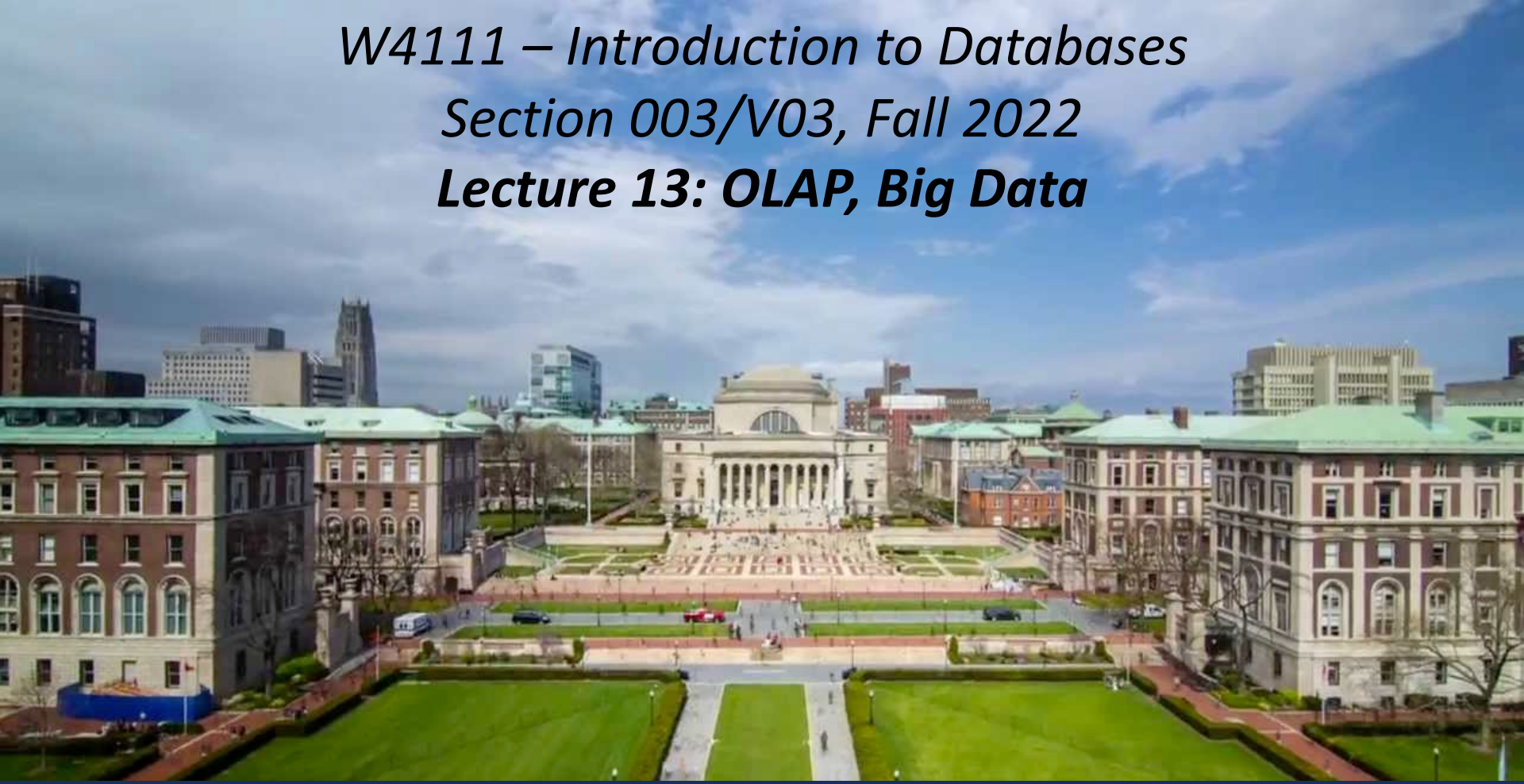
W4111 – Introduction to Databases
Section 003/V03, Fall 2022
Lecture 13: OLAP, Big Data



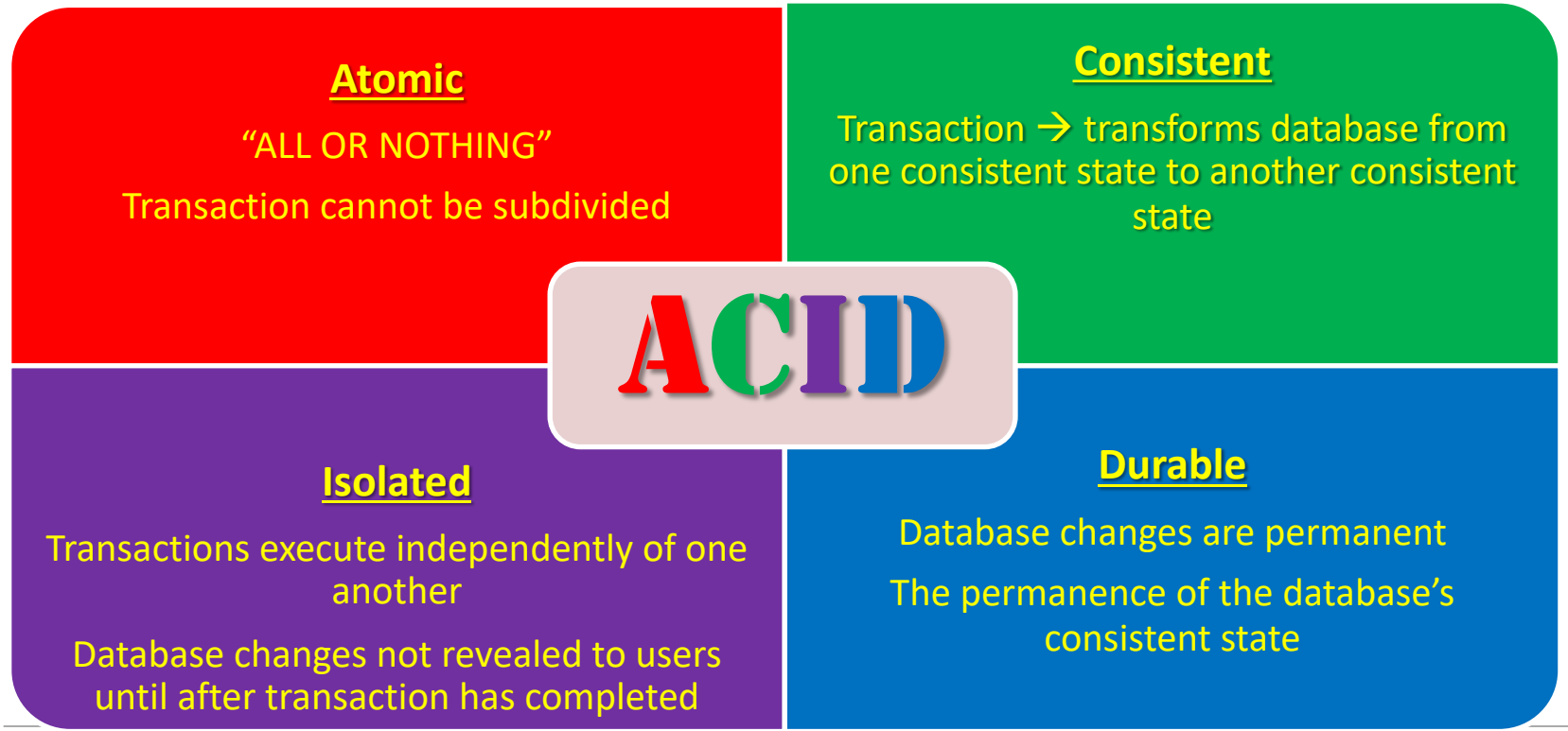
COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

Contents

CAP, BASE

Core Transaction Concept is ACID Properties

http://slideplayer.com/slide/9307681



CAP Theorem

- Consistency

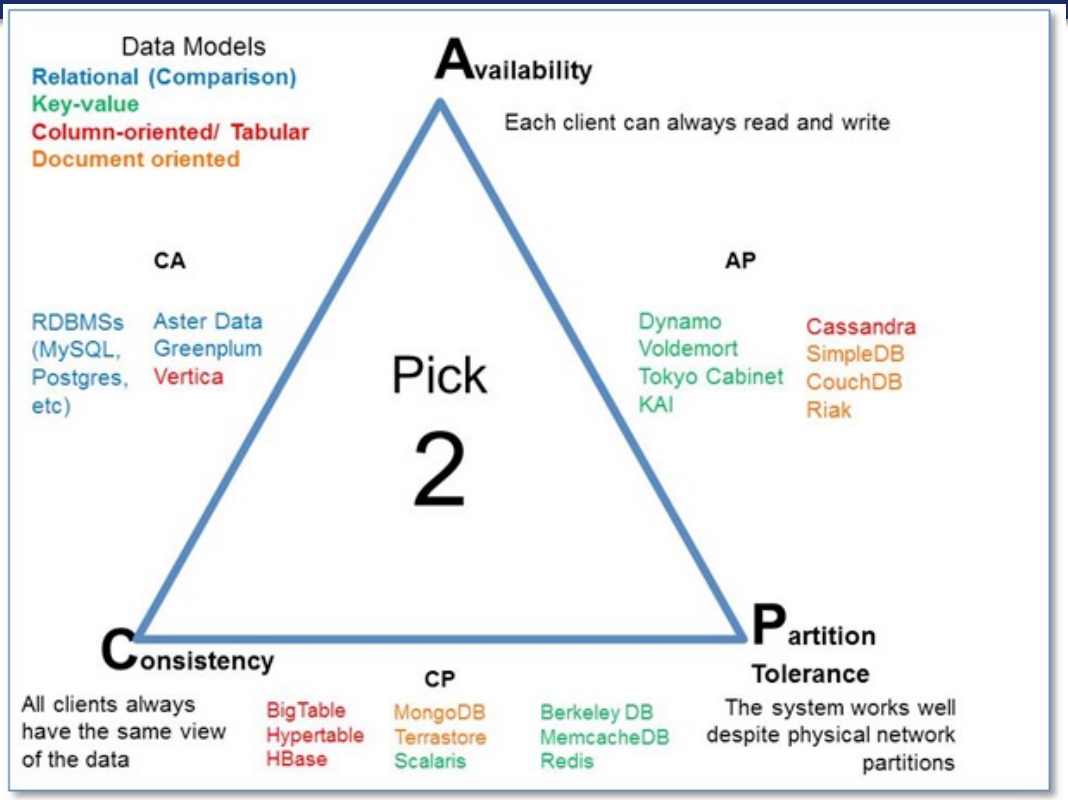
  Every read receives the most recent write or an error.

- Availability

  Every request receives a (non-error) response – without guarantee that it contains the most recent write.

- Partition Tolerance

  The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes



Consistency Models

- **STRONG CONSISTENCY:** Strong consistency is a consistency model where all subsequent accesses to a distributed system will always return the updated value after the update.
- **WEAK CONSISTENCY:** It is a consistency model used in distributed computing where subsequent accesses might not always be returning the updated value. There might be inconsistent responses.
- **EVENTUAL CONSISTENCY:** Eventual consistency is a special type of weak consistency method which informally guarantees that, if no new updates are made to a given data item, eventually all accesses to that item will return the last updated value.

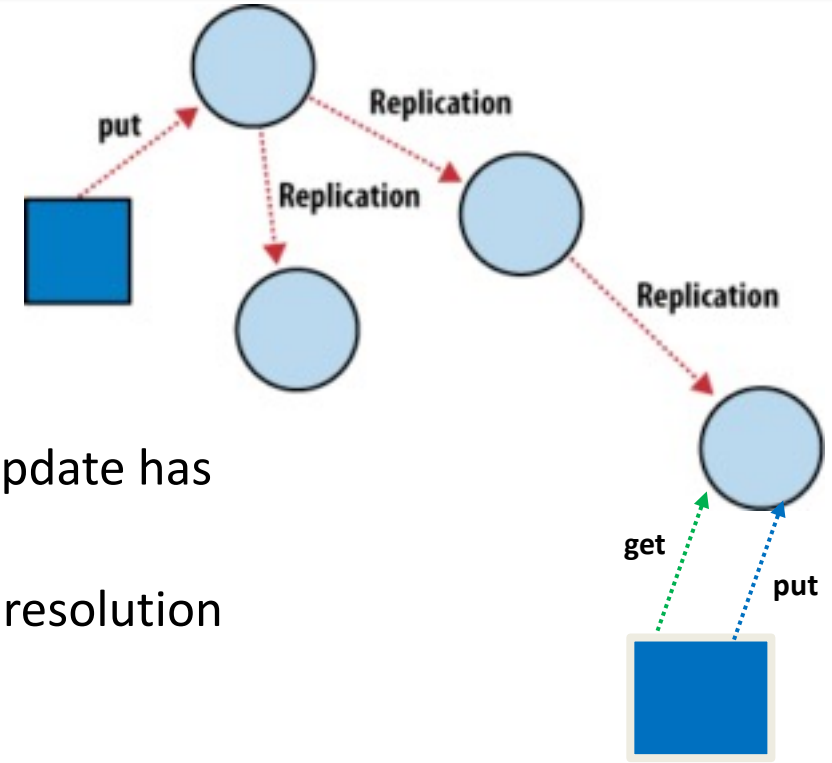
BASE

Eventually-consistent services are often classified as providing BASE (Basically Available, Soft state, Eventual consistency) semantics, in contrast to traditional ACID (Atomicity, Consistency, Isolation, Durability) guarantees. Rough definitions of each term in BASE:

- **Basically Available**: basic reading and writing operations are available as much as possible (using all nodes of a database cluster), but without any kind of consistency guarantees (the write may not persist after conflicts are reconciled, the read may not get the latest write)
- **Soft state**: without consistency guarantees, after some amount of time, we only have some probability of knowing the state, since it may not yet have converged
- **Eventually consistent**: If the system is functioning and we wait long enough after any given set of inputs, we will eventually be able to know what the state of the database is, and so any further reads will be consistent with our expectations

Eventual Consistency

- Availability and scalability via
  - Multiple, replicated data stores.
  - Read goes to “any” replica.
  - PUT/POST/DELETE
    - Goes to any replica
    - Change propagate asynchronously
- GET may not see the latest value if the update has not propagated to the replica.
- There are several algorithms for conflict resolution
  - Detect and handle in application.
  - Clock/change vectors/version numbers
  - … …



ACID – BASE (Simplistic Comparison)

ACID (relational)	BASE (NoSQL)
Strong consistency	Weak consistency
Isolation	Last write wins (Or other strategy)
Transaction	Program managed
Robust database	Simple database
Simple code (SQL)	Complex code
Available and consistent	Available and partition-tolerant
Scale-up (limited)	Scale-out (unlimited)
Shared (disk, mem, proc etc.)	Nothing shared (parallellizable)

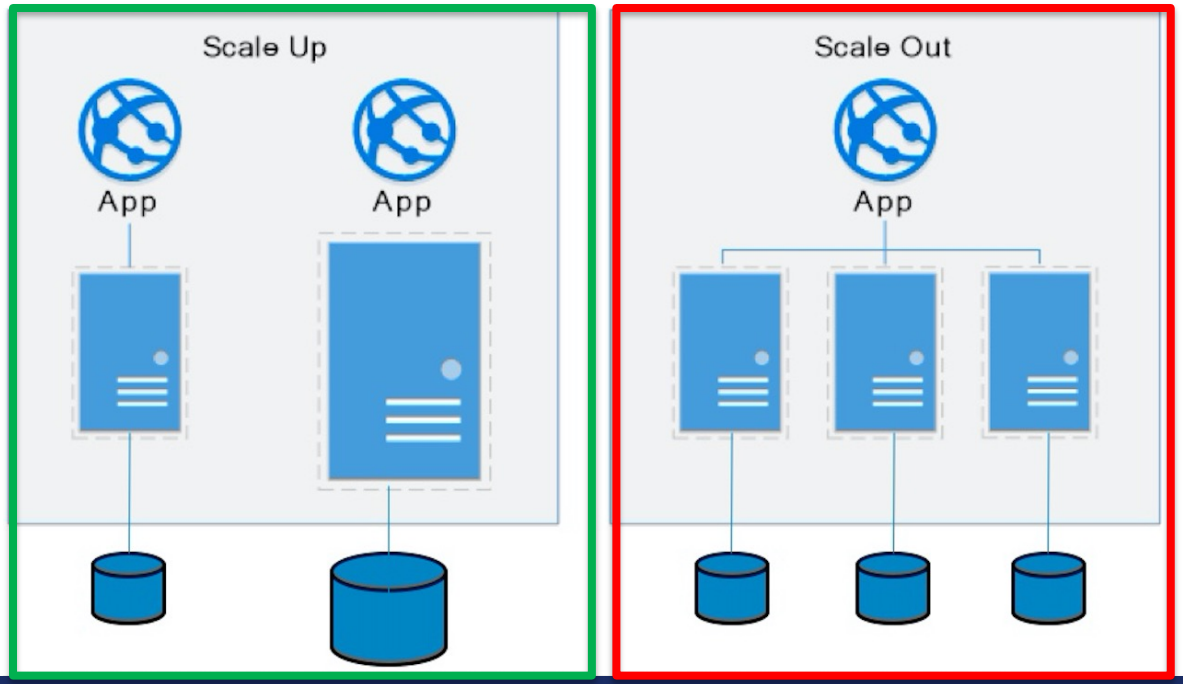
Scalability

Approaches to Scalability

Scalability is the property of a system to handle a growing amount of work by adding resources to the system.

Replace system with a bigger machine,
e.g. more memory, CPU, … …

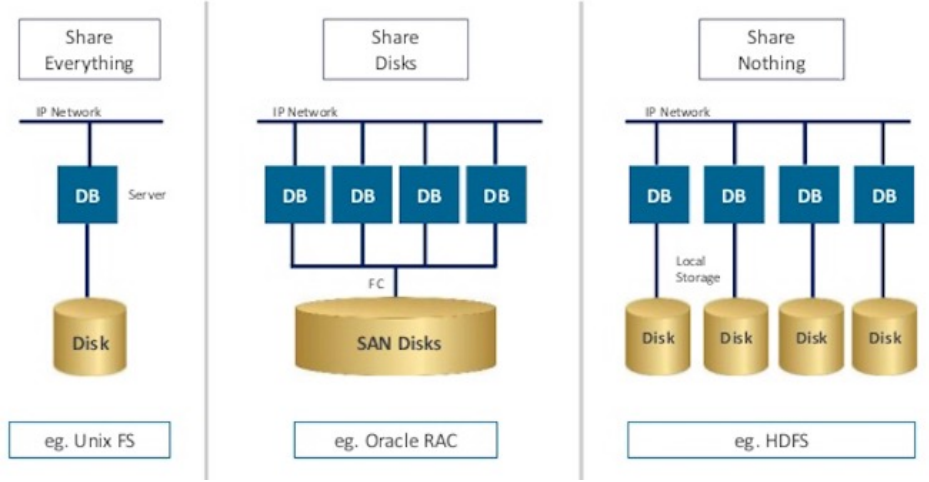
Add another system.



- Scale-up:
  - Less incremental.
  - More disruptive.
  - More expensive for extremely large systems.
  - Does not improve availability
- Scale-out:
  - Incremental cost.
  - Data replication enables availability.
  - Does not work well for functions like JOIN, referential integrity, … …

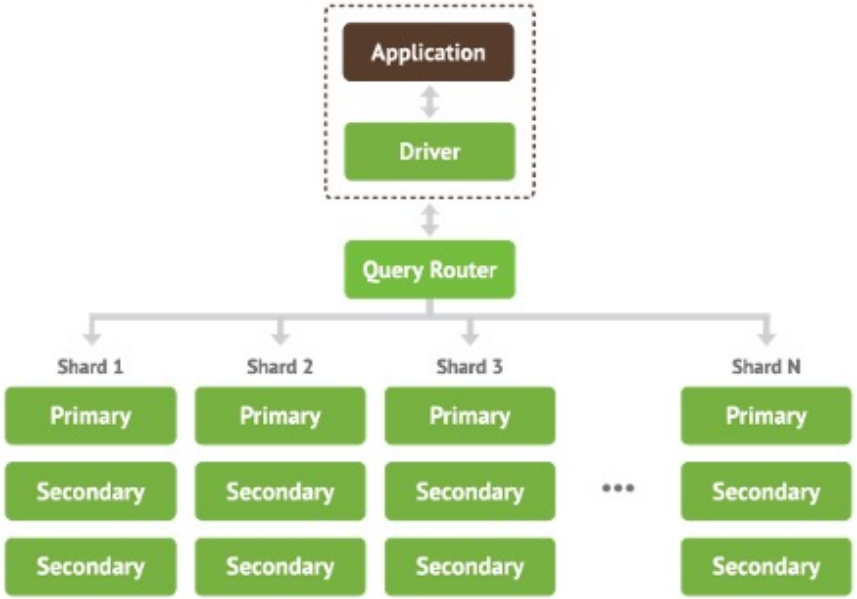
Disk Architecture for Scale-Out

- Share disks:
  - Is basically scale-up for data/disks. You can use NAS, SAN and RAID.
  - Isolation/Integrity requires distributed locking to control access from multiple database servers.
- Share nothing:
  - Is basically scale-out for disks.
  - Data is partitioned into *shards* based on a function f() applied to a key.
  - Can improve availability, at the code consistency, with data replication.
  - There is a router that sends requests to the proper shard based on the function.

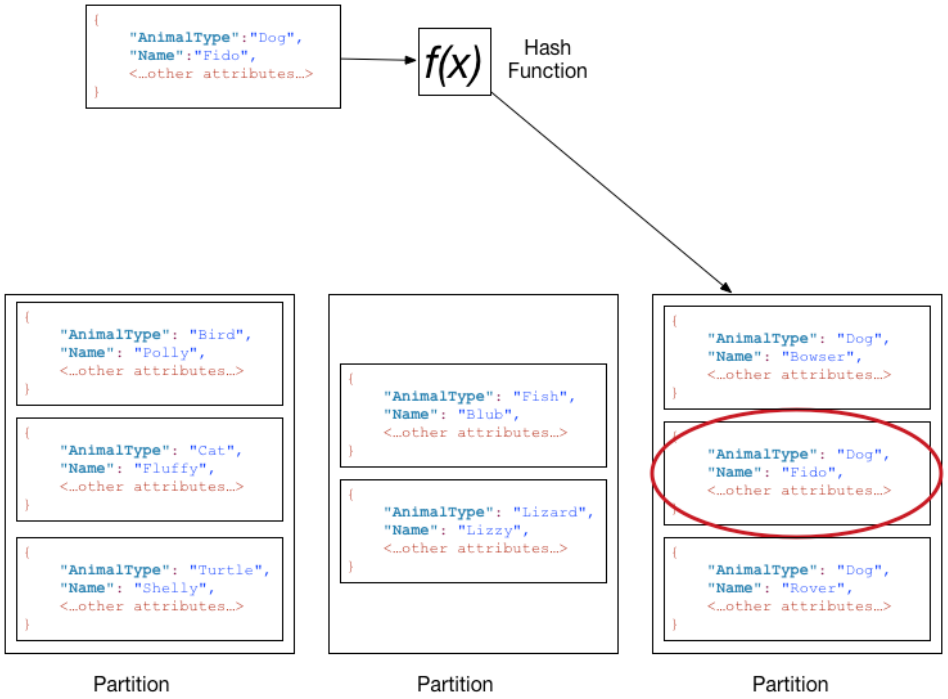


Shared Nothing, Scale-Out

MongoDB Sharding



DynamoDB Partitioning



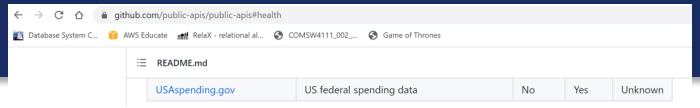
Data Models, REST, HATEOS

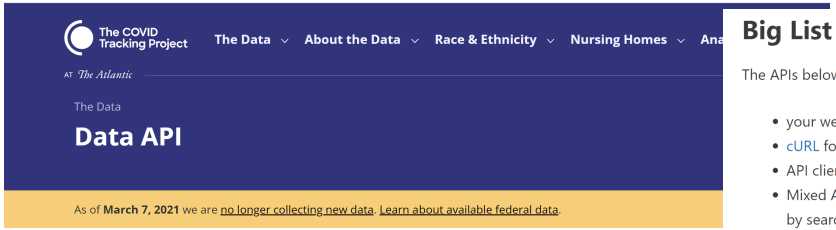
Data Modeling Concepts and the Web

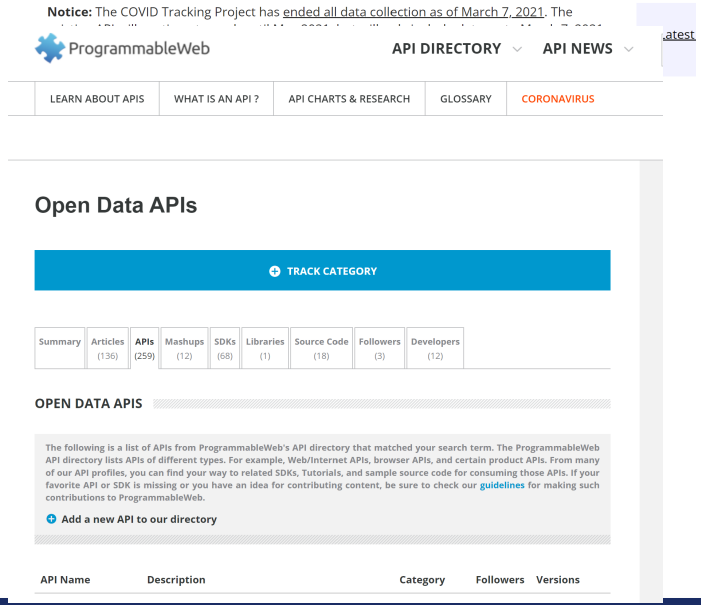
Almost any data model has the same core concepts:

- Types and instances:
  - Entity Type: A definition of a type of thing with properties and relationships.
  - Entity Instance: A specific instantiation of the Entity Type
  - Entity Set Instance: An Entity Type that:
    - Has properties and relationships like any entity, but ...
    - Has at least one *special relationship* – ***contains.***
- Operations, minimally CRUD, that manipulate entity types and instances:
  - Create (POST)
  - Retrieve (GET)
  - Update (PUT)
  - Delete (DELETE)
  - Reference/Identity/... ... (URL)

Data Web





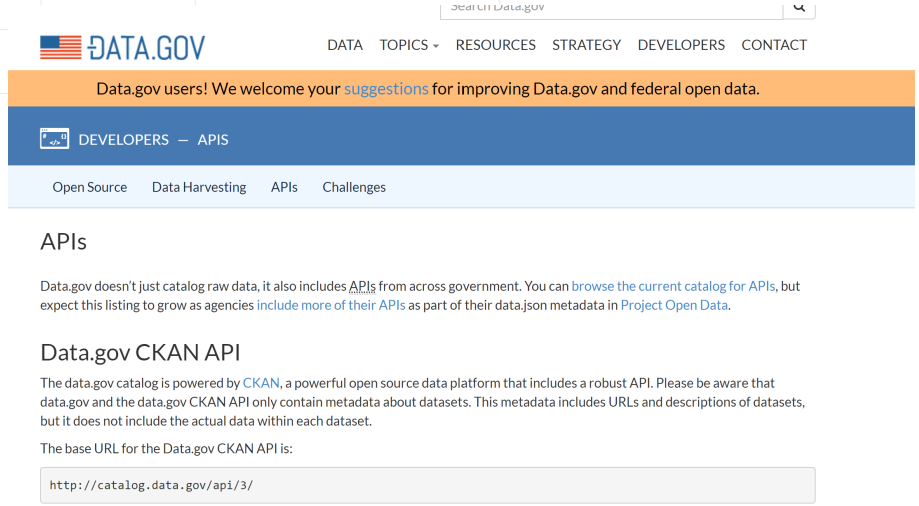


Big List of Free Open APIs

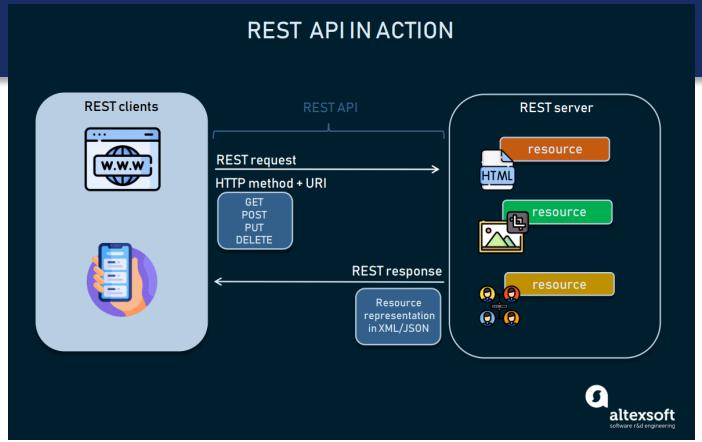
The APIs below can be accessed using any method:

- your web browser
- cURL for the command line
- API clients like Swagger and Postman
- Mixed Analytics' own API Connector for Google Sheets (all the below APIs can be found by searching for the keyword "Open" in the API field).

#	API NAME	DESCRIPTION	SAMPLE URL
1	7Timer!	Weather forecasts	http://www.7timer.info/bin/api.pl?lon=113.17&lat=23.09&product=astro&out
2	Agify.io		

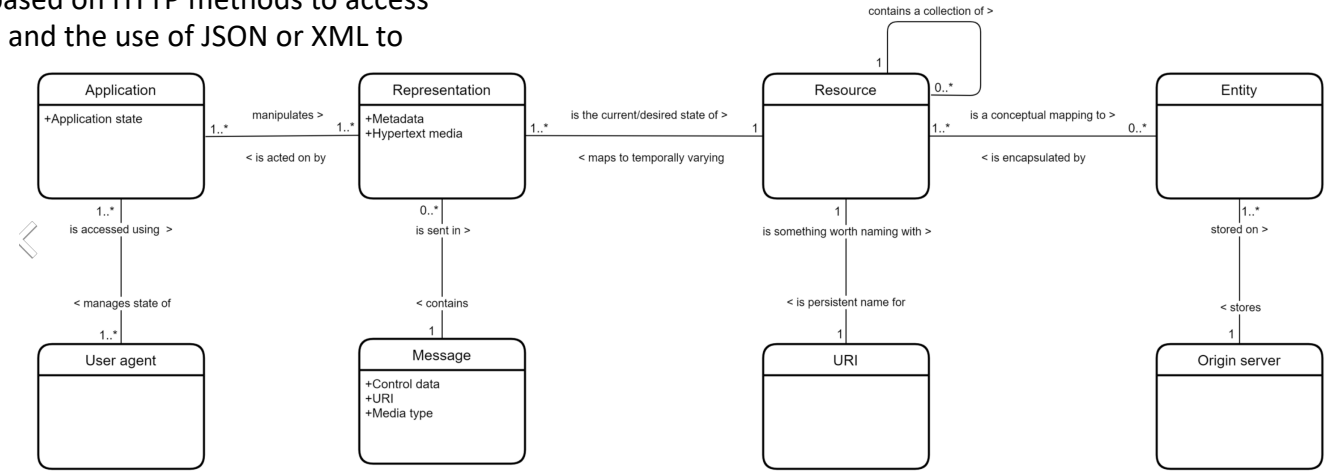


REST



"Representational state transfer (REST) is a software architectural style that was created to guide the design and development of the architecture for the World Wide Web. REST defines a set of constraints for how the architecture of an Internet-scale distributed hypermedia system, such as the Web, should behave. … …

REST has been employed throughout the software industry and is a widely accepted set of guidelines for creating stateless, reliable web APIs. A web API that obeys the REST constraints is informally described as RESTful. RESTful web APIs are typically loosely based on HTTP methods to access resources via URL-encoded parameters and the use of JSON or XML to transmit data."



HATEOAS

“Hypermedia as the Engine of Application State (HATEOAS) is a constraint of the REST application architecture that distinguishes it from other network application architectures.

With HATEOAS, a client interacts with a network application whose application servers provide information dynamically through hypermedia. A REST client needs little to no prior knowledge about how to interact with an application or server beyond a generic understanding of hypermedia.

The restrictions imposed by HATEOAS decouple client and server. This enables server functionality to evolve independently.”

Request

```
GET /accounts/12345 HTTP/1.1
Host: bank.example.com
```

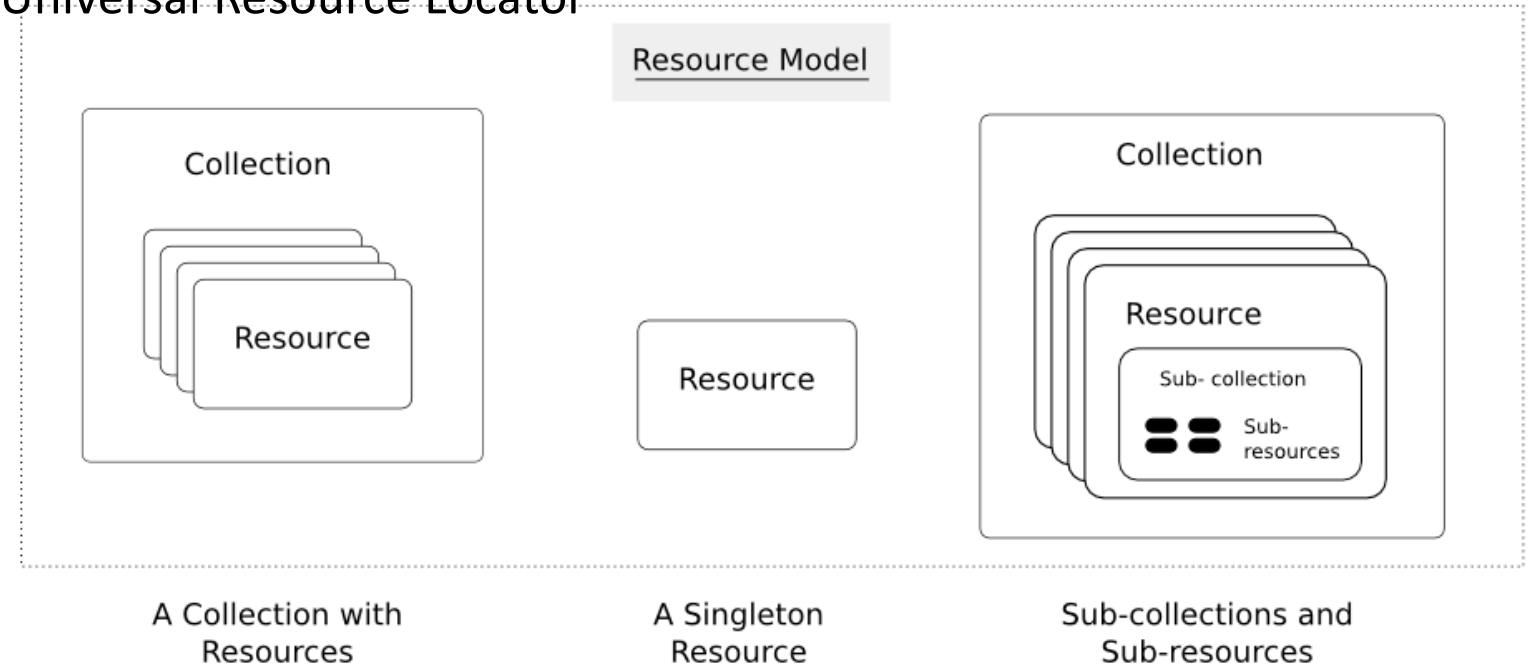
Response

```
HTTP/1.1 200 OK

{
    "account": {
        "account_number": 12345,
        "balance": {
            "currency": "usd",
            "value": 100.00
        },
        "links": {
            "deposits": "/accounts/12345/deposits",
            "withdrawals": "/accounts/12345/withdrawals",
            "transfers": "/accounts/12345/transfers",
            "close-requests": "/accounts/12345/close-requests"
        }
    }
}
```

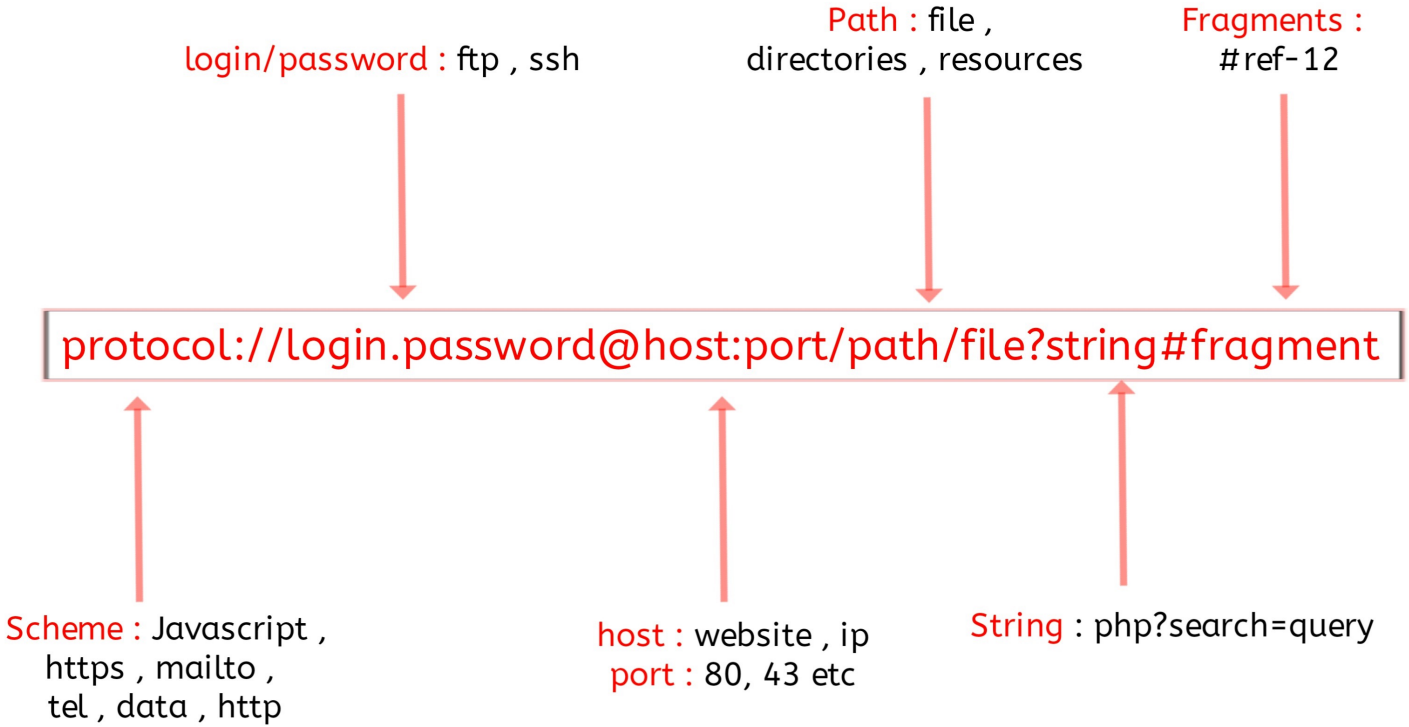
REST and Resources

- URL = Universal Resource Locator



- (URL, content-type) application/JSON, text/html, image/jpg

URLs

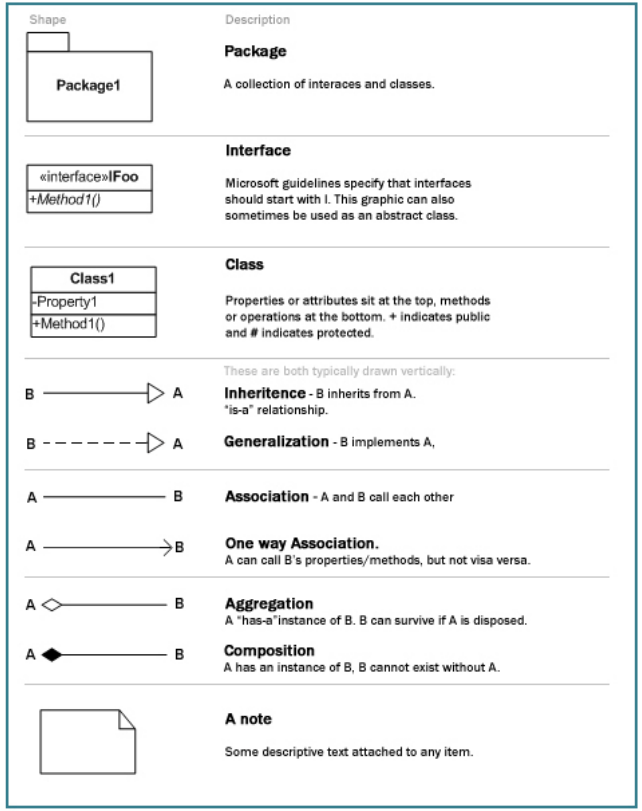


UML

Type of relationship	UML syntax source target	Brief semantics	Section
Dependency	---------->	The source element depends on the target element and may be affected by changes to it	9.5
Association	——	The description of a set of links between objects	9.4
Aggregation	◇——	The target element is a part of the source element	18.4
Composition	◆——	A strong (more constrained) form of aggregation	18.5
Containment	⊕——	The source element contains the target element	11.4
Generalization	——▷	The source element is a specialization of the more general target element and may be substituted for it	10.2
Realization	----------▷	The source element guarantees to carry out the contract specified by the target element	12.3

https://softwareengineering.stackexchange.com/questions/405247/what-is-the-difference-between-containment-and-aggregation-relationship-in-uml

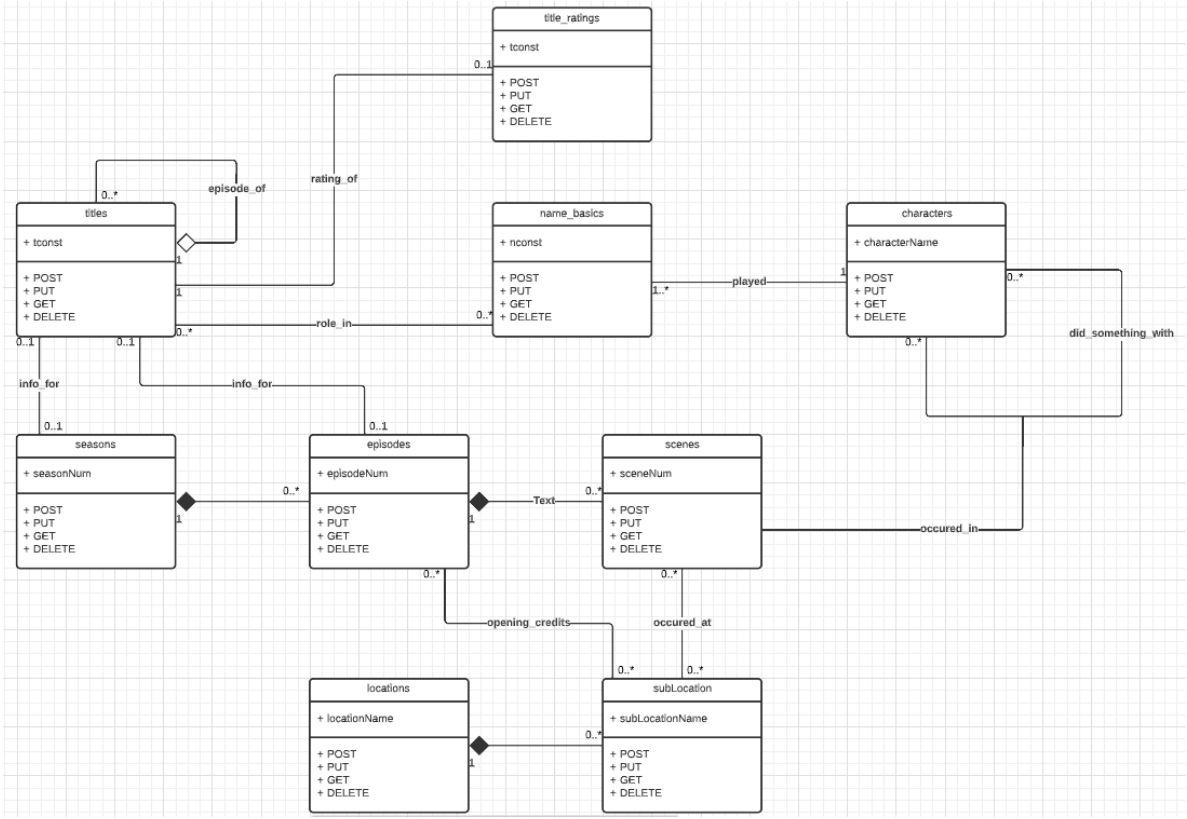
UML Cheatsheet



https://khalilstemmler.com/files/resources/umlcheatsheet.jpg

First Pass at a Resource Model using UML

- Did not include all attributes.
- May not be accurate
  - Did before coffee (really bad idea).
  - Made early this morning.
  - This is fun for about 10 minutes and then is boring.
- Like all modeling
  - You can go crazy doing it.
  - Just enough is the ideal.
- Original here.



This Would Lead to Resource Paths Link ... ...

- /seasons
- /seasons/{seasonNum}
- /seasons/{seasonNum}/episodes
- /seasons/{seasonNum}/episodes/{episodeNum}/scenes
- /characters
- /characters/{characterName}/scenes
- /characters/{characterName}/{relationshipName}
- /characters/{characterName}/actors
- /characters/{characterName}/actors/{nconst}/knownFor
- ... ...

Partial, initial mapping

Walk Through Some Code

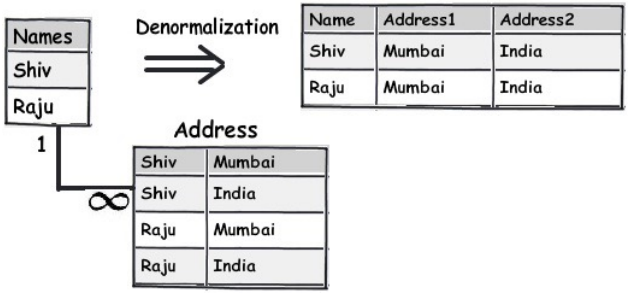
Decision Support

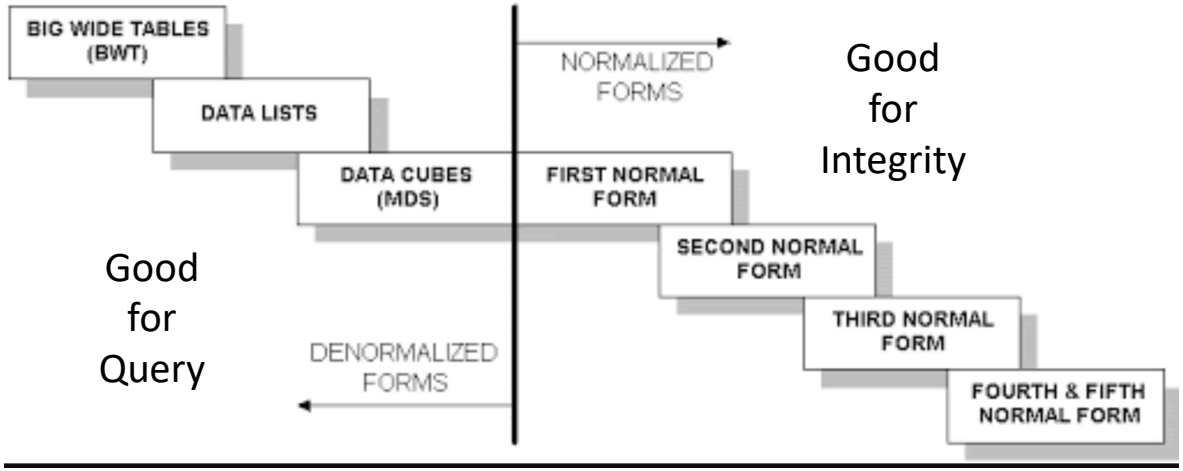
Concepts

Schema Design Tradeoffs

Data Warehouse, Data Lake, ETL

Wide Flat Tables





- Improve query performance by precomputing and saving:
  - JOINs
  - Aggregation
  - Derived/computed columns
- One of the primary strength of the relational model is maintaining "integrity" when applications create, update and delete data. This relies on:
  - The core capabilities of the relational model, e.g. constraints.
  - A well-design database (We will cover a formal definition – "normalization" in more detail later.
- Data models that are well designed for integrity are very bad for read only analysis queries.
  We will build and analyze wide flat tables as part of the analysis tasks in HW3, HW4 as projects.

Decision Support Systems

- **Decision-support systems** are used to make business decisions, often based on data collected by on-line transaction-processing systems.
- Examples of business decisions:
  - What items to stock?
  - What insurance premium to change?
  - To whom to send advertisements?
- Examples of data used for making decisions
  - Retail sales transaction details
  - Customer profiles (income, age, gender, etc.)

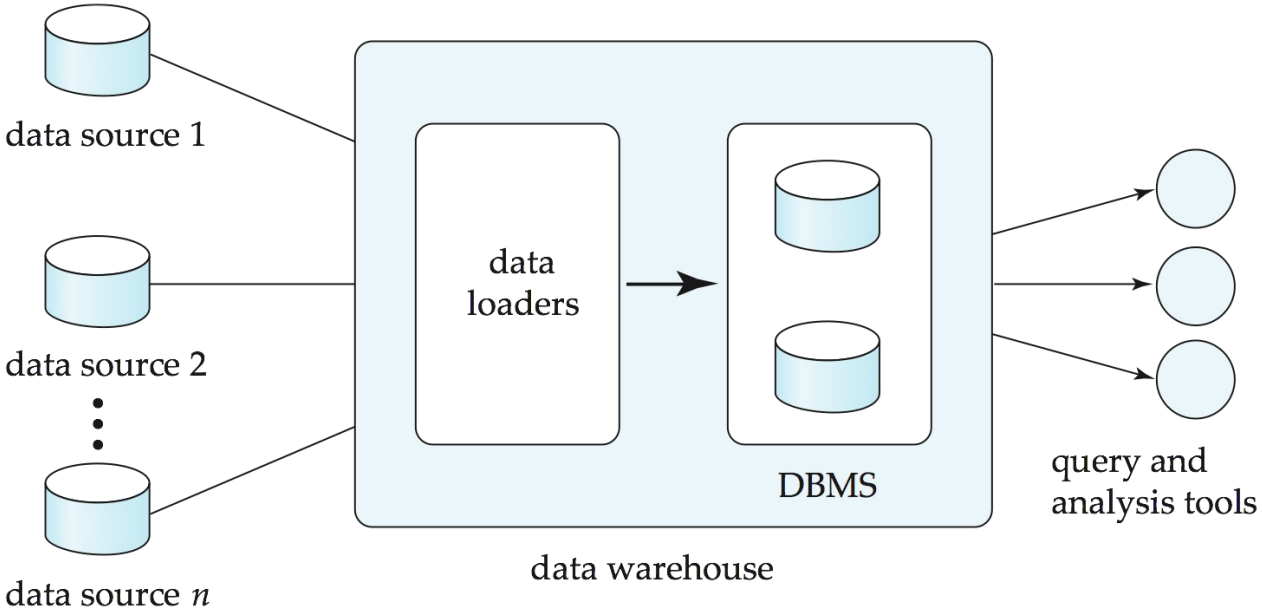
Decision-Support Systems: Overview

- **Data analysis** tasks are simplified by specialized tools and SQL extensions
  - Example tasks
    - For each product category and each region, what were the total sales in the last quarter and how do they compare with the same quarter last year
    - As above, for each product category and each customer category
- **Statistical analysis** packages (e.g., : S++) can be interfaced with databases
  - Statistical analysis is a large field, but not covered here
- **Data mining** seeks to discover knowledge automatically in the form of statistical rules and patterns from large databases.
- A **data warehouse** archives information gathered from multiple sources, and stores it under a unified schema, at a single site.
  - Important for large businesses that generate data from multiple divisions, possibly at multiple sites
  - Data may also be purchased externally

Data Warehousing

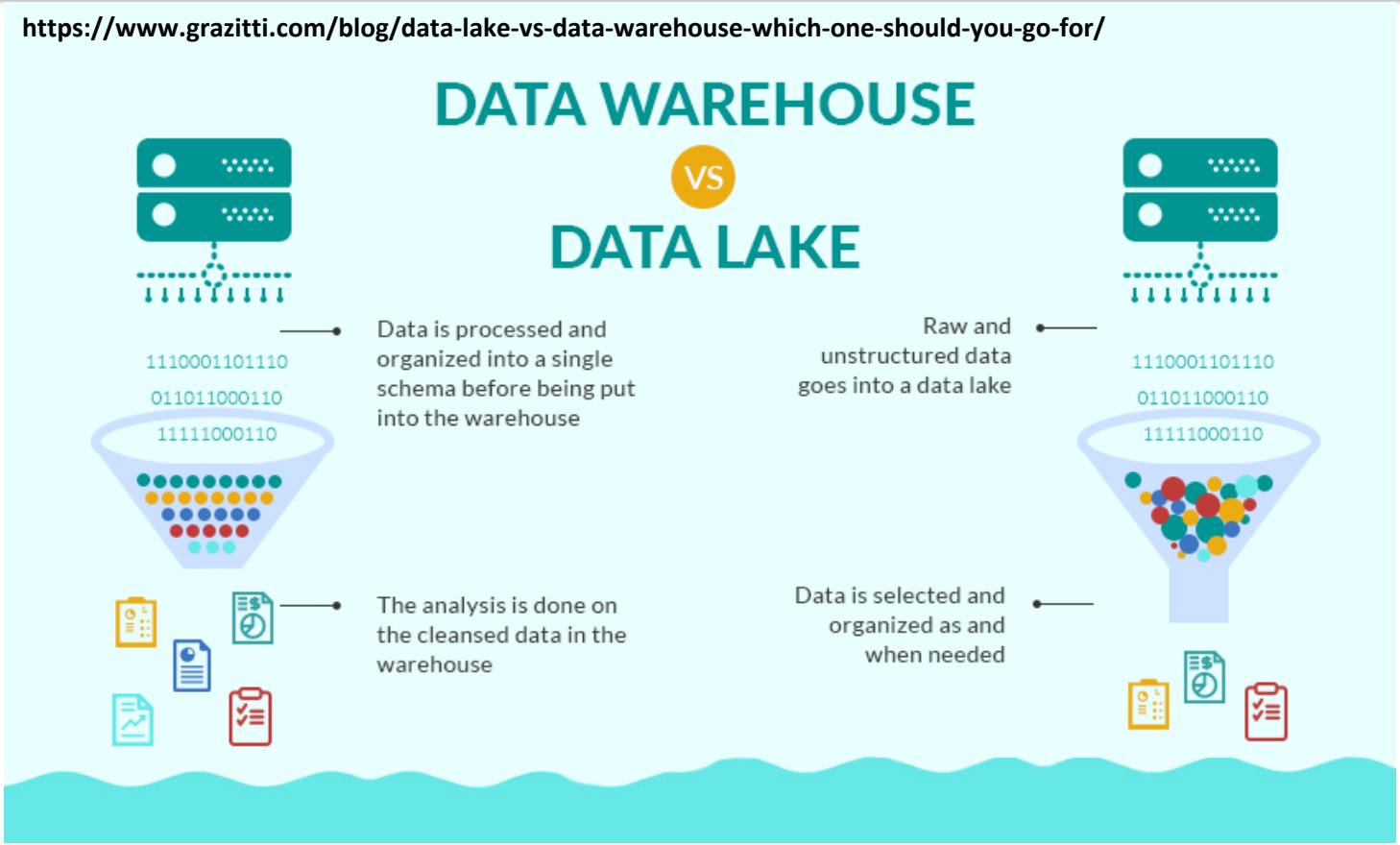
- Data sources often store only current data, not historical data
- Corporate decision making requires a unified view of all organizational data, including historical data
- A **data warehouse** is a repository (archive) of information gathered from multiple sources, stored under a unified schema, at a single site
  - Greatly simplifies querying, permits study of historical trends
  - Shifts decision support query load away from transaction processing systems

Data Warehousing



Data Warehouse vs Data Lake

https://www.grazitti.com/blog/data-lake-vs-data-warehouse-which-one-should-you-go-for/



Simplistic: Data Warehouse vs Data Lake

https://panoply.io/data-warehouse-guide/data-warehouse-vs-data-lake/

DATA WAREHOUSE	vs.	DATA LAKE
structured, processed	DATA	structured / semi-structured / unstructured, raw
schema-on-write	PROCESSING	schema-on-read
expensive for large data volumes	STORAGE	designed for low-cost storage
less agile, fixed configuration	AGILITY	highly agile, configure and reconfigure as needed
mature	SECURITY	maturing
business professionals	USERS	data scientists et. al.

Design Issues

- *When and how to gather data*
  - **Source driven architecture**: data sources transmit new information to warehouse, either continuously or periodically (e.g., at night)
  - **Destination driven architecture**: warehouse periodically requests new information from data sources
  - Keeping warehouse exactly synchronized with data sources (e.g., using two-phase commit) is too expensive
    - Usually OK to have slightly out-of-date data at warehouse
    - Data/updates are periodically downloaded form online transaction processing (OLTP) systems.
- *What schema to use*
  - Schema integration

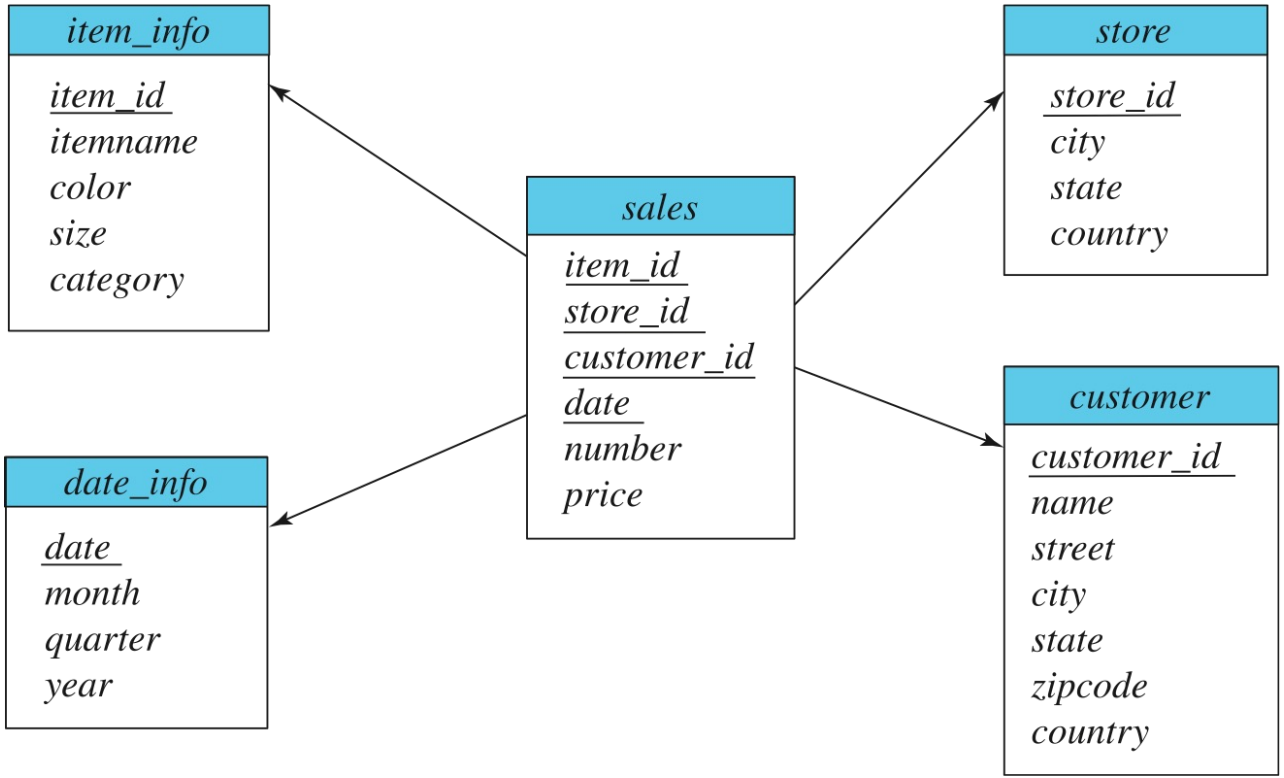
More Warehouse Design Issues

- *Data cleansing*
  - E.g., correct mistakes in addresses (misspellings, zip code errors)
  - **Merge** address lists from different sources and **purge** duplicates
- *How to propagate updates*
  - Warehouse schema may be a (materialized) view of schema from data sources
- *What data to summarize*
  - Raw data may be too large to store on-line
  - Aggregate values (totals/subtotals) often suffice
  - Queries on raw data can often be transformed by query optimizer to use aggregate values

Warehouse Schemas

- Dimension values are usually encoded using small integers and mapped to full values via dimension tables
- Resultant schema is called a **star schema**
  - More complicated schema structures
    - **Snowflake schema**: multiple levels of dimension tables
    - **Constellation**: multiple fact tables

Data Warehouse Schema

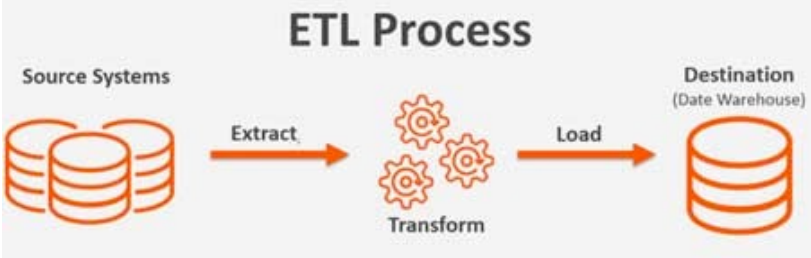


Overview (Cont.)

- Common steps in data analytics
  - Gather data from multiple sources into one location
    - Data warehouses also integrated data into common schema
    - Data often needs to be **extracted** from source formats, **transformed** to common schema, and **loaded** into the data warehouse
      - Can be done as **ETL (extract-transform-load)**, or **ELT (extract-load-transform)**
  - Generate aggregates and reports summarizing data
    - Dashboards showing graphical charts/reports
    - **Online analytical processing (OLAP) systems** allow interactive querying
    - Statistical analysis using tools such as R/SAS/SPSS
      - Including extensions for parallel processing of big data
  - Build **predictive models** and use the models for decision making

ETL Concepts

https://databricks.com/glossary/extract-transform-load



Extract

The first step of this process is extracting data from the target sources that could include an ERP, CRM, Streaming sources, and other enterprise systems as well as data from third-party sources. There are different ways to perform the extraction: **Three Data Extraction methods:**

1. Partial Extraction – The easiest way to obtain the data is if the if the source system notifies you when a record has been changed
2. Partial Extraction- with update notification - Not all systems can provide a notification in case an update has taken place; however, they can point those records that have been changed and provide an extract of such records.
3. Full extract – There are certain systems that cannot identify which data has been changed at all. In this case, a full extract is the only possibility to extract the data out of the system. This method requires having a copy of the last extract in the same format so you can identify the changes that have been made.

Transform

Next, the transform function converts the raw data that has been extracted from the source server. As it cannot be used in its original form in this stage it gets cleansed, mapped and transformed, often to a specific data schema, so it will meet operational needs. This process entails several transformation types that ensure the quality and integrity of data; below are the most common as well as advanced transformation types that prepare data for analysis:

- Basic transformations:
- Cleaning
- Format revision
- Data threshold validation checks
- Restructuring
- Deduplication
- Advanced transformations:
- Filtering
- Merging
- Splitting
- Derivation
- Summarization
- Integration
- Aggregation
- Complex data validation

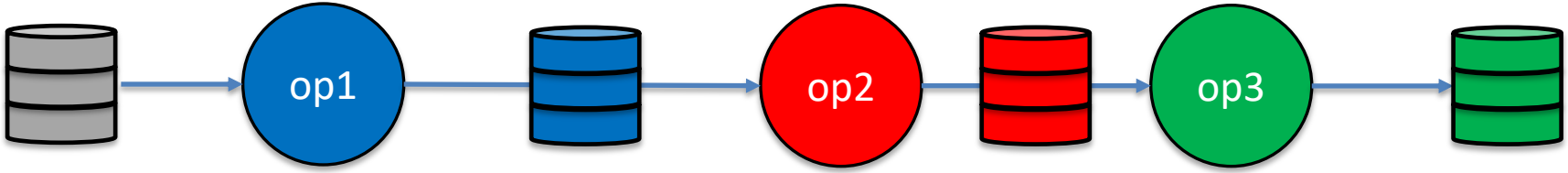
Load

Finally, the load function s is the process of writing converted data from a staging area to a target database, which may or may not have previously existed. Depending on the requirements of the application, this process may be either quite simple or intricate.

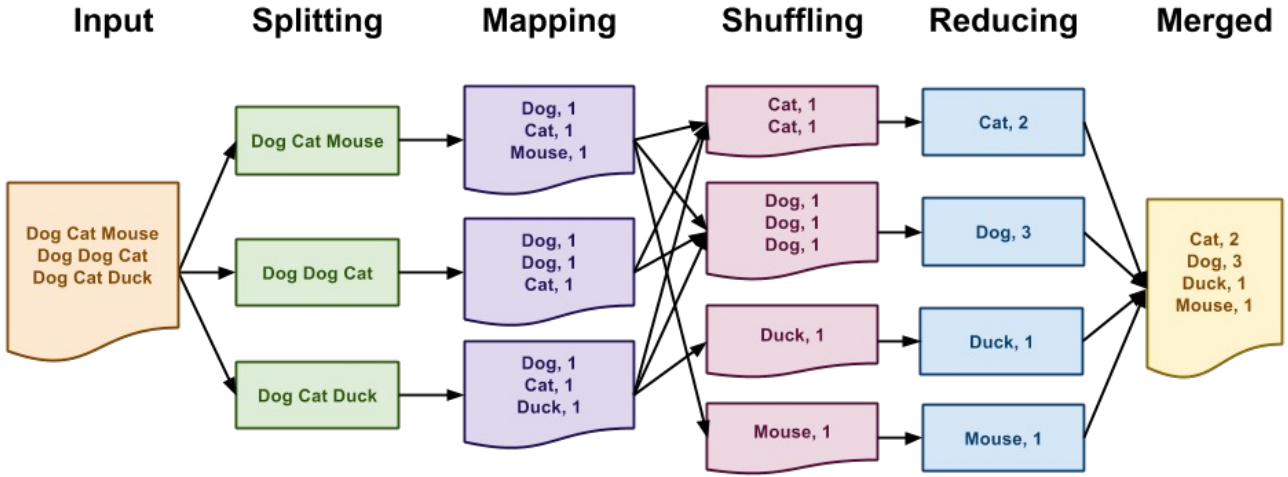
MapReduce, Spark (and ETL)

MapReduce

MapReduce is a data flow program with relatively simple operators on the data set.



With each operator implemented in parallel on multiple nodes for performance.



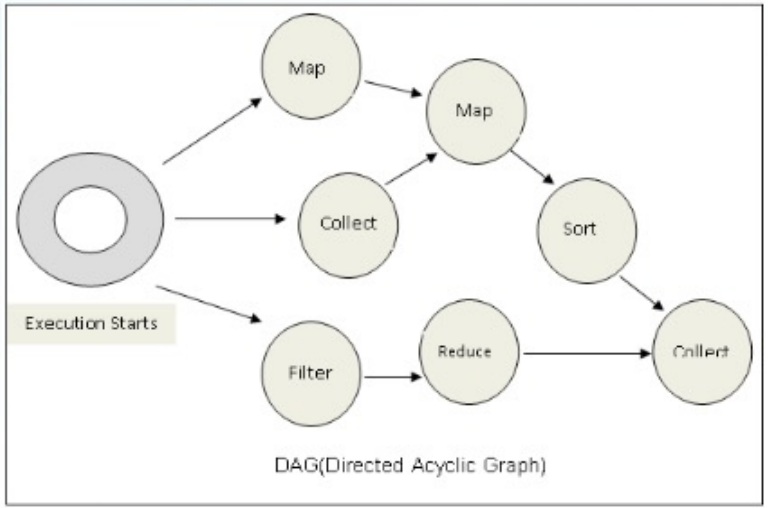
What if we want more complex "operators?"

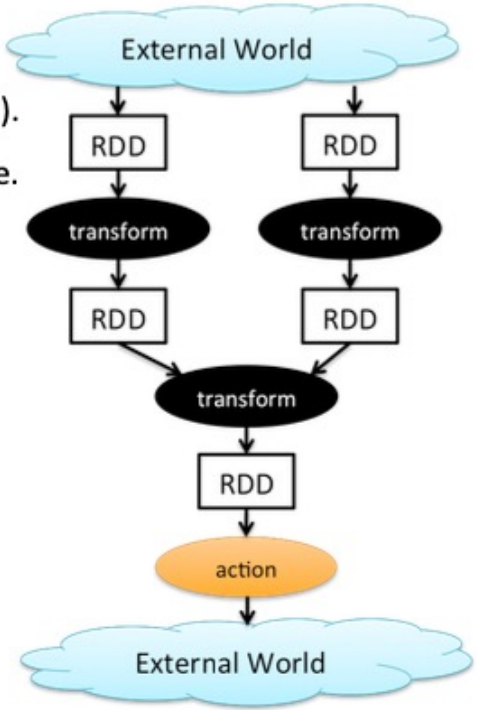
Algebraic Operations

- Current generation execution engines
  - natively support algebraic operations such as joins, aggregation, etc. natively.
  - Allow users to create their own algebraic operators
  - Support trees of algebraic operators that can be executed on multiple nodes in parallel
- E.g. Apache Tez, Spark
  - Tex provides low level API; Hive on Tez compiles SQL to Tez
  - Spark provides more user-friendly API

- In the relational model,
  - The are relations.
  - A fixed set of operators that produce relations from relations (closed).
  - Are declarative languages and compute the execution graph/plan.
- The Spark/... ... engines:
  - Enable programmers to develop and add new operators.
  - Operators convert reliable distributed datasets/data frames to new RDDs/data frames.
  - Data engineer explicitly defines the execution graph (data flow).

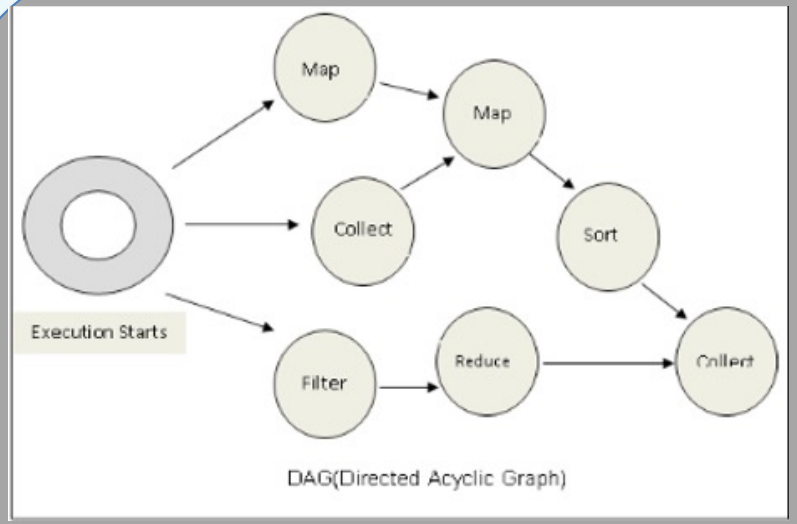
- All jobs in spark comprise a series of operators and run on a set of data.
- All the operators in a job are used to construct a DAG (Directed Acyclic Graph).
- The DAG is optimized by rearranging and combining operators where possible.





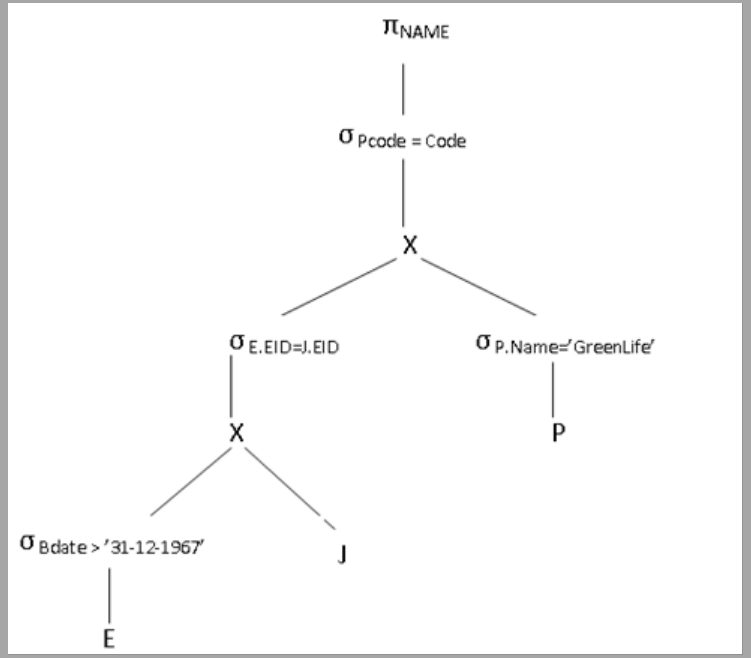
www.edureka.co/apache-spark-scala-training

Spark vs SQL



Conceptually similar to query evaluation graph, but ...

- You explicitly define the graph.
- You can develop your own operators.
- You can define levels of parallelism that the infrastructure manages.



SQL is declarative, and the query engine transparently produces the execution plan.

www.edureka.co/apache-spark-scala-training

Algebraic Operations in Spark

- **Resilient Distributed Dataset** (**RDD**) abstraction
  - Collection of records that can be stored across multiple machines
- RDDs can be created by applying algebraic operations on other RDDs
- RDDs can be lazily computed when needed
- ~~Spark programs can be written in Java/Scala/R~~
  - ~~Our examples are in Java~~
- ~~Spark makes use of Java 8 Lambda expressions; the code~~

  ~~s - > Arrays.asList(s.split(" ")).iterator()~~

  ~~defines unnamed function that takes argument s and executes the expression Arrays.asList(s.split(" ")).iterator() on the argument~~
- ~~Lambda functions are particularly convenient as arguments to map, reduce and other functions~~

Spark/PySpark

DataFrame

edureka!

- Inspired by DataFrames in R and Python (Pandas).
- DataFrames API is designed to make big data processing on tabular data easier.
- DataFrame is a distributed collection of data organized into named columns.
- Provides operations to filter, group, or compute aggregates, and can be used with Spark SQL.
- Can be constructed from structured data files, existing RDDs, tables in Hive, or external databases.

DataFrame features

edureka!

- Ability to scale from KBs to PBs
- Support for a wide array of data formats and storage systems
- State-of-the-art optimization and code generation through the spark SQL catalyst optimizer
- Seamless integration with all big data tooling and infrastructure via spark
- APIs for Python, Java, Scala, and R

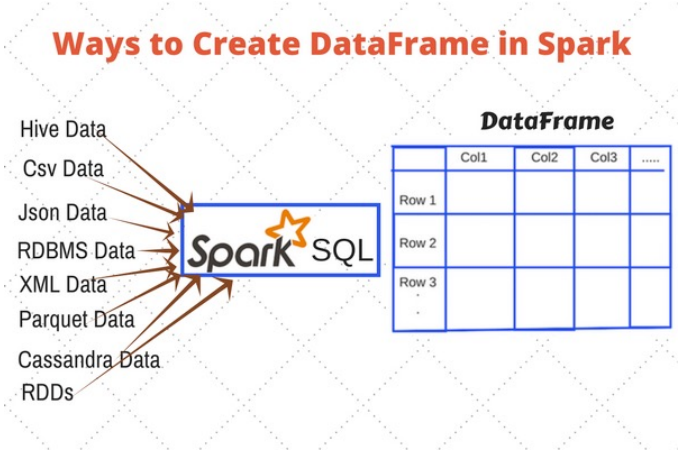
1. DataFrame in PySpark: Overview

In Apache Spark, a DataFrame is a distributed collection of rows under named columns. In simple terms, it is same as a table in relational database or an Excel sheet with Column headers. It also shares some common characteristics with RDD:

- **Immutable in nature** : We can create DataFrame / RDD once but can't change it. And we can transform a DataFrame / RDD after applying transformations.
- **Lazy Evaluations:** Which means that a task is not executed until an action is performed.
- **Distributed:** RDD and DataFrame both are distributed in nature.

My first exposure to DataFrames was when I learnt about Pandas. Today, it is difficult for me to run my data science workflow with out Pandas DataFrames. So, when I saw similar functionality in Apache Spark, I was excited about the possibilities it opens up!

https://www.analyticsvidhya.com/blog/2016/10/spark-dataframe-and-operations/



Conceptual Example

- Remember that really unpleasant set of queries and stored procedure we had to write for the midterm? You had to convert from … …

nconst	name	dateOfBirth	dateOfDeath	primaryProfession	knownFor
nm0000003	Brigitte Bardot	1934		actress,soundtrack,music_department	tt0057345,tt0059956,tt0054452,tt0049189
nm0000004	John Belushi	1949	1982	actor,soundtrack,writer	tt0077975,tt0072562,tt0080455,tt0078723
nm0000005	Ingmar Bergman	1918	2007	writer,director,actor	tt0050986,tt0083922,tt0060827,tt0050976
nm0000006	Ingrid Bergman	1915	1982	actress,soundtrack,producer	tt0038787,tt0036855,tt0034583,tt0038109
nm0000007	Humphrey Bogart	1899	1957	actor,soundtrack,producer	tt0043265,tt0040897,tt0034583,tt0037382
nm0000008	Marlon Brando	1924	2004	actor,soundtrack,director	tt0078788,tt0070849,tt0047296,tt0068646
nm0000009	Richard Burton	1925	1984	actor,soundtrack,producer	tt0087803,tt0061184,tt0059749,tt0057877
nm0000010	James Cagney	1899	1986	actor,soundtrack,director	tt0042041,tt0035575,tt0029870,tt0031867

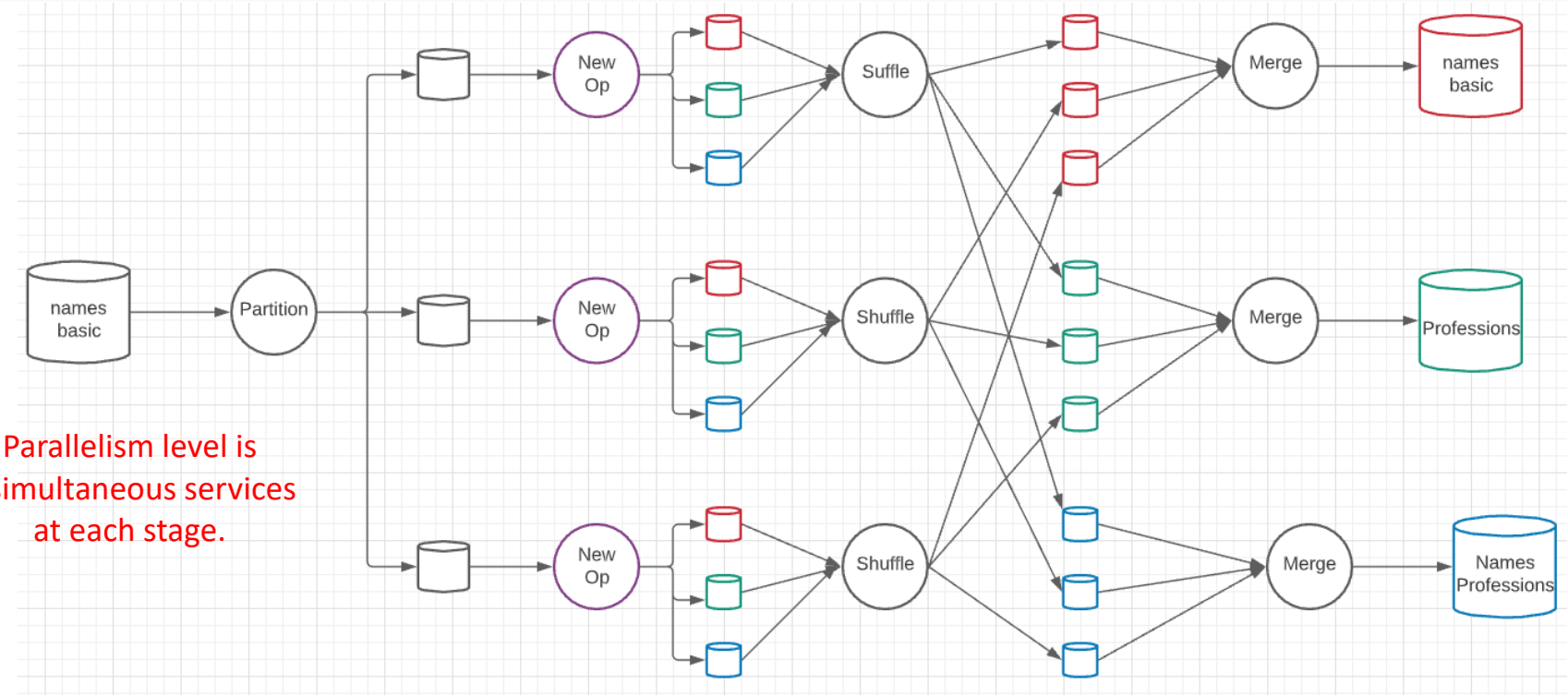
- To

nm0000001	Fred Astaire	1899	1987
nm0000002	Lauren Bacall	1924	2014
nm0000003	Brigitte Bardot	1934	
nm0000004	John Belushi	1949	1982
nm0000005	Ingmar Bergman	1918	2007
nm0000006	Ingrid Bergman	1915	1982
nm0000007	Humphrey Bogart	1899	1957
nm0000008	Marlon Brando	1924	2004
nm0000009	Richard Burton	1925	1984
nm0000010	James Cagney	1899	1986

nconst	profession_id
nm0000001	1
nm0000001	2
nm0000001	3
nm0000002	3
nm0000002	4
nm0000003	3
nm0000003	4
nm0000003	5
nm0000004	1
nm0000004	3

Profession	Profession_id
actor	1
miscellaneous	2
soundtrack	3
actress	4
music_department	5
writer	6
director	7
producer	8
make_up_department	9
composer	10
assistant_director	11

A New Algebraic Operator



Parallelism level is 3 simultaneous services at each stage.

- Input is a dataset and column ID
- Output is three datasets: dataset with column removed, table of distinct column values, associative entity.

OLAP

Data Analysis and OLAP

- **Online Analytical Processing (OLAP)**
  - Interactive analysis of data, allowing data to be summarized and viewed in different ways in an online fashion (with negligible delay)
- We use the following relation to illustrate OLAP concepts
  - *sales* (*item_name*, *color*, *clothes_size*, *quantity*)

  This is a simplified version of the *sales* fact table joined with the dimension tables, and many attributes removed (and some renamed)

Example sales relation

item_name	*color*	*clothes_size*	*quantity*
dress	dark	small	2
dress	dark	medium	6
dress	dark	large	12
dress	pastel	small	4
dress	pastel	medium	3
dress	pastel	large	3
dress	white	small	2
dress	white	medium	3
dress	white	large	0
pants	dark	small	14
pants	dark	medium	6
pants	dark	large	0
pants	pastel	small	1
pants	pastel	medium	0
pants	pastel	large	1
pants	white	small	3
pants	white	medium	0
pants	white	large	2
shirt	dark	small	2
shirt	dark	medium	6
shirt	dark	large	6
shirt	pastel	small	4
shirt	pastel	medium	1
shirt	pastel	large	2
shirt	white	small	17
shirt	white	medium	1
shirt	white	large	10
skirt	dark	small	2
skirt	dark	medium	5
...	...	...	...
...	...	...	...

Cross Tabulation of *sales* by *item_name* and *color*

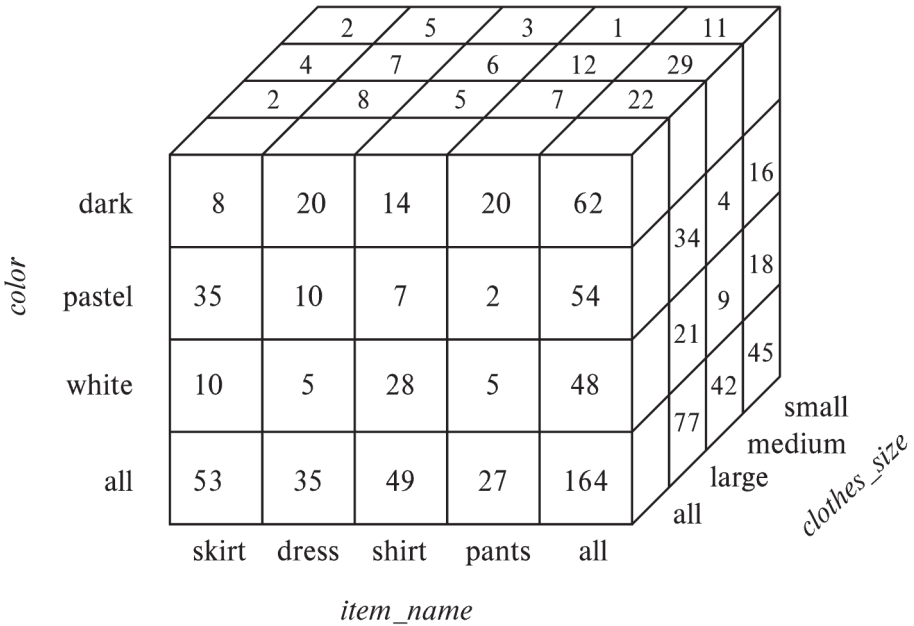
clothes_size **all**

item_name	*color* dark	pastel	white	total
skirt	8	35	10	53
dress	20	10	5	35
shirt	14	7	28	49
pants	20	2	5	27
total	62	54	48	164

- The table above is an example of a **cross-tabulation** (**cross-tab**), also referred to as a **pivot-table**.
  - Values for one of the dimension attributes form the row headers
  - Values for another dimension attribute form the column headers
  - Other dimension attributes are listed on top
  - Values in individual cells are (aggregates of) the values of the dimension attributes that specify the cell.

Data Cube

- A **data cube** is a multidimensional generalization of a cross-tab
- Can have n dimensions; we show 3 below
- Cross-tabs can be used as views on a data cube



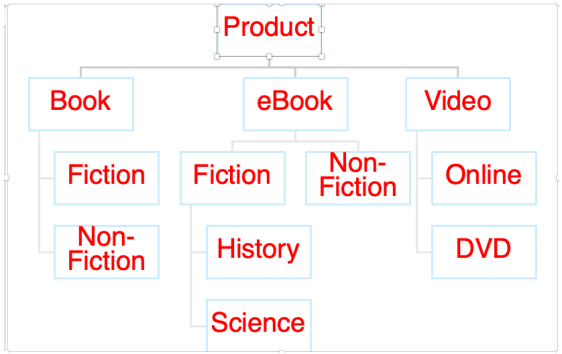
Online Analytical Processing Operations

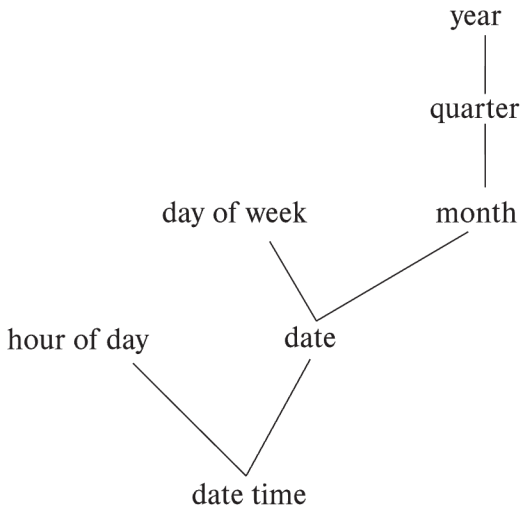
- **Pivoting:** changing the dimensions used in a cross-tab
  - E.g., moving colors to column names
- **Slicing:** creating a cross-tab for fixed values only
  - E.g., fixing color to white and size to small
  - Sometimes called **dicing**, particularly when values for multiple dimensions are fixed.
- **Rollup:** moving from finer-granularity data to a coarser granularity
  - E.g., aggregating away an attribute
  - E.g., moving from aggregates by day to aggregates by month or year
- **Drill down:** The opposite operation - that of moving from coarser-granularity data to finer-granularity data

Hierarchies on Dimensions

- **Hierarchy** on dimension attributes: lets dimensions be viewed at different levels of detail
- E.g., the dimension *datetime* can be used to aggregate by hour of day, date, day of week, month, quarter or year

Another dimension could be ...
Product category.

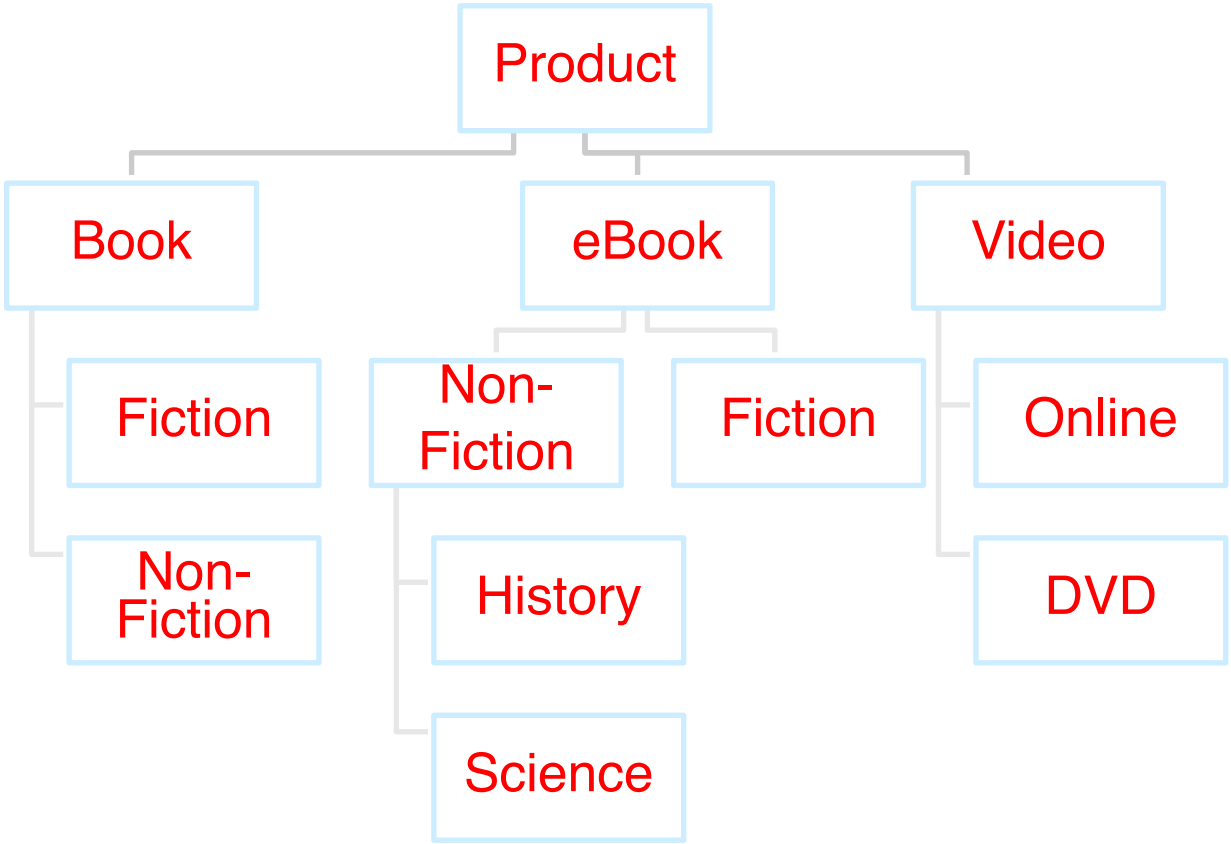




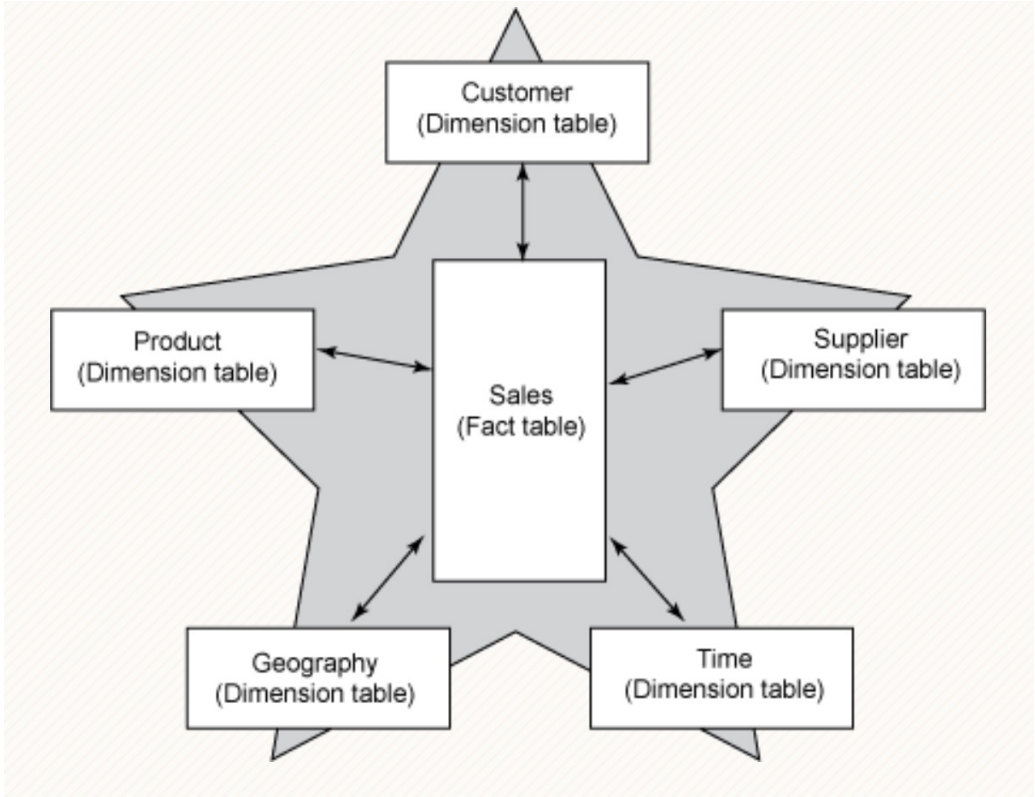
(a) time hierarchy

(b) location hierarchy

Another Dimension Example – Product Categories



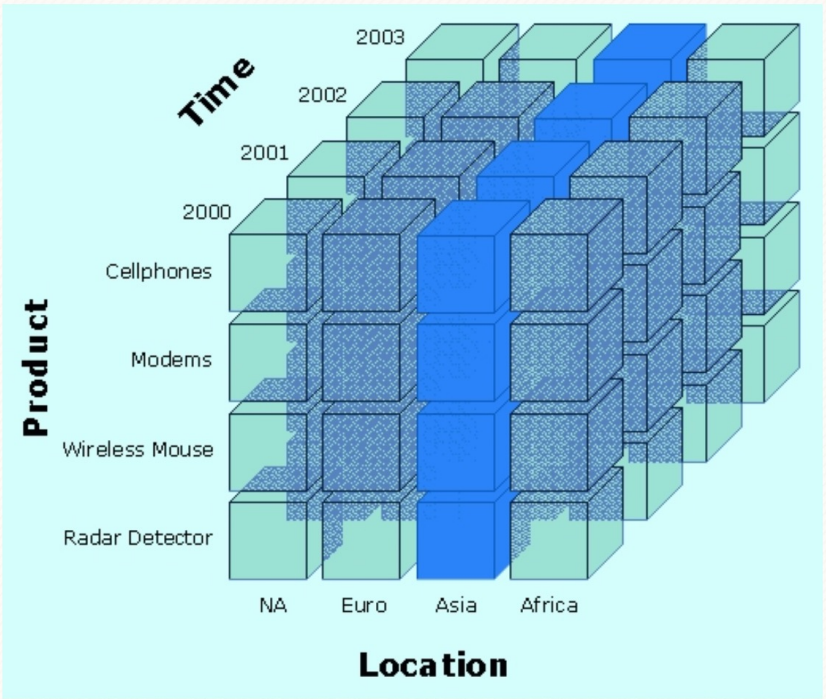
Facts and Dimensions



Slice

Slice:

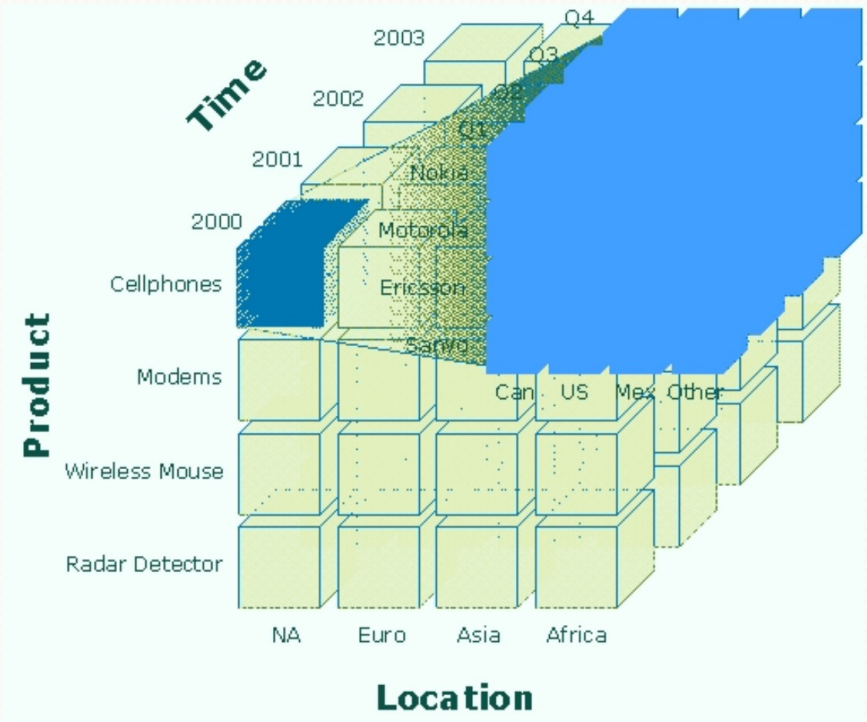
A slice is a subset of a multi-dimensional array corresponding to a single value for one or more members of the dimensions not in the subset.



Dice

Dice:

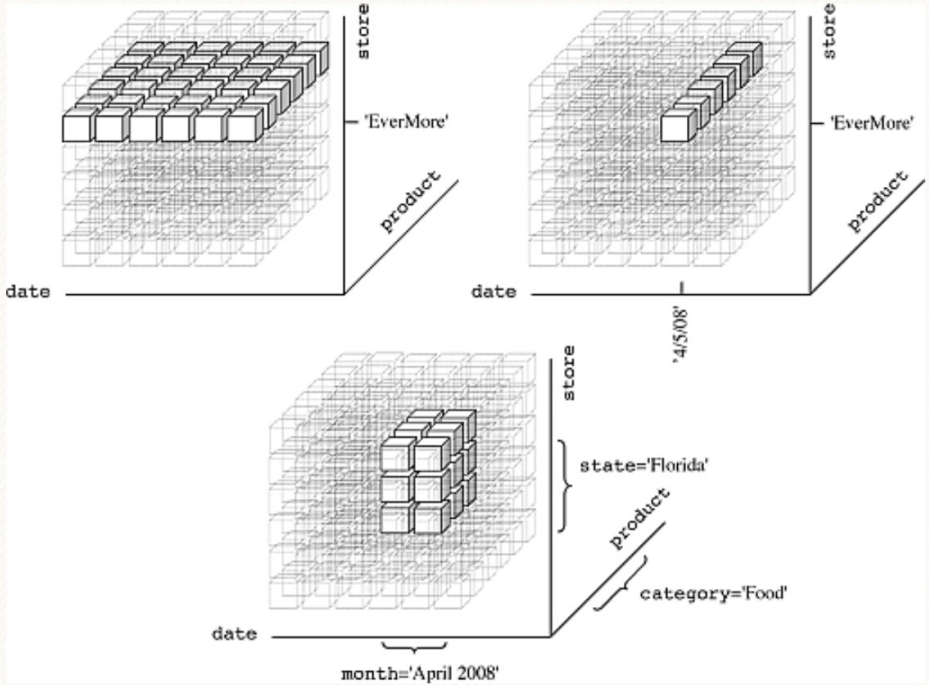
The dice operation is a slice on more than two dimensions of a data cube (or more than two consecutive slices).



Drilling

Drill Down/Up:

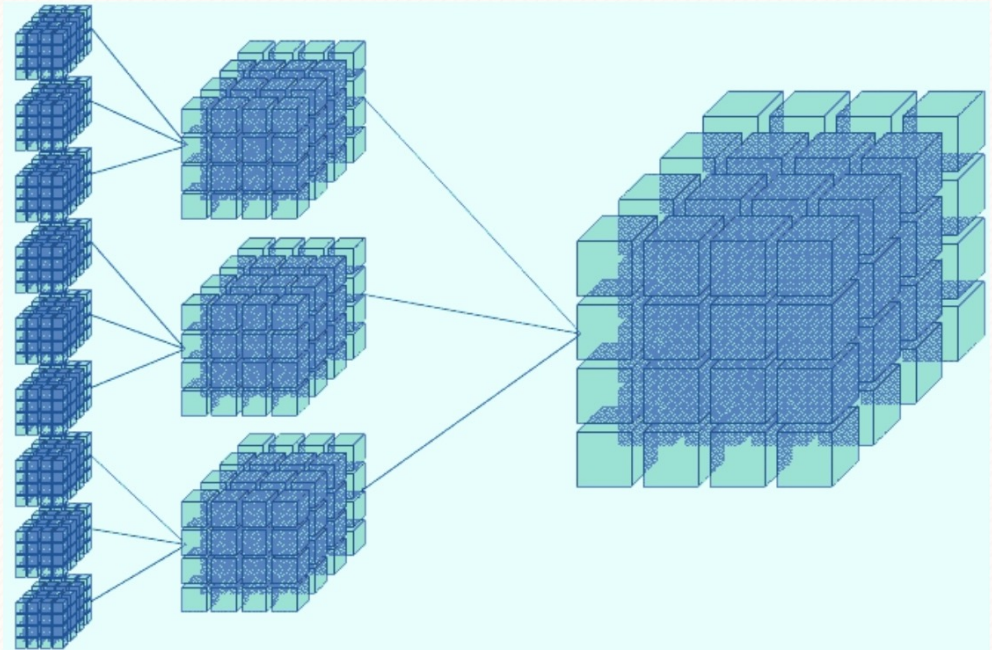
Drilling down or up is a specific analytical technique whereby the user navigates among levels of data ranging from the most summarized (up) to the most detailed (down).



Roll-up

Roll-up:

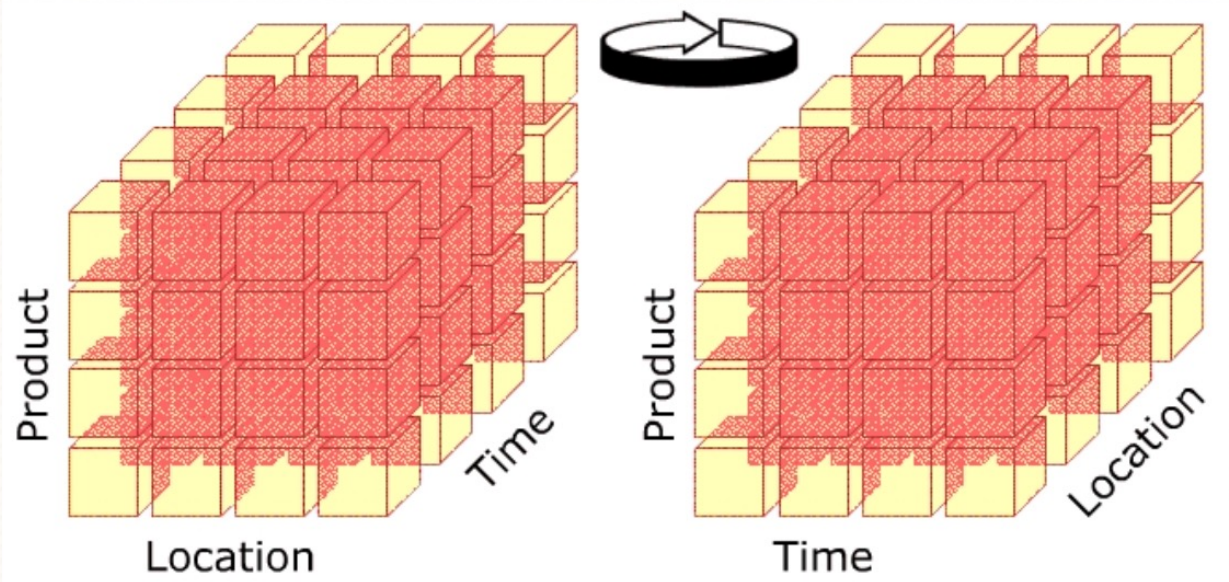
(Aggregate, Consolidate) A roll-up involves computing all of the data relationships for one or more dimensions. To do this, a computational relationship or formula might be defined.



Pivot

Pivot:

This operation is also called rotate operation. It rotates the data in order to provide an alternative presentation of data – the report or page display takes a different dimensional orientation.



Cross Tabulation With Hierarchy

- Cross-tabs can be easily extended to deal with hierarchies
- Can drill down or roll up on a hierarchy
- E.g. hierarchy: *item_name* → *category*

clothes_size: **all**

category	*item_name*	*color*				
		dark	pastel	white	total	
womenswear	skirt	8	8	10	53	
	dress	20	20	5	35	
	subtotal	28	28	15		88
menswear	pants	14	14	28	49	
	shirt	20	20	5	27	
	subtotal	34	34	33		76
total		62	62	48		164

Relational Representation of Cross-tabs

- Cross-tabs can be represented as relations
- We use the value **all** to represent aggregates.
- The SQL standard actually uses *null* values in place of **all**
  - Works with any data type
  - But can cause confusion with regular null values.

item_name	*color*	*clothes_size*	*quantity*
skirt	dark	**all**	8
skirt	pastel	**all**	35
skirt	white	**all**	10
skirt	**all**	**all**	53
dress	dark	**all**	20
dress	pastel	**all**	10
dress	white	**all**	5
dress	**all**	**all**	35
shirt	dark	**all**	14
shirt	pastel	**all**	7
shirt	white	**all**	28
shirt	**all**	**all**	49
pants	dark	**all**	20
pants	pastel	**all**	2
pants	white	**all**	5
pants	**all**	**all**	27
all	dark	**all**	62
all	pastel	**all**	54
all	white	**all**	48
all	**all**	**all**	164

Walk Through Some Code

- Wide Flat Tables
- Complex Mongo Query
- … …

Summary
Next Steps