CS 4604: Introduction to Database Management Systems

Data Mining and Warehousing

Virginia Tech CS 4604 Sprint 2021

Instructor: Yinlin Chen

Today’s Topics

- OLAP: Online Analytical Processing
- Data Mining
- Cloud Database

Introduction

Traditional database systems are tuned to many, small, simple queries

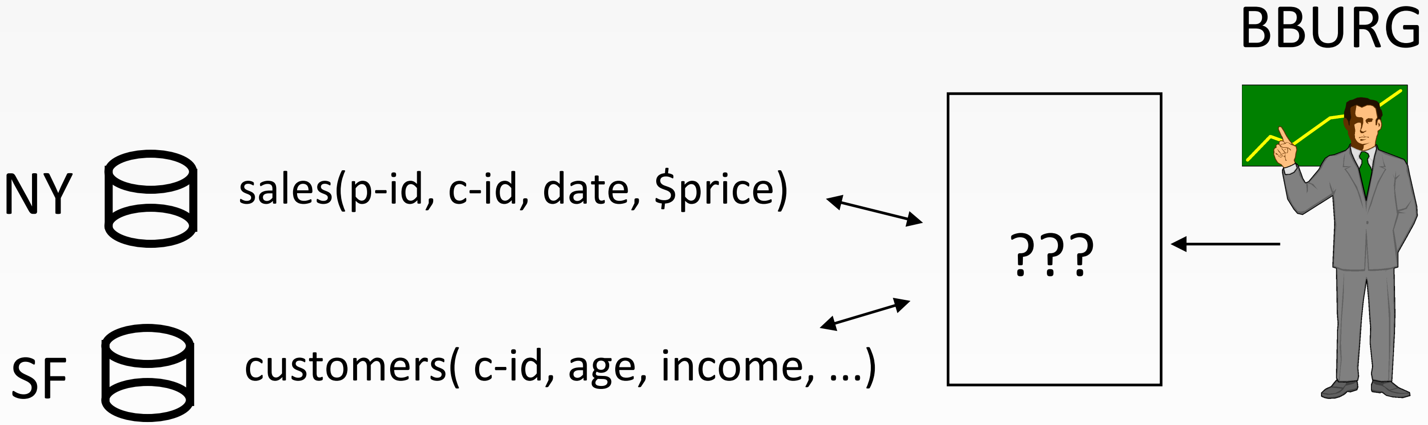
New applications use fewer, more time-consuming, *analytic* queries

New architectures have been developed to handle analytic queries efficiently

Problem

Given: multiple data sources

Find: patterns (classifiers, rules, clusters, outliers...)



BBURG

NY

sales(p-id, c-id, date, $price)

SF

customers(c-id, age, income, ...)

???

Data Ware-housing

Step 1: collect the data, in a single place (= Data Warehouse)

How?

A: Triggers/Materialized views

How often?

A: Depends

How about discrepancies / non-homegeneities?

A: Wrappers/Mediators

Step 2: collect counts. (DataCubes/OLAP)

The Data Warehouse

The most common form of data integration.

- Copy sources into a single DB (*warehouse*) and try to keep it up-to-date.
- Usual method: periodic reconstruction of the warehouse, perhaps overnight.
- Frequently essential for analytic queries.

OLTP

Most database operations involve *On-Line Transaction Processing* (OTLP).

- Short, simple, frequent queries and/or modifications, each involving a small number of tuples.
- Examples: Answering queries from a Web interface, sales at cash registers, selling airline tickets.

OLAP

On-Line Application Processing (OLAP, or “analytic”) queries are, typically:

– Few, but complex queries --- may run for hours.

– Queries do not depend on having an absolutely up-to-date database.

OLAP Examples

1. Amazon analyzes purchases by its customers to come up with an individual screen with products of likely interest to the customer.
2. Analysts at Wal-Mart look for items with increasing sales in some region.
   - Use empty trucks to move merchandise between stores.

Common Architecture

Databases at store branches handle OLTP.

Local store databases copied to a central warehouse overnight.

Analysts use the warehouse for OLAP.

Star Schemas

A *star schema* is a common organization for data at a warehouse. It consists of:

– *Fact table* : a very large accumulation of facts such as sales.

- Often "insert-only."

– *Dimension tables* : smaller, generally static information about the entities involved in the facts.

Example: Star Schema

Suppose we want to record in a warehouse information about every beer sale: the bar, the brand of beer, the drinker who bought the beer, the day, the time, and the price charged.

The fact table is a relation:

Sales(bar, beer, drinker, day, time, price)

Example -- Continued

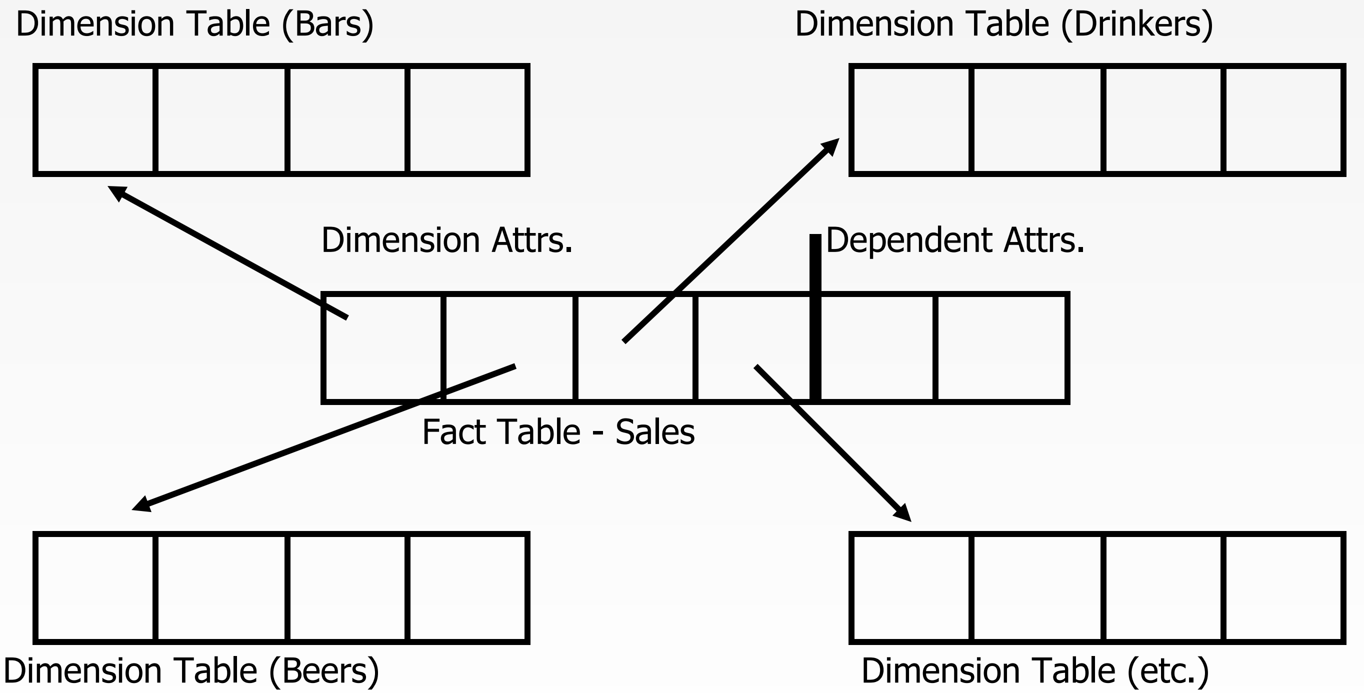
The dimension tables include information about the bar, beer, and drinker “dimensions”:

Bars(bar, addr, license)

Beers(beer, manf)

Drinkers(drinker, addr, phone)

Visualization – Star Schema



Dimension Table (Bars)

Dimension Table (Drinkers)

Dimension Attrs.

Dependent Attrs.

Fact Table - Sales

Dimension Table (Beers)

Dimension Table (etc.)

Dimensions and Dependent Attributes

Two classes of fact-table attributes:

1. *Dimension attributes* : the key of a dimension table.
2. *Dependent attributes* : a value determined by the dimension attributes of the tuple.

Example: Dependent Attribute

price is the dependent attribute of our example Sales relation.

It is determined by the combination of dimension attributes: bar, beer, drinker, and the time (combination of day and time-of-day attributes).

Approaches to Building Warehouses

1. *ROLAP* = “relational OLAP”: Tune a relational DBMS to support star schemas.
2. *MOLAP* = “multidimensional OLAP”: Use a specialized DBMS with a model such as the “data cube.”

ROLAP Techniques

1. *Bitmap indexes* : For each key value of a dimension table (e.g., each beer for relation Beers) create a bit-vector telling which tuples of the fact table have that value.
2. *Materialized views* : Store the answers to several useful queries (views) in the warehouse itself.

Typical OLAP Queries

Often, OLAP queries begin with a “star join”: the natural join of the fact table with all or most of the dimension tables.

Example:

```
SELECT *
FROM Sales, Bars, Beers, Drinkers
WHERE Sales.bar = Bars.bar AND
  Sales.beer = Beers.beer AND
  Sales.drinker = Drinkers.drinker;
```

Typical OLAP Queries --- (2)

The typical OLAP query will:

1. Start with a star join.
2. Select for interesting tuples, based on dimension data.
3. Group by one or more dimensions.
4. Aggregate certain attributes of the result.

Example: OLAP Query

For each bar in Blacksburg, find the total sale of each beer manufactured by Anheuser-Busch.

Filter: addr = “Blacksburg” and manf = “Anheuser-Busch”.

Grouping: by bar and beer.

Aggregation: Sum of price.

Example: In SQL

```
SELECT bar, beer, SUM(price)
FROM Sales NATURAL JOIN Bars
 NATURAL JOIN Beers
WHERE addr = ’Blacksburg’ AND
 manf = ’Anheuser-Busch’
GROUP BY bar, beer;
```

Using Materialized Views

A direct execution of this query from Sales and the dimension tables could take too long.

If we create a materialized view that contains enough information, we may be able to answer our query much **faster**.

Example: Materialized View

Which views could help with our query?

Key issues:

1. It must join Sales, Bars, and Beers, at least.
2. It must group by at least bar and beer.
3. It must not select out Blacksburg bars or Anheuser-Busch beers.
4. It must not project out addr or manf.

Example --- Continued

Here is a materialized view that could help:

```
CREATE VIEW BABMS(bar, addr,
   beer, manf, sales) AS
SELECT bar, addr, beer, manf,
   SUM(price) sales
FROM Sales NATURAL JOIN Bars
   NATURAL JOIN Beers
GROUP BY bar, addr, beer, manf;
```

Since bar -> addr and beer -> manf, there is no real grouping.   We need addr and manf in the SELECT.

Example --- Concluded

Here's our query using the materialized view BABMS:

```
SELECT bar, beer, sales
FROM BABMS
WHERE addr = 'Blacksburg' AND
   manf = 'Anheuser-Busch';
```

MOLAP and Data Cubes

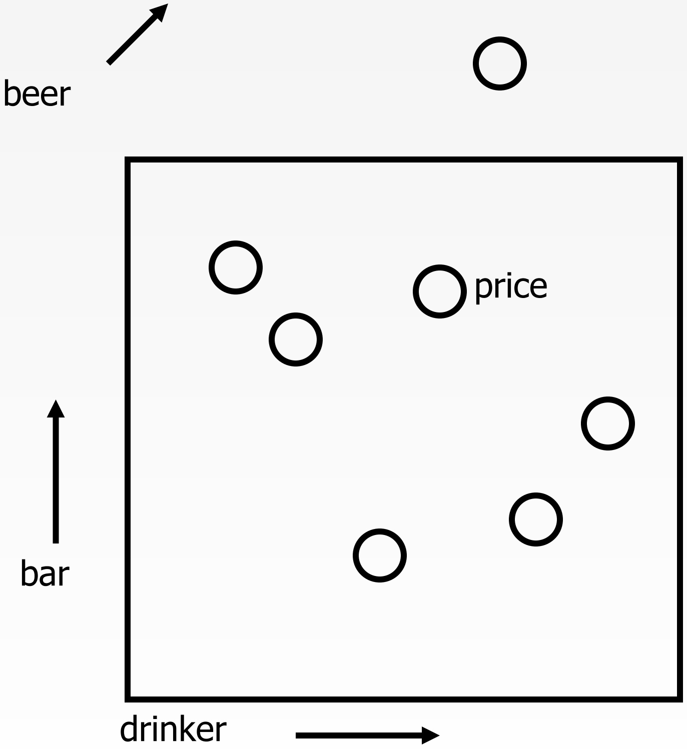
Keys of dimension tables are the dimensions of a hypercube. Example:

Sales(bar, beer, drinker, time, price)

– for the Sales data, the four dimensions are bar, beer, drinker, and time.

Dependent attributes (e.g., price) appear at the points of the cube.

Visualization -- Data Cubes

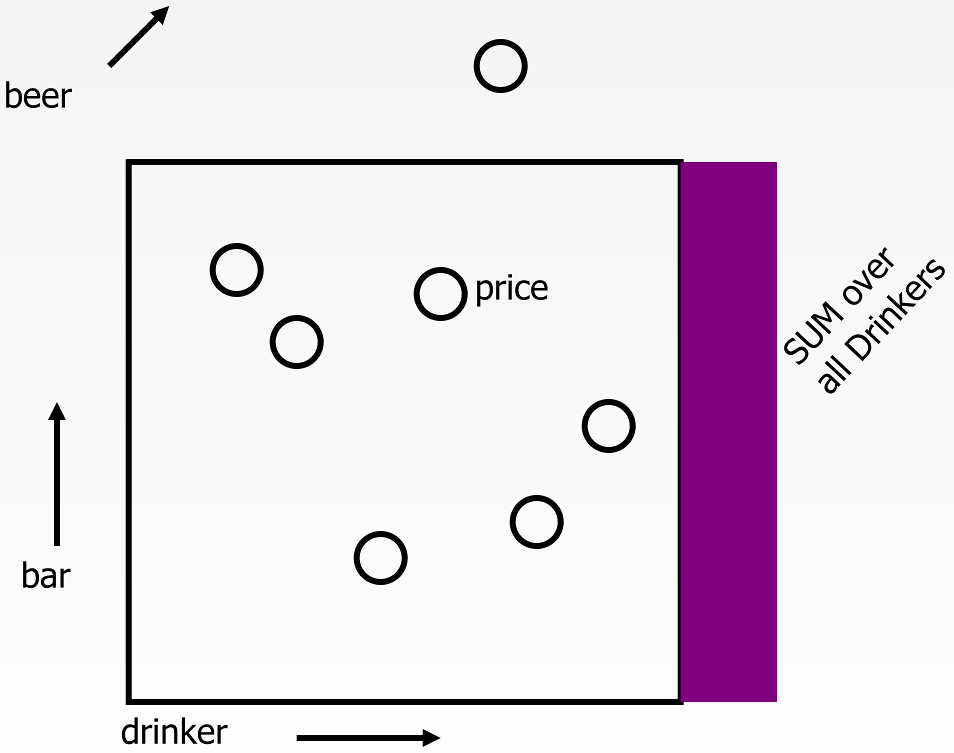


Marginals

The data cube also includes aggregation (typically SUM) along the margins of the cube.

The *marginals* include aggregations over one dimension, two dimensions,…

Visualization --- Data Cube w/Aggregation



Example: Marginals

Our 4-dimensional Sales cube includes the sum of price over each bar, each beer, each drinker, and each time unit (perhaps days).

It would also have the sum of price over all bar-beer pairs, all bar-drinker-day triples,…

Marginals

Think of each dimension as having an additional value *.

A point with one or more *'s in its coordinates aggregates over the dimensions with the *'s.

Example: ("Joe's Bar", "Bud", *, *) holds the sum, over all drinkers and all time, of the Bud consumed at Joe's.

Drill-Down

Drill-down = “de-aggregate” = break an aggregate into its constituents.

Example: having determined that Joe’s Bar sells very few Anheuser-Busch beers, break down his sales by particular A.-B. beer.

Roll-Up

Roll-up = aggregate along one or more dimensions.

Example: given a table of how much Bud each drinker consumes at each bar, roll it up into a table giving total amount of Bud consumed by each drinker.

Example: Roll Up and Drill Down

$ of Anheuser-Busch by drinker/bar

	Jim	Bob	Mary
Joe's Bar	45	33	30
Bull & Bones	50	36	42
Blue Chalk	38	31	40

Roll up by Bar

$ of A-B / drinker

Jim	Bob	Mary
133	100	112

Drill down by Beer

$ of A-B Beers / drinker

	Jim	Bob	Mary
Bud	40	29	40
M'lob	45	31	37
Bud Light	48	40	35

Structure of the Data Cube

CUBE(F) of fact table F is *roughly* === the Fact table (F) + aggregations across all dimensions (i.e. marginals)

– Note CUBE(F) is a relation itself!

CUBE in SQL: Example

For our Sales example:

Sales(bar, beer, drinker, time, price)

```
CREATE MATERIALIZED VIEW SalesCube AS
 SELECT bar, beer, drinker, time, SUM(price)
 FROM Sales
 GROUP BY bar, beer, drinker, time WITH CUBE;
```

Tuples in SalesCube

Tuples implied by the standard GROUP-BY:

```
(Joes, Bud, John, 4/19/13, 3.00)
```

And those tuples of that are constructed by rolling-up the dimensions in GROUP-BY (== marginals, NULL == *). E.g:

```
(Joes, NULL, John, 4/19/13, 10.00)
(Joes, NULL, John, NULL, 200.00)
(Joes, NULL, NULL, NULL, 200000.00)
(NULL, NULL, NULL, NULL, 2000000.00)
```

Tuples in SalesCube

Tuples implied by the standard GROUP-BY:

```
(Joes, Bud, John, 4/19/13, 3.00)
```

And those tuples of that are constructed by rolling-up the dimensions in GROUP-BY (== marginals, NULL == *). E.g:

```
(Joes, NULL, John, 4/19/13, 10.00)
(Joes, NULL, John, NULL, 200.00)
(Joes, NULL, NULL, NULL, 200000.00)
(NULL, NULL, NULL, NULL, 2000000.00)
```

Total spent by John at Joes on Apr 19.

Tuples in SalesCube

Tuples implied by the standard GROUP-BY:

```
(Joes, Bud, John, 4/19/13, 3.00)
```

And those tuples of that are constructed by rolling-up the dimensions in GROUP-BY (== marginals, NULL == *). E.g:

```
(Joes, NULL, John, 4/19/13, 10.00)
(Joes, NULL, John, NULL, 200.00)
(Joes, NULL, NULL, NULL, 200000.00)
(NULL, NULL, NULL, NULL, 2000000.00)
```

Total spent by John at Joes ever.

Tuples in SalesCube

Tuples implied by the standard GROUP-BY:

```
(Joes, Bud, John, 4/19/13, 3.00)
```

And those tuples of that are constructed by rolling-up the dimensions in GROUP-BY (== marginals, NULL == *). E.g:

```
(Joes, NULL, John, 4/19/13, 10.00)
(Joes, NULL, John, NULL, 200.00)
(Joes, NULL, NULL, NULL, 200000.00)
(NULL, NULL, NULL, NULL, 2000000.00)
```

Total spent by everyone at Joes ever.

Tuples in SalesCube

Tuples implied by the standard GROUP-BY:

```
(Joes, Bud, John, 4/19/13, 3.00)
```

And those tuples of that are constructed by rolling-up the dimensions in GROUP-BY (== marginals, NULL == *). E.g:

```
(Joes, NULL, John, 4/19/13, 10.00)
(Joes, NULL, John, NULL, 200.00)
(Joes, NULL, NULL, NULL, 200000.00)
(NULL, NULL, NULL, NULL, 2000000.00)
```

Total spent by everyone at every bar ever.

Compare ROLAP vs MOLAP

ROLAP Solution

```
CREATE VIEW BABMS(bar, addr,
          beer, manf, sales) AS
SELECT bar, addr, beer, manf,
     SUM(price) sales
FROM Sales NATURAL JOIN Bars
          NATURAL JOIN Beers
GROUP BY bar, addr, beer, manf;
```

A specific view for a specific type of query (note the join)

MOLAP (Data Cube) Solution

```
CREATE MATERIALIZED VIEW SalesCube AS
SELECT bar, beer, drinker, time,
   SUM(price)
FROM Sales
GROUP BY bar, beer, drinker, time WITH
   CUBE;
```

A generalized view which stores marginals as well (no join)

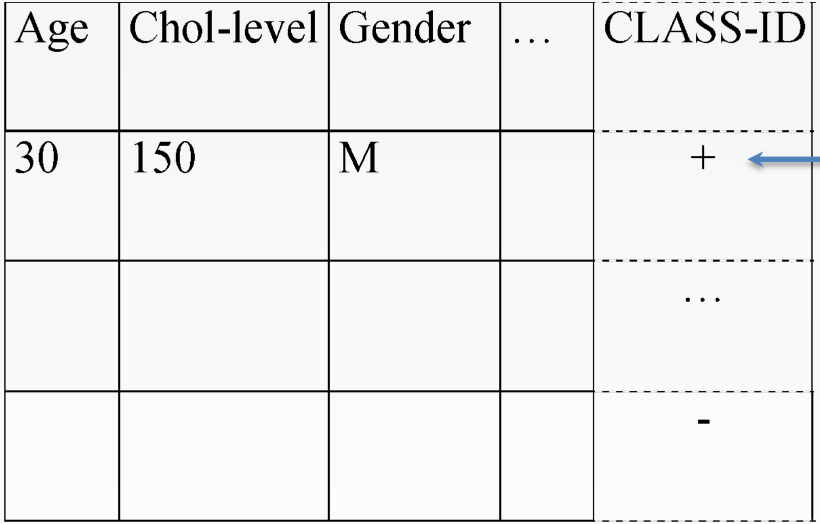
Data Mining

Data mining is a popular term for techniques to summarize big data sets in useful ways.

Examples:

1. Clustering all Web pages by topic.
2. Finding characteristics of fraudulent credit-card use.

Supervised Learning: Decision Trees: Problem



Age	Chol-level	Gender	…	CLASS-ID
30	150	M		+
				…
				-

Has heart disease

Supervised Learning: Decision Trees: Problem

Age	Chol-level	Gender	…	CLASS-ID
30	150	M		+
				…
				-
15	90	F	…	??

What is the label for this new patient?

Supervised Learning: Decision Trees: Problem

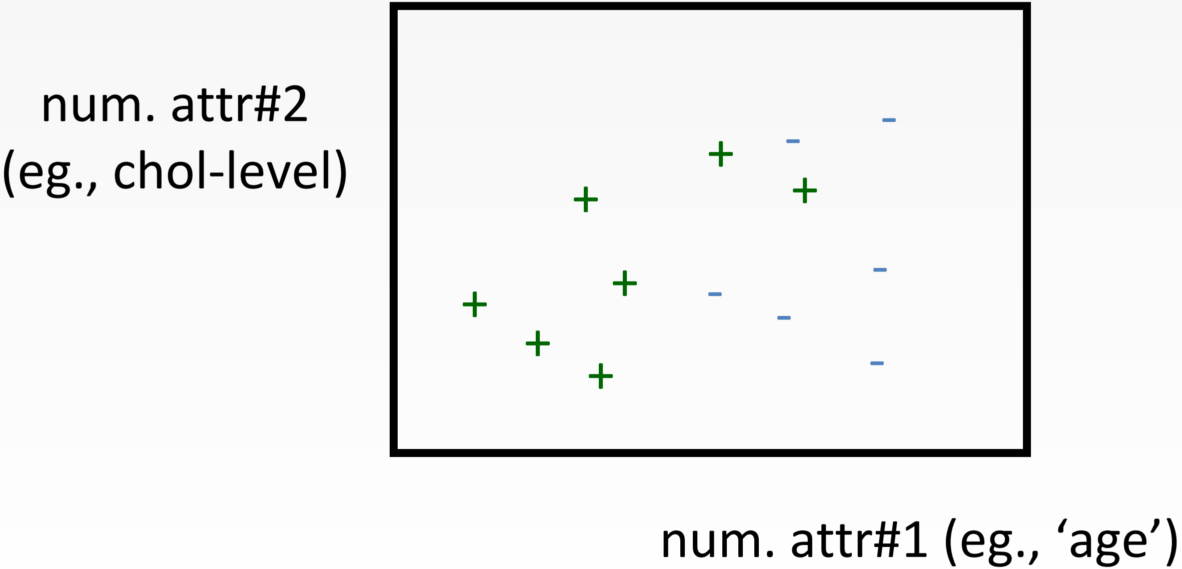
Training Data Set

Age	Chol-level	Gender	…	CLASS-ID
30	150	M		+
				…
				-
15	90	F	…	??

What is the label for this new patient?

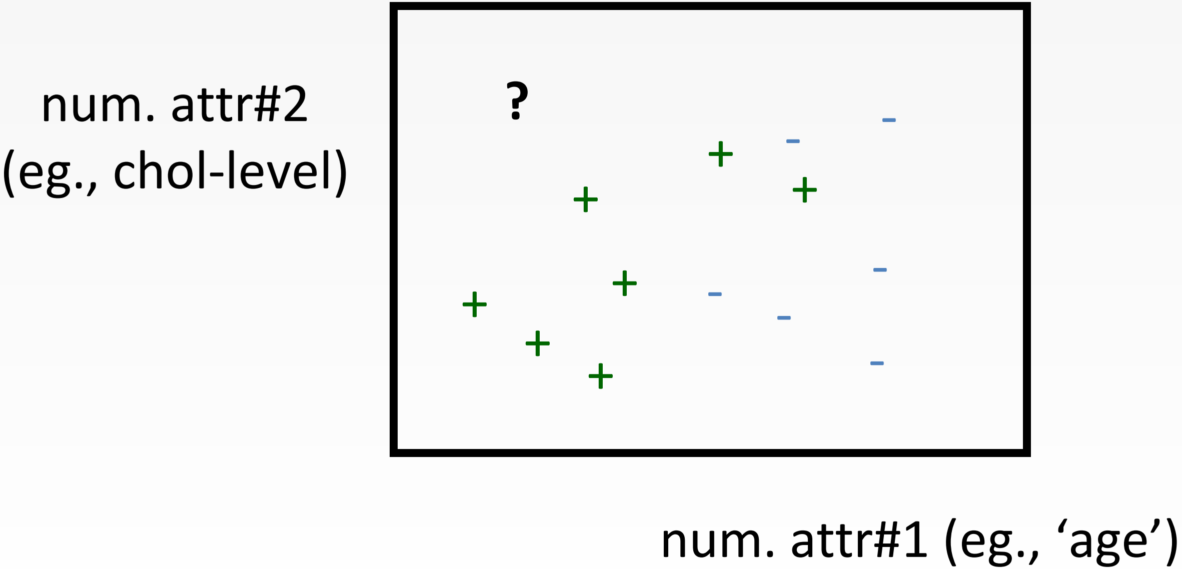
Decision trees

Pictorially, we have



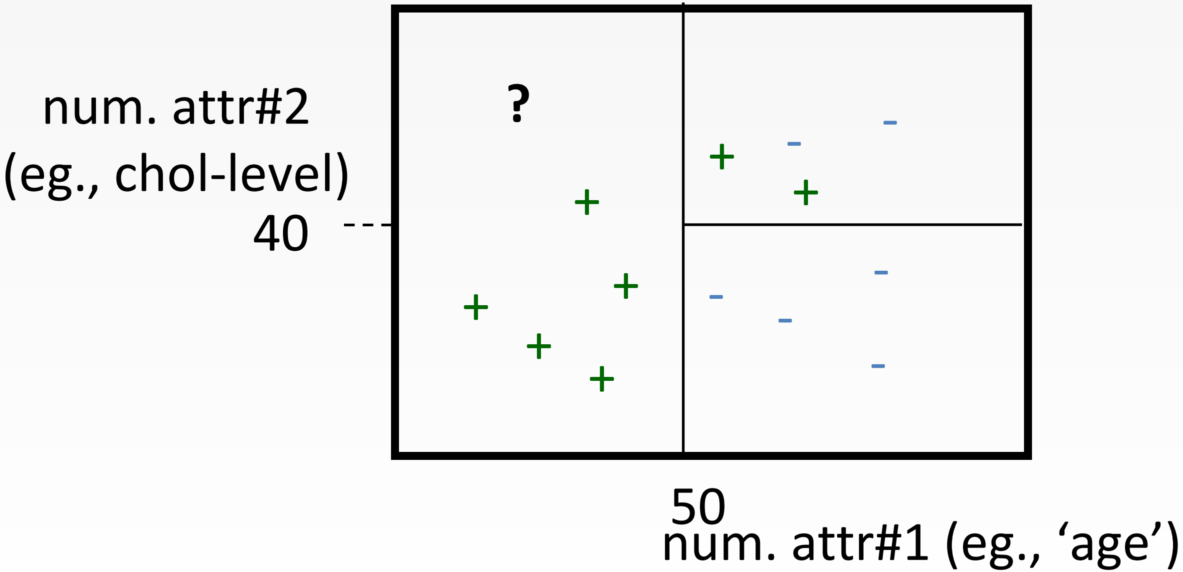
Decision trees

and we want to label ‘**?**’



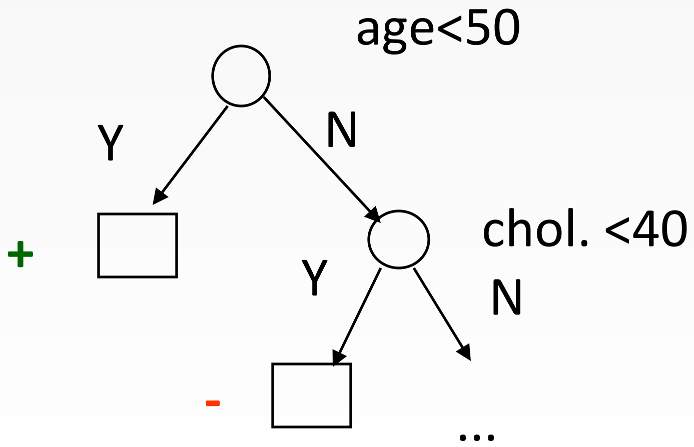
Decision trees

so we build a decision tree:



Decision trees

so we build a decision tree:



Conclusions

Data Mining: of high commercial interest (think BIG data)

DM = DB + Machine Learning + Stats.

Data Warehousing/OLAP: to get the data

Tree Classifiers

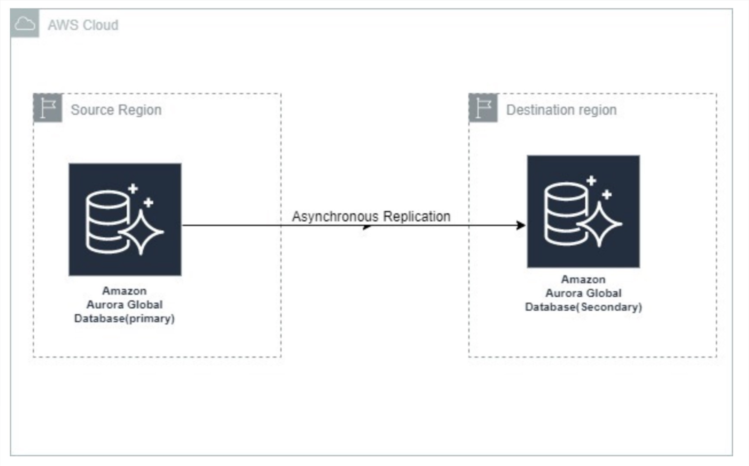
Association Rules

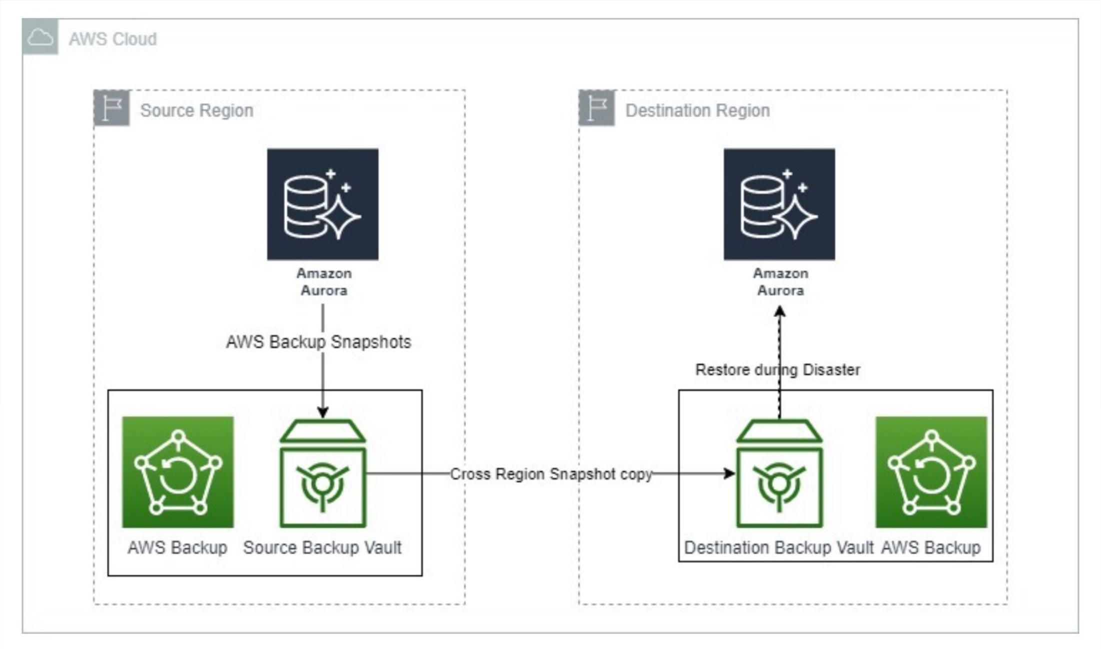
….. (like clustering etc.)

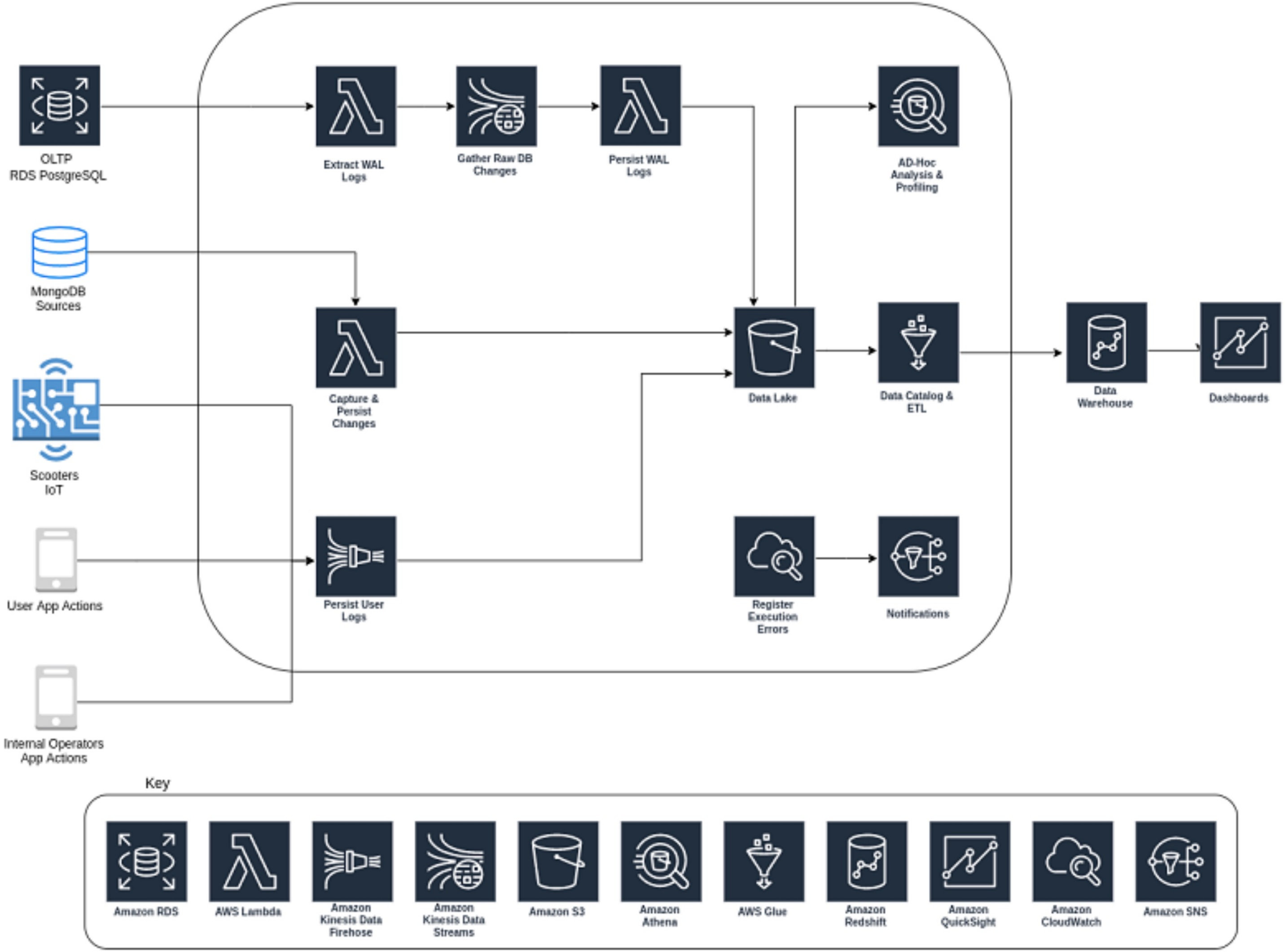
Summary

- For OLAP, column-oriented storage trumps row-oriented storage
  - Many tricks beyond splitting columns up
  - compression, late materialization, redundant layouts, efficient write processing
- For OLTP, the costs of many random accesses for updating columns makes a columnar layout not worth it
  - Hence OLTP systems look more traditional, and typically opt for row-oriented storage
- Many systems are now opting for hybrid layouts to try to support both OLAP and OLTP in the same system

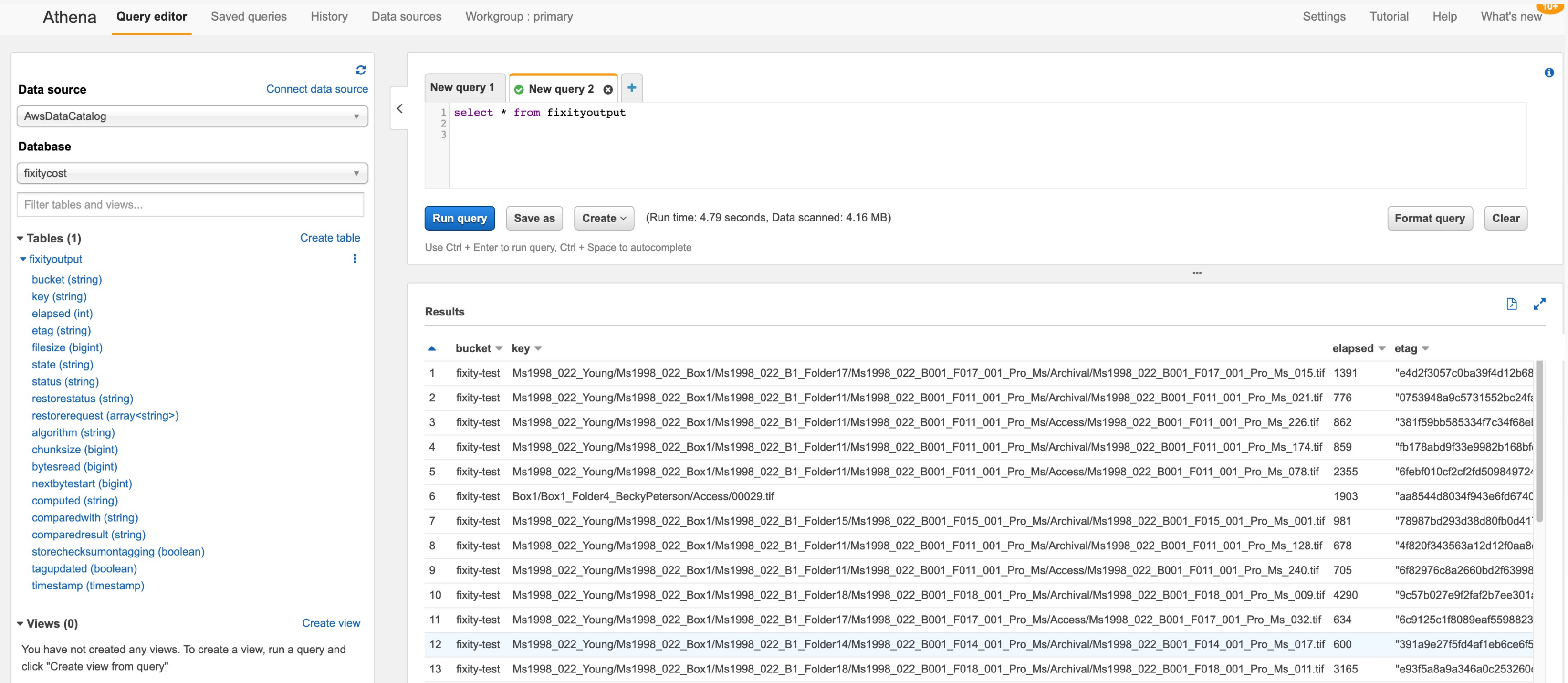
Cloud Database


AWS Cloud
Source Region
Destination region
Asynchronous Replication
Amazon Aurora Global Database(primary)
Amazon Aurora Global Database(Secondary)




OLTP
RDS PostgreSQL
MongoDB
Sources
Scooters
IoT
User App Actions
Internal Operators
App Actions
Extract WAL
Logs
Gather Raw DB
Changes
Persist WAL
Logs
AD-Hoc
Analysis &
Profiling
Capture &
Persist
Changes
Data Lake
Data Catalog &
ETL
Data
Warehouse
Dashboards
Persist User
Logs
Register
Execution
Errors
Notifications
Key
Amazon RDS
AWS Lambda
Amazon
Kinesis Data
Firehose
Amazon
Kinesis Data
Streams
Amazon S3
Amazon
Athena
AWS Glue
Amazon
Redshift
Amazon
QuickSight
Amazon
CloudWatch
Amazon SNS

AWS Athena



Data source: AwsDataCatalog

Database: fixitycost

Tables (1)

fixityoutput

- bucket (string)
- key (string)
- elapsed (int)
- etag (string)
- filesize (bigint)
- state (string)
- status (string)
- restorestatus (string)
- restorerequest (array<string>)
- algorithm (string)
- chunksize (bigint)
- bytesread (bigint)
- nextbytestart (bigint)
- computed (string)
- comparedwith (string)
- comparedresult (string)
- storechecksumontagging (boolean)
- tagupdated (boolean)
- timestamp (timestamp)

Views (0)

You have not created any views. To create a view, run a query and click "Create view from query"

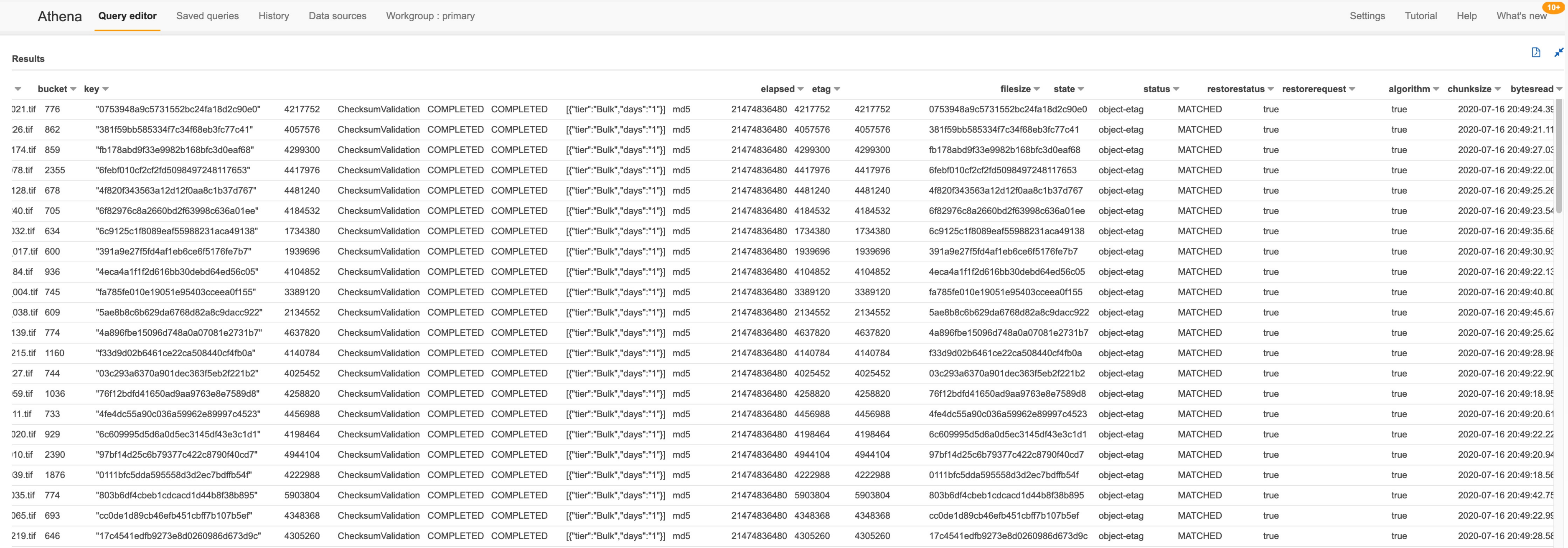
```
select * from fixityoutput
```

(Run time: 4.79 seconds, Data scanned: 4.16 MB)

Results

	bucket	key	elapsed	etag
1	fixity-test	Ms1998_022_Young/Ms1998_022_Box1/Ms1998_022_B1_Folder17/Ms1998_022_B001_F017_001_Pro_Ms/Archival/Ms1998_022_B001_F017_001_Pro_Ms_015.tif	1391	"e4d2f3057c0ba39f4d12b68
2	fixity-test	Ms1998_022_Young/Ms1998_022_Box1/Ms1998_022_B1_Folder11/Ms1998_022_B001_F011_001_Pro_Ms/Archival/Ms1998_022_B001_F011_001_Pro_Ms_021.tif	776	"0753948a9c5731552bc24f
3	fixity-test	Ms1998_022_Young/Ms1998_022_Box1/Ms1998_022_B1_Folder11/Ms1998_022_B001_F011_001_Pro_Ms/Access/Ms1998_022_B001_F011_001_Pro_Ms_226.tif	862	"381f59bb585334f7c34f68e
4	fixity-test	Ms1998_022_Young/Ms1998_022_Box1/Ms1998_022_B1_Folder11/Ms1998_022_B001_F011_001_Pro_Ms/Archival/Ms1998_022_B001_F011_001_Pro_Ms_174.tif	859	"fb178abd9f33e9982b168bf
5	fixity-test	Ms1998_022_Young/Ms1998_022_Box1/Ms1998_022_B1_Folder11/Ms1998_022_B001_F011_001_Pro_Ms/Access/Ms1998_022_B001_F011_001_Pro_Ms_078.tif	2355	"6febf010cf2cf2fd50984972
6	fixity-test	Box1/Box1_Folder4_BeckyPeterson/Access/00029.tif	1903	"aa8544d8034f943e6fd6740
7	fixity-test	Ms1998_022_Young/Ms1998_022_Box1/Ms1998_022_B1_Folder15/Ms1998_022_B001_F015_001_Pro_Ms/Archival/Ms1998_022_B001_F015_001_Pro_Ms_001.tif	981	"78987bd293d38d80fb0d41
8	fixity-test	Ms1998_022_Young/Ms1998_022_Box1/Ms1998_022_B1_Folder11/Ms1998_022_B001_F011_001_Pro_Ms/Archival/Ms1998_022_B001_F011_001_Pro_Ms_128.tif	678	"4f820f343563a12d12f0aa8
9	fixity-test	Ms1998_022_Young/Ms1998_022_Box1/Ms1998_022_B1_Folder11/Ms1998_022_B001_F011_001_Pro_Ms/Access/Ms1998_022_B001_F011_001_Pro_Ms_240.tif	705	"6f82976c8a2660bd2f63998
10	fixity-test	Ms1998_022_Young/Ms1998_022_Box1/Ms1998_022_B1_Folder18/Ms1998_022_B001_F018_001_Pro_Ms/Archival/Ms1998_022_B001_F018_001_Pro_Ms_009.tif	4290	"9c57b027e9f2faf2b7ee301
11	fixity-test	Ms1998_022_Young/Ms1998_022_Box1/Ms1998_022_B1_Folder17/Ms1998_022_B001_F017_001_Pro_Ms/Access/Ms1998_022_B001_F017_001_Pro_Ms_032.tif	634	"6c9125c1f8089eaf5598823
12	fixity-test	Ms1998_022_Young/Ms1998_022_Box1/Ms1998_022_B1_Folder14/Ms1998_022_B001_F014_001_Pro_Ms/Archival/Ms1998_022_B001_F014_001_Pro_Ms_017.tif	600	"391a9e27f5fd4af1eb6ce6f5
13	fixity-test	Ms1998_022_Young/Ms1998_022_Box1/Ms1998_022_B1_Folder18/Ms1998_022_B001_F018_001_Pro_Ms/Archival/Ms1998_022_B001_F018_001_Pro_Ms_011.tif	3165	"e93f5a8a9a346a0c253260

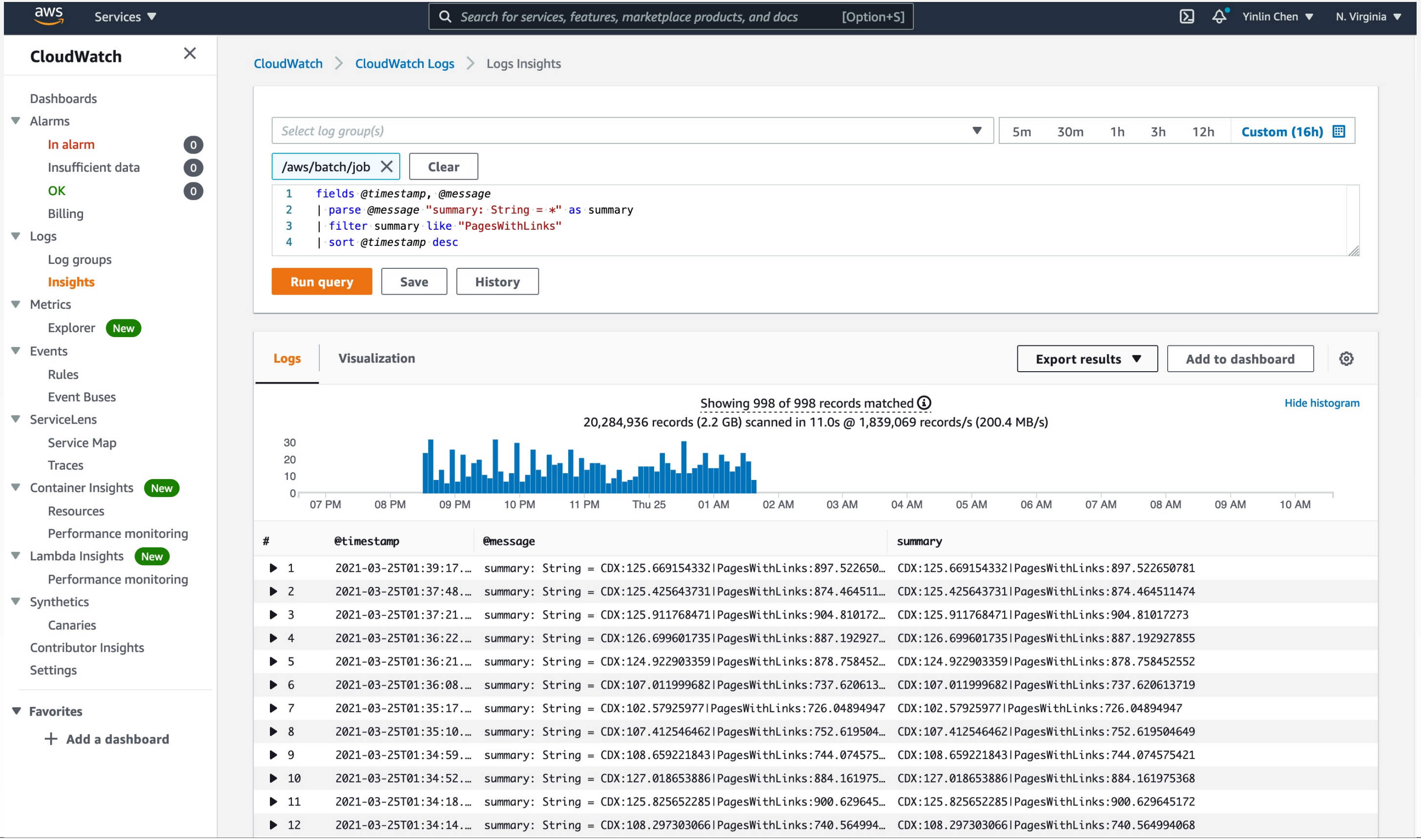
AWS Athena



Results

	bucket	key							elapsed	etag		filesize	state	status	restorestatus	restorerequest	algorithm	chunksize	bytesread
021.tif	776	"0753948a9c5731552bc24fa18d2c90e0"	4217752	ChecksumValidation	COMPLETED	COMPLETED	[{"tier":"Bulk","days":"1"}]	md5	21474836480	4217752	4217752	0753948a9c5731552bc24fa18d2c90e0	object-etag	MATCHED	true		true	2020-07-16 20:49:24.39	
26.tif	862	"381f59bb585334f7c34f68eb3fc77c41"	4057576	ChecksumValidation	COMPLETED	COMPLETED	[{"tier":"Bulk","days":"1"}]	md5	21474836480	4057576	4057576	381f59bb585334f7c34f68eb3fc77c41	object-etag	MATCHED	true		true	2020-07-16 20:49:21.11	
174.tif	859	"fb178abd9f33e9982b168bfc3d0eaf68"	4299300	ChecksumValidation	COMPLETED	COMPLETED	[{"tier":"Bulk","days":"1"}]	md5	21474836480	4299300	4299300	fb178abd9f33e9982b168bfc3d0eaf68	object-etag	MATCHED	true		true	2020-07-16 20:49:27.03	
78.tif	2355	"6febf010cf2cf2fd5098497248117653"	4417976	ChecksumValidation	COMPLETED	COMPLETED	[{"tier":"Bulk","days":"1"}]	md5	21474836480	4417976	4417976	6febf010cf2cf2fd5098497248117653	object-etag	MATCHED	true		true	2020-07-16 20:49:22.00	
128.tif	678	"4f820f343563a12d12f0aa8c1b37d767"	4481240	ChecksumValidation	COMPLETED	COMPLETED	[{"tier":"Bulk","days":"1"}]	md5	21474836480	4481240	4481240	4f820f343563a12d12f0aa8c1b37d767	object-etag	MATCHED	true		true	2020-07-16 20:49:25.26	
40.tif	705	"6f82976c8a2660bd2f63998c636a01ee"	4184532	ChecksumValidation	COMPLETED	COMPLETED	[{"tier":"Bulk","days":"1"}]	md5	21474836480	4184532	4184532	6f82976c8a2660bd2f63998c636a01ee	object-etag	MATCHED	true		true	2020-07-16 20:49:23.54	
032.tif	634	"6c9125c1f8089eaf55988231aca49138"	1734380	ChecksumValidation	COMPLETED	COMPLETED	[{"tier":"Bulk","days":"1"}]	md5	21474836480	1734380	1734380	6c9125c1f8089eaf55988231aca49138	object-etag	MATCHED	true		true	2020-07-16 20:49:35.68	
017.tif	600	"391a9e27f5fd4af1eb6ce6f5176fe7b7"	1939696	ChecksumValidation	COMPLETED	COMPLETED	[{"tier":"Bulk","days":"1"}]	md5	21474836480	1939696	1939696	391a9e27f5fd4af1eb6ce6f5176fe7b7	object-etag	MATCHED	true		true	2020-07-16 20:49:30.93	
84.tif	936	"4eca4a1f1f2d616bb30debd64ed56c05"	4104852	ChecksumValidation	COMPLETED	COMPLETED	[{"tier":"Bulk","days":"1"}]	md5	21474836480	4104852	4104852	4eca4a1f1f2d616bb30debd64ed56c05	object-etag	MATCHED	true		true	2020-07-16 20:49:22.13	
004.tif	745	"fa785fe010e19051e95403cceea0f155"	3389120	ChecksumValidation	COMPLETED	COMPLETED	[{"tier":"Bulk","days":"1"}]	md5	21474836480	3389120	3389120	fa785fe010e19051e95403cceea0f155	object-etag	MATCHED	true		true	2020-07-16 20:49:40.80	
038.tif	609	"5ae8b8c6b629da6768d82a8c9dacc922"	2134552	ChecksumValidation	COMPLETED	COMPLETED	[{"tier":"Bulk","days":"1"}]	md5	21474836480	2134552	2134552	5ae8b8c6b629da6768d82a8c9dacc922	object-etag	MATCHED	true		true	2020-07-16 20:49:45.67	
139.tif	774	"4a896fbe15096d748a0a07081e2731b7"	4637820	ChecksumValidation	COMPLETED	COMPLETED	[{"tier":"Bulk","days":"1"}]	md5	21474836480	4637820	4637820	4a896fbe15096d748a0a07081e2731b7	object-etag	MATCHED	true		true	2020-07-16 20:49:25.62	
215.tif	1160	"f33d9d02b6461ce22ca508440cf4fb0a"	4140784	ChecksumValidation	COMPLETED	COMPLETED	[{"tier":"Bulk","days":"1"}]	md5	21474836480	4140784	4140784	f33d9d02b6461ce22ca508440cf4fb0a	object-etag	MATCHED	true		true	2020-07-16 20:49:28.98	
27.tif	744	"03c293a6370a901dec363f5eb2f221b2"	4025452	ChecksumValidation	COMPLETED	COMPLETED	[{"tier":"Bulk","days":"1"}]	md5	21474836480	4025452	4025452	03c293a6370a901dec363f5eb2f221b2	object-etag	MATCHED	true		true	2020-07-16 20:49:22.90	
59.tif	1036	"76f12bdfd41650ad9aa9763e8e7589d8"	4258820	ChecksumValidation	COMPLETED	COMPLETED	[{"tier":"Bulk","days":"1"}]	md5	21474836480	4258820	4258820	76f12bdfd41650ad9aa9763e8e7589d8	object-etag	MATCHED	true		true	2020-07-16 20:49:18.95	
11.tif	733	"4fe4dc55a90c036a59962e89997c4523"	4456988	ChecksumValidation	COMPLETED	COMPLETED	[{"tier":"Bulk","days":"1"}]	md5	21474836480	4456988	4456988	4fe4dc55a90c036a59962e89997c4523	object-etag	MATCHED	true		true	2020-07-16 20:49:20.61	
020.tif	929	"6c609995d5d6a0d5ec3145df43e3c1d1"	4198464	ChecksumValidation	COMPLETED	COMPLETED	[{"tier":"Bulk","days":"1"}]	md5	21474836480	4198464	4198464	6c609995d5d6a0d5ec3145df43e3c1d1	object-etag	MATCHED	true		true	2020-07-16 20:49:22.22	
10.tif	2390	"97bf14d25c6b79377c422c8790f40cd7"	4944104	ChecksumValidation	COMPLETED	COMPLETED	[{"tier":"Bulk","days":"1"}]	md5	21474836480	4944104	4944104	97bf14d25c6b79377c422c8790f40cd7	object-etag	MATCHED	true		true	2020-07-16 20:49:20.94	
39.tif	1876	"0111bfc5dda595558d3d2ec7bdffb54f"	4222988	ChecksumValidation	COMPLETED	COMPLETED	[{"tier":"Bulk","days":"1"}]	md5	21474836480	4222988	4222988	0111bfc5dda595558d3d2ec7bdffb54f	object-etag	MATCHED	true		true	2020-07-16 20:49:18.56	
035.tif	774	"803b6df4cbeb1cdcacd1d44b8f38b895"	5903804	ChecksumValidation	COMPLETED	COMPLETED	[{"tier":"Bulk","days":"1"}]	md5	21474836480	5903804	5903804	803b6df4cbeb1cdcacd1d44b8f38b895	object-etag	MATCHED	true		true	2020-07-16 20:49:42.75	
065.tif	693	"cc0de1d89cb46efb451cbff7b107b5ef"	4348368	ChecksumValidation	COMPLETED	COMPLETED	[{"tier":"Bulk","days":"1"}]	md5	21474836480	4348368	4348368	cc0de1d89cb46efb451cbff7b107b5ef	object-etag	MATCHED	true		true	2020-07-16 20:49:22.99	
219.tif	646	"17c4541edfb9273e8d0260986d673d9c"	4305260	ChecksumValidation	COMPLETED	COMPLETED	[{"tier":"Bulk","days":"1"}]	md5	21474836480	4305260	4305260	17c4541edfb9273e8d0260986d673d9c	object-etag	MATCHED	true		true	2020-07-16 20:49:28.58	

AWS CloudWatch Insight



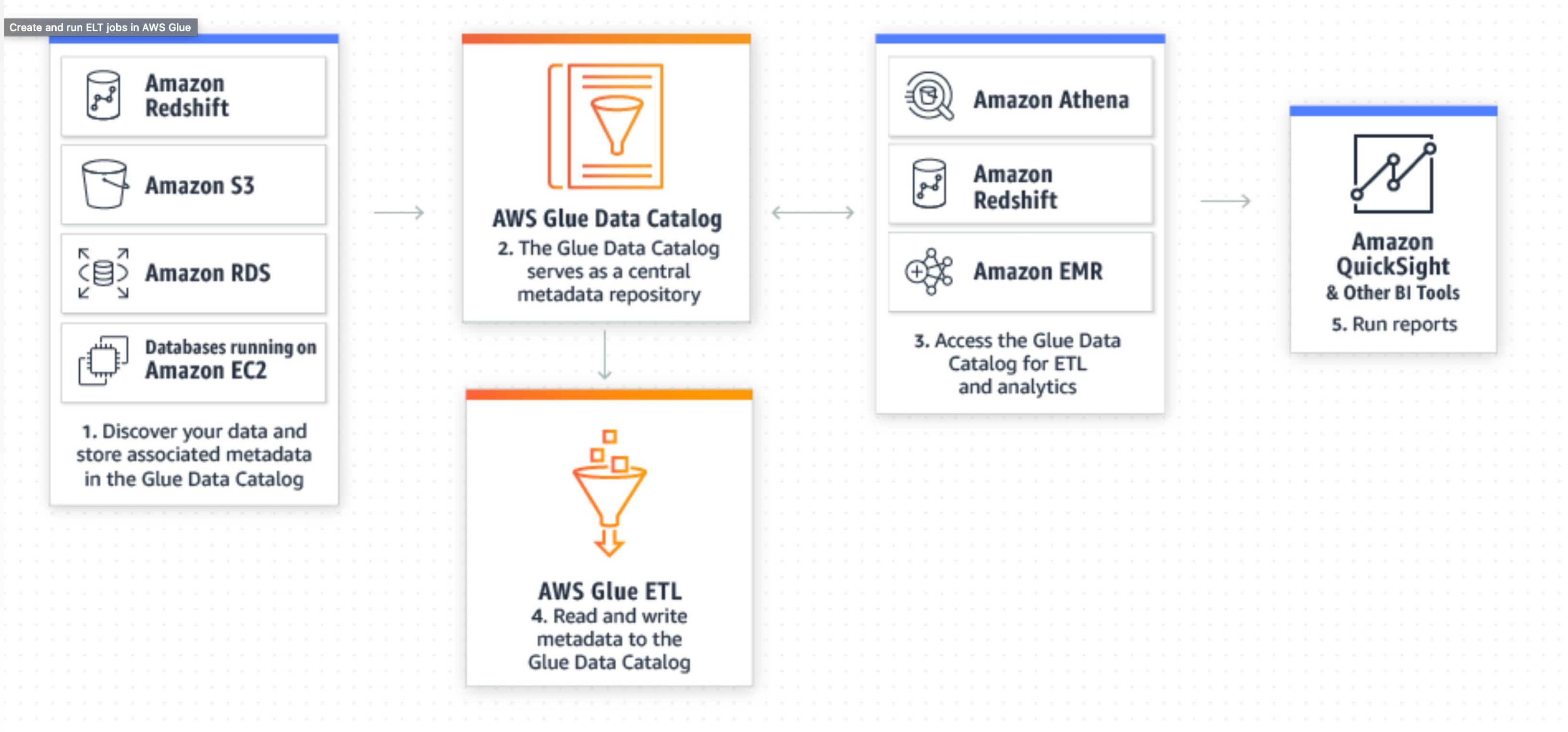
```
fields @timestamp, @message
| parse @message "summary: String = *" as summary
| filter summary like "PagesWithLinks"
| sort @timestamp desc
```

Showing 998 of 998 records matched

20,284,936 records (2.2 GB) scanned in 11.0s @ 1,839,069 records/s (200.4 MB/s)

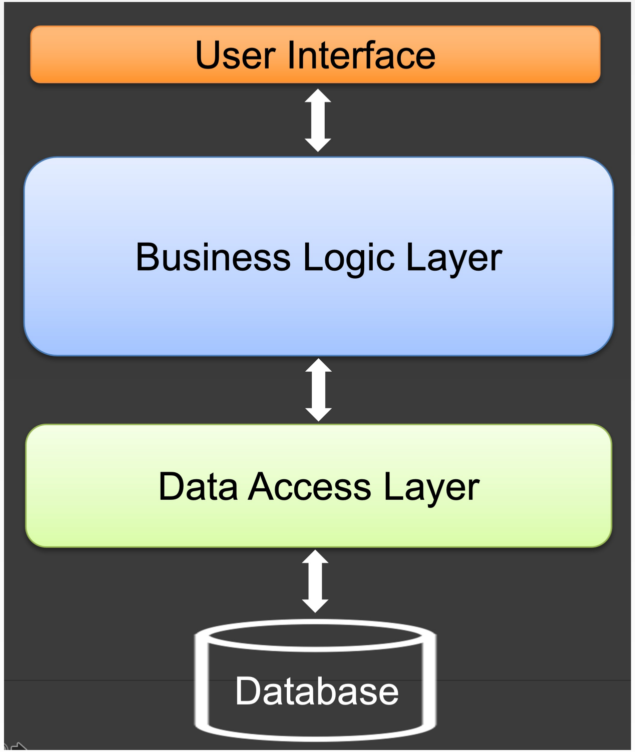
```
#     @timestamp               @message                                                      summary
▶ 1   2021-03-25T01:39:17....  summary: String = CDX:125.669154332|PagesWithLinks:897.522650…  CDX:125.669154332|PagesWithLinks:897.522650781
▶ 2   2021-03-25T01:37:48....  summary: String = CDX:125.425643731|PagesWithLinks:874.464511…  CDX:125.425643731|PagesWithLinks:874.464511474
▶ 3   2021-03-25T01:37:21....  summary: String = CDX:125.911768471|PagesWithLinks:904.810172…  CDX:125.911768471|PagesWithLinks:904.81017273
▶ 4   2021-03-25T01:36:22....  summary: String = CDX:126.699601735|PagesWithLinks:887.192927…  CDX:126.699601735|PagesWithLinks:887.192927855
▶ 5   2021-03-25T01:36:21....  summary: String = CDX:124.922903359|PagesWithLinks:878.758452…  CDX:124.922903359|PagesWithLinks:878.758452552
▶ 6   2021-03-25T01:36:08....  summary: String = CDX:107.011999682|PagesWithLinks:737.620613…  CDX:107.011999682|PagesWithLinks:737.620613719
▶ 7   2021-03-25T01:35:17....  summary: String = CDX:102.57925977|PagesWithLinks:726.04894947  CDX:102.57925977|PagesWithLinks:726.04894947
▶ 8   2021-03-25T01:35:10....  summary: String = CDX:107.412546462|PagesWithLinks:752.619504…  CDX:107.412546462|PagesWithLinks:752.619504649
▶ 9   2021-03-25T01:34:59....  summary: String = CDX:108.659221843|PagesWithLinks:744.074575…  CDX:108.659221843|PagesWithLinks:744.074575421
▶ 10  2021-03-25T01:34:52....  summary: String = CDX:127.018653886|PagesWithLinks:884.161975…  CDX:127.018653886|PagesWithLinks:884.161975368
▶ 11  2021-03-25T01:34:18....  summary: String = CDX:125.825652285|PagesWithLinks:900.629645…  CDX:125.825652285|PagesWithLinks:900.629645172
▶ 12  2021-03-25T01:34:14....  summary: String = CDX:108.297303066|PagesWithLinks:740.564994…  CDX:108.297303066|PagesWithLinks:740.564994068
```

AWS Glue

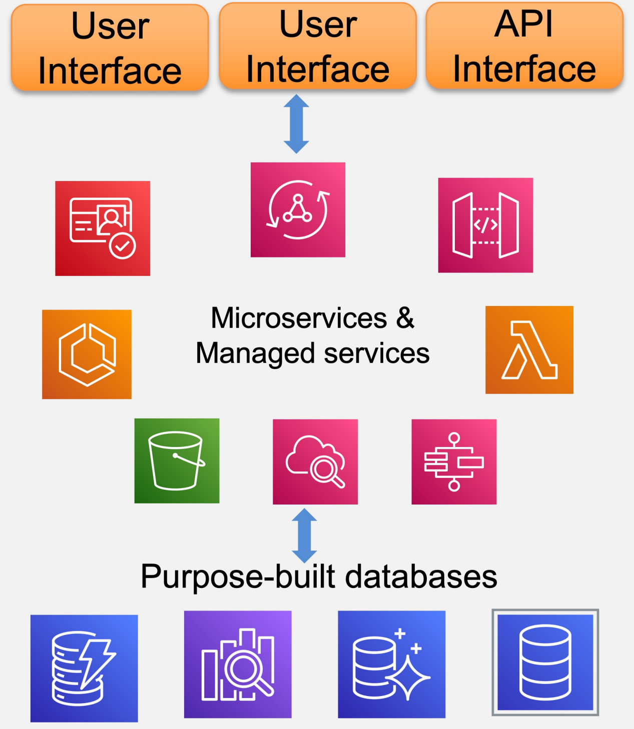

Create and run ELT jobs in AWS Glue

Amazon Redshift
Amazon S3
Amazon RDS
Databases running on Amazon EC2
1. Discover your data and store associated metadata in the Glue Data Catalog
AWS Glue Data Catalog
2. The Glue Data Catalog serves as a central metadata repository
AWS Glue ETL
4. Read and write metadata to the Glue Data Catalog
Amazon Athena
Amazon Redshift
Amazon EMR
3. Access the Glue Data Catalog for ETL and analytics
Amazon QuickSight & Other BI Tools
5. Run reports

Monolithic Architecture



Serverless Architecture



Reading and Next Class

- Data Mining and Warehousing
- Cloud Database
- Next: Final Review