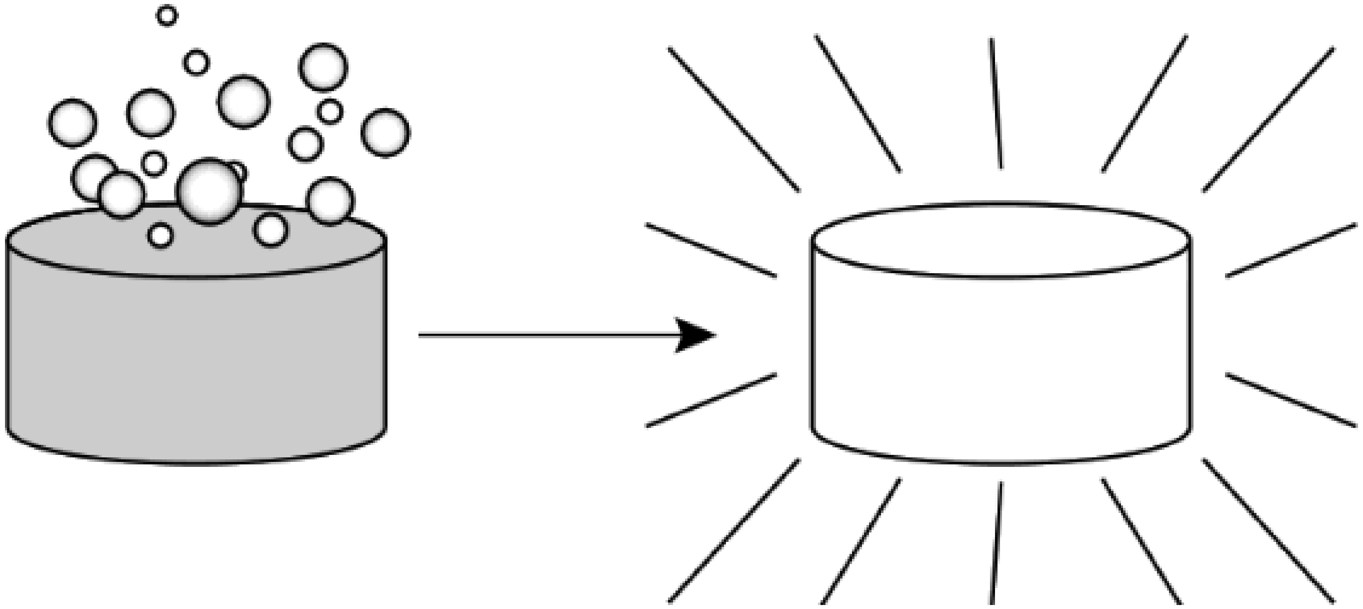
# Data Engineering

MG-GY 8441

# Processing Data

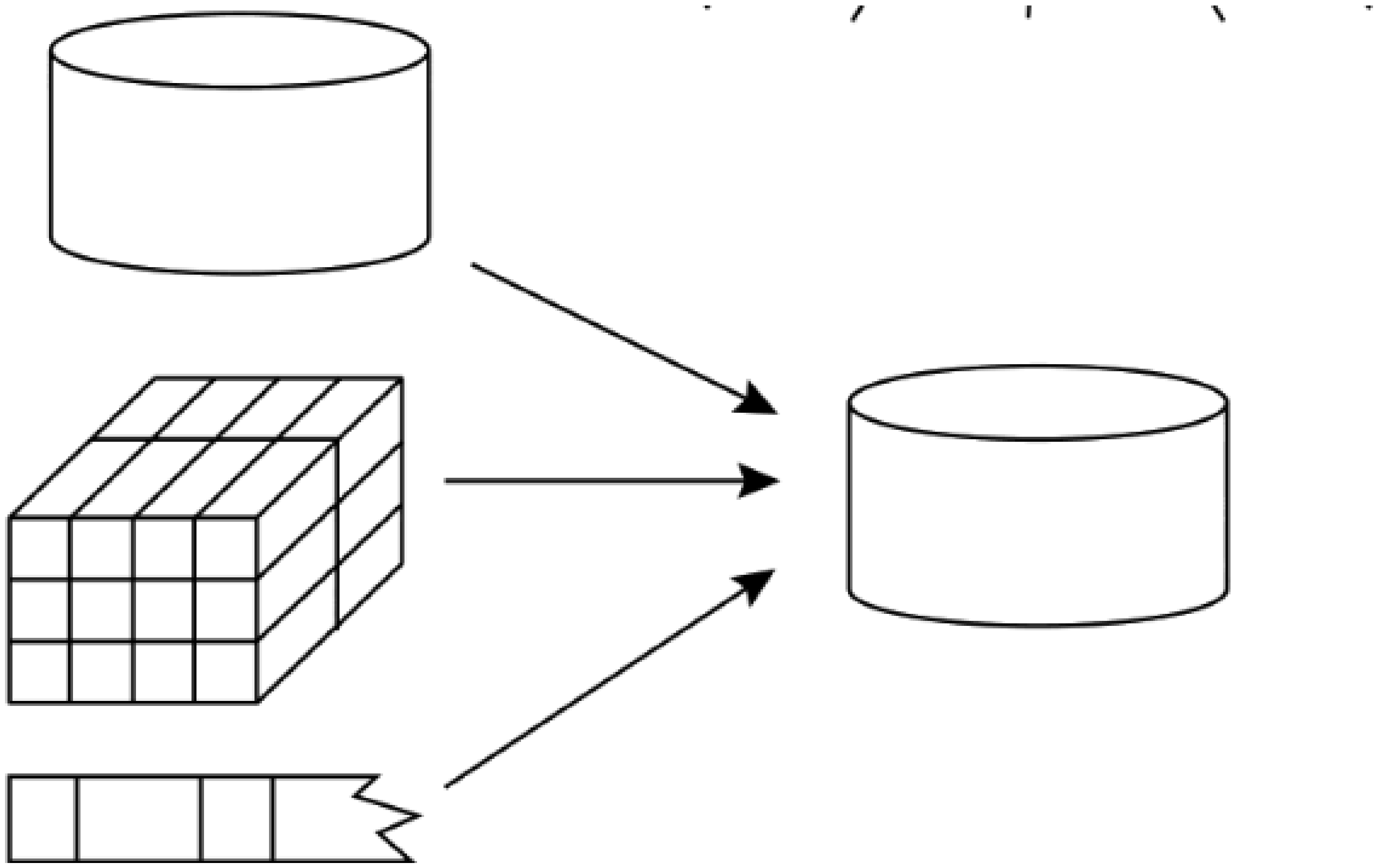
**Data cleaning**



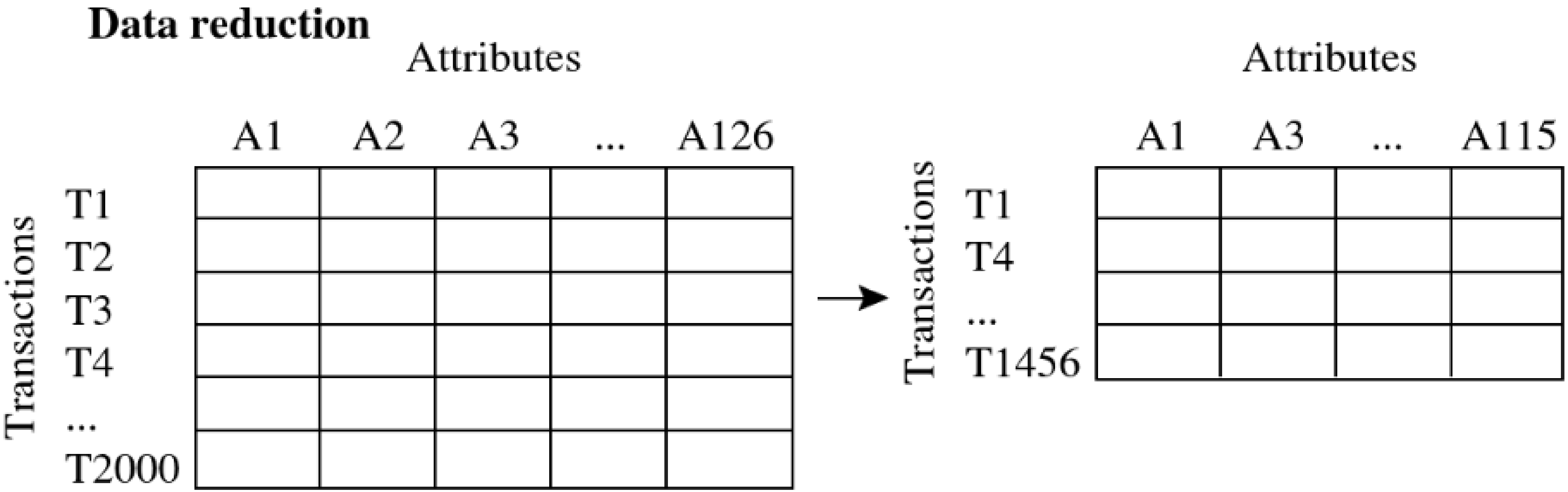
**Data transformation** −2, 32, 100, 59, 48 → −0.02, 0.32, 1.00, 0.59, 0.48

# Processing Data

**Data integration**



# Processing Data



**Data reduction**

# Storing Data

- Agenda
  - Data Formats
  - Database Management Systems
  - Online Analytical Processing
  - Structured Query Language
- References
  - Han, Kamber, Pei, *Data Mining: Concepts and Techniques* (Chapter 4.1 – 4.3)
  - (Optional) Garcia-Molina, Ullman, Widom, *Database Systems: The Complete Book* (Chapter 2)

# Example

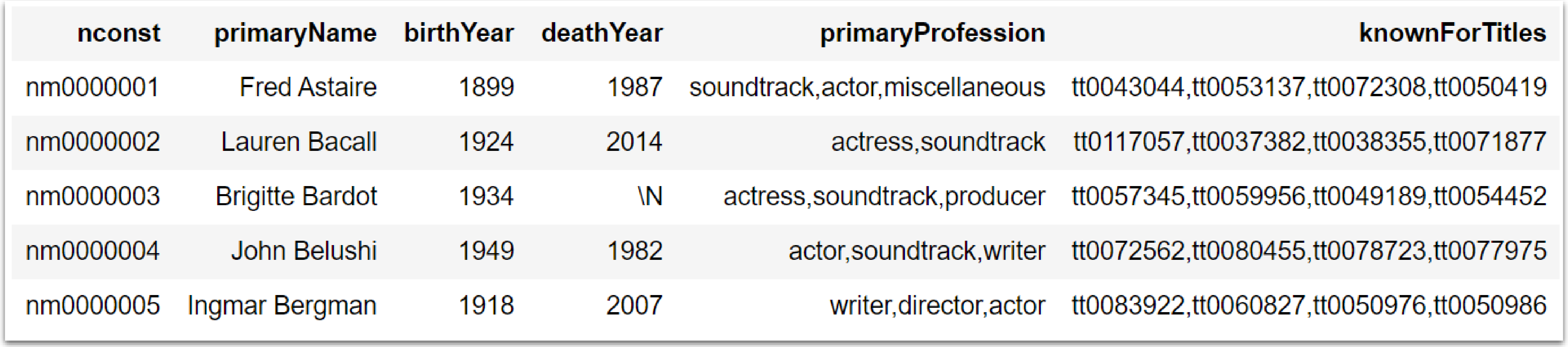
**Example from Commerce**

Suppose you are a business analyst within the accounting group of a retailer.

Your group has been studying sales from movie streaming. You want to find a competitive advantage through offering popular movies.

You decide to integrate data from your sales records and the IMDB database.

# Example



| nconst | primaryName | birthYear | deathYear | primaryProfession | knownForTitles |
|---|---|---|---|---|---|
| nm0000001 | Fred Astaire | 1899 | 1987 | soundtrack,actor,miscellaneous | tt0043044,tt0053137,tt0072308,tt0050419 |
| nm0000002 | Lauren Bacall | 1924 | 2014 | actress,soundtrack | tt0117057,tt0037382,tt0038355,tt0071877 |
| nm0000003 | Brigitte Bardot | 1934 | \N | actress,soundtrack,producer | tt0057345,tt0059956,tt0049189,tt0054452 |
| nm0000004 | John Belushi | 1949 | 1982 | actor,soundtrack,writer | tt0072562,tt0080455,tt0078723,tt0077975 |
| nm0000005 | Ingmar Bergman | 1918 | 2007 | writer,director,actor | tt0083922,tt0060827,tt0050976,tt0050986 |

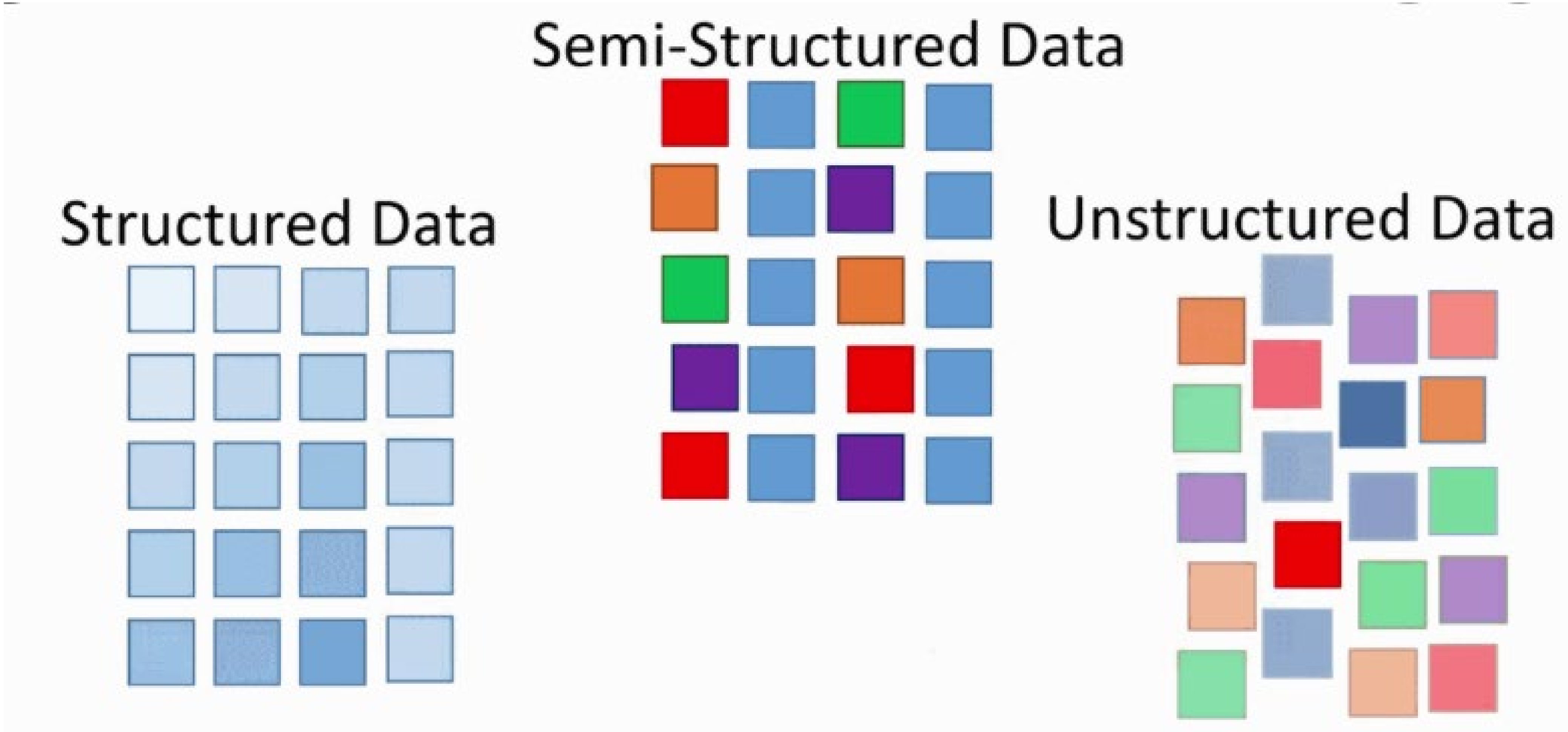
# Example

**Example from Commerce**

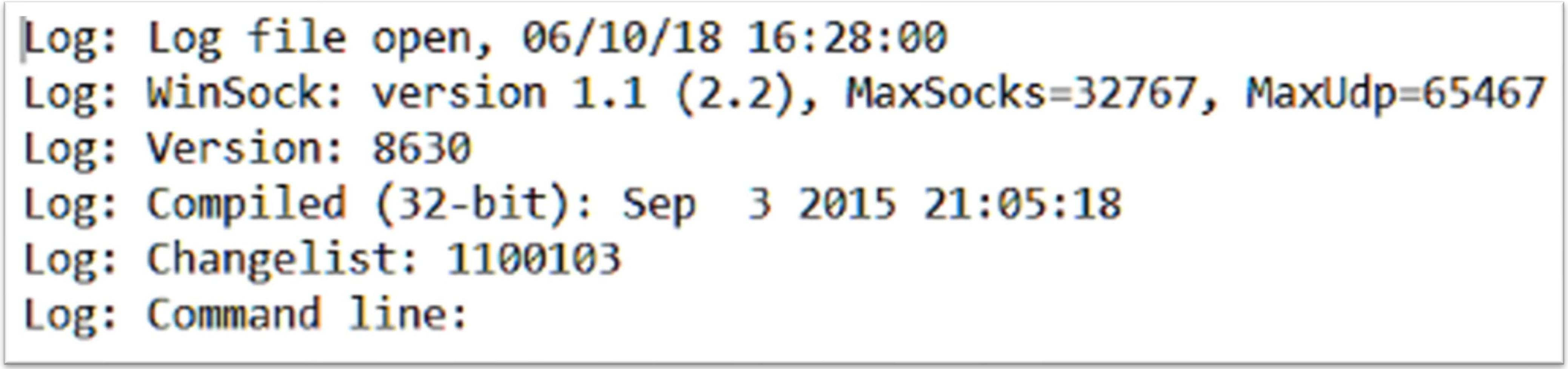
You want to compare the revenue generated by a movie and the content of the movie

- How does genre impact sales?
- Do some directors more popular?
- Will customers respond to indications of ratings?

# Data Formats

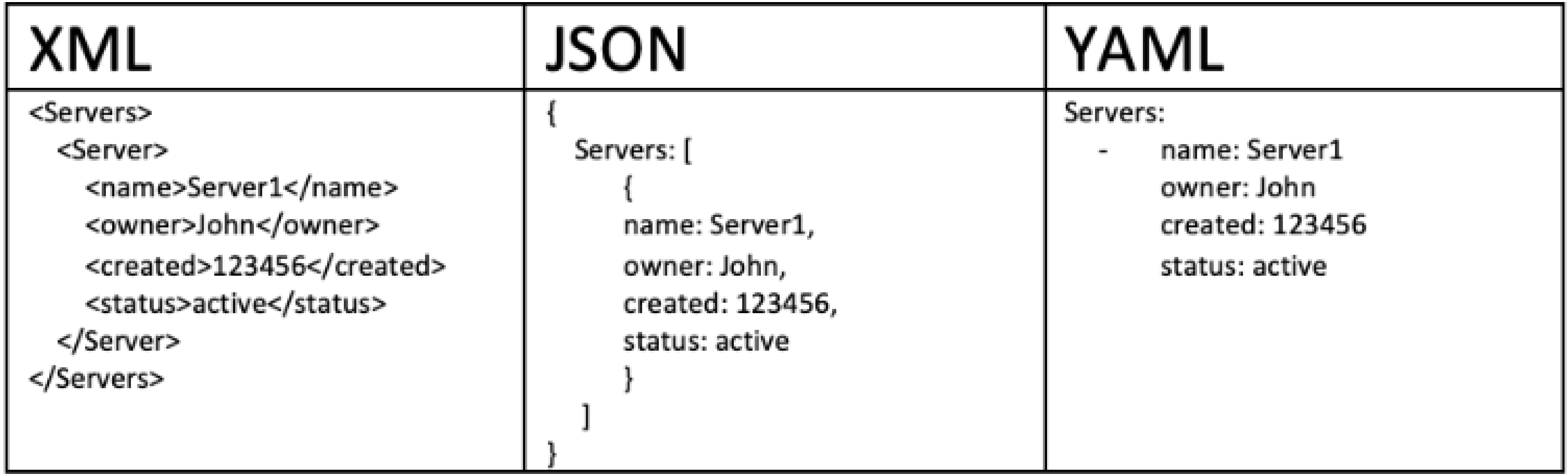


# Unstructured Data



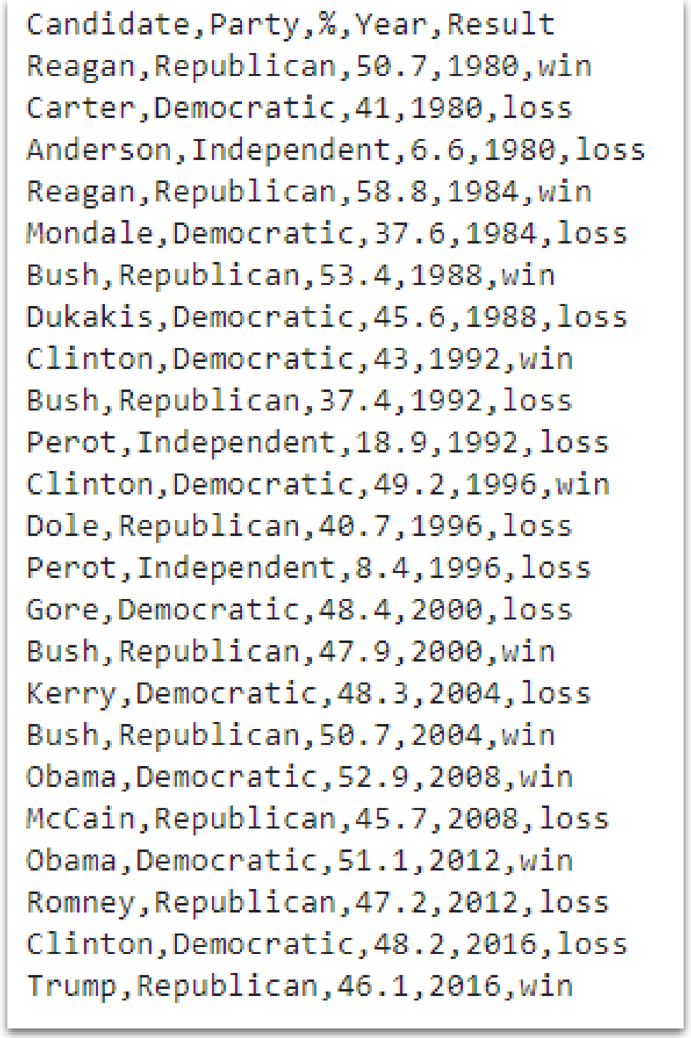
```
Log: Log file open, 06/10/18 16:28:00
Log: WinSock: version 1.1 (2.2), MaxSocks=32767, MaxUdp=65467
Log: Version: 8630
Log: Compiled (32-bit): Sep  3 2015 21:05:18
Log: Changelist: 1100103
Log: Command line:
```

# Semi-Structured Data



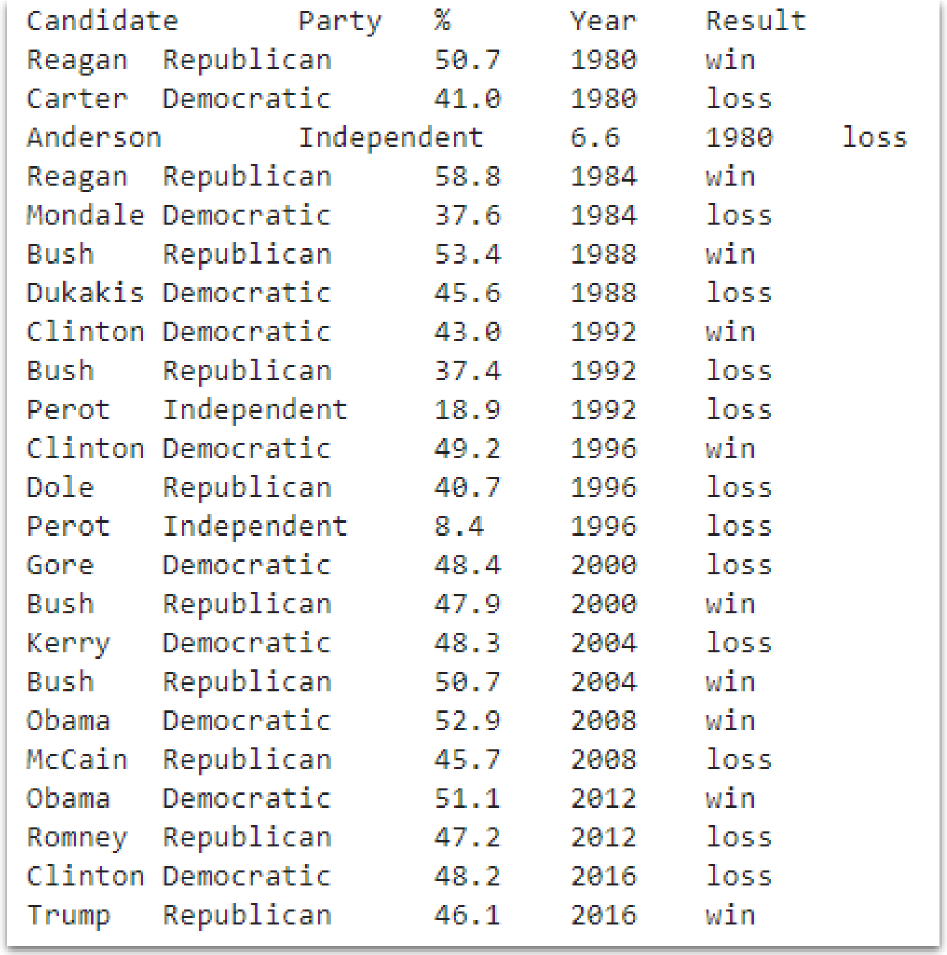
| XML | JSON | YAML |
|---|---|---|
| <Servers><br>  <Server><br>    <name>Server1</name><br>    <owner>John</owner><br>    <created>123456</created><br>    <status>active</status><br>  </Server><br></Servers> | {<br>  Servers: [<br>    {<br>    name: Server1,<br>    owner: John,<br>    created: 123456,<br>    status: active<br>    }<br>  ]<br>} | Servers:<br>  -    name: Server1<br>       owner: John<br>       created: 123456<br>       status: active |

# Structured Data



```
Candidate,Party,%,Year,Result
Reagan,Republican,50.7,1980,win
Carter,Democratic,41,1980,loss
Anderson,Independent,6.6,1980,loss
Reagan,Republican,58.8,1984,win
Mondale,Democratic,37.6,1984,loss
Bush,Republican,53.4,1988,win
Dukakis,Democratic,45.6,1988,loss
Clinton,Democratic,43,1992,win
Bush,Republican,37.4,1992,loss
Perot,Independent,18.9,1992,loss
Clinton,Democratic,49.2,1996,win
Dole,Republican,40.7,1996,loss
Perot,Independent,8.4,1996,loss
Gore,Democratic,48.4,2000,loss
Bush,Republican,47.9,2000,win
Kerry,Democratic,48.3,2004,loss
Bush,Republican,50.7,2004,win
Obama,Democratic,52.9,2008,win
McCain,Republican,45.7,2008,loss
Obama,Democratic,51.1,2012,win
Romney,Republican,47.2,2012,loss
Clinton,Democratic,48.2,2016,loss
Trump,Republican,46.1,2016,win
```

csv

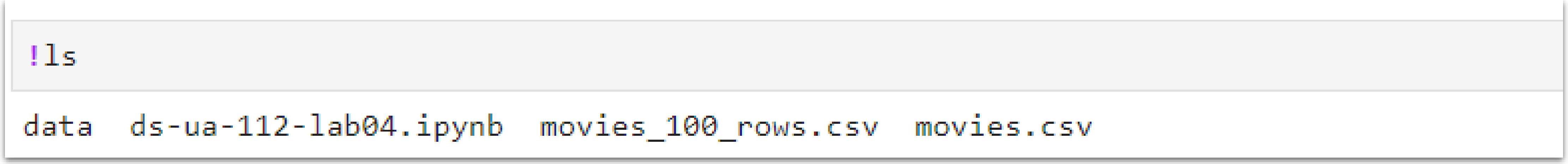


```
Candidate       Party   %       Year    Result
Reagan  Republican      50.7    1980    win
Carter  Democratic      41.0    1980    loss
Anderson        Independent     6.6     1980    loss
Reagan  Republican      58.8    1984    win
Mondale Democratic      37.6    1984    loss
Bush    Republican      53.4    1988    win
Dukakis Democratic      45.6    1988    loss
Clinton Democratic      43.0    1992    win
Bush    Republican      37.4    1992    loss
Perot   Independent     18.9    1992    loss
Clinton Democratic      49.2    1996    win
Dole    Republican      40.7    1996    loss
Perot   Independent     8.4     1996    loss
Gore    Democratic      48.4    2000    loss
Bush    Republican      47.9    2000    win
Kerry   Democratic      48.3    2004    loss
Bush    Republican      50.7    2004    win
Obama   Democratic      52.9    2008    win
McCain  Republican      45.7    2008    loss
Obama   Democratic      51.1    2012    win
Romney  Republican      47.2    2012    loss
Clinton Democratic      48.2    2016    loss
Trump   Republican      46.1    2016    win
```

tsv

# File Systems

- Computers provide access to a command line interface. Users input commands to perform operations on the computer especially files.



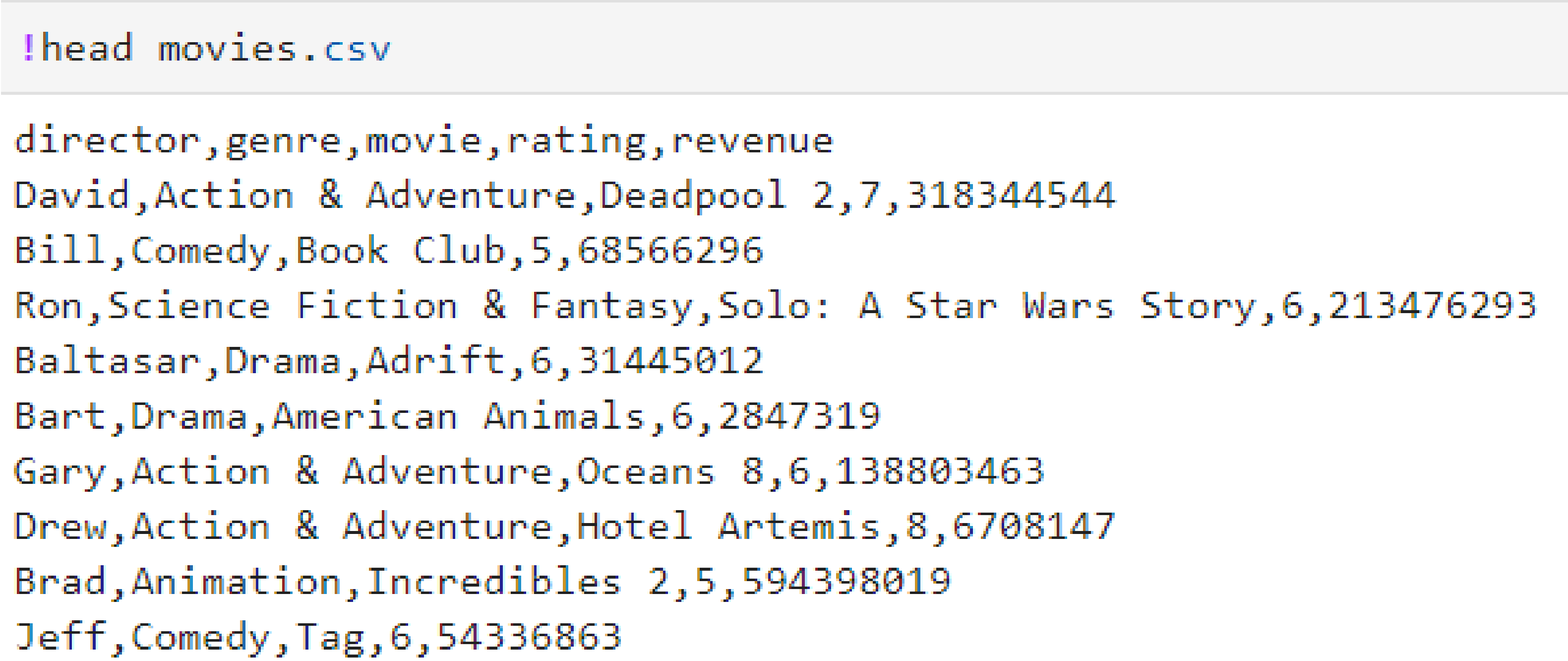
```
!ls
```

```
data  ds-ua-112-lab04.ipynb  movies_100_rows.csv  movies.csv
```

- We can enter commands in Jupyter notebook using exclamation point
- Note that the commands differ across operating systems. Here we have the commands for the Linux operating system on JupyterHub.

# File Systems

- Some commands for accessing files include
  - head
    - Returns the first 10 rows of the file
  - tail
    - Returns the last 10 rows of the file
  - cat
    - Returns all rows of the file

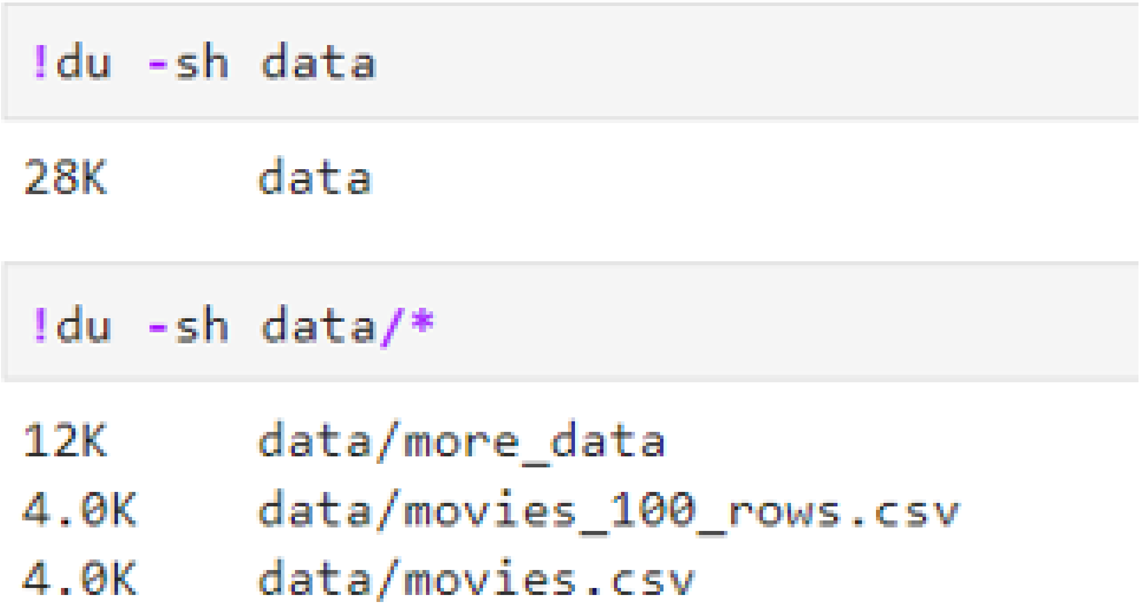


```
!head movies.csv
```

```
director,genre,movie,rating,revenue
David,Action & Adventure,Deadpool 2,7,318344544
Bill,Comedy,Book Club,5,68566296
Ron,Science Fiction & Fantasy,Solo: A Star Wars Story,6,213476293
Baltasar,Drama,Adrift,6,31445012
Bart,Drama,American Animals,6,2847319
Gary,Action & Adventure,Oceans 8,6,138803463
Drew,Action & Adventure,Hotel Artemis,8,6708147
Brad,Animation,Incredibles 2,5,594398019
Jeff,Comedy,Tag,6,54336863
```

# File Systems

- A command for determining the size of files is
  - du -sh
    - Will calculate the size of files or folders



```
!du -sh data
```

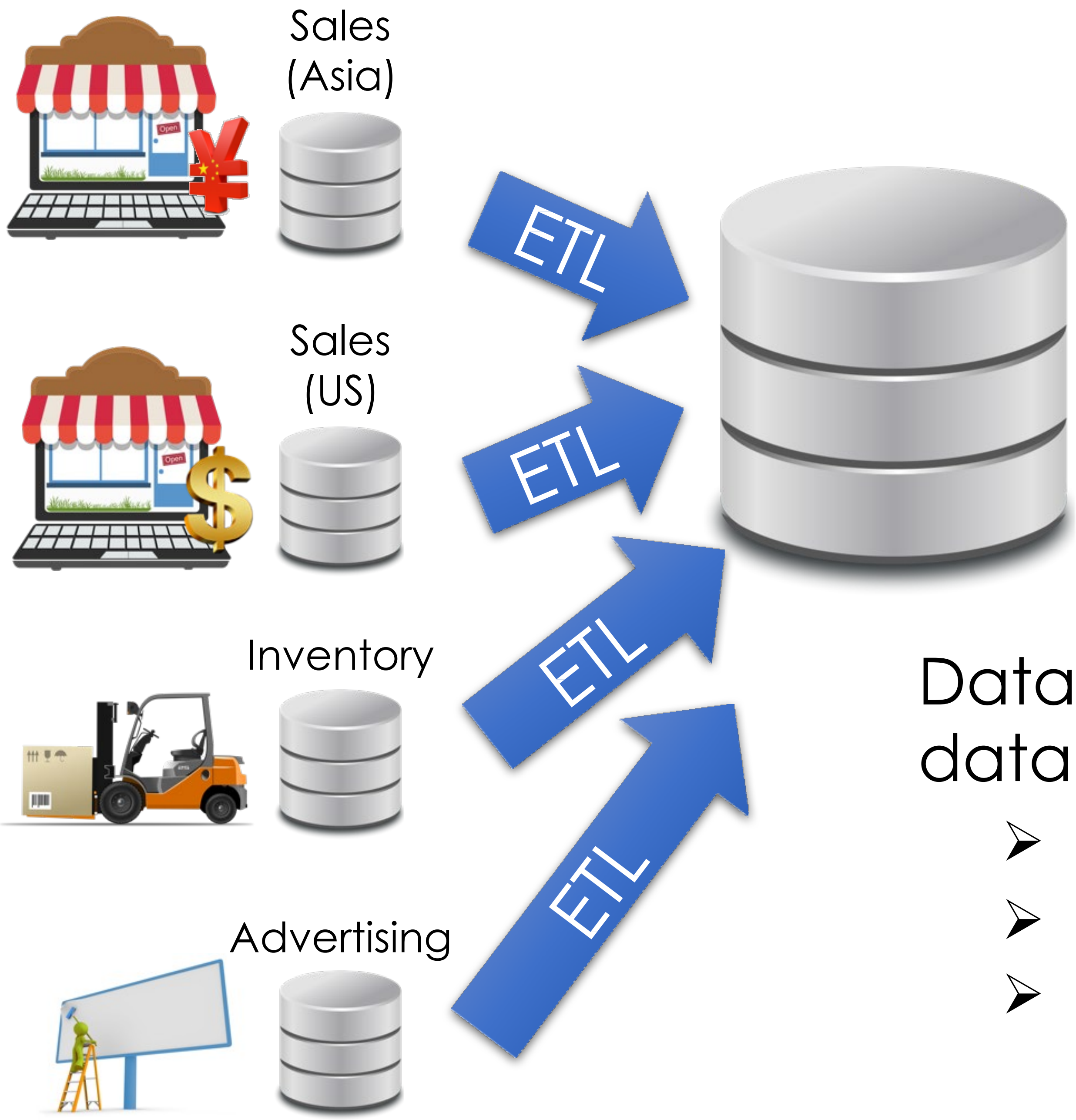
```
28K	data
```

```
!du -sh data/*
```

```
12K	data/more_data
4.0K	data/movies_100_rows.csv
4.0K	data/movies.csv
```

# Database Management Systems

# Data Warehouse



Collects and organizes historical data from multiple sources

Data is *periodically* **ETL**ed into the data warehouse:

- **Extracted** from remote sources
- **Transformed** to standard schemas
- **Loaded** into the (typically) relational (SQL) data system

# Extract → Transform → Load (ETL)

**Extract & Load:** provides a snapshot of operational data

- Historical snapshot
- Data in a single system
- Isolates analytics queries (e.g., Deep Learning) from business critical services (e.g., processing user purchase)
- Easy!

**Transform:** clean and prepare data for analytics in a unified representation

- **Difficult** → often requires specialized code and tools
- Different schemas, encodings, granularities

# Table

| pname | category | price | qty | date | day | city | state | country |
|---|---|---|---|---|---|---|---|---|
| Corn | Food | 25 | 25 | 3/30/16 | Wed. | Omaha | NE | USA |
| Corn | Food | 25 | 8 | 3/31/16 | Thu. | Omaha | NE | USA |
| Corn | Food | 25 | 15 | 4/1/16 | Fri. | Omaha | NE | USA |
|  |  |  |  |  | Wed. | Omaha | NE | USA |
|  |  | 18 | 20 | 3/31/16 | Thu. | Omaha | NE | USA |
|  |  |  |  |  | Fri. | Omaha | NE | USA |
| Galaxy 1 | Phones | 18 | 8 | 1/30/16 | Wed. | Omaha | NE | USA |
| Peanuts | Food | 2 | 45 | 3/31/16 | Thu. | Seoul |  | Korea |

- Substantial redundancy → expensive to store
- Could we organize the data more efficiently?

# Multidimensional Data Model

## *Sales* Fact Table

| pid | timeid | locid | sales |
|---|---|---|---|
| 11 | 1 | 1 | 25 |
| 11 | 2 | 1 | 8 |
| 11 | 3 | 1 | 15 |
| 12 | 1 | 1 | 30 |
| 12 | 2 | 1 | 20 |
| 12 | 3 | 1 | 50 |
| 12 | 1 | 1 | 8 |
| 13 | 2 | 1 | 10 |
| 13 | 3 | 1 | 10 |
| 11 | 1 | 2 | 35 |
| 11 | 2 | 2 | 22 |
| 11 | 3 | 2 | 10 |
| 12 | 1 | 2 | 26 |

## Locations

| locid | city | state | country |
|---|---|---|---|
| 1 | Omaha | Nebraska | USA |
| 2 | Seoul | | Korea |
| 5 | Richmond | Virginia | USA |

## Products

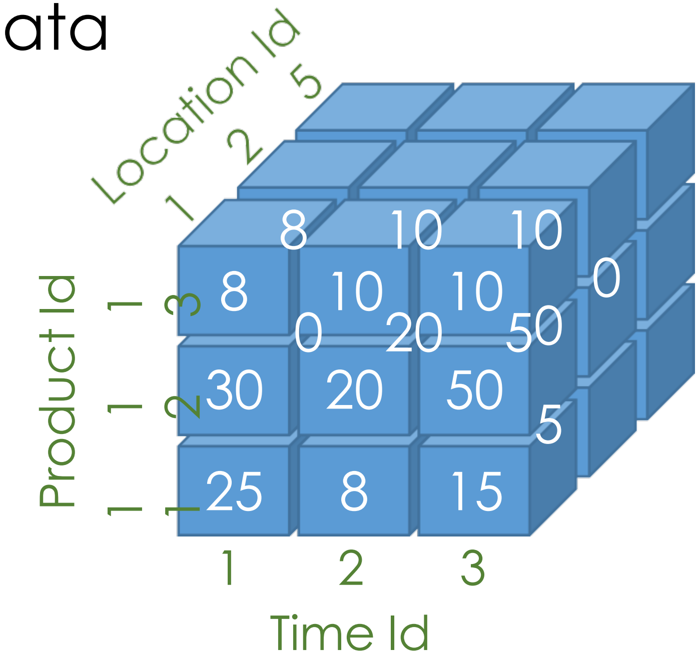
| pid | pname | category | price |
|---|---|---|---|
| 11 | Corn | Food | 25 |
| 12 | Galaxy 1 | Phones | 18 |
| 13 | Peanuts | Food | 2 |

## Time

| timeid | Date | Day |
|---|---|---|
| 1 | 3/30/16 | Wed. |
| 2 | 3/31/16 | Thu. |
| 3 | 4/1/16 | Fri. |

## Dimension Tables

- Multidimensional "Cube" of data



# Multidimensional Data Model

## *Sales* **Fact Table**

| pid | timeid | locid | sales |
|---|---|---|---|
| 11 | 1 | 1 | 25 |
| 11 | 2 | 1 | 8 |
| 11 | 3 | 1 | 15 |
| 12 | 1 | 1 | 30 |
| 12 | 2 | 1 | 20 |
| 12 | 3 | 1 | 50 |
| 12 | 1 | 1 | 8 |
| 13 | 2 | 1 | 10 |
| 13 | 3 | 1 | 10 |
| 11 | 1 | 2 | 35 |
| 11 | 2 | 2 | 22 |
| 11 | 3 | 2 | 10 |
| 12 | 1 | 2 | 26 |

## Locations

| locid | city | state | country |
|---|---|---|---|
| 1 | Omaha | Nebraska | USA |
| 2 | Seoul |  | Korea |
| 5 | Richmond | Virginia | USA |

## Products

| pid | pname | category | price |
|---|---|---|---|
| 11 | Corn | Food | 25 |
| 12 | Galaxy 1 | Phones | 18 |
| 13 | Peanuts | Food | 2 |

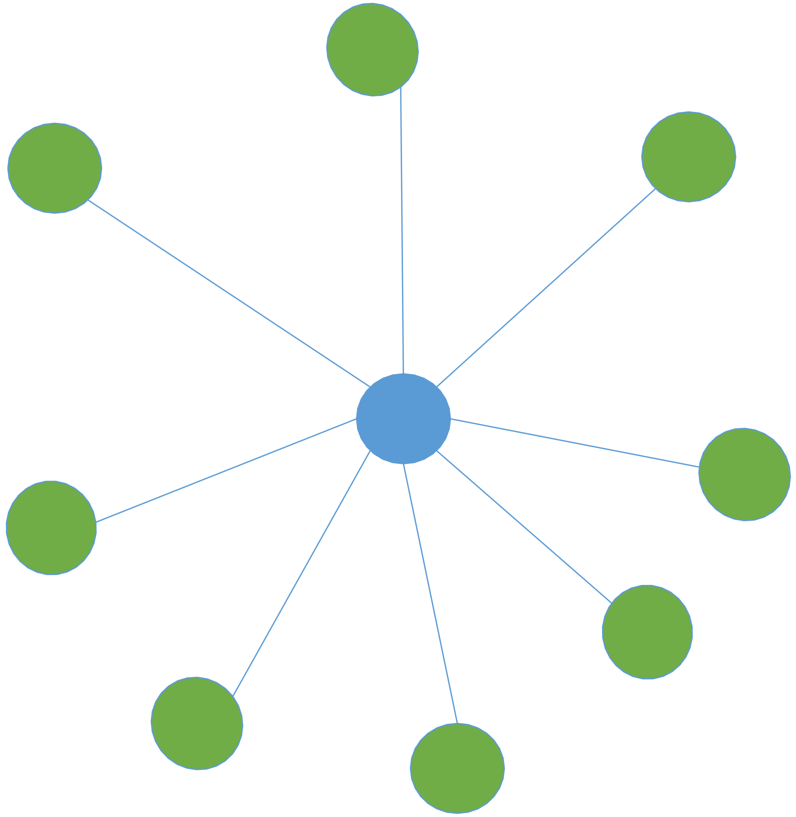
## Time

| timeid | Date | Day |
|---|---|---|
| 1 | 3/30/16 | Wed. |
| 2 | 3/31/16 | Thu. |
| 3 | 4/1/16 | Fri. |

## Dimension Tables

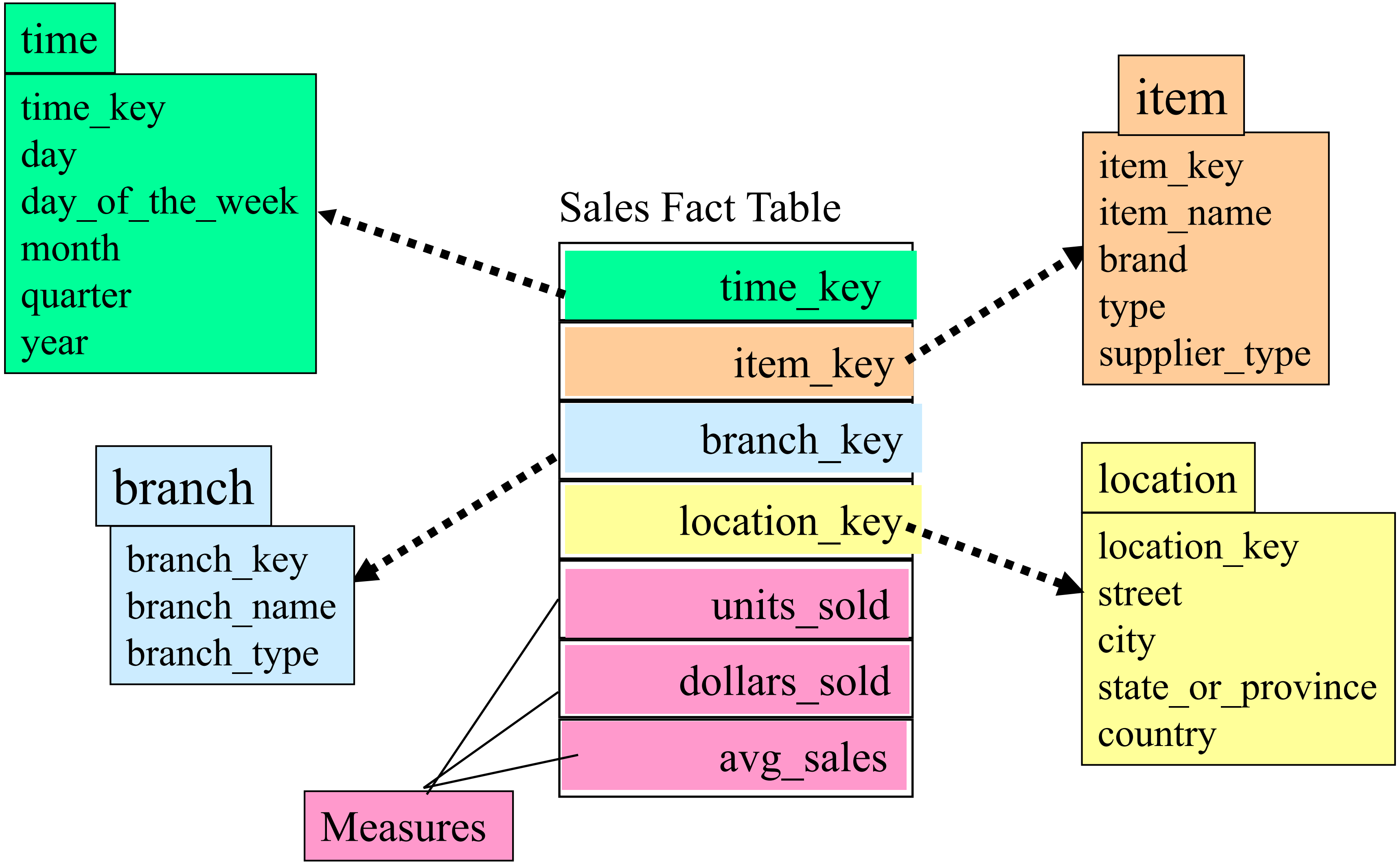
- Fact Table
  - minimizes redundant info.
  - Reduces data errors
- Dimensions
  - easy to manage, summarize and rename
- How can we combine tables?
  - Joins

# The Star Schema

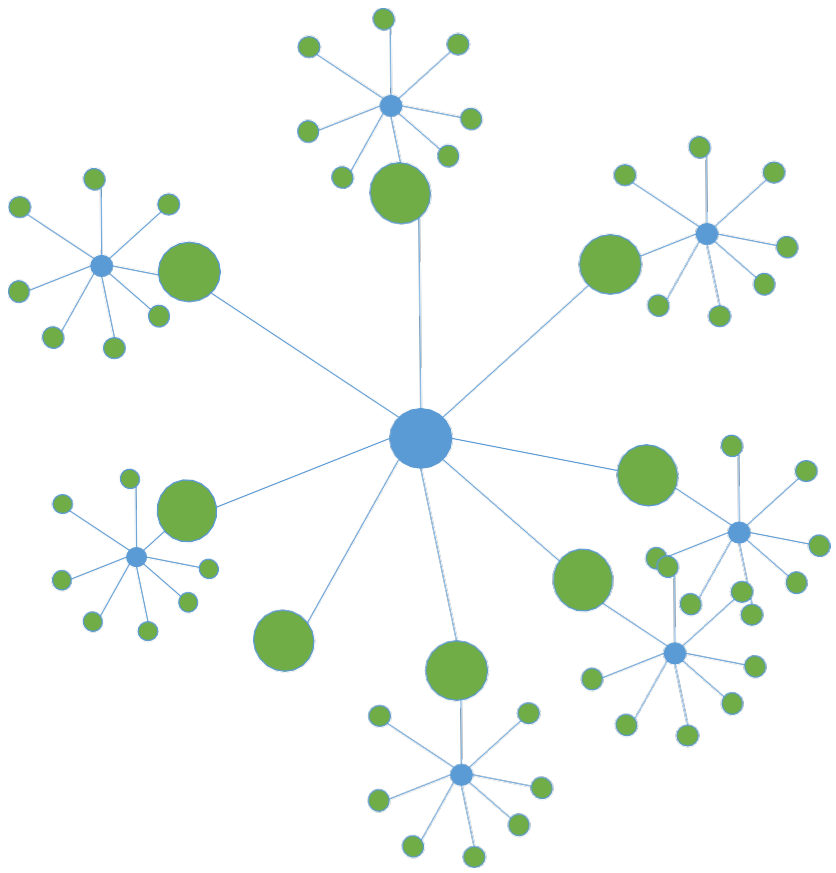


← This looks like a star …

# Star Schema: An Example

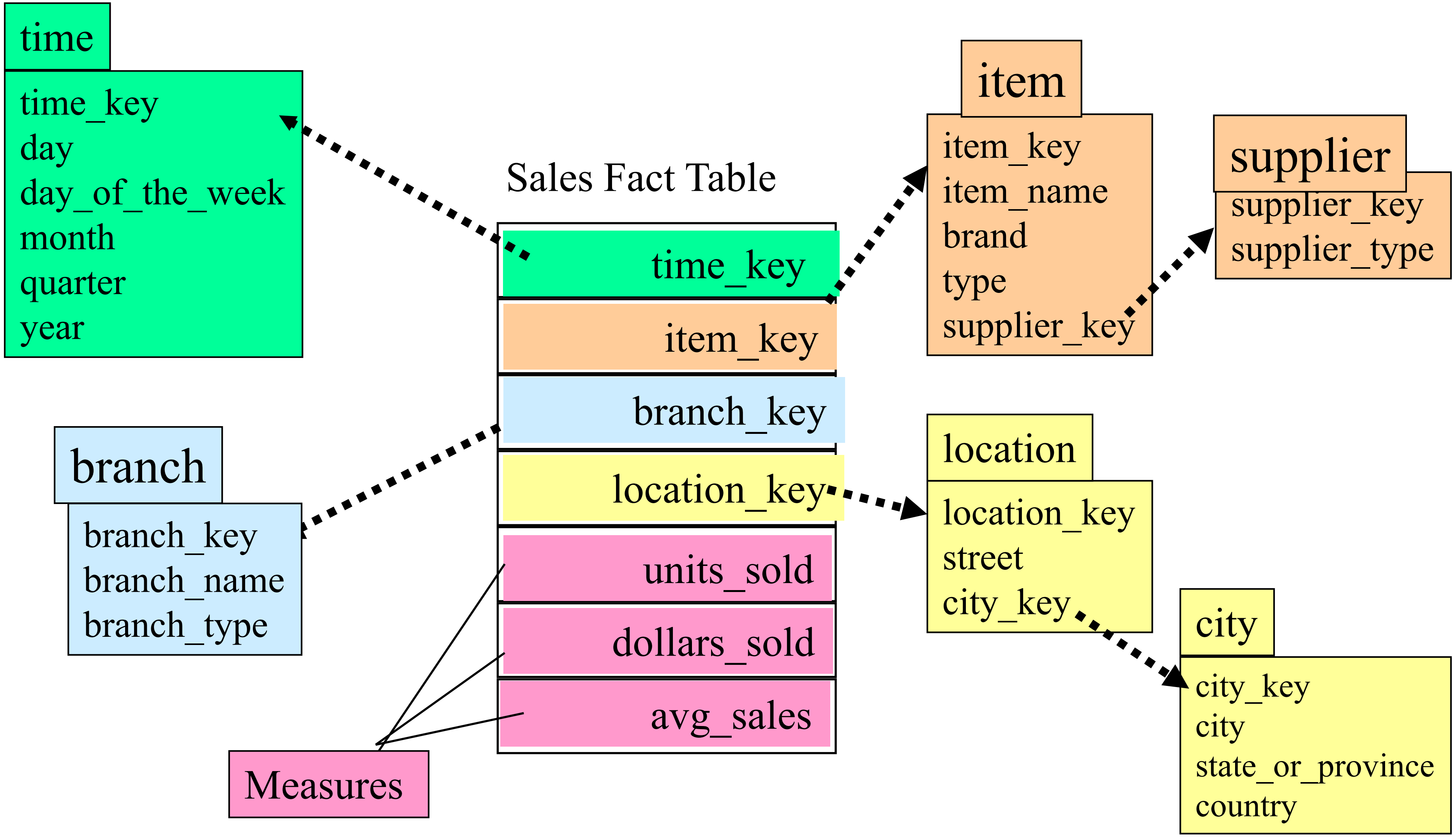


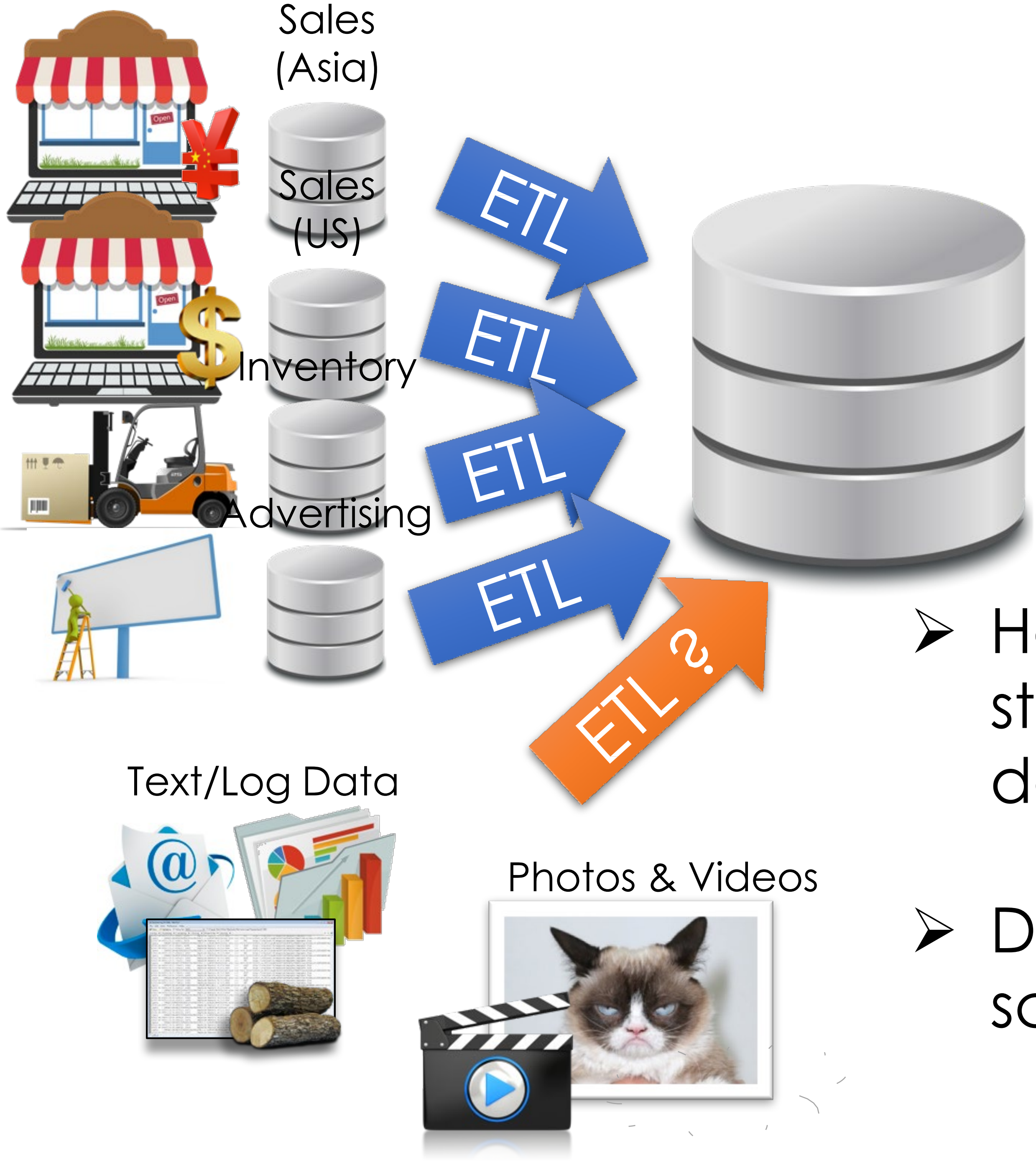
# The Snowflake Schema



← This looks like a snowflake
…sort of

# Snowflake Schema: An Example





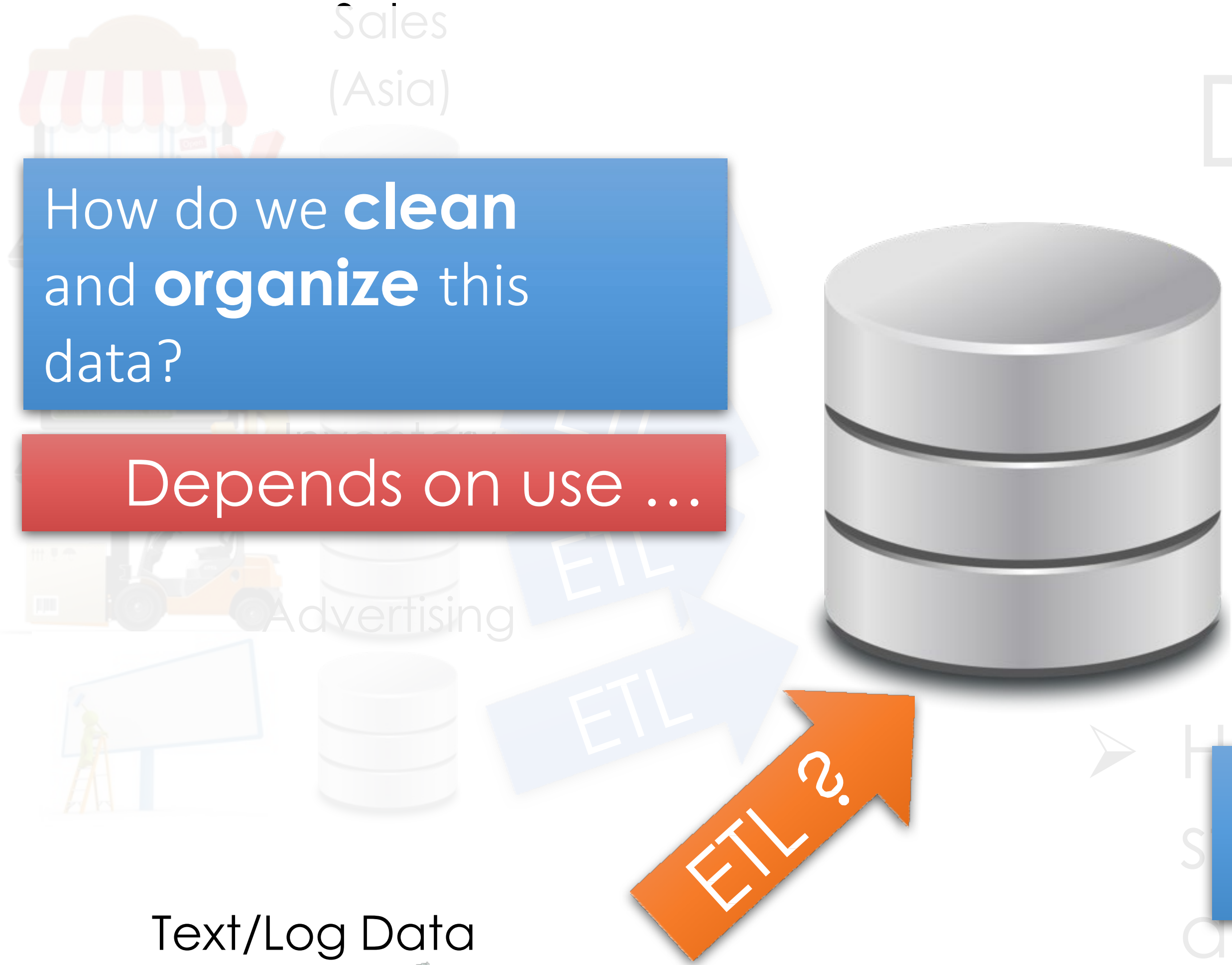
# Data Warehouse

Collects and organizes historical data from multiple sources

- How do we deal with semi-structured and unstructured data?
- Do we really want to force a schema on load?

# Data Warehouse

Collects and organizes historical data from multiple sources



How do we **clean** and **organize** this data?

Depends on use …

- How do we deal with semi-s… data?

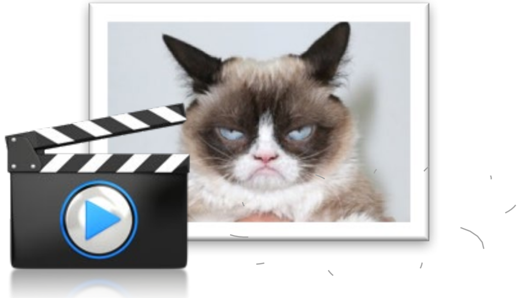
How do we **load** and **process** this data in a relational system?

- Do we re… schema …

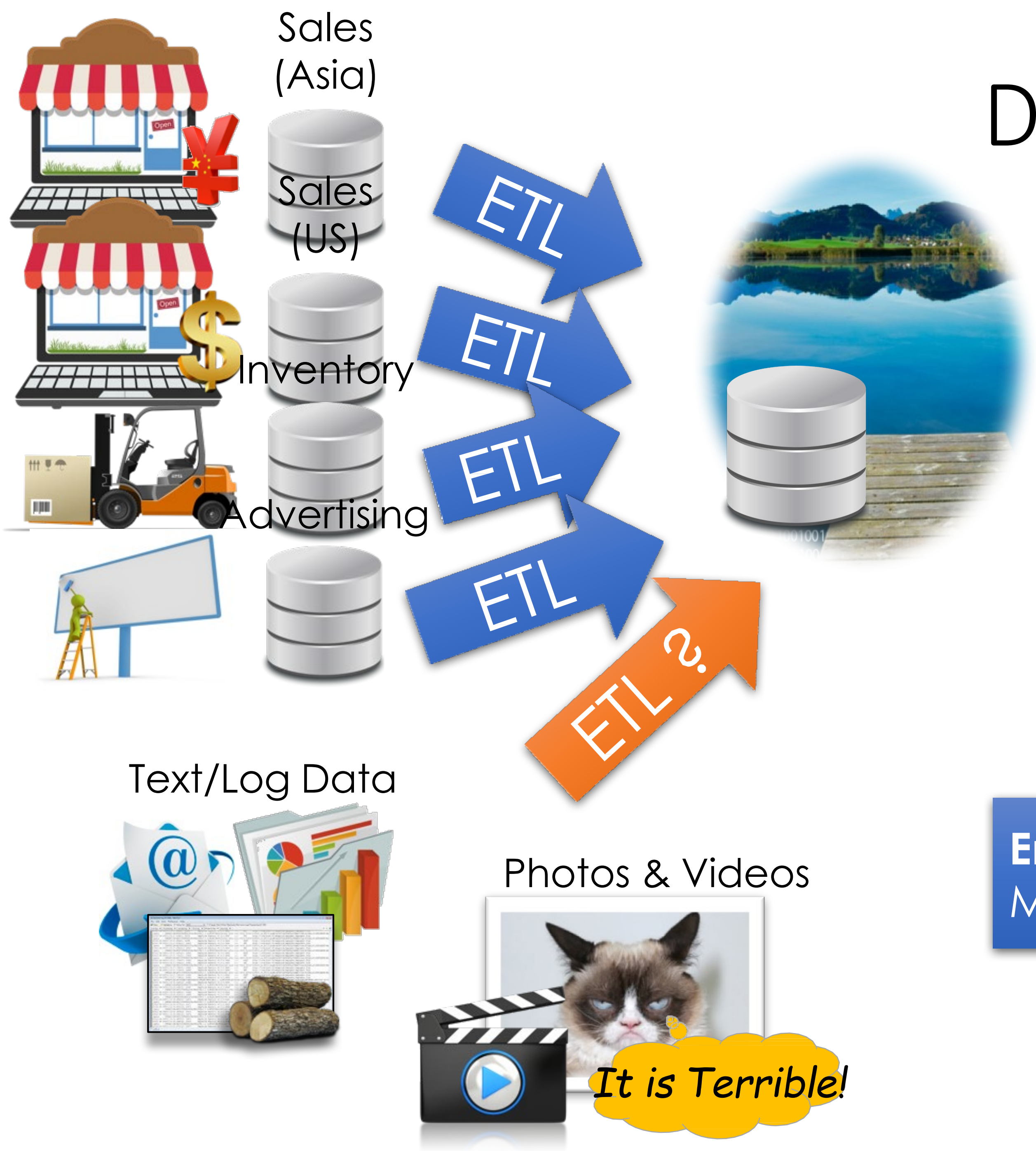
Depends on use …
Can be difficult …
Requires thought …

Text/Log Data

Photos & Videos



# Data Lake



Store a copy of all the data

- in one place
- in its original "natural" form

Enable data consumers to choose how to transform and use data.

- *Schema on Read*

**Enabled by new Tools:**
Map-Reduce & Distributed Filesystems

What could go wrong?

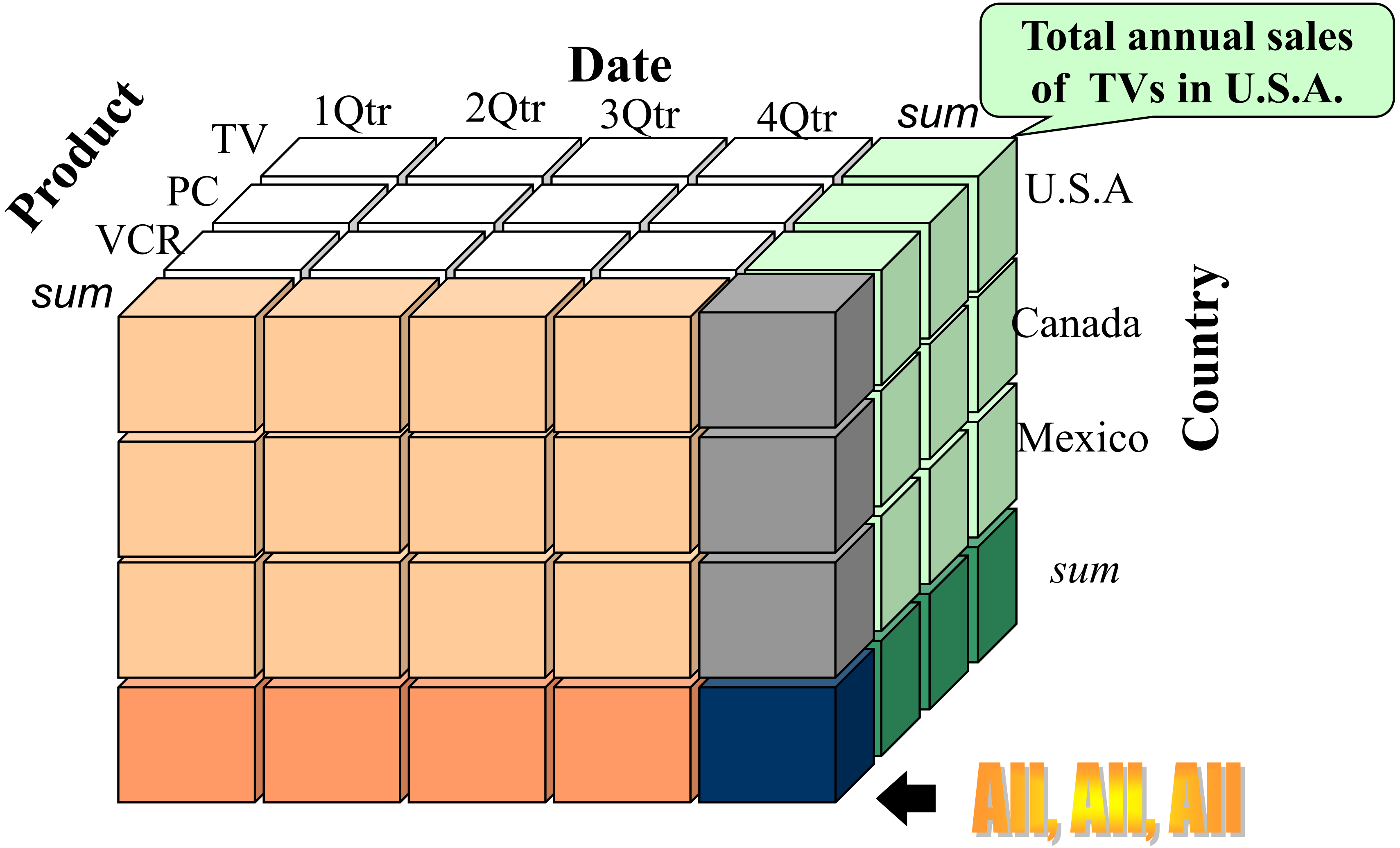
# Online Analytical Processing

# OLTP vs. OLAP

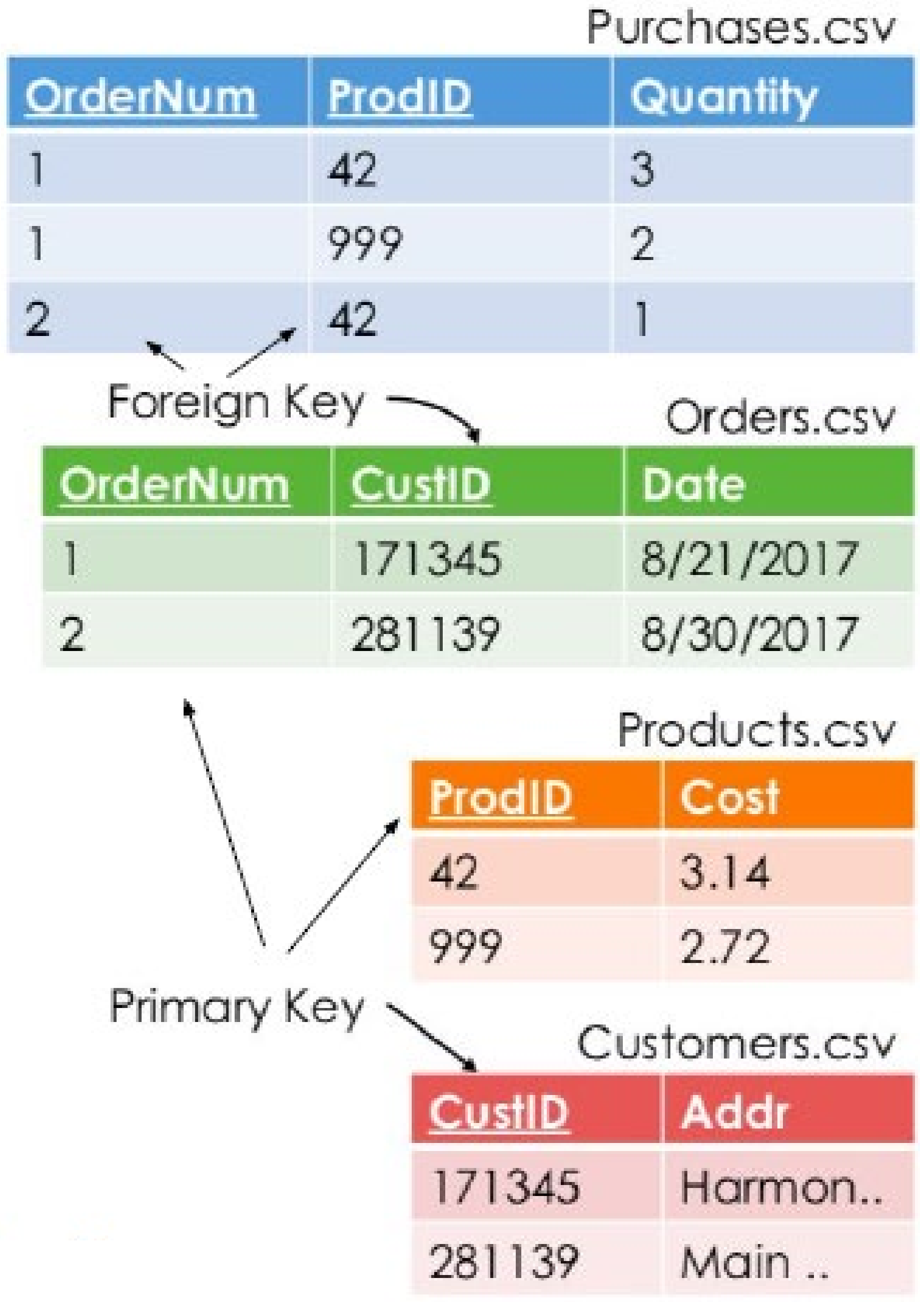
- OLTP: Online transactional processing
  - DBMS operations
  - Query and transactional processing
- OLAP: Online analytical processing
  - Data warehouse operations
  - Drilling, slicing, dicing, etc.

| | **OLTP** | **OLAP** |
|---|---|---|
| **users** | clerk, IT professional | knowledge worker |
| **function** | day to day operations | decision support |
| **DB design** | application-oriented | subject-oriented |
| **data** | current, up-to-date<br>detailed, flat relational<br>isolated | historical,<br>summarized, multidimensional<br>integrated, consolidated |
| **usage** | repetitive | ad-hoc |
| **access** | read/write<br>index/hash on prim. key | lots of scans |
| **unit of work** | short, simple transaction | complex query |
| **# records accessed** | tens | millions |
| **#users** | thousands | hundreds |
| **DB size** | 100MB-GB | 100GB-TB |
| **metric** | transaction throughput | query throughput, response |

# Data Cube


Total annual sales
of TVs in U.S.A.
Date
Product
1Qtr
2Qtr
3Qtr
4Qtr
sum
TV
PC
VCR
sum
U.S.A
Canada
Mexico
sum
Country
All, All, All

# Keys



Purchases.csv

| OrderNum | ProdID | Quantity |
|---|---|---|
| 1 | 42 | 3 |
| 1 | 999 | 2 |
| 2 | 42 | 1 |

Orders.csv

| OrderNum | CustID | Date |
|---|---|---|
| 1 | 171345 | 8/21/2017 |
| 2 | 281139 | 8/30/2017 |

Products.csv

| ProdID | Cost |
|---|---|
| 42 | 3.14 |
| 999 | 2.72 |

Customers.csv

| CustID | Addr |
|---|---|
| 171345 | Harmon.. |
| 281139 | Main .. |

# Index

**Base table**

| Cust | Region | Type |
|---|---|---|
| C1 | Asia | Retail |
| C2 | Europe | Dealer |
| C3 | Asia | Dealer |
| C4 | America | Retail |
| C5 | Europe | Dealer |

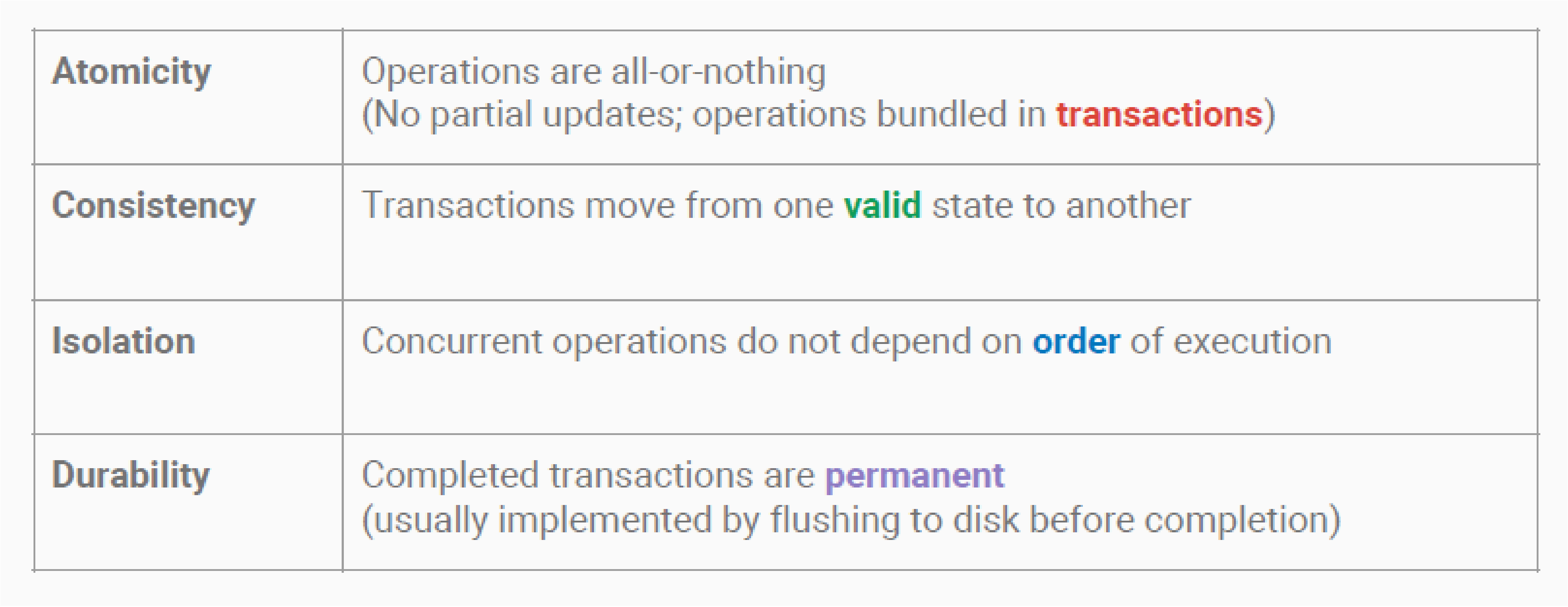
**Index on Region**

| RecID | Asia | Europe | America |
|---|---|---|---|
| 1 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 |
| 3 | 1 | 0 | 0 |
| 4 | 0 | 0 | 1 |
| 5 | 0 | 1 | 0 |

**Index on Type**

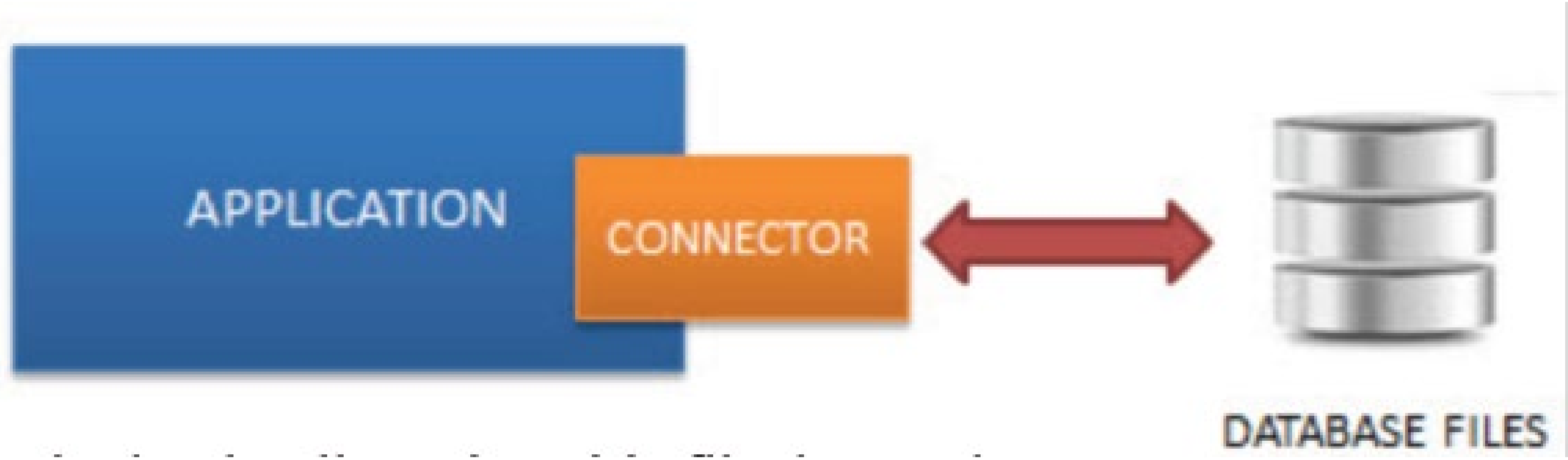
| RecID | Retail | Dealer |
|---|---|---|
| 1 | 1 | 0 |
| 2 | 0 | 1 |
| 3 | 0 | 1 |
| 4 | 1 | 0 |
| 5 | 0 | 1 |

# ACID



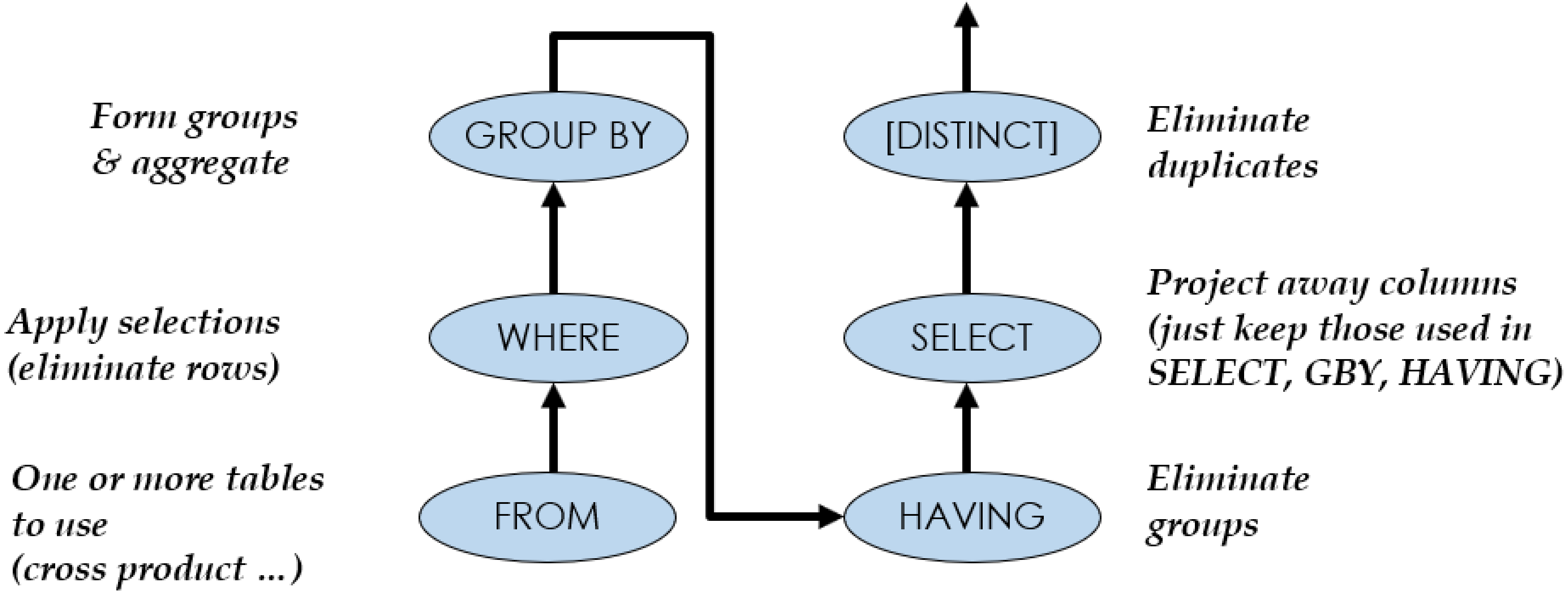
| | |
|---|---|
| **Atomicity** | Operations are all-or-nothing<br>(No partial updates; operations bundled in **transactions**) |
| **Consistency** | Transactions move from one **valid** state to another |
| **Isolation** | Concurrent operations do not depend on **order** of execution |
| **Durability** | Completed transactions are **permanent**<br>(usually implemented by flushing to disk before completion) |

# Structured Query Language



# Commands

```
SELECT       [DISTINCT]  target-list
FROM         relation-list
WHERE        qualification
GROUP BY     grouping-list
HAVING       group-qualification
```



*Form groups & aggregate*

GROUP BY

[DISTINCT]

*Eliminate duplicates*

*Apply selections (eliminate rows)*

WHERE

SELECT

*Project away columns (just keep those used in SELECT, GBY, HAVING)*

*One or more tables to use (cross product …)*

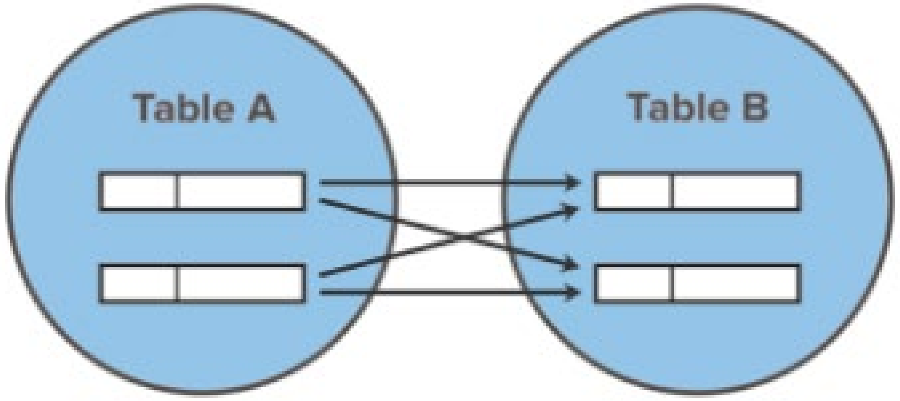
FROM

HAVING

*Eliminate groups*

# Cross Join

- Cross Join pairs each of the rows in the tables regardless of the entries in the columns.
- All pairs of rows appear in the Join.



s

| M | U |
|---|---|
| 1 | W |
| 2 | X |
| 3 | X |
| 4 | Y |

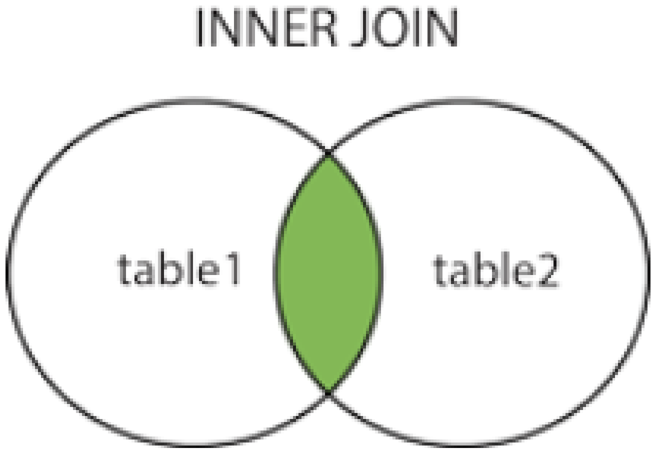
t

| N | V |
|---|---|
| A | X |
| B | X |
| C | Y |
| D | Z |

| M | U | N | V |
|---|---|---|---|
| 1 | W | A | X |
| 2 | X | A | X |
| 3 | X | A | X |
| 4 | Y | A | X |
| 1 | W | B | X |
| 2 | X | B | X |
| 3 | X | B | X |
| 4 | Y | B | X |
| 1 | W | C | Y |
| 2 | X | C | Y |
| 3 | X | C | Y |
| 4 | Y | C | Y |
| 1 | W | D | Z |
| 2 | X | D | Z |
| 3 | X | D | Z |
| 4 | Y | D | Z |

# Inner Join

- Inner Join pairs each of the rows in the tables depending on the entries in specific columns.
- The entries of columns must match for the pair to appear in the Join.



s

| M | U |
|---|---|
| ~~1~~ | ~~W~~ |
| 2 | X |
| 3 | X |
| 4 | Y |

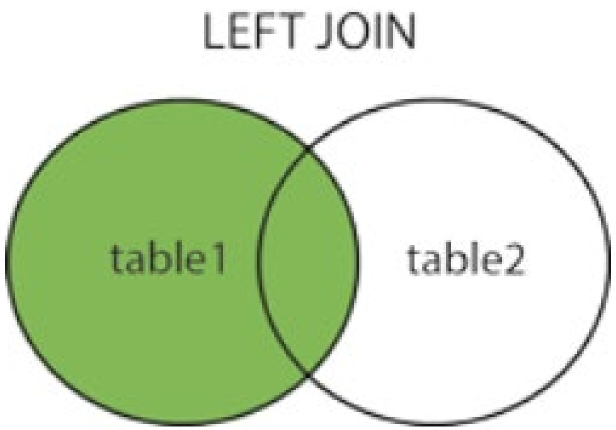
t

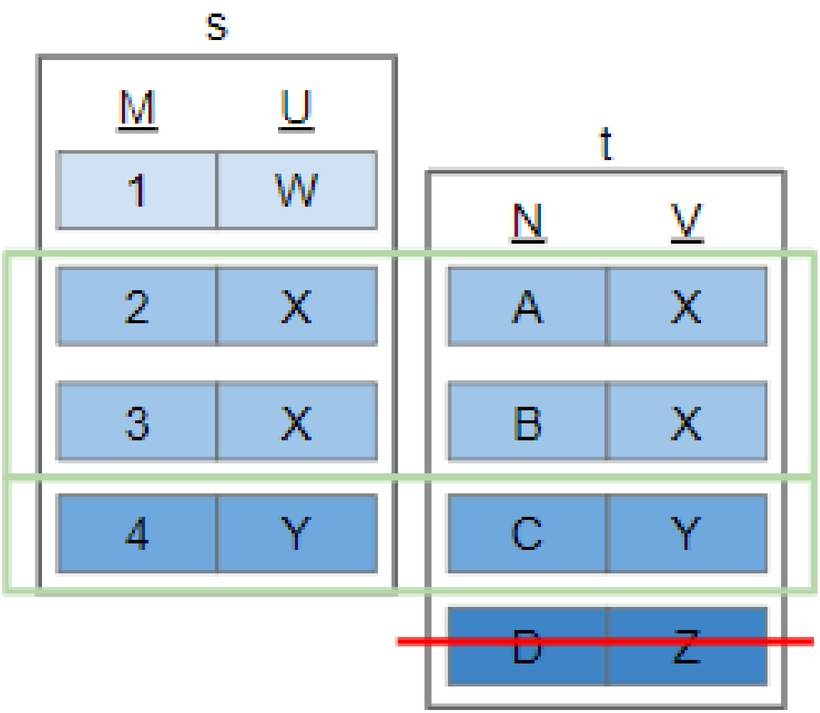
| N | V |
|---|---|
| A | X |
| B | X |
| C | Y |
| ~~D~~ | ~~Z~~ |

| M | U | N | V |
|---|---|---|---|
| 2 | X | A | X |
| 3 | X | A | X |
| 2 | X | B | X |
| 3 | X | B | X |
| 4 | Y | C | Y |

# Left Join

- Left Join pairs each of the rows in the left table to rows in the right table depending on the entries in specific columns.
- The entries of columns in the right table must match for the pair to appear in the Join.





s

| M | U |
|---|---|
| 1 | W |
| 2 | X |
| 3 | X |
| 4 | Y |

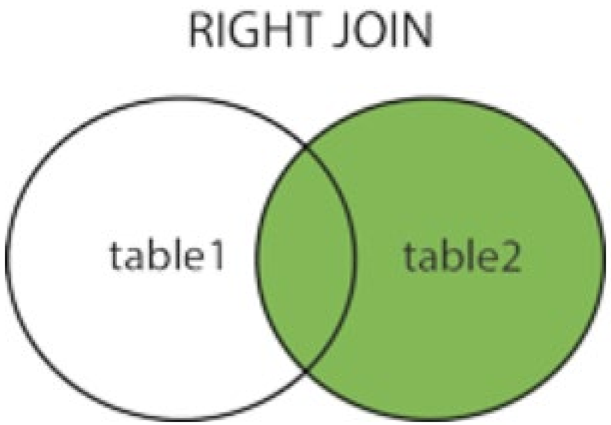
t

| N | V |
|---|---|
| A | X |
| B | X |
| C | Y |
| ~~D~~ | ~~Z~~ |

| | | | |
|---|---|---|---|
| 1 | W | null | null |
| 2 | X | A | X |
| 3 | X | A | X |
| 2 | X | B | X |
| 3 | X | B | X |
| 4 | Y | C | Y |

# Right Join

- Right Join pairs each of the rows in the right table to rows in the left table depending on the entries in specific columns.
- The entries of columns in the left table must match for the pair to appear in the Join.



s

| M | U |
|---|---|
| ~~1~~ | ~~W~~ |
| 2 | X |
| 3 | X |
| 4 | Y |

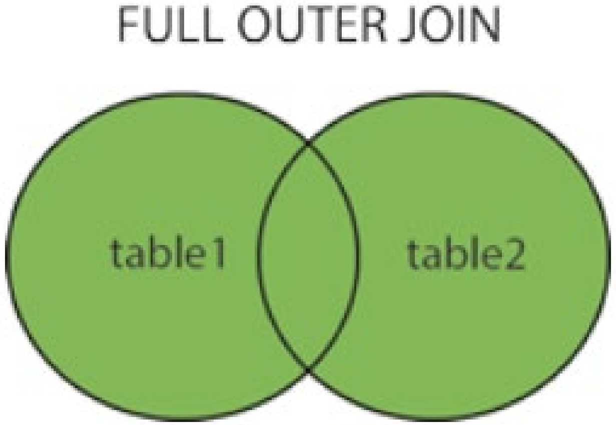
t

| N | V |
|---|---|
| A | X |
| B | X |
| C | Y |
| D | Z |

| | | | |
|---|---|---|---|
| 2 | X | A | X |
| 3 | X | A | X |
| 2 | X | B | X |
| 3 | X | B | X |
| 4 | Y | C | Y |
| null | null | D | Z |

# Outer Join

- Outer Join combines Left Join and Right Join
- Note that Outer Join does not contain duplicate entries for the matching rows, that is, the rows contained in the Inner Join.



s

| M | U |
|---|---|
| 1 | W |
| 2 | X |
| 3 | X |
| 4 | Y |

t

| N | V |
|---|---|
| A | X |
| B | X |
| C | Y |
| D | Z |

| | | | |
|---|---|---|---|
| 1 | W | null | null |
| 2 | X | A | X |
| 3 | X | A | X |
| 2 | X | B | X |
| 3 | X | B | X |
| 4 | Y | C | Y |
| null | null | D | Z |

# Summary

- Data Formats
    - structured, unstructured, semi-structured
- Database Management Systems
    - extract, transform, load
- Online Analytical Processing
    - data cube, keys and indices, slicing/drilling
- Structured Query Language
    - joins