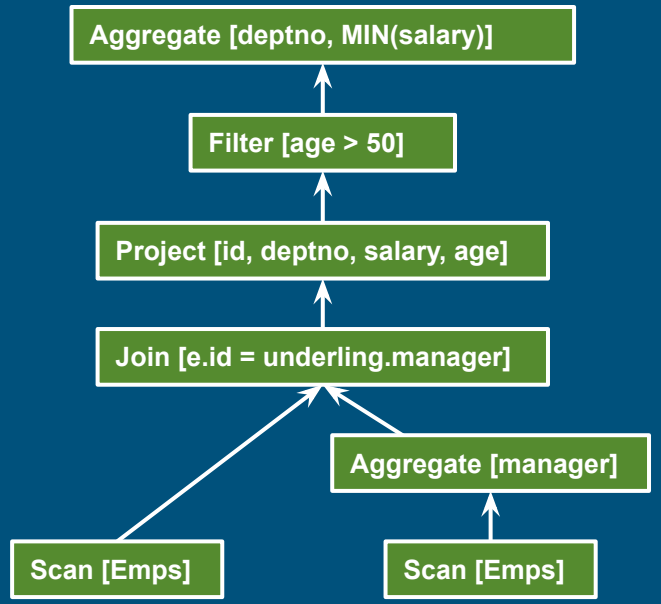

```
CREATE VIEW Managers AS
SELECT *
FROM Emps AS e
WHERE EXISTS (
  SELECT *
  FROM Emps AS underling
  WHERE underling.manager = e.id)
```



View query (after expansion)

```
SELECT deptno, MIN(salary)
FROM Managers
WHERE age > 50
GROUP BY deptno
```

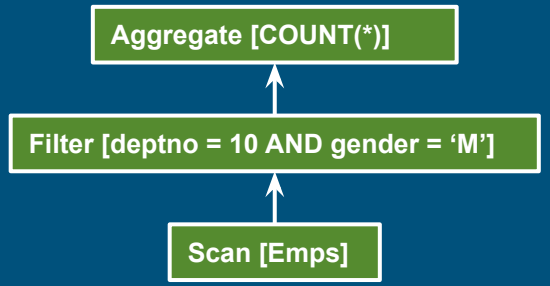
```
CREATE VIEW Managers AS
SELECT *
FROM Emps AS e
WHERE EXISTS (
  SELECT *
  FROM Emps AS underling
  WHERE underling.manager = e.id)
```



Materialized view

```
CREATE MATERIALIZED VIEW
  EmpSummary AS
SELECT deptno, gender,
  COUNT(*) AS c, SUM(sal) AS s
FROM Emps
GROUP BY deptno, gender
```

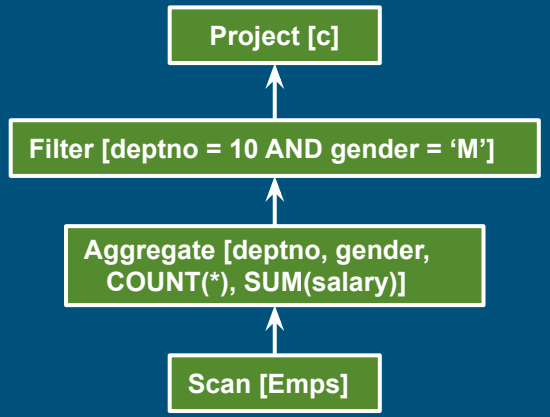

```
SELECT COUNT(*) AS c
FROM Emps
WHERE deptno = 10
AND gender = 'M'
```



Materialized view: rewrite query to match

```
CREATE MATERIALIZED VIEW
  EmpSummary AS
SELECT deptno, gender,
  COUNT(*) AS c, SUM(sal) AS s
FROM Emps
GROUP BY deptno, gender
```

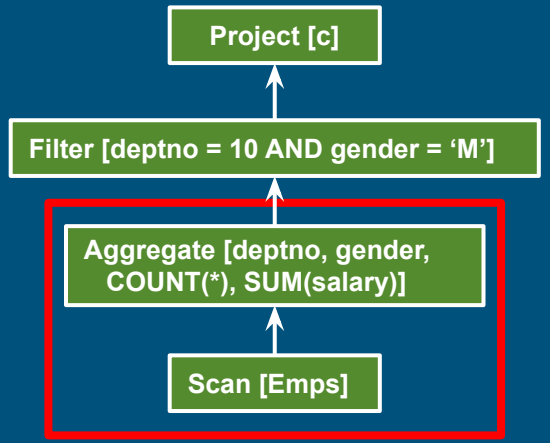

```
SELECT COUNT(*) AS c
FROM Emps
WHERE deptno = 10
AND gender = 'M'
```



Materialized view: rewrite query to match

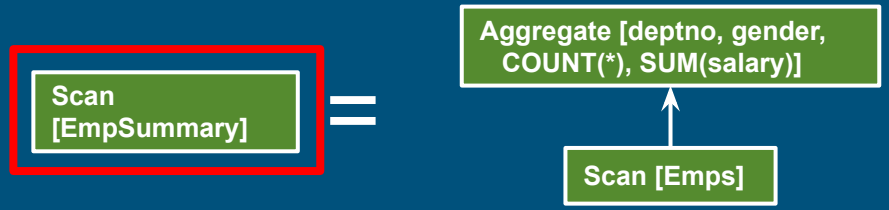
```
CREATE MATERIALIZED VIEW
  EmpSummary AS
SELECT deptno, gender,
  COUNT(*) AS c, SUM(sal) AS s
FROM Emps
GROUP BY deptno, gender
```


```
SELECT COUNT(*) AS c
FROM Emps
WHERE deptno = 10
AND gender = 'M'
```

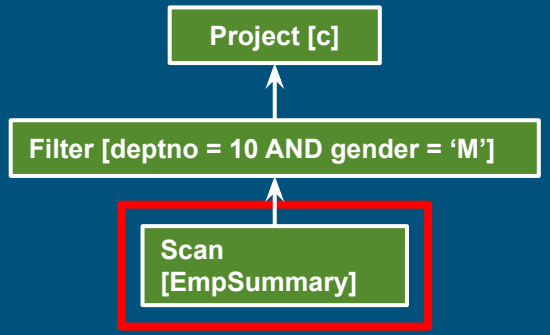


Materialized view: substitute table scan

```
CREATE MATERIALIZED VIEW
  EmpSummary AS
SELECT deptno, gender,
  COUNT(*) AS c, SUM(sal) AS s
FROM Emps
GROUP BY deptno, gender
```



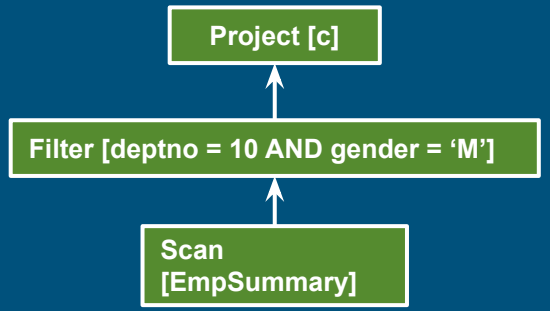

```
SELECT COUNT(*) AS c
FROM Emps
WHERE deptno = 10
AND gender = 'M'
```



Materialized view: substitute table scan

```
CREATE MATERIALIZED VIEW
  EmpSummary AS
SELECT deptno, gender,
  COUNT(*) AS c, SUM(sal) AS s
FROM Emps
GROUP BY deptno, gender
```


```
SELECT c
FROM EmpSummary
WHERE deptno = 10
AND gender = 'M'
```



4. Analytics

“orders” view in LookML

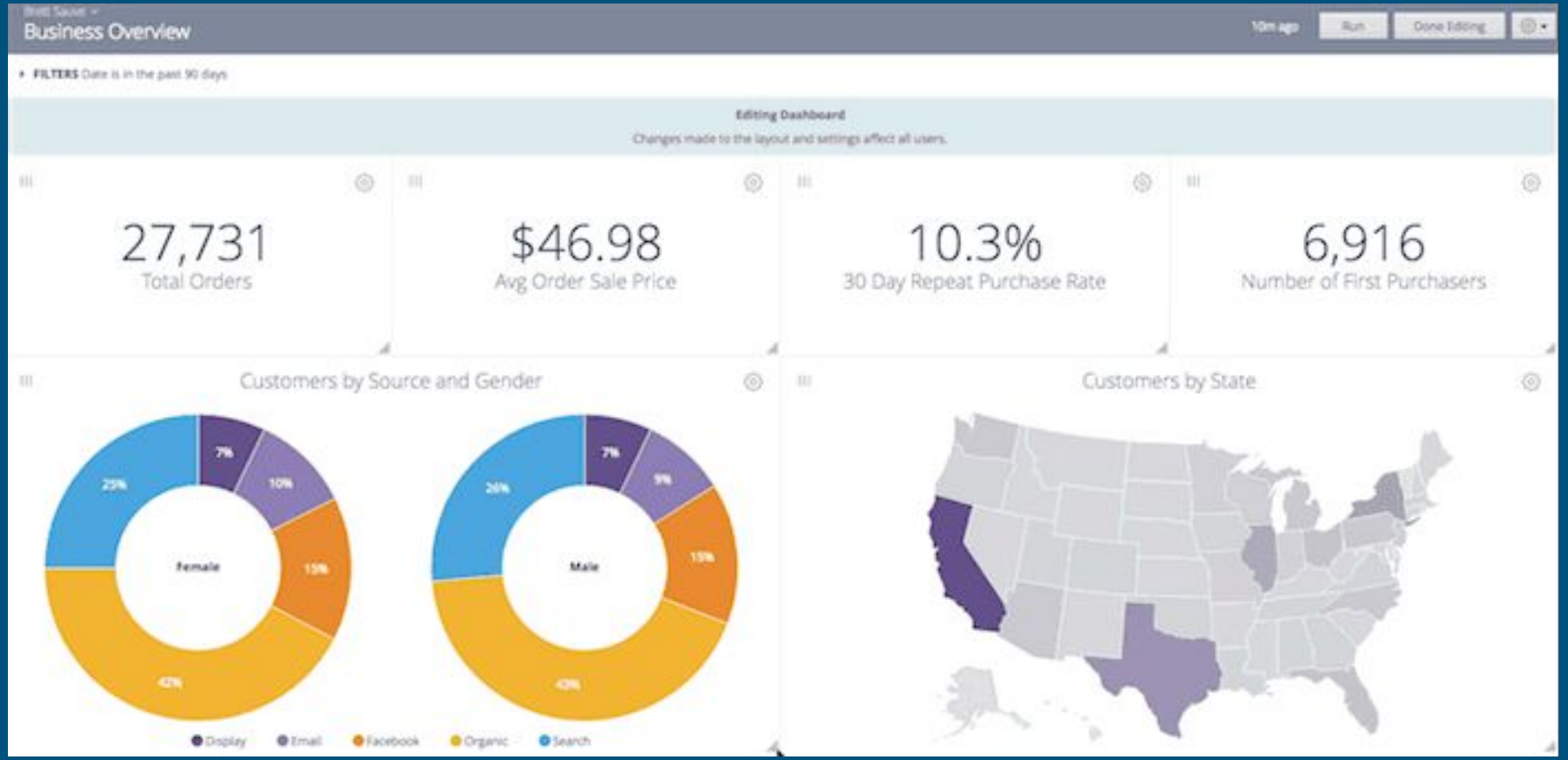
```
view: orders {
  dimension: id {
    primary_key: yes
    type: number
    sql: ${TABLE}.id ;;
  }

  dimension: customer_id {       # field: orders.customer_id
    sql: ${TABLE}.customer_id ;;
  }

  dimension: amount {            # field: orders.amount
    type: number
    value_format: "0.00"
    sql: ${TABLE}.amount ;;
  }

  measure: count {               # field: orders.count
    type: count                  # creates a sql COUNT(*)
  }

  measure: total_amount {
    type: sum
    sql: ${amount} ;;
  }
}
```


Business Overview
10m ago
Run
Done Editing
FILTERS Date is in the past 90 days
Editing Dashboard
Changes made to the layout and settings affect all users.
27,731
Total Orders
$46.98
Avg Order Sale Price
10.3%
30 Day Repeat Purchase Rate
6,916
Number of First Purchasers
Customers by Source and Gender
7%
10%
15%
42%
25%
female
7%
9%
15%
43%
26%
Male
Display
Email
Facebook
Organic
Search
Customers by State

Orders by Date and Category

US Pacific (America - Los Angeles) · 31 rows · from cache · 2m ago

Run

Order Items

Search

All Fields | Dimensions | Measures

- Inventory Items
- Order Facts
- Order Items

FILTER-ONLY FIELDS

Brand Select

DIMENSIONS

Brand Comparitor
Gross Margin
Gross Margin Tier
Id
Item Gross Margin Percentage
Item Gross Margin Percentage Tier
Return Date
Returned (Yes / No)
Sale Price

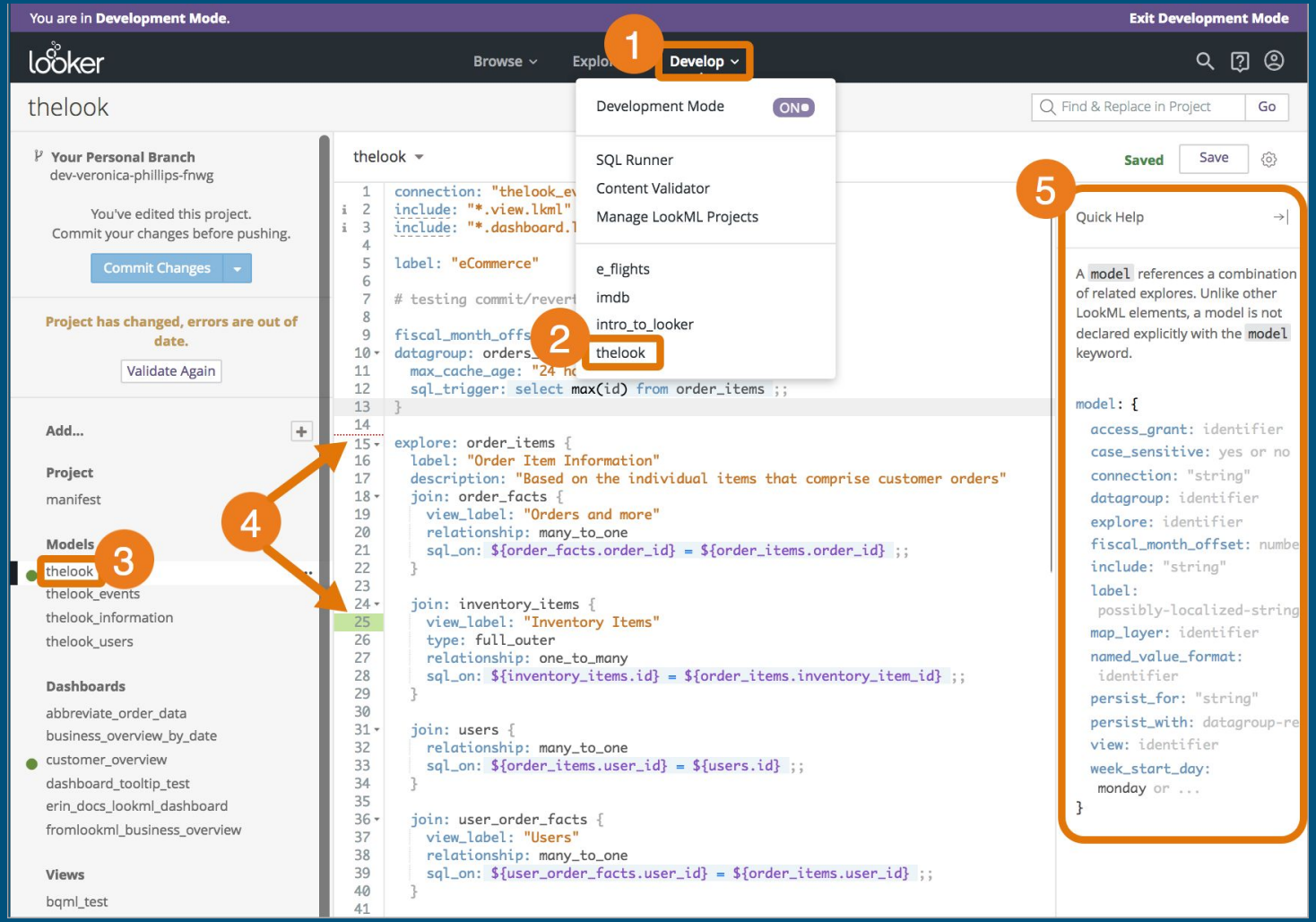
MEASURES

FILTERS Orders **Created Date** "2014/03/01 to 2014/04/01" Products **Category Name** is "Accesso

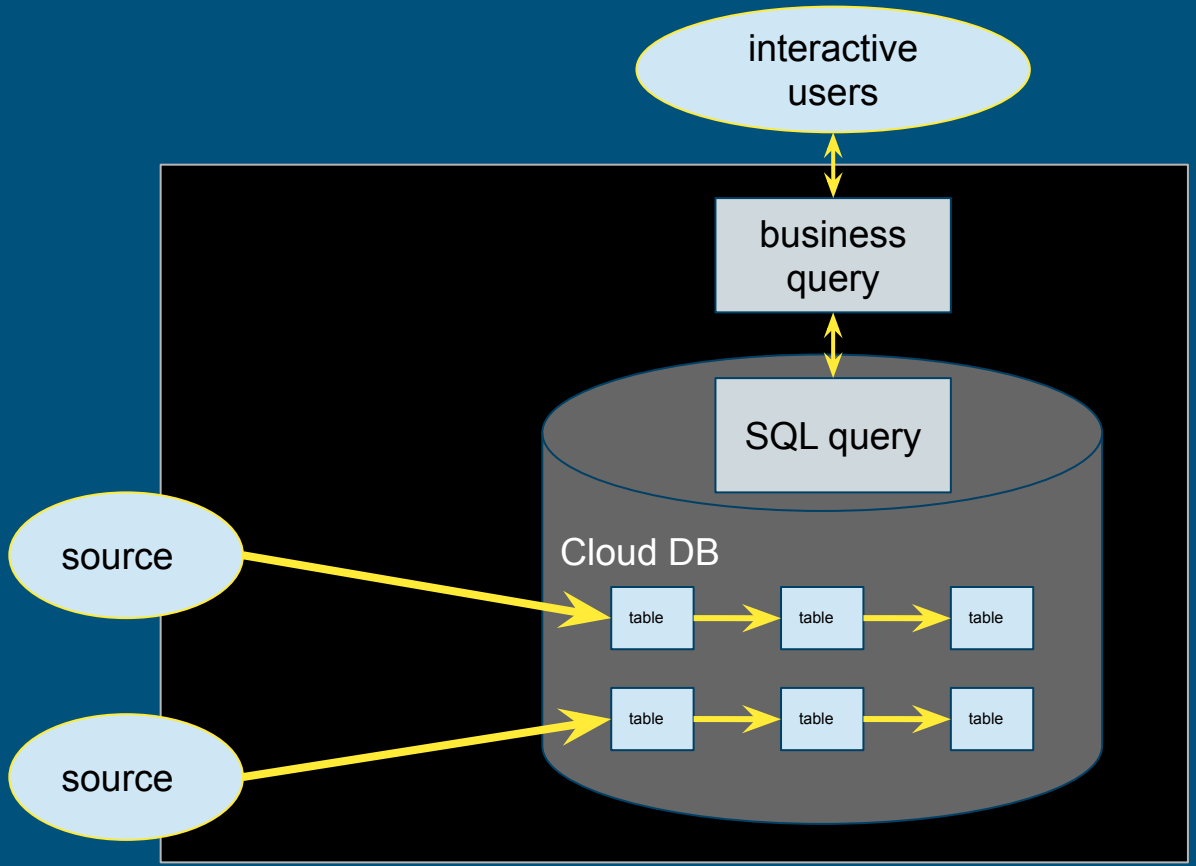
VISUALIZATION EDIT

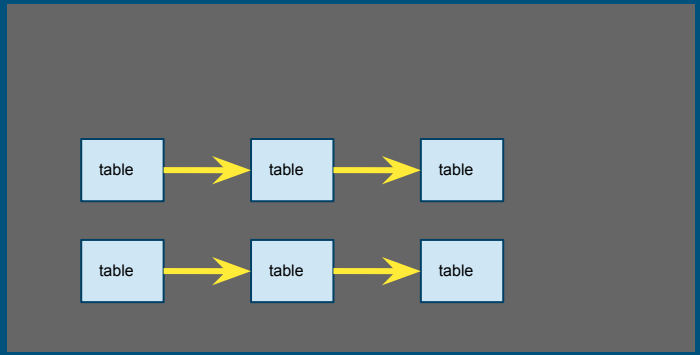
DATA RESULTS SQL

	Products **Category Name**	Accessories	Blazers & Jackets	Fashion Hoodies & Sweatshirts	Pants	Shorts	Sweaters
	Orders **Created Date**	Order Items **Count**	Order Items **Count**	Order Items **Count**	Order Items **Count**	Order Items **Count**	Order Items **Count**
1	2014-03-31	∅	2	3	∅	2	3
2	2014-03-30	1	∅	∅	1	1	∅
3	2014-03-29	1	∅	2	1	2	2
4	2014-03-28	2	3	1	1	1	3

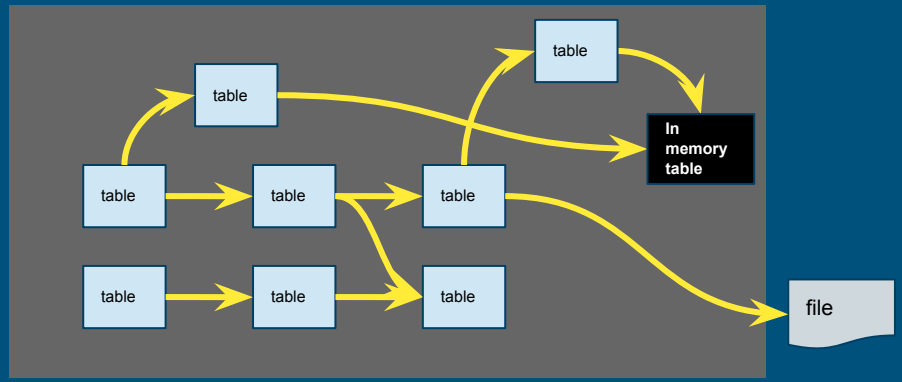

You are in Development Mode.
Exit Development Mode
looker
Browse
Explo
Develop
1
thelook
Development Mode
ON
Find & Replace in Project
Go
SQL Runner
Content Validator
Manage LookML Projects
e_flights
imdb
intro_to_looker
2
thelook
Your Personal Branch
dev-veronica-phillips-fnwg
You've edited this project.
Commit your changes before pushing.
Commit Changes
Project has changed, errors are out of date.
Validate Again
thelook
Saved
Save
5
Quick Help
A model references a combination of related explores. Unlike other LookML elements, a model is not declared explicitly with the model keyword.
model: {
access_grant: identifier
case_sensitive: yes or no
connection: "string"
datagroup: identifier
explore: identifier
fiscal_month_offset: numbe
include: "string"
label:
possibly-localized-string
map_layer: identifier
named_value_format:
identifier
persist_for: "string"
persist_with: datagroup-re
view: identifier
week_start_day:
monday or ...
}
Add...
Project
manifest
Models
3
thelook
thelook_events
thelook_information
thelook_users
4
Dashboards
abbreviate_order_data
business_overview_by_date
customer_overview
dashboard_tooltip_test
erin_docs_lookml_dashboard
fromlookml_business_overview
Views
bqml_test
connection: "thelook_ev
include: "*.view.lkml"
include: "*.dashboard.l
label: "eCommerce"
testing commit/revert
fiscal_month_offs
datagroup: orders_
max_cache_age: "24 ho
sql_trigger: select max(id) from order_items ;;
}
explore: order_items {
label: "Order Item Information"
description: "Based on the individual items that comprise customer orders"
join: order_facts {
view_label: "Orders and more"
relationship: many_to_one
sql_on: ${order_facts.order_id} = ${order_items.order_id} ;;
}
join: inventory_items {
view_label: "Inventory Items"
type: full_outer
relationship: one_to_many
sql_on: ${inventory_items.id} = ${order_items.inventory_item_id} ;;
}
join: users {
relationship: many_to_one
sql_on: ${order_items.user_id} = ${users.id} ;;
}
join: user_order_facts {
view_label: "Users"
relationship: many_to_one
sql_on: ${user_order_facts.user_id} = ${order_items.user_id} ;;
}

5. Evolving the data pipeline

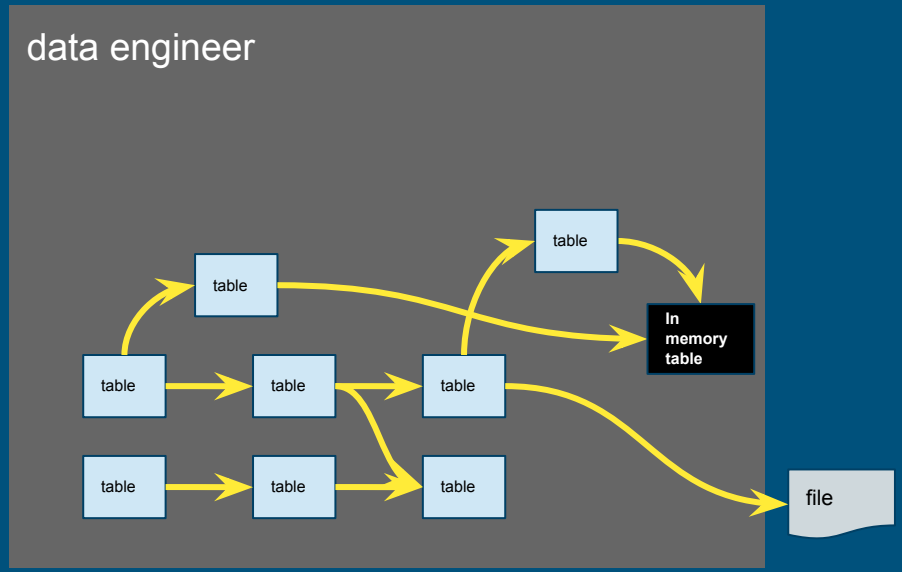

interactive users
business query
SQL query
Cloud DB
source
source
table
table
table
table
table
table



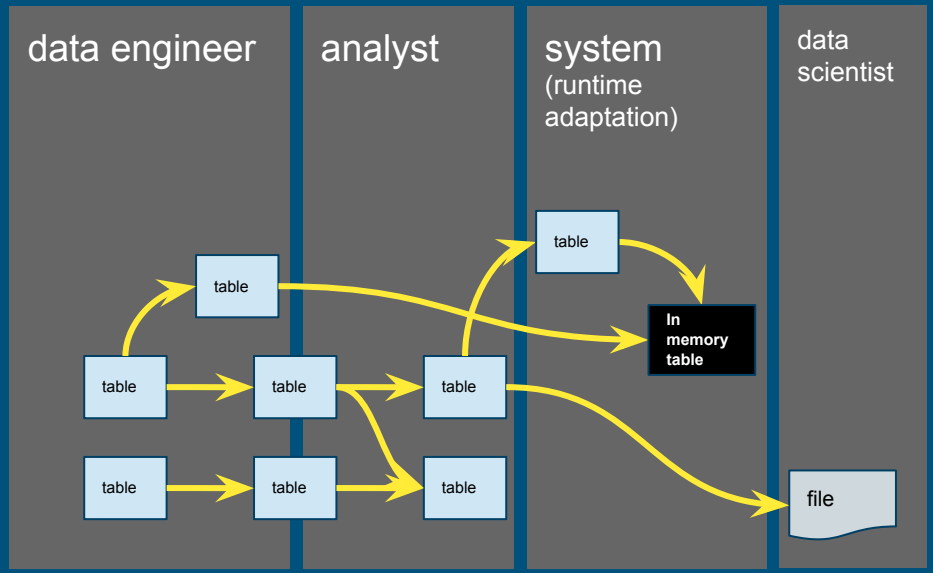
Data engineering



Data engineering is not a static problem



Who is responsible for data engineering?



Data engineering - empower users, reduce friction

LookML - derived table (based on SQL)

```
view: customer_order_facts {
  derived_table: {
    sql:
     SELECT customer_id,
       MIN(DATE(time)) AS first_order_date,
       SUM(amount) AS lifetime_amount
     FROM order
     GROUP BY customer_id ;;
  }

  dimension: customer_id {
    type: number
    primary_key: yes
    sql: ${TABLE}.customer_id ;;
  }

  dimension_group: first_order {
    type: time
    timeframes: [date, week, month]
    sql: ${TABLE}.first_order_date ;;
  }

  dimension: lifetime_amount {
    type: number
    value_format: "0.00"
    sql: ${TABLE}.lifetime_amount ;;
  }
}
```

LookML - derived table (based on an Explore)

```
view: customer_order_facts {
 derived_table: {
   explore_source: orders {
    column: customer_id {
      field: order.customer_id
    }

    column: first_order {
      field: order.first_order
    }

    column: lifetime_amount {
      field: order.lifetime_amount
    }
   }
 }

 dimension: customer_id {
   type: number
   primary_key: yes
   sql: ${TABLE}.customer_id ;;
 }

 dimension_group: first_order {
   type: time
   timeframes: [date, week, month]
   sql: ${TABLE}.first_order_date ;;
 }
```

Flavors of derived table

Derived table flavor	Purpose	SQL equivalent
Ephemeral	Query expansion	CREATE VIEW
Persistent	Query is executed once, used by several queries until it expires	CREATE TABLE AS SELECT
Transparent	Populated as persistent DT, but can be used even if the business query does not reference it by name	CREATE MATERIALIZED VIEW

Each flavor comes can be based on either an Explore or SQL

Building materialized views

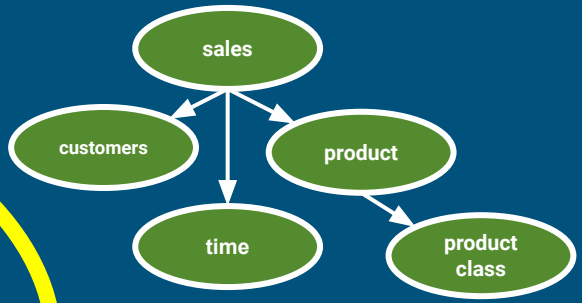
Challenges:

- **Design** Which materializations to create?
- **Populate** Load them with data
- **Maintain** Incrementally populate when data changes
- **Rewrite** Transparently rewrite queries to use materializations
- **Adapt** Design and populate new materializations, drop unused ones
- **Express** Need a rich algebra, to model how data is derived

Initial focus: summary tables (materialized views over star schemas)

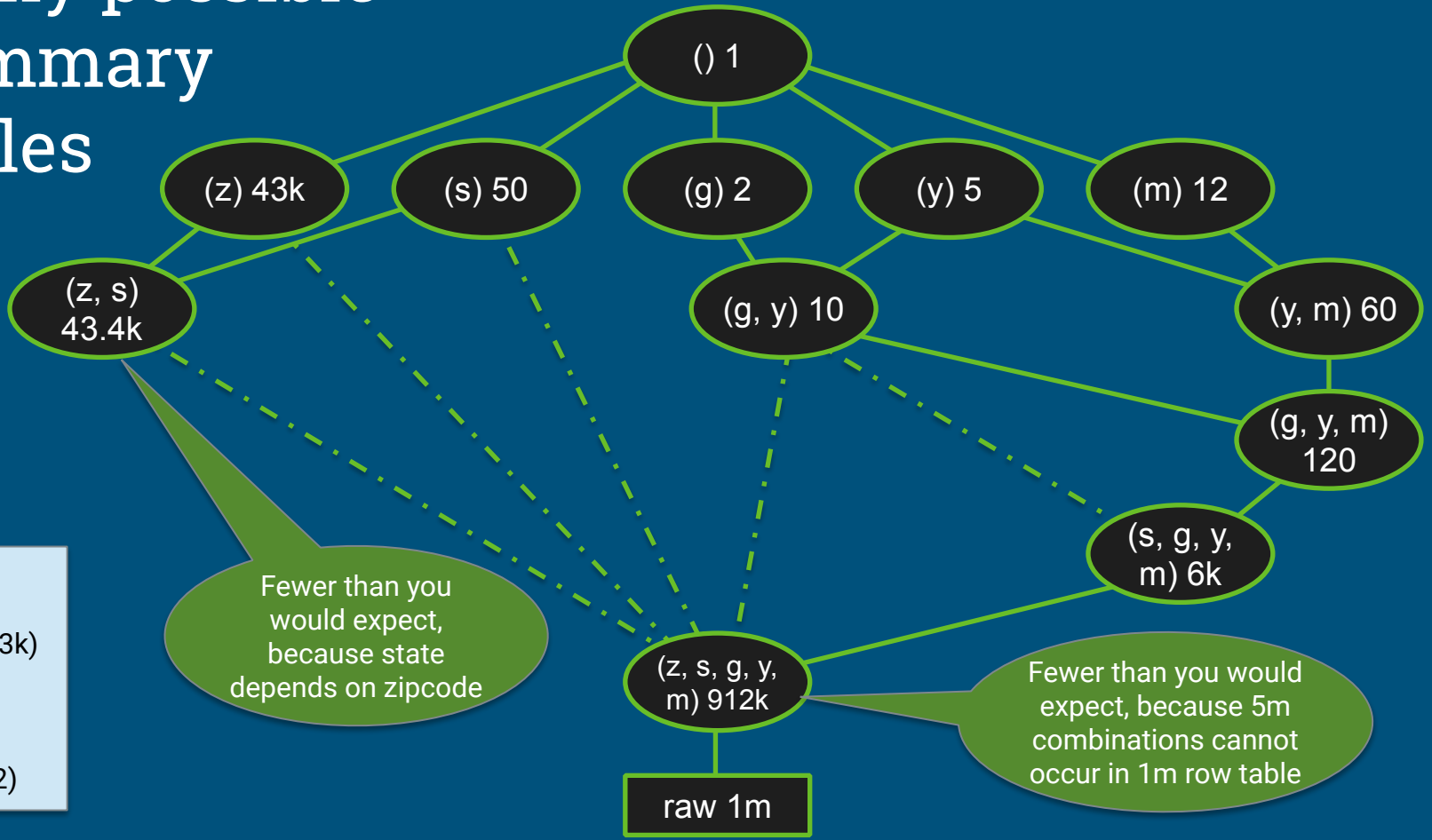
Designing summary tables via lattices

```
CREATE MATERIALIZED VIEW SalesYearZipcode AS
SELECT t.year, c.state, c.zipcode,
  COUNT(*), SUM(units)
FROM Sales AS s
JOIN Time AS t USING (timeId)
JOIN Customers AS c USING (customerId)
GROUP BY 1, 2, 3;
```




```
CREATE LATTICE Sales AS
SELECT t.*, c.*, COUNT(*), SUM(s.units)
FROM Sales AS s
JOIN Time AS t USING (timeId)
JOIN Customers AS c USING (customerId)
JOIN Products AS p USING (productId);
```

Many possible summary tables



Algorithm: Design summary tables

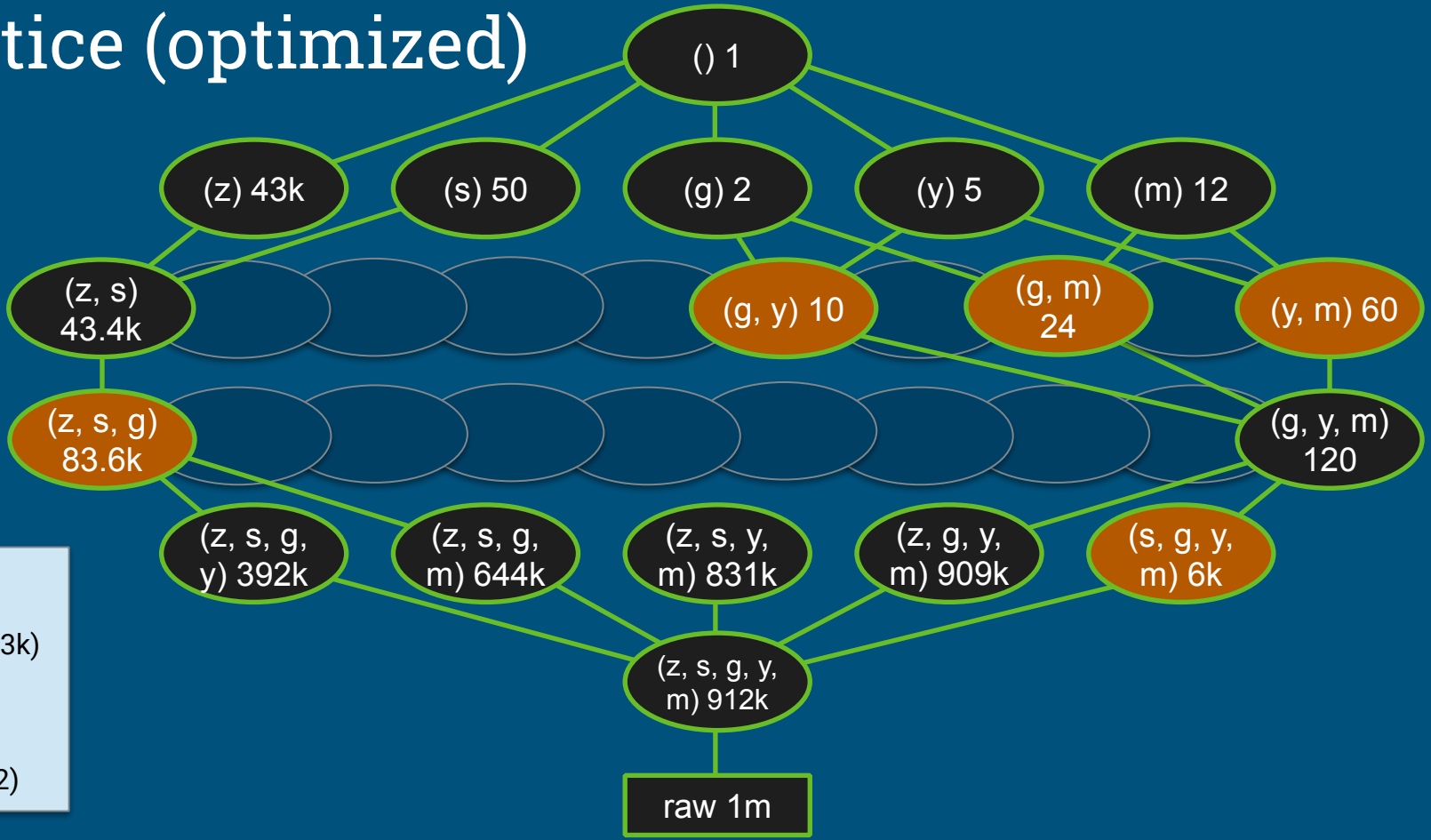
Given a database with 30 columns, 10M rows. Find X summary tables with under Y rows that improve query response time the most.

AdaptiveMonteCarlo algorithm [1]:

- Based on research [2]
- Greedy algorithm that takes a combination of summary tables and tries to find the table that yields the greatest cost/benefit improvement
- Models “benefit” of the table as query time saved over simulated query load
- The “cost” of a table is its size

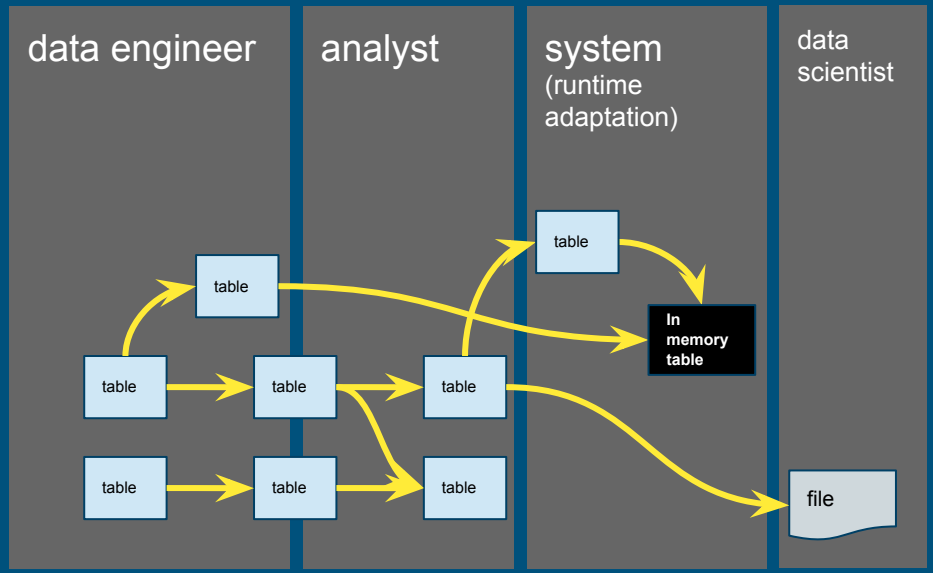
[1] org.pentaho.aggdes.algorithm.impl.AdaptiveMonteCarloAlgorithm
[2] Harinarayan, Rajaraman, Ullman (1996). “Implementing data cubes efficiently”

Lattice (optimized)

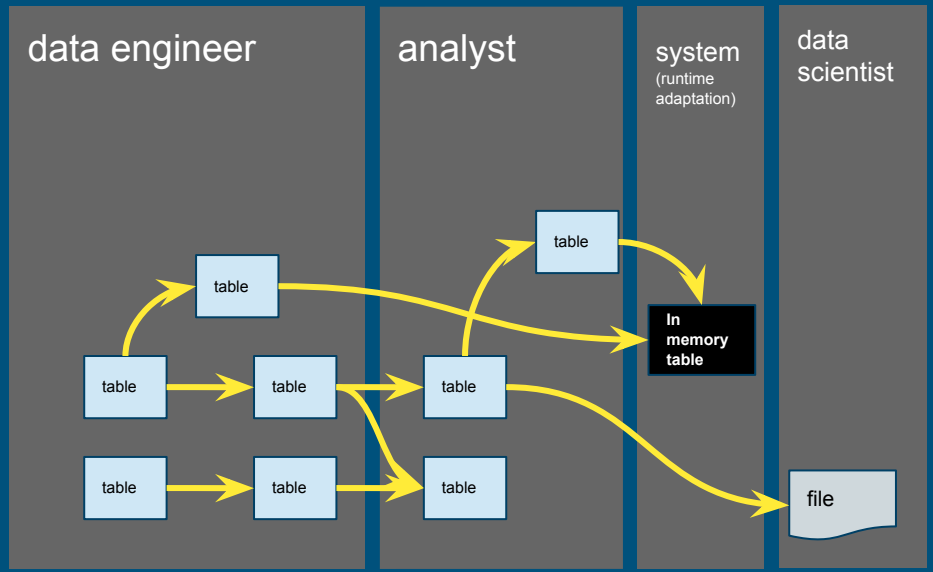


Key

z zipcode (43k)
s state (50)
g gender (2)
y year (5)
m month (12)



Data engineering - empower users, reduce friction



Data engineering - productionize

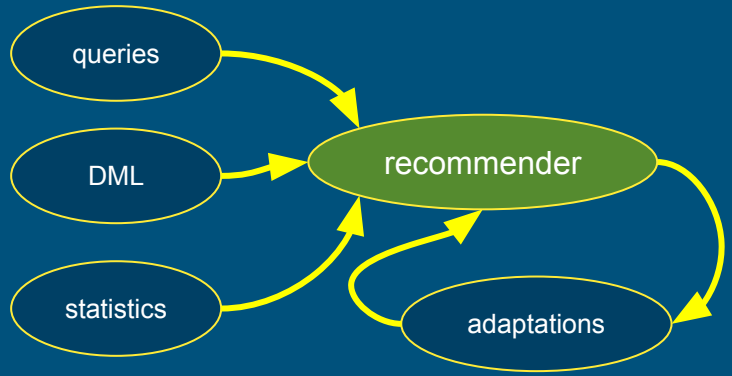
Adaptive data systems

Goals

- Improve response time, throughput, storage cost
- Predictable, adaptive (short and long term), allow human intervention

How?

- Humans
- Adaptive systems
- Smart algorithms



Example adaptations

- Cache disk blocks in memory
- Cached query results
- Data organization, e.g. partition on a different key
- Secondary structures, e.g. b-tree and r-tree indexes

Thank you! Any questions?

@julianhyde
www.looker.com
calcite.apache.org