Data Warehousing

http://goo.gl/qRBY58

CS5200 DBMS
Bruce Chhay

Data Warehousing

L1. Data Warehousing and ETLs

Data Warehouse

- Data storage repository used for making decisions.
- Enables reporting for key metrics as well as ad-hoc analysis.
  - Decision making for: sales, marketing, business processes, budgeting, forecasting, financial reporting, etc.
  - Knowledge discovery: statistical model/inference, machine learning, etc.
  - Reporting: visualizations, dashboarding, alerting, etc.

Data Warehouse

- Data storage repository used for making business decisions.
- Enables reporting for key metrics as well as ad-hoc analysis.
  - Decision making for: sales, marketing, business processes, budgeting, forecasting, financial reporting, etc.
  - Knowledge discovery: statistical model/inference, machine learning, etc.
  - Reporting: visualizations, dashboarding, alerting, etc.

Organizing Data

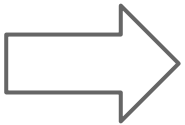
- Pivot tables: summarization tool to re-organize data by aggregation and reducing variables.

Date	Location	Article	Price
4/1/2017	Seattle	Jean Jacket	$100.00
4/1/2017	Seattle	Khakis	$50.00
4/1/2017	Bellevue	Cardigan	$60.00
4/2/2017	Toronto	Flannel Shirt	$90.00
4/2/2017	Toronto	Flannel Shirt	$80.00
4/2/2017	Seattle	T-shirt	$30.00
4/2/2017	Bellevue	Tie	$20.00

Organizing Data

- Pivot tables: summarization tool to re-organize data by aggregation and reducing variables.

Date	Location	Article	Price
4/1/2017	Seattle	Jean Jacket	$100.00
4/1/2017	Seattle	Khakis	$50.00
4/1/2017	Bellevue	Cardigan	$60.00
4/2/2017	Toronto	Flannel Shirt	$90.00
4/2/2017	Toronto	Flannel Shirt	$80.00
4/2/2017	Seattle	T-shirt	$30.00
4/2/2017	Bellevue	Tie	$20.00



Sum(Price)	Date		
Location	**4/1/2017**	**4/2/2017**	**Subtotal**
Bellevue	$60.00	$20.00	$80.00
Seattle	$150.00	$30.00	$180.00
Toronto	$0.00	$170.00	$170.00
Subtotal	$210.00	$220.00	$430.00

Organizing Data

- Compare to SELECT aggregation

Date	Location	Article	Price
4/1/2017	Seattle	Jean Jacket	$100.00
4/1/2017	Seattle	Khakis	$50.00
4/1/2017	Bellevue	Cardigan	$60.00
4/2/2017	Toronto	Flannel Shirt	$90.00
4/2/2017	Toronto	Flannel Shirt	$80.00
4/2/2017	Seattle	T-shirt	$30.00
4/2/2017	Bellevue	Tie	$20.00

```
SELECT Location, Date, SUM(Price) AS Subtotal
FROM Sales
GROUP BY Location, Date WITH ROLLUP
ORDER BY Location, Date
```

Location	Date	Subtotal
Bellevue	4/1/2017	$60.00
Bellevue	4/2/2017	$20.00
Bellevue	NULL	$80.00
Seattle	4/1/2017	$150.00
Seattle	4/2/2017	$30.00
Seattle	NULL	$180.00
Toronto	4/2/2017	$170.00
Toronto	NULL	$170.00
NULL	NULL	$430.00

Organizing Data

- Compare to SELECT aggregation and handling each unique date as separate column.

Date	Location	Article	Price
4/1/2017	Seattle	Jean Jacket	$100.00
4/1/2017	Seattle	Khakis	$50.00
4/1/2017	Bellevue	Cardigan	$60.00
4/2/2017	Toronto	Flannel Shirt	$90.00
4/2/2017	Toronto	Flannel Shirt	$80.00
4/2/2017	Seattle	T-shirt	$30.00
4/2/2017	Bellevue	Tie	$20.00

```
SELECT Location,
  SUM(IF(Date='2017-04-01', Price, 0) AS '4/1/2017',
  SUM(IF(Date='2017-04-02', Price, 0) AS '4/2/2017',
  SUM(Price) AS Subtotal
FROM Sales
GROUP BY Location WITH ROLLUP
```

Location	4/1/2017	4/2/2017	Subtotal
Bellevue	$60.00	$20.00	$80.00
Seattle	$150.00	$30.00	$180.00
Toronto	$0	$170.00	$170.00
NULL	$210.00	$220.00	$430.00

Organizing Data

- Pivot support in MS SQL Server.
  https://docs.microsoft.com/en-us/sql/t-sql/queries/from-using-pivot-and-unpivot?view=sql-server-2017

Date	Location	Article	Price
4/1/2017	Seattle	Jean Jacket	$100.00
4/1/2017	Seattle	Khakis	$50.00
4/1/2017	Bellevue	Cardigan	$60.00
4/2/2017	Toronto	Flannel Shirt	$90.00
4/2/2017	Toronto	Flannel Shirt	$80.00
4/2/2017	Seattle	T-shirt	$30.00
4/2/2017	Bellevue	Tie	$20.00

```
SELECT Location, Date, Price
FROM Sales
PIVOT (
  SUM(Price)
  FOR Date IN ([4/1/2017], [4/2/2017])) AS PVT
ORDER BY Location
```

Location	4/1/2017	4/2/2017
Bellevue	$60.00	$20.00
Seattle	$150.00	$30.00
Toronto	$0.00	$170.00

Organizing Data

- Star schema:
  - Fact table: central table containing aggregate measures/metrics. Foreign key references to dimensions.
  - Dimension tables: dimensions are the granularity used to label the measures. Each dimension table contains the entire domain of values for that dimension.

MonthID	Month
3	March
4	April

Sales	MonthID	CountryID	ArticleID
$50.00	3	1	1
$20.00	4	1	1
$900.00	3	2	1
$700.00	4	2	1

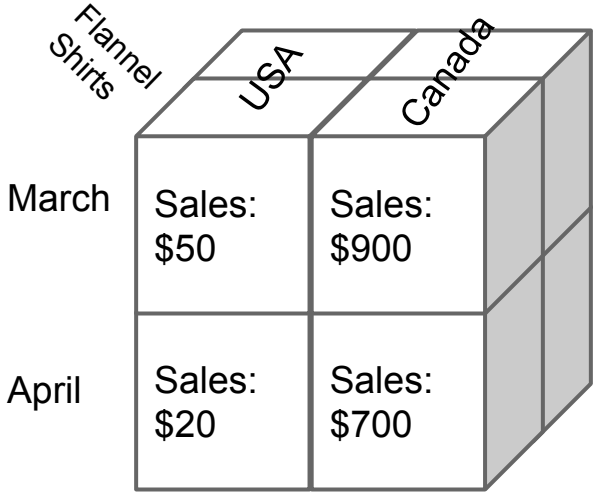
CountryID	Country
1	USA
2	Canada

ArticleID	Article
1	Flannel Shirt

Compare to normalized (3NF) relational model:
Star schema is easy to use and fast to query; if dependencies amongst non-prime attributes, then data inconsistency for updates.
3NF minimizes inconsistency/redundancy, relations and relationships are easier to update.

Organizing Data

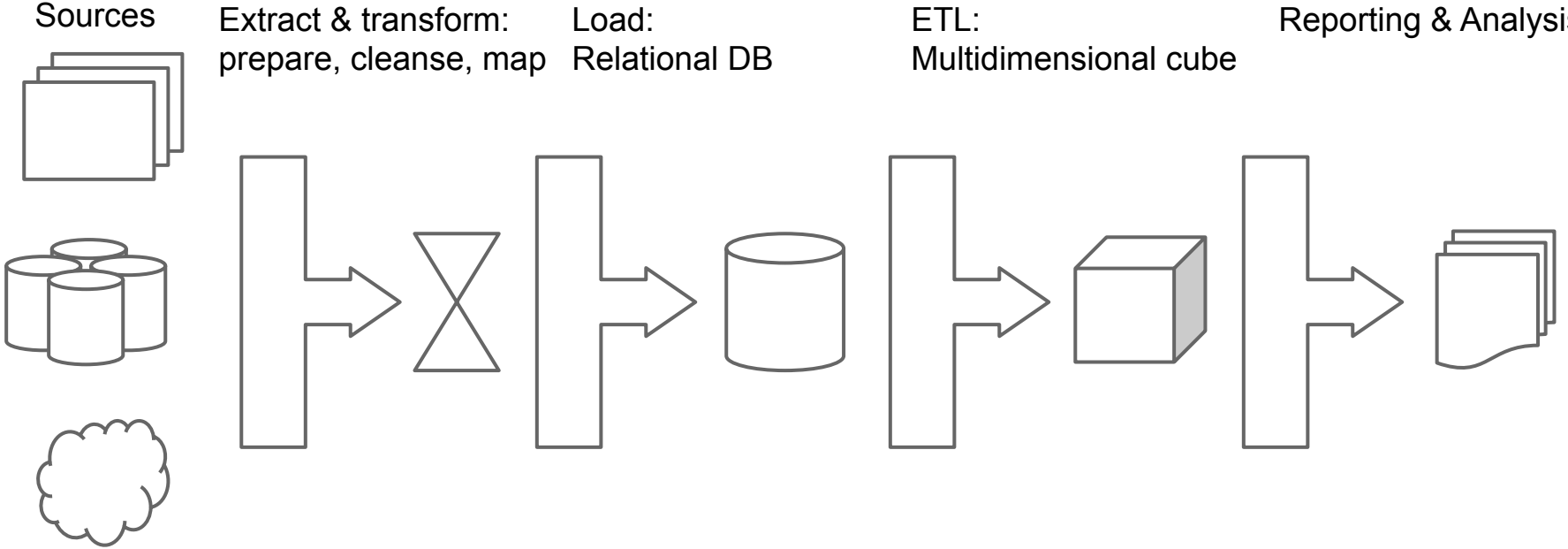
- Multidimensional (OLAP) cubes: consists of measures and dimensions, where a measure is an aggregation described by a set of dimensions.



Limited to three dimensions for illustration purposes, but actually we have a lot more (for example, we can easily include year).

Data Pipeline

- ETL: extract, transform, load.
- Extract from a source, transform according to business rules, load to data warehouse.



Tools

- Microsoft SQL Server Business Intelligence (http://www.microsoft.com/en-us/server-cloud/solutions/business-intelligence/, http://www.microsoft.com/en-us/powerbi/default.aspx), HP Vertica (http://www.vertica.com/about/), IBM InfoSphere (http://www-03.ibm.com/software/products/en/category/SWP00), Informatica (www.informatica.com), Talend (http://www.talend.com/download), Tableau (http://public.tableau.com/s/), Pentaho (http://community.pentaho.com/), etc.

Data Warehousing

L2. Exercise 1: BlogApplication

Setup

- Install CloverDX: https://www.cloverdx.com/trial-cloverdx
- Documentation: https://learn.cloverdx.com/quickstart/
- See “Quick Start Guide” and “Examples” in CloverDX Welcome page

Exercise 1: BlogApplication

- Goal: how many BlogPosts are created per day, and are there any trends?

Exercise 1: BlogApplication

- Let’s seed the data first.
- Recreate the model: http://goo.gl/86a11H
- Run:

```
USE BlogApplication;
INSERT INTO Persons(UserName,FirstName,LastName) VALUES('foo','first','last');
INSERT INTO BlogUsers(UserName,DoB,StatusLevel) VALUES('foo','1990-04-01','novice');
INSERT INTO BlogPosts(Title,Content,Published,UserName,Created) VALUES('cats1','cats',True,'foo','2016-11-18 01:00:00');
INSERT INTO BlogPosts(Title,Content,Published,UserName,Created) VALUES('cats2','cats',True,'foo','2016-11-19 01:00:00');
INSERT INTO BlogPosts(Title,Content,Published,UserName,Created) VALUES('cats3','cats',True,'foo','2016-11-19 01:00:00');
INSERT INTO BlogPosts(Title,Content,Published,UserName,Created) VALUES('cats4','cats',True,'foo','2016-11-20 01:00:00');
INSERT INTO BlogPosts(Title,Content,Published,UserName,Created) VALUES('cats5','cats',True,'foo','2016-11-20 01:00:00');
INSERT INTO BlogPosts(Title,Content,Published,UserName,Created) VALUES('cats6','cats',True,'foo','2016-11-20 01:00:00');
INSERT INTO BlogPosts(Title,Content,Published,UserName,Created) VALUES('cats7','cats',True,'foo','2016-11-20 01:00:00');
INSERT INTO BlogPosts(Title,Content,Published,UserName,Created) VALUES('cats8','cats',True,'foo','2016-11-20 01:00:00');
DROP TABLE IF EXISTS DWPost;
CREATE TABLE DWPost (DWPostId INT AUTO_INCREMENT,Created DATE,Count INT,CONSTRAINT pk_DWPost_DWPostId PRIMARY KEY (DWPostId));
```

Exercise 1: BlogApplication

- Similar to “Accessing a Database” example
  - Should be DatabaseAccess.grf under RealWorldExamples > graph
- Create a new CloverETL Project, new ETL Graph (MySql.grf)
- Add Readers > DatabaseReader
  - Configure new connection (and validate!) (URL: jdbc:mysql://localhost:3306/BlogApplication)
  - Build the query (and validate) (SELECT BlogPosts.Created FROM BlogPosts ORDER BY Created)
  - Extract Metadata (Set delimiter, format timestamp to date “MM/dd/yyyy”)

Exercise 1: BlogApplication

- Add Transformers > Aggregate
- Add Writers > Trash
  - Debug print: true
- Add edge and create metadata (Aggregate output fields).
  - Add Created (date) and Count (integer) fields
- Update Aggregate properties
  - Aggregate key: Created
  - Aggregation mapping: $Created:=$Created;$Count:=count(); (Also check out the mapping builder)
  - Click “?” for more details
- Save, run, and view output!

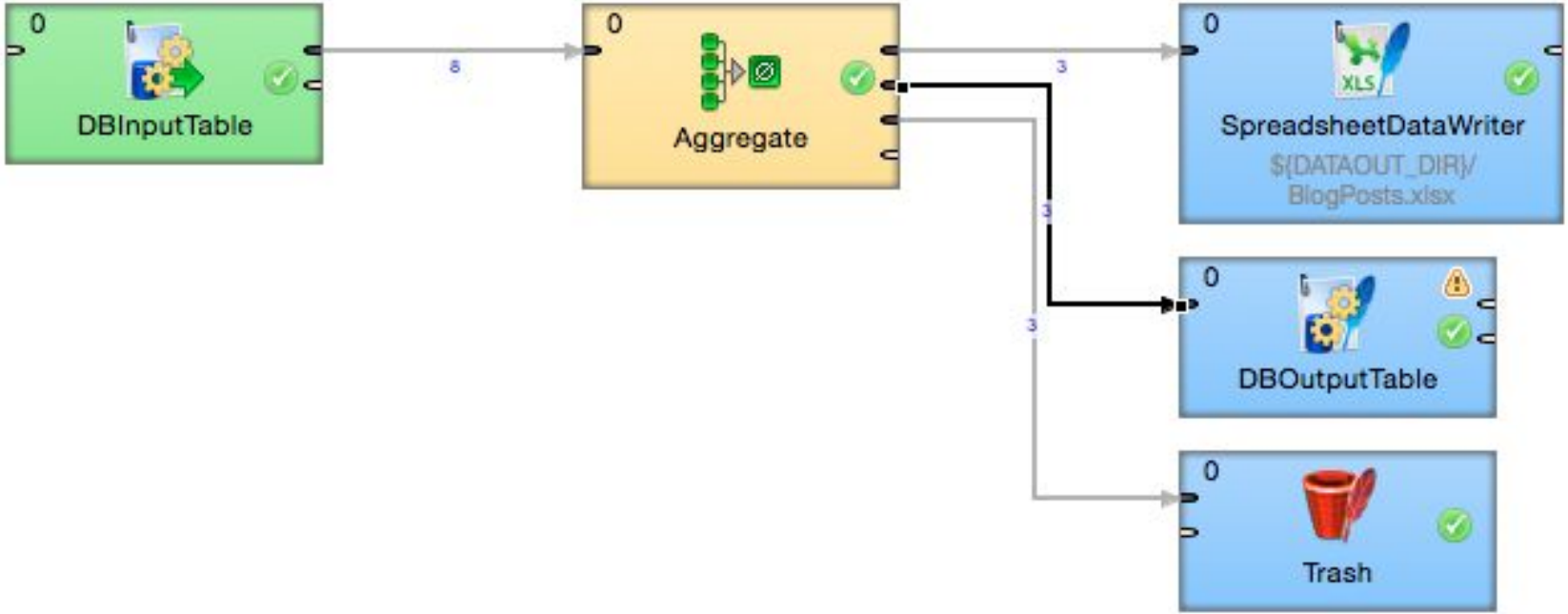
Exercise 1: BlogApplication

- Add Writers > SpreadsheetDataWriter
  - Add edge from Aggregate to SpreadsheetDataWriter
  - File URL: data-out/BlogPosts.xlsx
  - Sheet: Sheet0
  - Mapping: use builder
  - Write mode: overwrite
  - Existing sheets: replace
- Save, run, and view XLSX.
  - Format Created to Date
  - Create a chart
  - Trend?

Exercise 1: BlogApplication

- Add Writers > DatabaseWriter
  - Add edge from Aggregate to DatabaseWriter.
  - DB Connection: re-use the MySQL connection
  - DB Table: DWPost
  - Field Mapping: use builder ($Created:=Created;$Count:=Count;)
- Save, run, and view DWPost table.
  - What other dimensions can we add?
  - What if we run this nightly?

Exercise 1


0
DBInputTable
8
0
Aggregate
3
3
3
0
SpreadsheetDataWriter
${DATAOUT_DIR}/
BlogPosts.xlsx
0
DBOutputTable
0
Trash

Data Warehousing

L3. Exercise 2: Bike Weather

Exercise 2: Bike Weather

- Goal: does temperature affect bike crossings on the Fremont Bridge?

seattletimes.com - Fremont bridge bike counter

Exercise 2: Bike Weather

- Download the data (as CSV):
  - Bike count: https://data.seattle.gov/Transportation/Fremont-Bridge-Hourly-Bicycle-Counts-by-Month-Octo/65db-xm6k
  - Weather: https://data.seattle.gov/Transportation/Road-Weather-Information-Stations/egc4-d24i
    - Filter StationName “AuroraBridge”
  - Also available at: Shared Downloads
- Open the CSVs to understand the format.
  - How can they be combined?
  - Bike, 2015 Oct - 2012 Oct, Hour. (27K records)
  - Weather, 2015 Nov - 2014 Mar, Minute. (1M records)
  - 2015 Oct - 2014 Mar, Hour; AirTemperature vs. E+W Bike

Exercise 2: Bike Weather

- Similar to “Joining & Aggregating” example
  - Should be Joining_Aggregating.grf under RealWorldExamples > graph
- Create a new ETL Graph (BikeWeather.grf)
- Move CSV files to data-in directory.

Exercise 2: Bike Weather

- Add Readers > FlatFileReader
  - File URL: data-in/Bikes.csv
  - Data Policy: Lenient
  - Trim Strings: true
  - Quoted Strings: false
  - (Max number of records: - useful for debugging, use 15K)
  - Skip first line: true

Exercise 2: Bike Weather

- Add Readers > FlatFileReader
  - File URL: data-in/Weather.csv
  - Data Policy: Lenient
  - Trim Strings: true
  - Quoted Strings: true
  - Quote character: "
  - (Max number of records: - useful for debugging, use 100K)
  - Skip first line: true

Exercise 2: Bike Weather

- FlatFileReader (Bikes.csv) > Extract Metadata
  - Use the builder to view and parse data from CSV
  - Change “Date” field to date data type
    - Change format to “MM/dd/yyyy hh:mm:ss a”
    - Java date time symbols: https://docs.oracle.com/javase/8/docs/api/java/time/format/DateTimeFormatter.html
  - Change bike count fields to “West” and “East”
  - Verify parsed data looks correct!

Exercise 2: Bike Weather

- FlatFileReader (Weather.csv) > Extract Metadata
  - Use the builder to view and parse data from CSV
  - Change Quote char to " and click “Reparse”
  - Change “DateTime” field to date data type
    - Change format to “MM/dd/yyyy hh:mm:ss a”
    - Java date time symbols: https://docs.oracle.com/javase/8/docs/api/java/time/format/DateTimeFormatter.html
  - Verify parsed data looks correct!

Exercise 2: Bike Weather

- Add 2X Writers > Trash
  - Debug print: true
- Add edges and use the created metadata
- Set FlatFileReader Max number of records: 100
- Save, run, and view output

Exercise 2: Bike Weather

- Add Joiners > ExtHashJoin
- Add edges and use the created metadata
- Add Writers > Trash
  - Debug print: true
- Add edge and create metadata (metadata can be created from ExtHashJoin, too).
  - Date date (format: “MM/dd/yyyy hh:mm:ss a”)
  - BikeCount integer
  - AirTemperature decimal
  - Set delimiter to “,”

Exercise 2: Bike Weather

- Update ExtHashJoin properties
  - Join key: use builder to map Date and DateTime ($Date=$DateTime)
  - Join type: inner join
  - Transform: use builder to map:
    - Date -> Date (safe: nvl($in.0.Date, createDate(2014,1,1)))
    - West + East -> BikeCount (safe: nvl($in.0.West,0) + nvl($in.0.East,0))
    - AirTemperature -> AirTemperature
      (safe: nvl($in.1.AirTemperature,0.0))
- Set Bike/Weather Max number of records: 15,000/100,000
  - Or gradually turn up the volume until you view output
- Save, run, and view output

Exercise 2: Bike Weather

- Create a DW table to store results

```
Use BlogApplication;
DROP TABLE IF EXISTS BikeWeather;
CREATE TABLE BikeWeather(BikeWeatherId INT AUTO_INCREMENT, Date TIMESTAMP, BikeCount INT, AirTemperature
DECIMAL, CONSTRAINT pk_BikeWeather_BikeWeatherId PRIMARY KEY (BikeWeatherId));
```

- Add Writer > DatabaseWriter
  - Add edge from ExtHashJoin to DatabaseWriter.
  - DB Connection: re-create the MySQL connection
  - DB Table: BikeWeather
  - Field Mapping: use builder

    ($Date:=Date;$BikeCount:=BikeCount;$AirTemperature:=AirTemperature;)

Exercise 2: Bike Weather

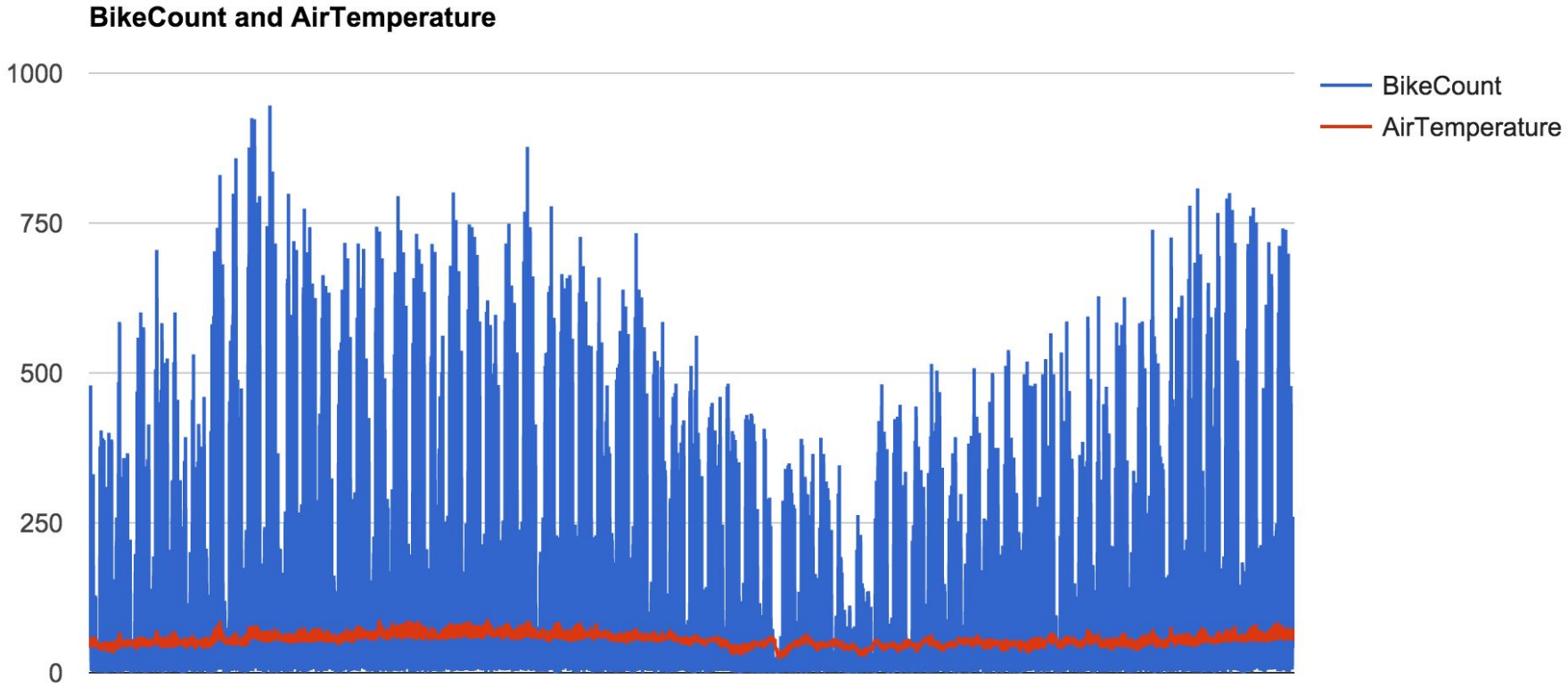
- Now create phase 1 to read from table and generate report
- Add Reader > DatabaseReader
  - DB Connection: re-use the MySQL connection
  - SQL Query: use the builder?

    SELECT Date, BikeCount, AirTemperature FROM BlogApplication.BikeWeather ORDER BY Date
  - Phase: 1
  - Extract metadata
    - Set delimiter to “,”

Exercise 2: Bike Weather

- Add Writer > SpreadsheetDataWriter
  - Add edge from DatabaseReader to SpreadsheetDataWriter
  - File URL: data-out/BikeWeather.xlsx
  - Sheet: Sheet0
  - Mapping: use builder
  - Write mode: overwrite
  - Existing sheets: clear target sheet
  - Phase: 1
- Save, run, and generate chart!
  - Format date if necessary
  - Conclusive?

Exercise 2: Bike Weather



Exercise 2: Bike Weather

- Now create phase 2 to clean up the results
  - Total bike count per day; Weekdays
- Add Reader > DatabaseReader
  - DB Connection: re-use the MySQL connection
  - SQL Query: use the builder?

```
SELECT Date(Date) AS Day, SUM(BikeCount) AS TotalBikes, MAX(AirTemperature) AS MaxWeather
FROM BlogApplication.BikeWeather
WHERE DAYOFWEEK(Date) > 1 AND DAYOFWEEK(Date) < 7
GROUP BY Date(Date)
ORDER BY Day
```

  - Phase: 2
  - Extract metadata
    - Set delimiter to “,”

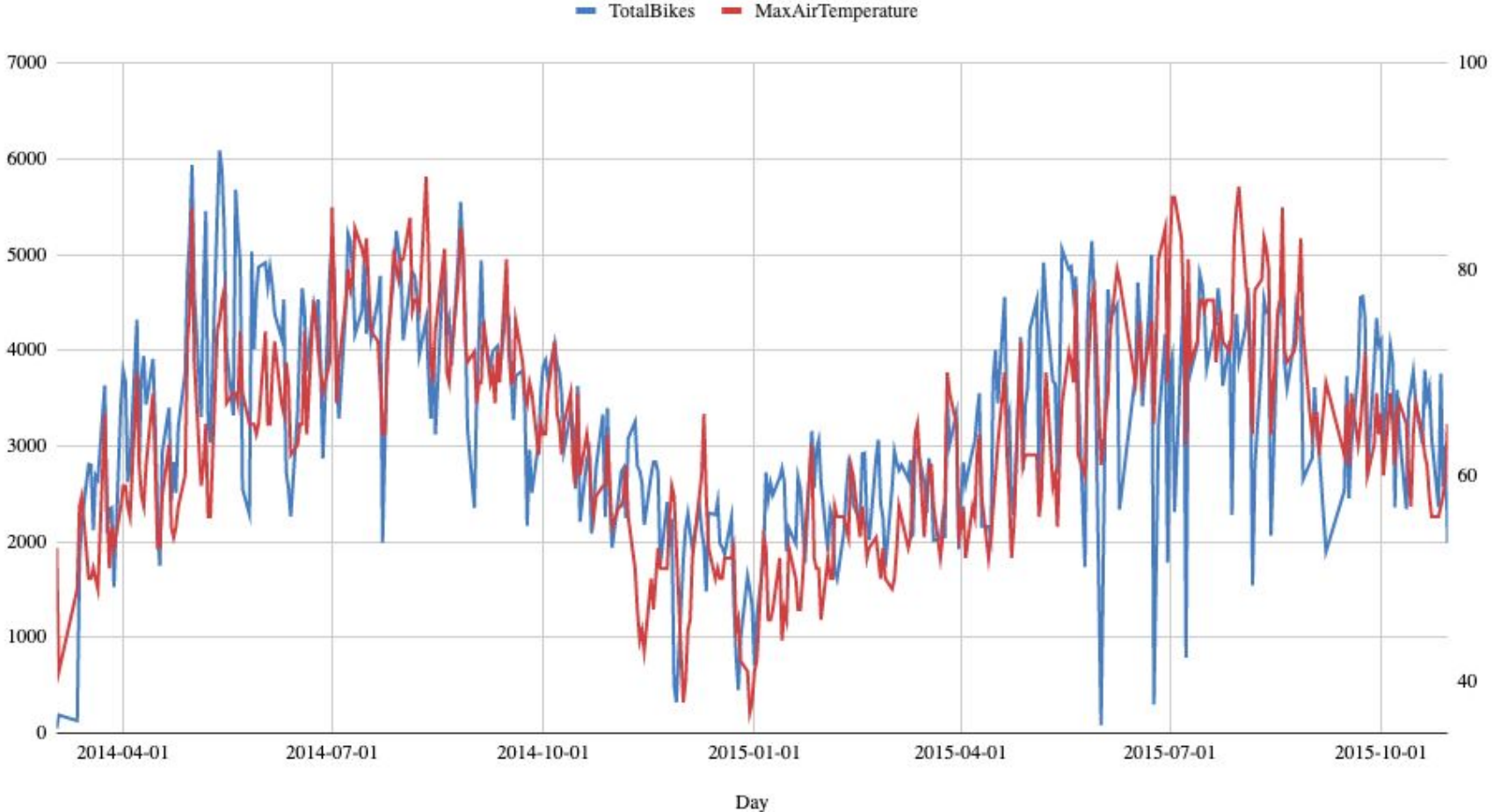
Exercise 2: Bike Weather

- Add Writer > SpreadsheetDataWriter
  - Add edge from DatabaseReader to SpreadsheetDataWriter
  - File URL: data-out/BikeWeather.xlsx
  - Sheet: Sheet1
  - Mapping: use builder
  - Write mode: overwrite
  - Existing sheets: clear target sheet
  - Phase: 2
- Drop & recreate BikeWeather table each time before running
- Save, run, and generate chart!
  - Format date if necessary

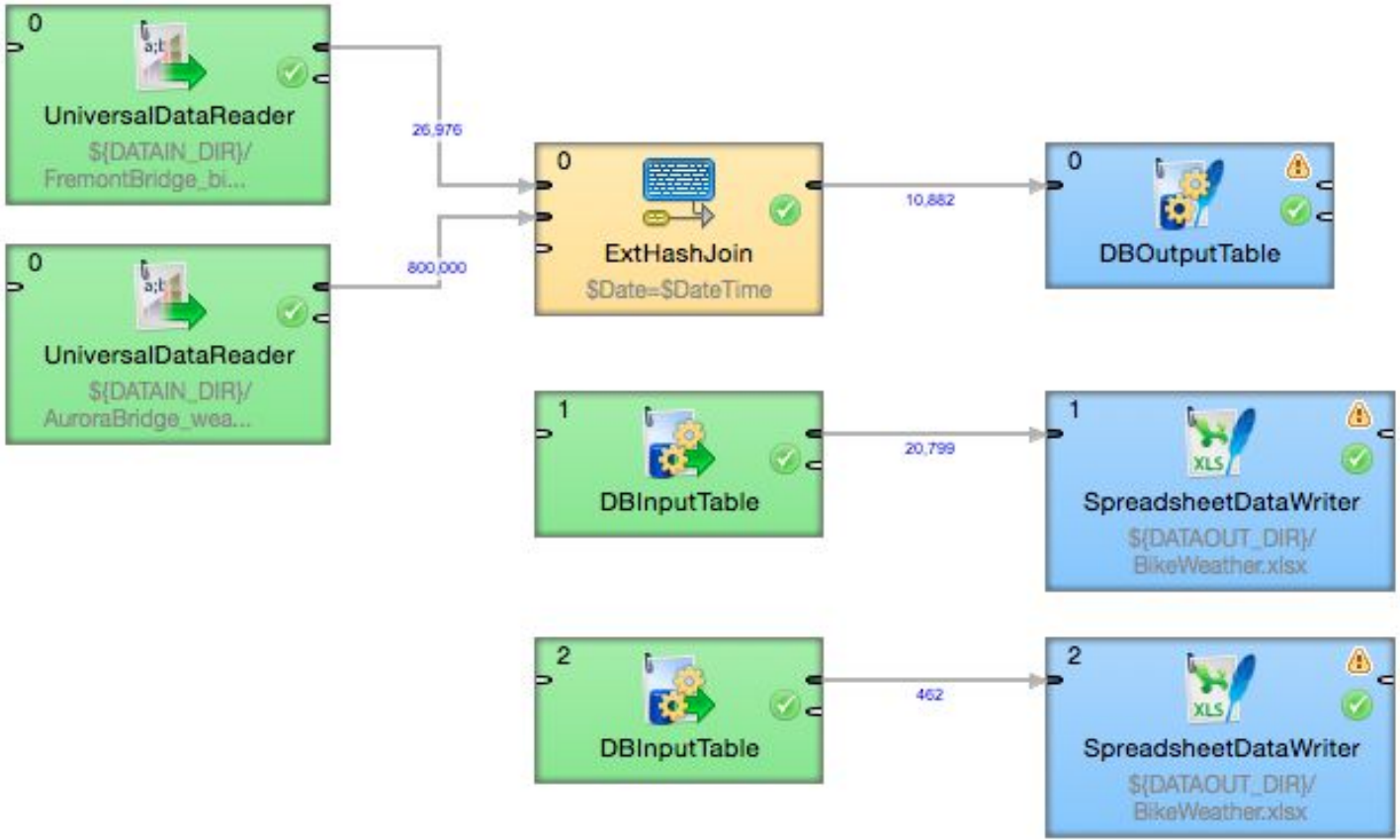
Exercise 2: Bike Weather

- Update FlatFileReader
  - Remove Max number of records to parse full data source
  - You might run out of memory… process incrementally and/or continue to use Max number records (e.g. 800K for weather)
- Drop & recreate BikeWeather table each time before running
- Save, run, and generate chart!
  - Format date if necessary
  - How do the results look now?

Exercise 2: Bike Weather



Exercise 2: Bike Weather



New Models

- What happens when you have more data than you can process in an ETL and store in a data warehouse?
- What if the structure of the data is always changing?
- What if you want to process data real-time?

Data Warehousing

Discussion: PM5

PM5

- Project Milestone 5: find two external data sources that can provide value when combined with your data.
- Examples:
  - City of Seattle: https://data.seattle.gov
  - Socrata: http://www.opendatanetwork.com/, http://www.socrata.com/customer-stories/
  - Government Open Data: http://www.data.gov/
  - Health Data: http://www.healthdata.gov/, http://www.who.int/gho/database/en/
  - Global socio-economic data: https://data.worldbank.org/data-catalog
  - Kaggle: https://www.kaggle.com/