COMM1822

Term 2 2022

Introduction to Databases for Business Analytics

Week 9 Big Data 2

Lecturer-in-Charge: Kam-Fung (Henry) Cheung
Email: kf.cheung@unsw.edu.au
Tutors: Theresa Tran
Liam Li Chen
Kathy Xu
PASS Leader: Srilekha Chandrashekara Kolaki

Copyright

- There are some file-sharing websites that specialise in buying and selling academic work to and from university students.

- If you upload your original work to these websites, and if another student downloads and presents it as their own either wholly or partially, **you might be found guilty of collusion — even years after graduation**.

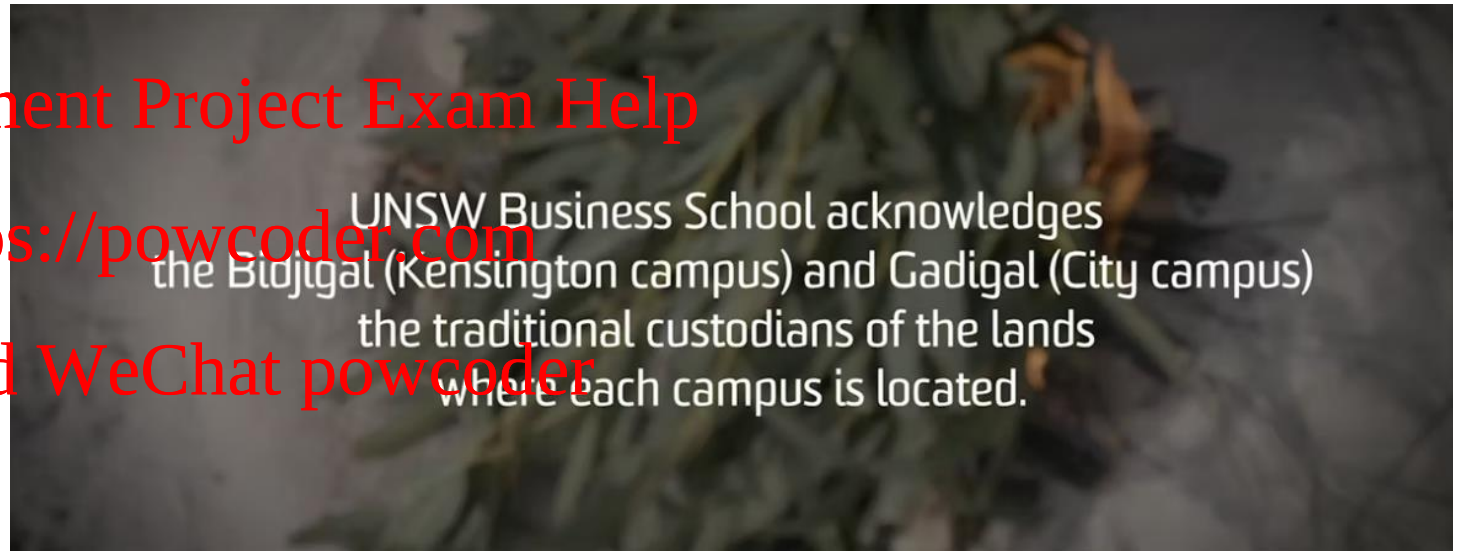
- These file-sharing websites may also accept purchase of course materials, **such as copies of <u>lecture slides</u> and <u>tutorial handouts</u>. By law, the copyright on course materials, developed by UNSW staff in the course of their employment, belongs to UNSW. It constitutes copyright infringement, if not academic misconduct, to trade these materials.**

Acknowledgement of Country

UNSW Business School acknowledges the Bidjigal (Kensington campus) and Gadigal (City campus) the traditional custodians of the lands where each campus is located.

We acknowledge all Aboriginal and Torres Strait Islander Elders, past and present and their communities who have shared and practiced their teachings over thousands of years including business practices.

We recognise Aboriginal and Torres Strait Islander people's ongoing leadership and contributions, including to business, education and industry.



UNSW Business School. (2022, May 7). *Acknowledgement of Country* [online video]. Retrieved from https://vimeo.com/369229957/d995d8087f

At UNSW you are free to...

- Respectfully disagree about anything
- Express different opinions
- Write your beliefs
- Show your beliefs
- Leave any club or organisation

UNSW SYDNEY

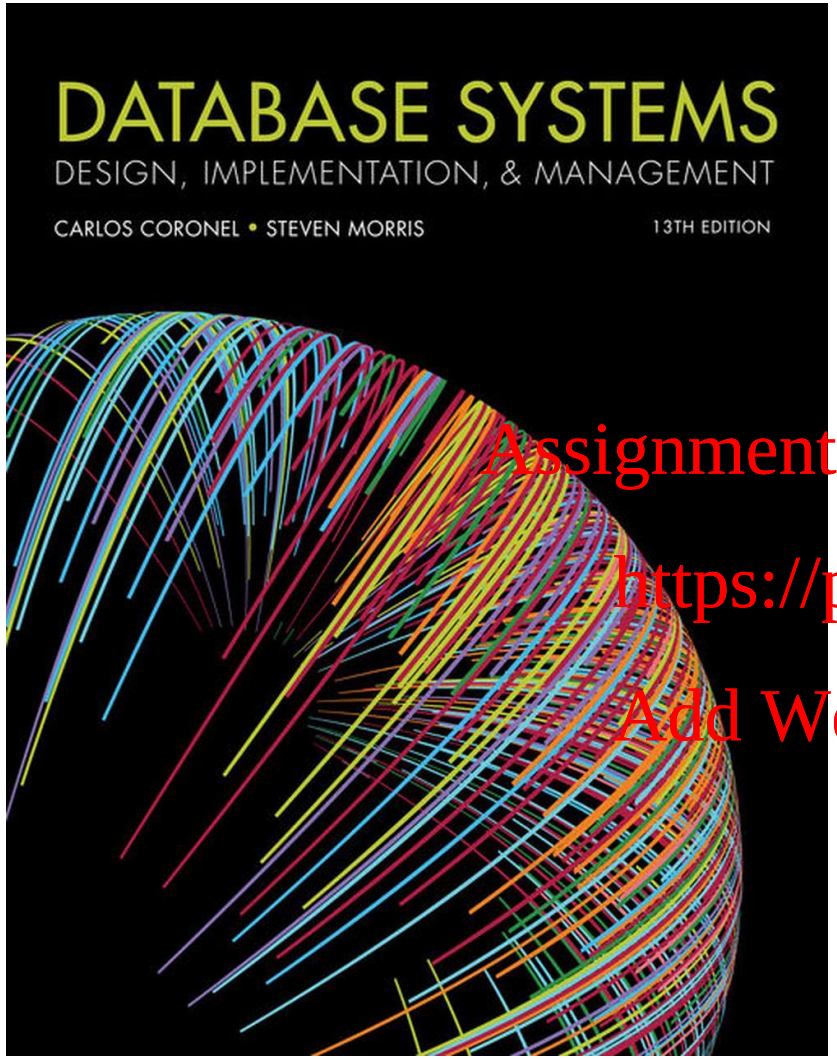
It's not acceptable to...

- Attempt to censor opinions
- Use hate speech
- Make threats or instil fear
- Make false accusations
- Access or share others private information without consent

We are here to help...

- Tell a teacher
- Tell UNSW Psychology and Wellness
- Report to UNSW Complaints
- Report to UNSW Security
- Report a crime to police

Find out more



Chapter 14

Big Data and NoSQL

W9 Learning Outcomes

- **Big Data Technologies**
  - Hadoop Ecosystem
    - Hadoop Distributed File System (HDFS)
    - MapReduce
    - Pig
    - Hive
    - HBase
    - Impala
- **NoSQL Database Types**
  - Key-value databases
  - Document databases
  - Column-oriented databases
  - Graph databases
- **Big Data Strategies**

Big Data Technologies

Big Data Infrastructure Challenges

- **Linear scalability**
  - To accommodate for the scalability of processing, thereby the storage management and architecture of traditional data management techniques become obsolete.

- **High throughput**
  - Infrastructure that is extremely fast across input/output (I/O), processing, and storage.

- **Fault tolerance**
  - Any portion of the processing architecture should be able to take over and resume processing from the point of failure in any other part of the system.

Big Data Infrastructure Challenges

- **Auto recovery**
  - The processing architecture should be self-managing and recover from failure without manual intervention.

- **High degree of parallelism**
  - Distribute the load across multiple machines, each having its own copy of the same data, but processing a different program. e.g., data analysis using different methods: linear regression, random forests

- **Distributed data processing**
  - The underlying platform must be able to process distributed data to achieve extreme scalability.

What is Hadoop?

- Hadoop is an open-source framework for storing and analyzing massive amounts of distributed, unstructured data.

- Hadoop was created by Doug Cutting and Mike Cafarella in 2005.

- Hadoop clusters run on inexpensive commodity hardware so projects can scale-out inexpensively.

- Open source - hundreds of contributors continuously improve the core technology.

- What is Hadoop? - https://www.youtube.com/watch?v=9s-vSeWej1U

Hadoop

- Not a single product, not a single database.
- A **collection of big data applications**.
- A **framework**, platform and ecosystem.
- Consisting of **different components** / modules.

- Most important components:
  - **Hadoop Distributed File System (HDFS)**
  - **MapReduce**
  - **Pig**
  - **Hive**
  - **HBase**
  - **Impala**

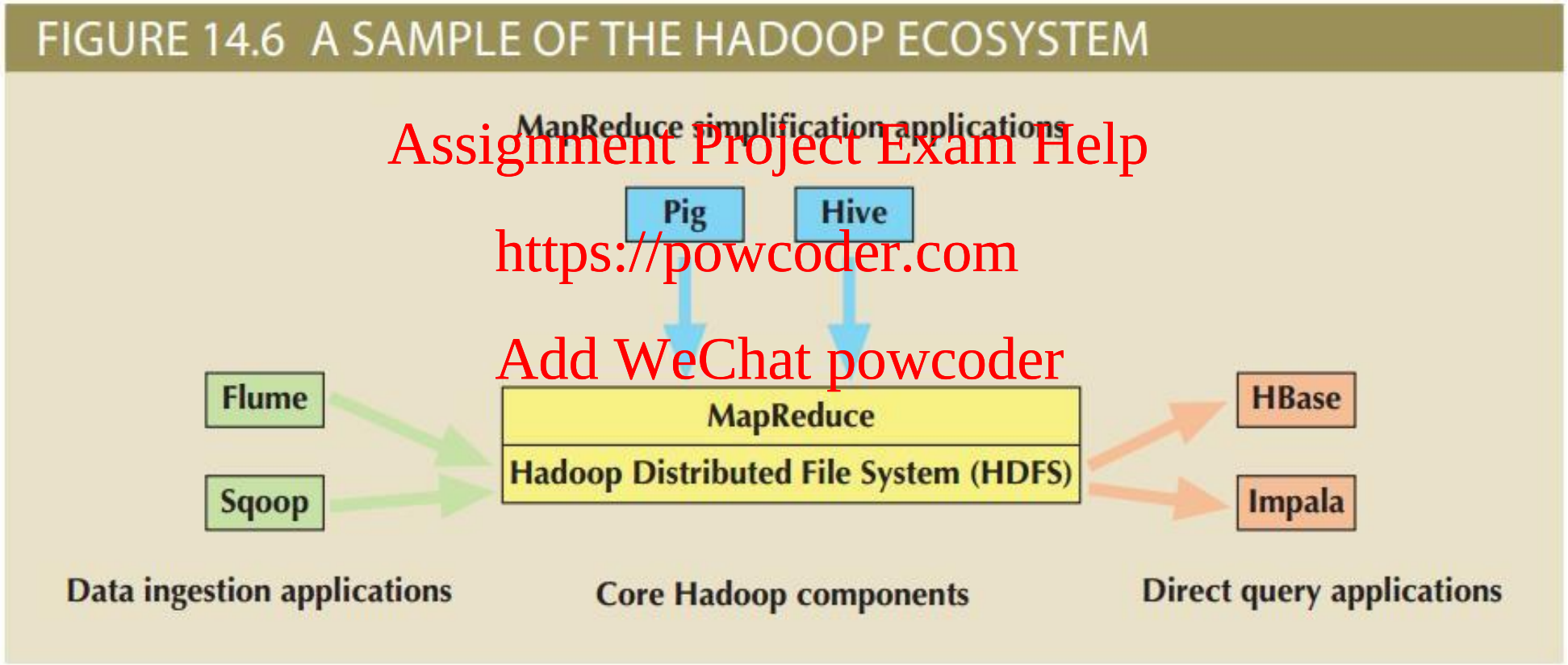
Why Hadoop?

- **Problems with relational database management system (RDBMS):**
  - Insufficiently scalable for big data
  - Insufficient speed for live data
  - Lack of sophisticated aggregation/analytics
  - Essentially a design based on the premise of a single CPU and RAM (you can easily "scale up" to an extent, but not easily "scale out")

- **Polyglot persistence**: The coexistence of a variety of data storage and data management technologies within an organization's infrastructure.

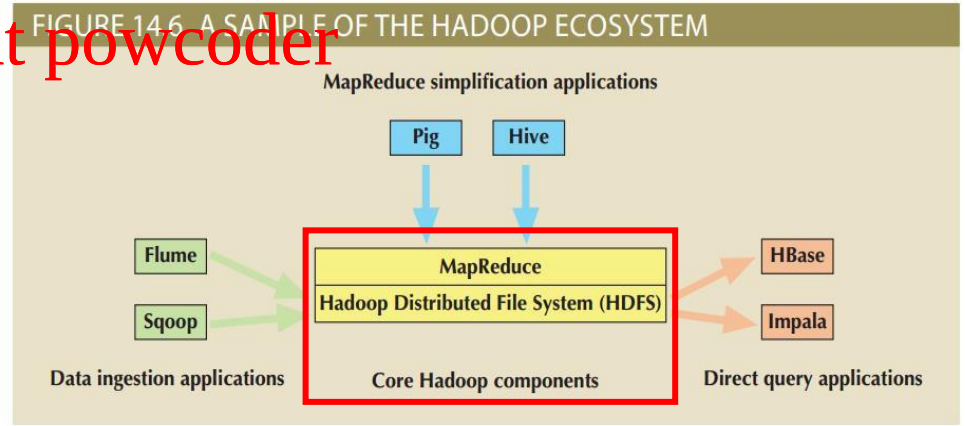
Structured: Customer's data, e.g., date of birth, address, bank account, …
Unstructured: Customer's feedback (in text), …

Hadoop Ecosystem



Hadoop Ecosystem – Core Components

- **Hadoop Distributed File System (HDFS)**
- **MapReduce**



Hadoop Distributed File System (HDFS)

- **Hadoop** stores **files across networks** using **Hadoop Distributed File System (HDFS)**

- **Hence, Hadoop** is **not a single file**, it is **not a classical database**, it is **a distributed file system** (with many added functions and tools in its ecosystem)

- **Networks** can be very large, **10,000s of computers**

- HDFS is a low-level **distributed file processing system** (can be used directly for data storage)

Hadoop Distributed File System (HDFS)

HDFS/Hadoop approach based on several **key assumptions**:

- **High volume:** Default physical **block sizes is 64 MB**, hence much fewer blocks per file (files are assumed to be very large)
- **Write-once, read-many:** Model simplifies concurrency issues and improves data throughput
- **Streaming access:** Hadoop is optimized for batch processing of entire files as a continuous stream of data
- **Fault tolerance:** HDFS is designed to replicate data across many different devices so that when one fails, data is still available from another device (default **replication factor of three**)

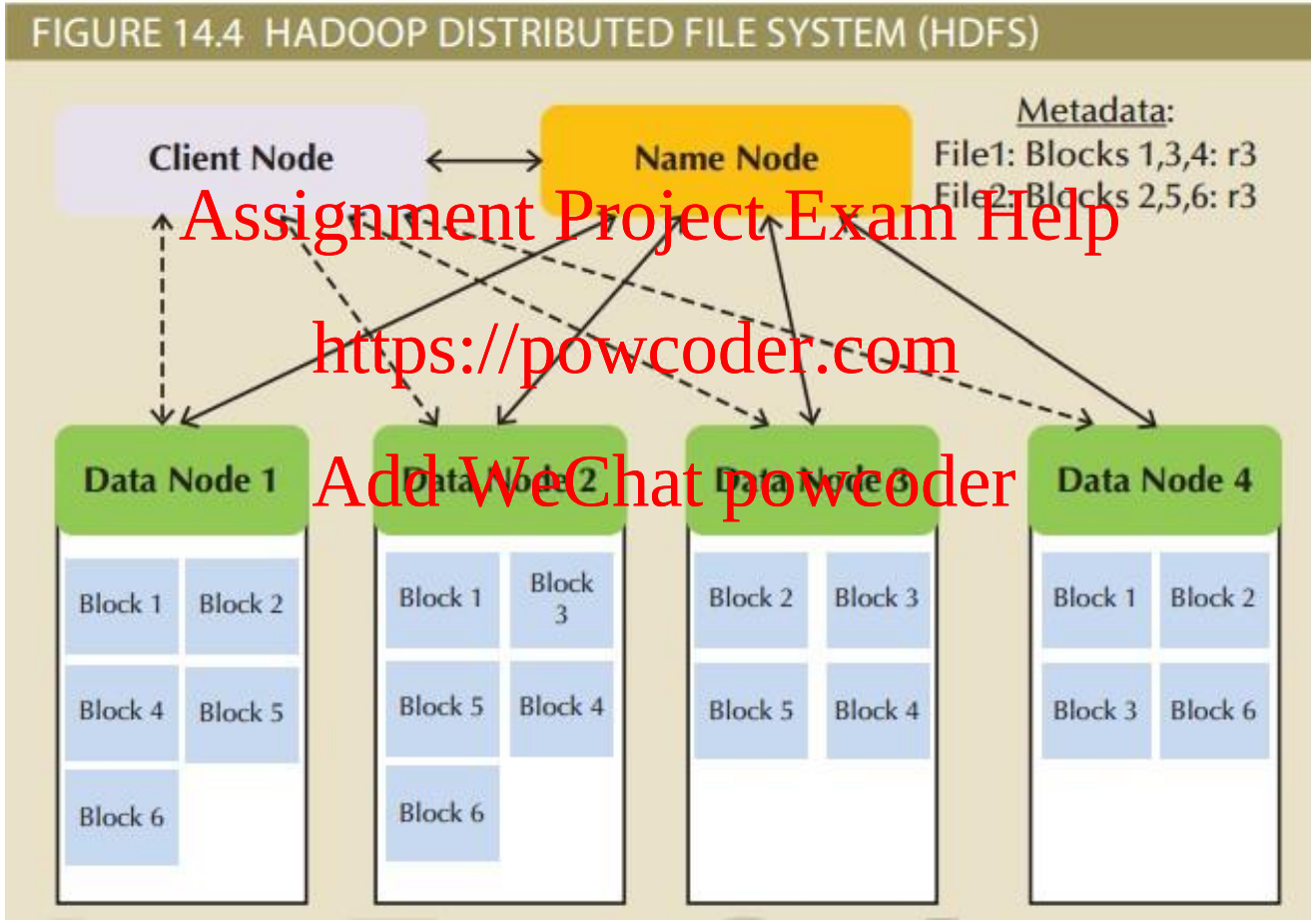
Hadoop Distributed File System (HDFS)

- HDFS uses several types of nodes (computers): (see figure next slide)
  - **Data node** stores the actual file data
  - **Name node** contains file system metadata
  - **Client node** makes requests to the file system as needed to support user applications

- **Same computer** can fulfil **several node types functions**.

- Data node communicates with name node by regularly sending block reports (list of blocks, every 6 hours) and heartbeats (every 3 seconds)
  - If heartbeat stops, data blocks of that node are replicated elsewhere

Hadoop Distributed File System (HDFS)



How Does HDFS Work? [Writing]

1. The **client node** needs to create a new file, and communicates with the **name node**.
2. The **name node**
   - adds the new file name to the metadata;
   - determines a new (first) block number for the file;
   - determines a list of on which data nodes the new block will be stored;
   - and passes that information back to the client node.
3. The **client node**
   - contacts the first data node specified by the name node and begins writing;
   - sends the data node the list of replicating data nodes.
4. First **data node** contacts the second data node in the list for replication while receiving it from the client node.
5. The **client node** gets further block numbers from the name node … until file is written.

MapReduce

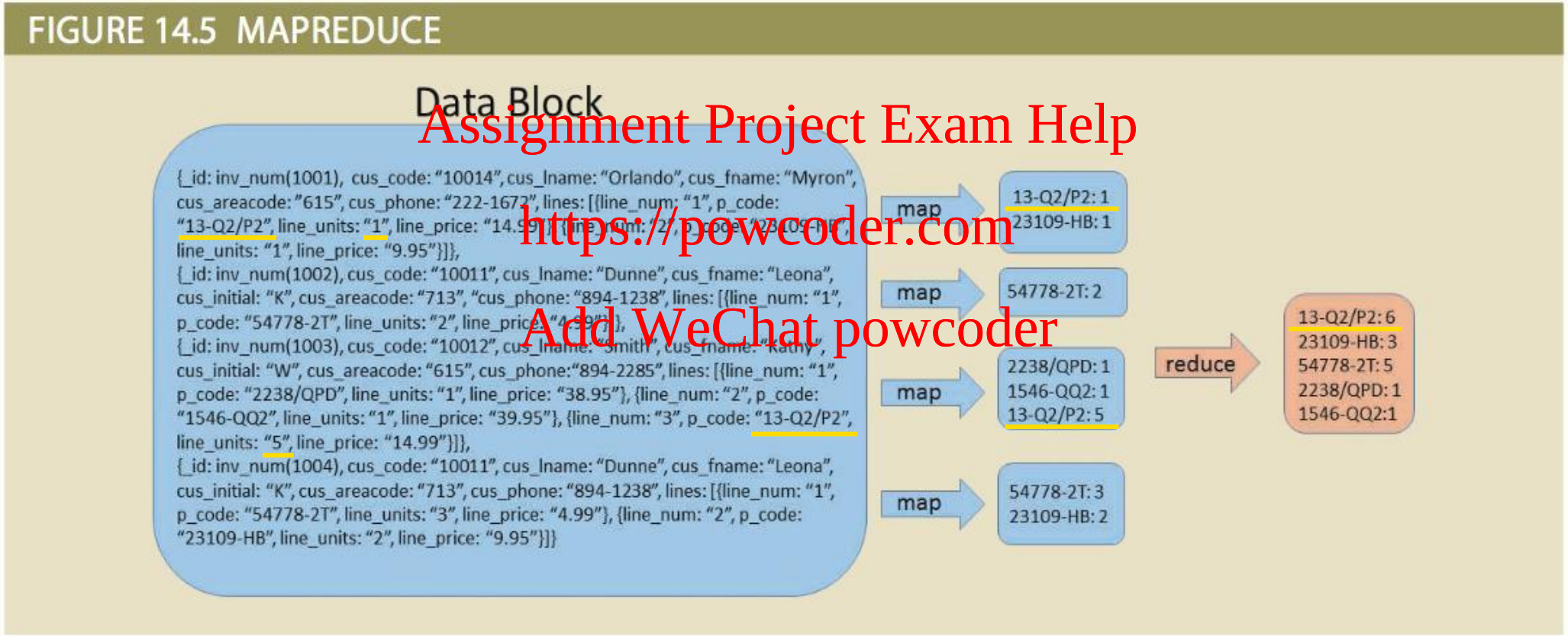
- Implementation **complements HDFS structure**.
- **Open-source** application programming interface (API).
- **Framework** used to process large data sets across clusters.
- **"Divide and conquer"** strategy: breaks down task into smaller subtasks, performed at node level in parallel and then aggregated to final result.
- Based on **batch processing**, runs tasks from beginning to end with no user interaction.

- **YARN** (Yet Another Resource Negotiator), or MapReduce 2, can do
  - Batch processing
  - **Stream processing** (for data that comes in/out continuously)
  - **Graph processing** (for social networks)

MapReduce

- **Map function** takes a collection of data and sorts and filters it into a set of key-value pairs.
  - **Mapper** program performs the map function
- **Reduce function** summaries results of map function to produce a single result.
  - **Reducer** program performs the reduce function
- **Map and reduce functions** are written as **Java** programs.
- Instead of central program retrieving the data for processing in a central location, **copies of the program are "pushed" to the nodes**.
- Typically **1 mapper *per block***, **1 reducer *per node***.

MapReduce



MapReduce

- **Job tracker** or central control program to accept, distribute, monitor and report on jobs in a Hadoop environment
  - Typically on **name node**.

- **Task tracker** is a program in MapReduce responsible for reducing tasks on a node
  - Typically on **data node**.

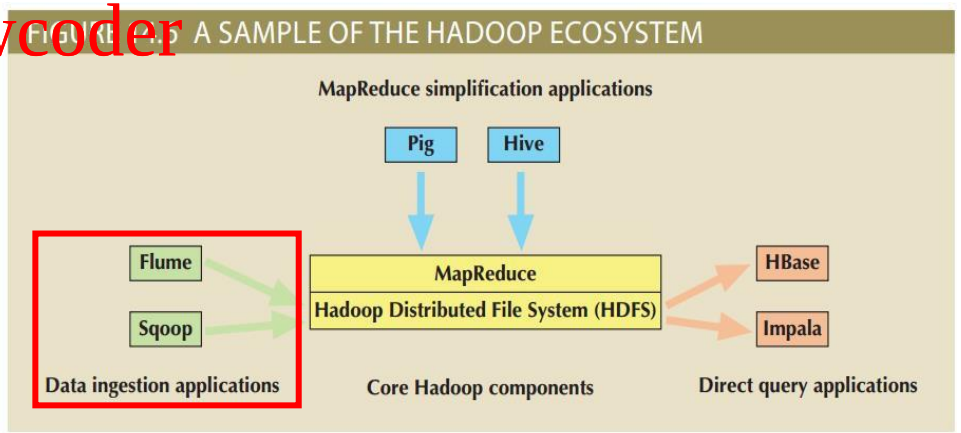
How Does MapReduce Work? [Reading/Analyzing]

1. A **client node** (client application) submits a **MapReduce job** to the job tracker.
2. The **job tracker** (on server that is also the **name node**):
   - communicates with name node to determine the relevant data node;
   - determines which task trackers are available for work (could be busy);
   - send portions of work to task trackers.
3. The **task tracker** (on server that is also a **data node**)
   - runs map and reduce functions (in virtual machine);
   - sends heartbeat ("still working") and "complete" message to job tracker.
4. The **client** node
   - periodically queries job tracker if all task trackers are completed;
   - receives completed job.

Hadoop Ecosystem – Data Ingestion Applications

- **Flume**
- **Sqoop**

- Why? **Help getting data from existing systems into Hadoop clusters**. These tools "ingest" or gather data into Hadoop.



Flume

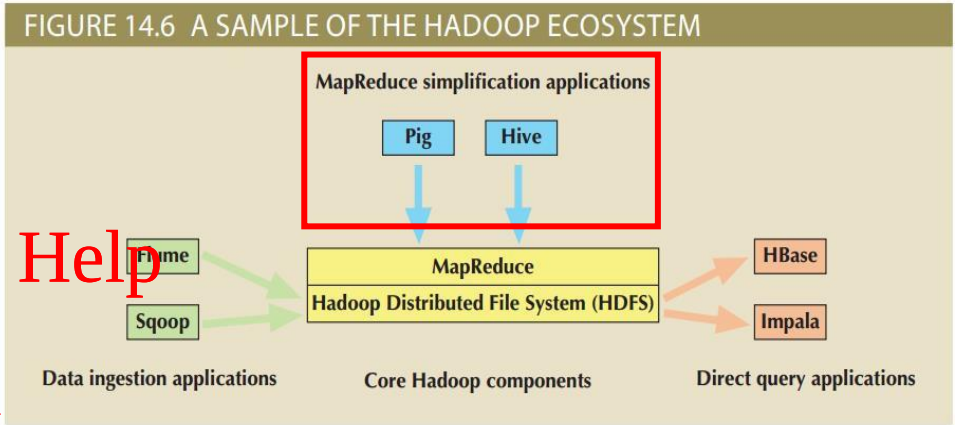
- Flume is a component for **ingesting data in Hadoop**.
- Primarily for harvesting large sets of data such **clickstream data/server logs**.
- Simple query processing component to performing **some transformation**.
- Can move data into **HDFS** or **HBase**.

Sqoop

- **"SQL-to-Hadoop."**
- **Sqoop** is a tool for converting data back and forth between **relational databases** and **HDFS** (both directions).
- Works with Oracle, MySQL, SQL Server, ...
- Example of Hadoop-to-SQL: MapReduce results imported back into a traditional (relational) data warehouse.

Hadoop Ecosystem – MapReduce Simplification Applications

- ❑ **Hive**
- ❑ **Pig**



- ❑ Why? **They help creating MapReduce jobs**.
  - Creating MapReduce jobs requires significant programming skills.
  - As the mapper and reducer programs become more complex, the skill requirements increase and the time to produce the programs becomes significant.

Hive

- **Hive** is a **data warehousing system** that sites on top of HDFS.
  - Supports its own **SQL-like language: HiveQL** (declarative / non-procedural)
  - **Summarizes queries, analyzes data**

This is the component that most people are going to use in terms of how to actually work with the data!

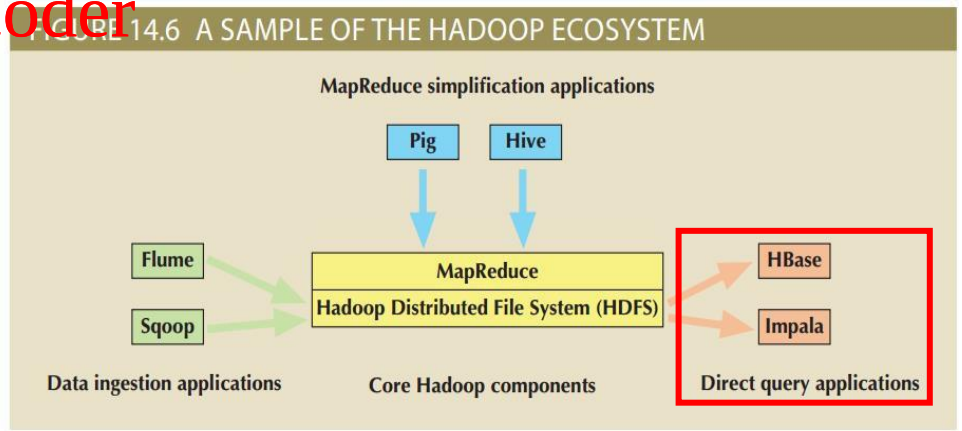
Pig

- Hadoop platform to **write MapReduce programs.**
- Has its own **high-level scripting/programming language: Pig Latin** (procedural).
- **Pig** compiles Pig Latin scripts **into MapReduce jobs** for executing in Hadoop.

Hadoop Ecosystem – Direct Query Applications

- **HBase**
- **Impala**

- Why? **To provide faster query access directly to HDFS** (without going through the MapReduce processing layer).



HBase

- **HBase** is a **NoSQL** database
- **Column-oriented**
- Designed to **sit on top of HDFS**
- **Quickly** processes **smaller subsets** of the data
- **No SQL** support, instead uses **Java**

Impala

- First **SQL-on-Hadoop** application
- Produced by **Cloudera**
- **SQL queries** directly against the data while it is still in **HDFS**
- Makes heavy use of **in-memory caching on data nodes**

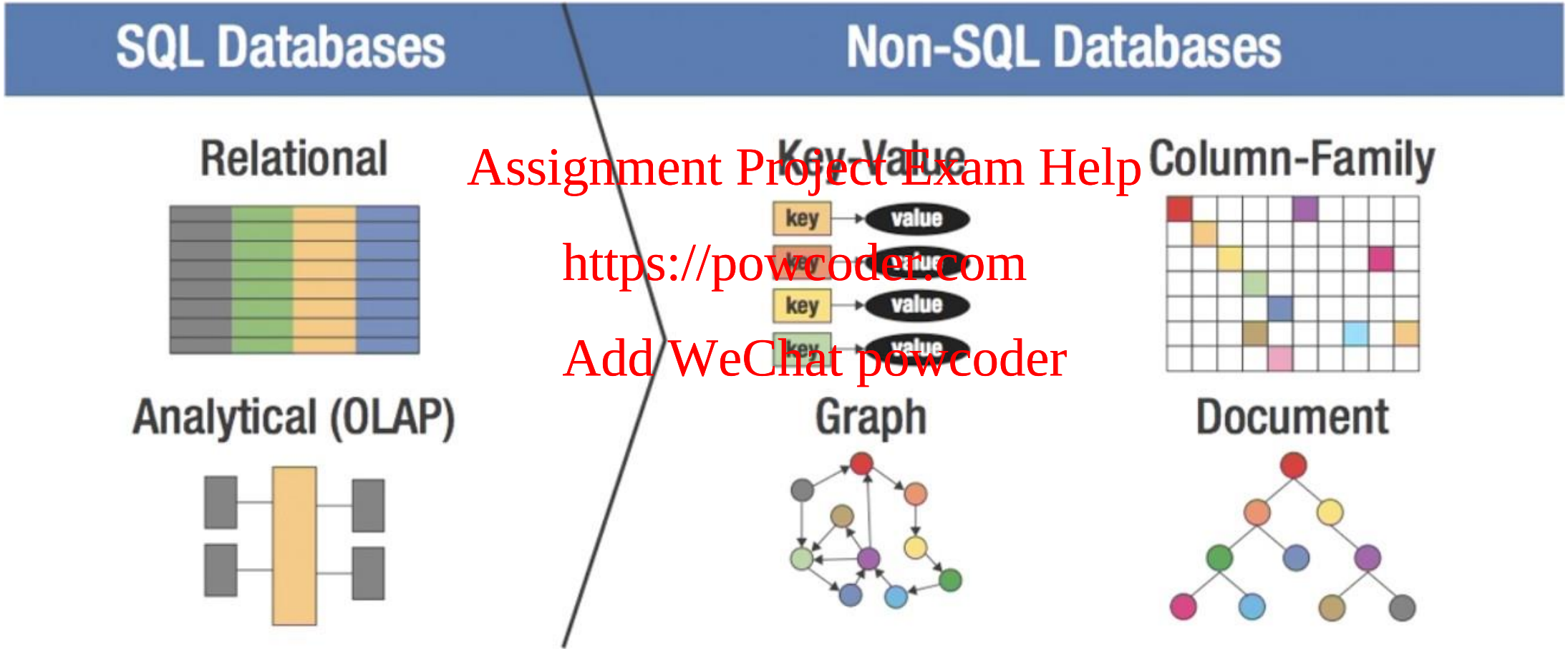
NoSQL Database Types

NoSQL

- **Non-relational database technologies** developed to address **Big Data challenges**
- **NoSQL** = “not modelled using relational model” (“non-SQL” / “not-only SQL”)
- **Category** emerged from organizations such as **Google**, **Amazon** and **Facebook** that faced problems of their data sets reached enormous sizes
- **Much larger data** volumes can be stored
- **Flexible structure** and often **faster**
- No standardized query language – no SQL! (maybe in the future)

- Less adopted than RDBMS:
  - Was at peak in 2015-2016
  - Survey 2016, **16%** of companies use **NoSQL** databases and **79%** of companies use **relational databases**
- *NoSQL seems to be in decline nowadays ??!!*

NoSQL



NoSQL

TABLE 14.2

NoSQL DATABASES

NoSQL CATEGORY	EXAMPLE DATABASES
Key-value database	Dynamo Riak Redis Voldemort
Document databases	MongoDB CouchDB OrientDB RavenDB
Column-oriented databases	HBase Cassandra Hypertable
Graph databases	Neo4J ArangoDB GraphBase

NoSQL – Key-Value Database

- Store data as a collection of **key-value pairs** (keys ~ primary keys, there are no foreign keys)
- Key-value pairs are organized in logical groupings, **buckets** (buckets ~ tables)
- Key values must be unique (only) within a bucket.
- **Queries** are based on **buckets and keys** (not values)
- ***get*, *store* and *delete*** operations

FIGURE 24.7 KEY-VALUE DATABASE STORAGE

Bucket = Customer

Key	Value
10010	"LName Ramas FName Alfred Initial A Areacode 615 Phone 844-2573 Balance 0"
10011	"LName Dunne FName Leona Initial K Areacode 713 Phone 894-1238 Balance 0"
10014	"LName Orlando FName Myron Areacode 615 Phone 222-1672 Balance 0"

NoSQL – Document Databases

- **Document databases** store data in **key-value pairs** in which the value components are **tag-encoded documents.**
- Document can be encoded in **XML**, **JSON** or **BSON** (Binary JSON).
- Have tags, but still **schema-less** (not schemas, documents may have different tags).
- Documents are grouped into logical groups called **collections** (buckets).
- Tags can be queried (e.g., where balance = 0).

FIGURE 14.8 DOCUMENT DATABASE TAGGED FORMAT

Collection = Customer

Key	Document
10010	{LName: "Ramas", FName: "Alfred", Initial: "A", Areacode: "615", Phone: "844-2573", Balance: "0"}
10011	{LName: "Dunne", FName: "Leona", Initial: "K", Areacode: "713", Phone: "894-1238", Balance: "0"}
10014	{LName: "Orlando", FName: "Myron", Areacode: "615", Phone: "222-1672", Balance: "0"}

NoSQL – Column-Centric Databases

- **Column-centric (columnar) databases** focuses on storing data in columns, not rows, but still relational logic.
- **Column-centric storage:** Data stored in blocks which hold data from **a single column across many rows**
- **Row-centric storage:** Data stored in blocks which hold data from **all columns of a given set of rows**

FIGURE 14.9 COMPARISON OF ROW-CENTRIC AND COLUMN-CENTRIC STORAGE

CUSTOMER relational table

Cus_Code	Cus_LName	Cus_FName	Cus_City	Cus_State
10010	Ramas	Alfred	Nashville	TN
10011	Dunne	Leona	Miami	FL
10012	Smith	Kathy	Boston	MA
10013	Olowski	Paul	Nashville	TN
10014	Orlando	Myron		
10015	O'Brian	Amy	Miami	FL
10016	Brown	James		
10017	Williams	George	Mobile	AL
10018	Farriss	Anne	Opp	AL
10019	Smith	Olette	Nashville	TN

Row-centric storage

Block 1
10010,Ramas,Alfred,Nashville,TN
10011,Dunne,Leona,Miami,FL

Block 2
10012,Smith,Kathy,Boston,MA
10013,Olowski,Paul,Nashville,TN

Block 3
10014,Orlando,Myron,NULL,NULL
10015,O'Brian,Amy,Miami,FL

Block 4
10016,Brown,James,NULL,NULL
10017,Williams,George,Mobile,AL

Block 5
10018,Farriss,Anne,OPP,AL
10019,Smith,Olette,Nashville,TN

Column-centric storage

Block 1
10010,10011,10012,10013,10014
10015,10016,10017,10018,10019

Block 2
Ramas,Dunne,Smith,Olowski,Orlando
O'Brian,Brown,Williams,Farriss,Smith

Block 3
Alfred,Leona,Kathy,Paul,Myron
Amy,James,George,Anne,Olette

Block 4
Nashville,Miami,Boston,Nashville,NULL
Miami,NULL,Mobile,Opp,Nashville

Block 5
TN,FL,MA,TN,NULL,
FL,NULL,AL,AL,TN

NoSQL – Column-Centric Databases

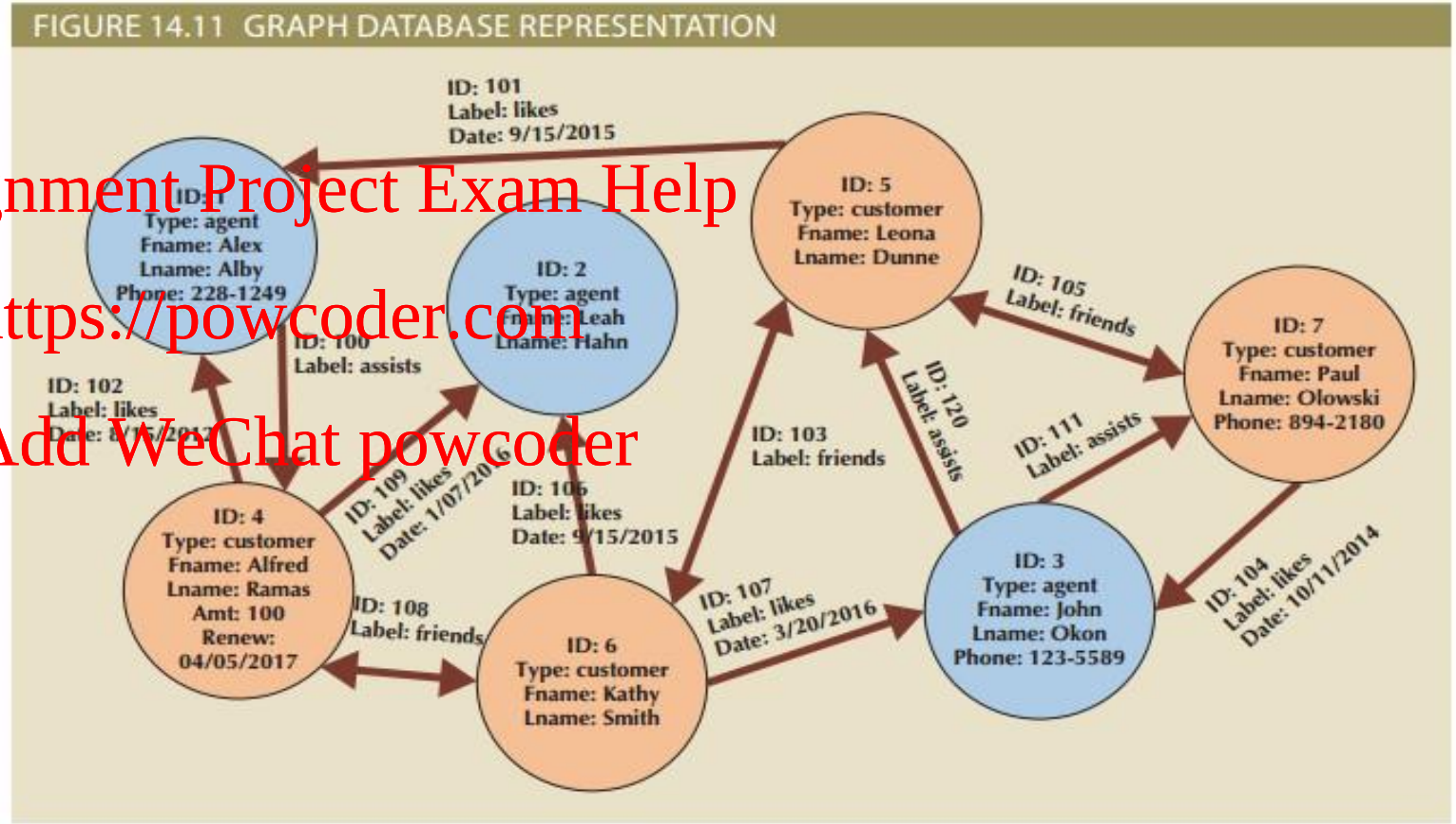
- **Column-oriented (column family) databases** in NoSQL:
  - Organizes data in key-value pairs.
  - Keys are mapped to columns in the value component.
  - The columns vary by row.
- **Key-value pair:** name of the column as key + data as value. Example: “cus_lname: Ramas”. (~**cell** in relational model)
- **Super column:** group of columns that are logically related (~**composite attribute**)
- **Rows keys**: created to identify objects (~**entity instances**) in the environment
- **Column family:** All of the columns (or super columns) that describe objects are grouped (~**table**)

FIGURE 14.10 COLUMN FAMILY DATABASE

Column Family Name	CUSTOMERS
Key	Rowkey 1
Columns	City: Nashville Fname: Alfred Lname: Ramas State: TN
Key	Rowkey 2
Columns	Balance: 345.86 Fname: Kathy Lname: Smith
Key	Rowkey 3
Columns	Company: Local Markets, Inc. Lname: Dunne

NoSQL – Graph Databases

- Suitable for **relationship-rich data**
- A collection of **nodes and edges**
- **Properties** are the **attributes of a node or edge** of interest to a user
- **Traversal** is a **query** in a graph databases



Applications of NoSQL

- Twitter app generating 7 Tbs+ of daily tweets and displaying it back.
- Property details in a real-estate website, redundant in nature but accessed in huge numbers.
- Online coupon sites distributing coupons to open market.
- Update of railway schedules and accessed by thousands of users at peak time.
- Real time score update of baseball / cricket match.

Big Data Strategies

What is Big Data Strategy?

A Big Data strategy defines and lays out a comprehensive vision across the enterprise and sets a foundation for the organization to employ data-related or data-dependent capabilities.

Week 9 - Big Data II

Pre-Class Activities

- How Do You Create A Data Strategy?
- How to Define a Big Data Strategy
- How to Develop a Data Strategy (Bernard Marr)
- Types and Examples of NoSQL Databases
- Week 9 Pre-Class Tasks

Source: https://www.bigdataframework.org/formulating-a-big-data-strategy/

Challenges of Implementing Big Data Strategy

- Technological
  - Lack of managerial analytics knowledge
  - Technical misunderstandings between managers and data scientists
  - Inherent challenges related to Big Data (e.g., 3Vs)
  - Technical requirements in compliance with data ownership and privacy regulations (e.g., NSW Transport data liberation could lead to app deluge https://www.itnews.com.au/news/nsw-%20transport-data-liberation-could-lead-to-app-deluge-418406)
  - Costly data management tools

(Tabesh et al. 2019)

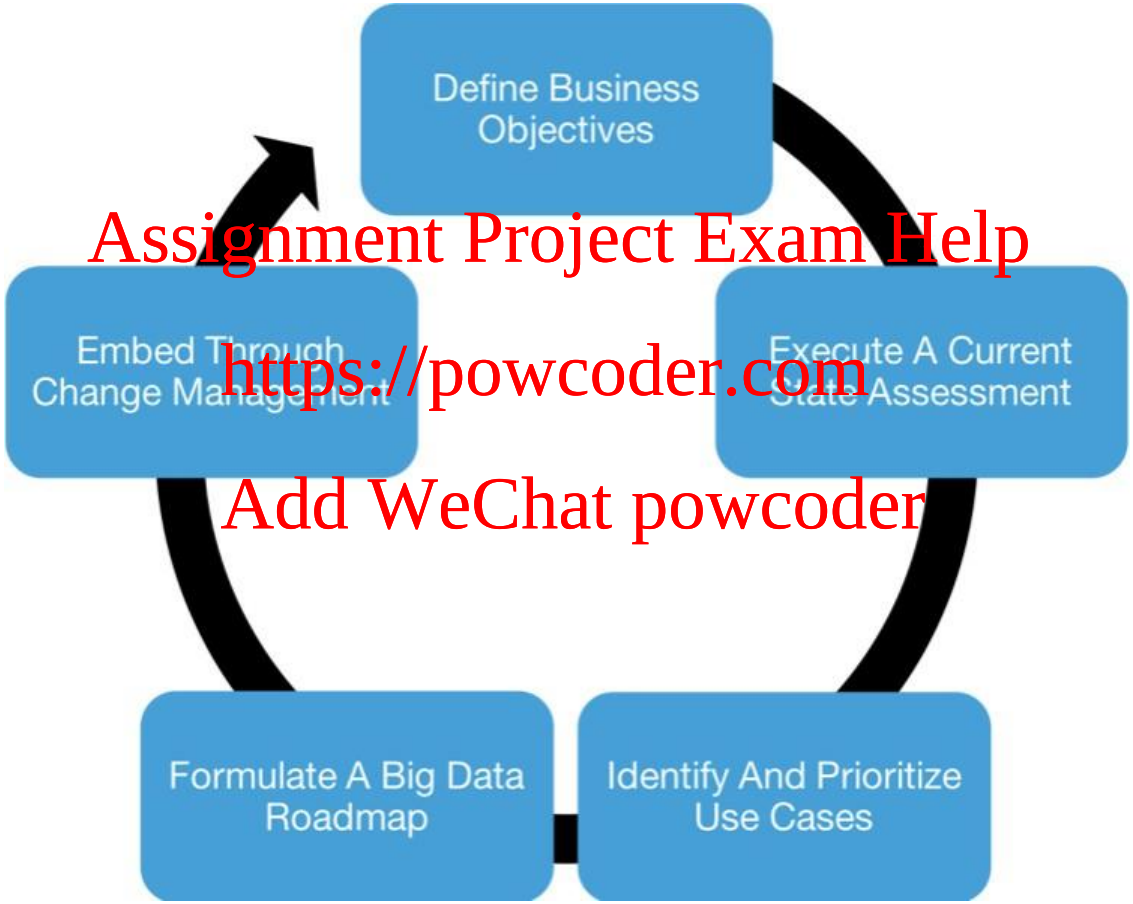
Challenges of Implementing Big Data Strategy

- Cultural
  - Extensive reliance on intuitive or experiential decision-making approaches
  - Dominance of management in the decision-making process
  - Lack of a shared understanding of Big Data and its goals

(Tabesh et al. 2019)

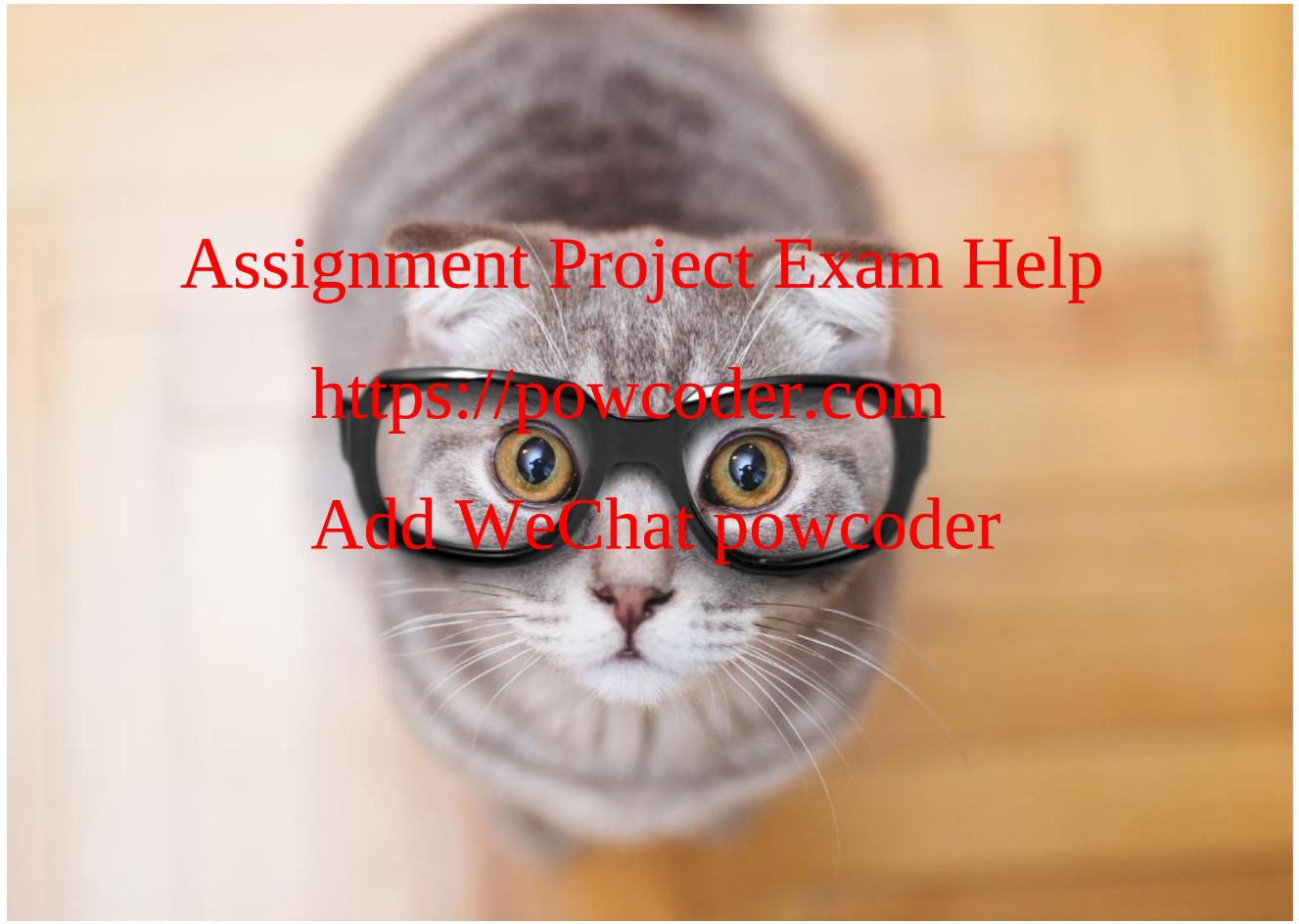
Reference:
Tabesh, P., Mousavidin, E. and Hasani, S., 2019. Implementing big data strategies: A managerial perspective. *Business Horizons, 62*(3), pp.347-358. https://doi.org/10.1016/j.bushor.2019.02.001

Implementing Big Data Strategy



Source: https://www.bigdataframework.org/formulating-a-big-data-strategy/

Questions



Source: stacker.com