



MIS 381N INTRO. TO DATABASE MANAGEMENT

Enterprise Data Architecture

NoSQL

Tayfun Keskin

Visiting Clinical Professor, The University of Texas at Austin, McCombs School of Business

Associate Teaching Professor, University of Washington Seattle, Foster School of Business

QUESTIONS

- Any questions before we begin?



AGENDA



Lecture

Ent. Data Architecture
NoSQL



Discussion

NoSQL
Case Analysis



Looking Forward

Homework 5
Final Project





QUESTION

What is the purpose of a database?

Hint: What do we store in a database?

Which data type?

Relational Databases are an incredible commercial success

Simple &
Powerful Model

- **Intuitive and elegant entity-relationship model**
- Ability to represent complex, real-world business scenarios
- Ability to generalize into software products and customize for specific industries and businesses

Standardization

High
Performance &
Secure

Innovation



Relational Databases are an incredible commercial success

Simple &
Powerful Model

Standardization

High
Performance &
Secure

Innovation

- SQL is for the most part standard and portable
- SQL can be used by people with “moderately” technical skills
- Integration of SQL with programming languages
- Robust commercial offerings (Oracle, MS-SQL, ...)
- Robust open source offerings (MySQL, Postgres, ...)



Relational Databases are an incredible commercial success

Simple &
Powerful Model

Standardization

High
Performance &
Secure

Innovation

- Feature rich security models (authentication, **authorization, role/content based access controls, encryption ...**)
- Optimized physical implementations for large data volumes (partitioning, **indexing, ...**) and thousands of concurrent users
- **Widely deployed business applications**



Relational Databases are an incredible commercial success

Simple &
Powerful Model

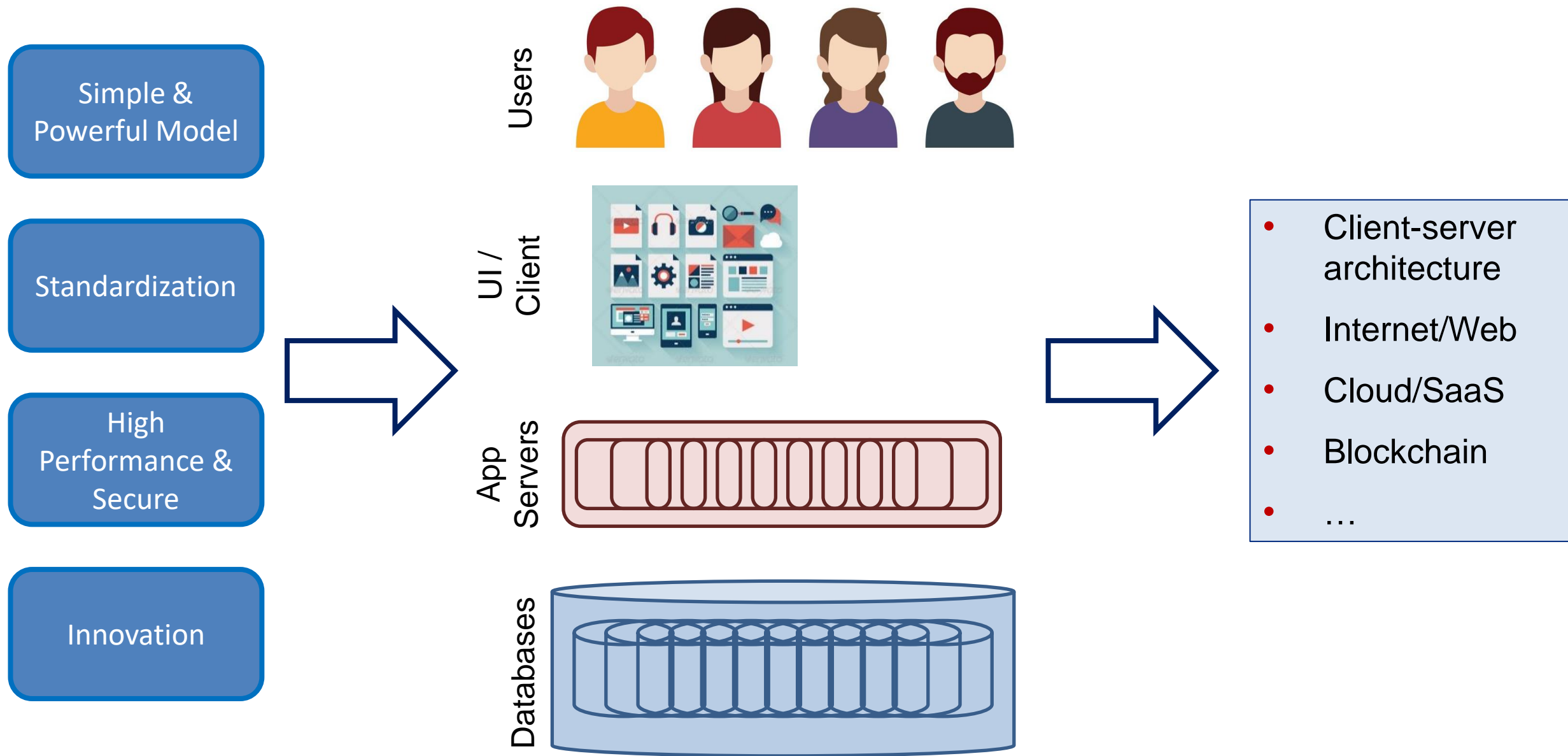
Standardization

High
Performance &
Secure

Innovation

- Research by industry and academia continues
- Continuous innovation for the last 40 years
- Consolidation has not prevented a competitive market

RDBMS an incredible commercial success through all trends





QUESTION

What type of data could be challenging to store in a table?

What other problems do you predict with RDBMS?

However, there are several issues with Relational databases

Change
Management
Complexity

- Changing an in-use data model is difficult, time-consuming, and expensive
- Maintenance of database code such as stored procedures is complex
- Diagnosing performance changes in a database is complex
- Maintaining user roles and privileges requires elevated skillset (i.e., matrix)

Business
Analytics

Real-world
modeling
challenges

Scalability Costs

However, there are several issues with Relational databases

Change
Management
Complexity

Business
Analytics

- Data needed for analytics is distributed across multiple databases
- Often data from these multiple databases cannot be combined easily
- Lack of uniform naming standards or definitions of data across multiple DBs

Real-world
modeling
challenges

Scalability Costs

However, there are several issues with Relational databases

Change
Management
Complexity

Business
Analytics

Real-world
modeling
challenges

Scalability Costs

- Limited data types – an issue for text, video, audio, location data
- Entities artificially separate “data” from “behavior”
- Real-world scenarios such as graphs (e.g., social networks) difficult to model
- Difficult to represent “semantics” of relationships
- Difficult to deal with constant changes in fields and types of data

However, there are several issues with Relational databases

Change
Management
Complexity

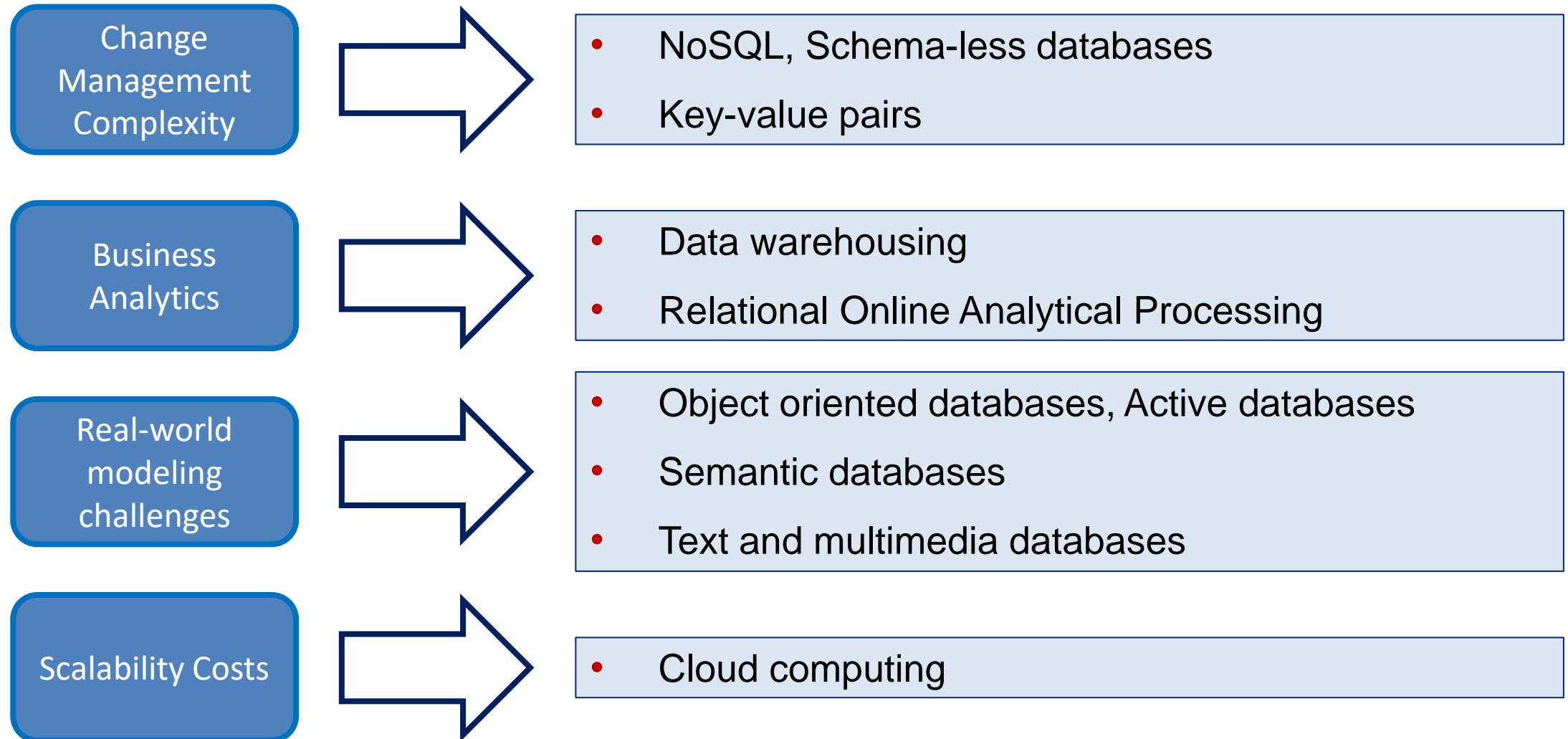
Business
Analytics

Real-world
modeling
challenges

Scalability Costs

- Commercial DB products are expensive with increasing costs
- Highly skilled labor required for maintenance
- Expensive hardware infrastructure needed for high/peak performance

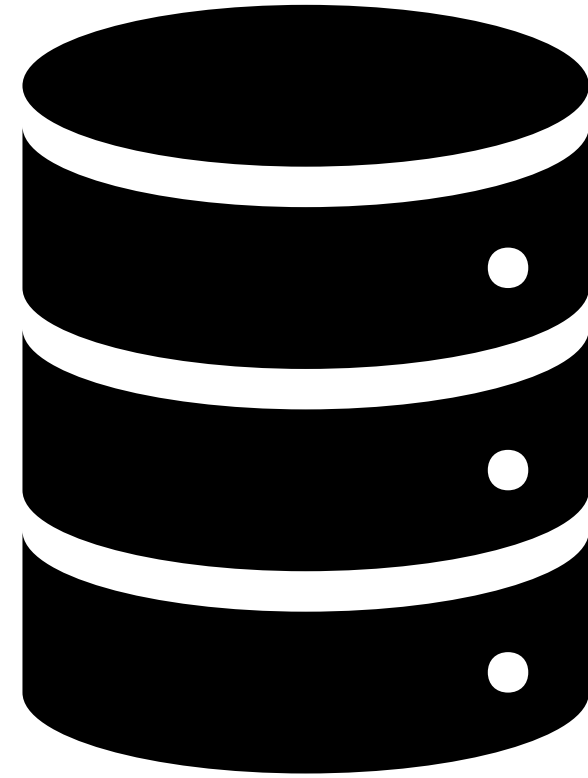
Possible Solutions to RDBMS issues





What happened?

**Specialized solutions
have been developed
over the past 40+
years to address the
limitations of relational
database management
systems**





REVIEW QUESTION

What are pros and cons of normalization?

Pros and cons of normalization

Pros

1. Reduce redundant data – leads to increased data integrity
2. Separates entities into intuitive design which aids query design
3. Segment sensitive data
4. Grant granular permission by table

Cons

1. More difficult for business folks to understand (i.e., ERD level)
2. Queries require joins which are hard unless trained in them
3. Query performance impacted as you join more tables



REVIEW QUESTION

What are pros and cons of denormalization?

Pros and cons of denormalization

Pros

- 1. Performance** – Data is combined in one table so that reading and updating the data does not require joins
- 2. Analysis** – Data is combined in one table so that it can be analyzed without knowledge of the database structure.

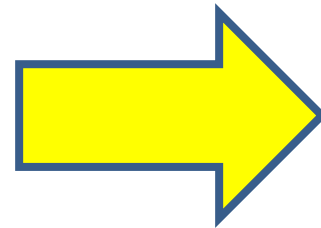
Cons

- 1.** Redundant data impacts data quality and integrity
- 2.** Higher data storage costs (due to higher amount of dup data)
- 3.** More difficult for some to govern/understand data as a whole. All entities in one place instead of broken out

Online Transaction Processing (i.e., Enterprise Apps) Need

- High performances
- Large number of concurrent users
- Queries are known
- Data domains are limited
- Create, read, update and delete data
- More interested in current data rather than historical data

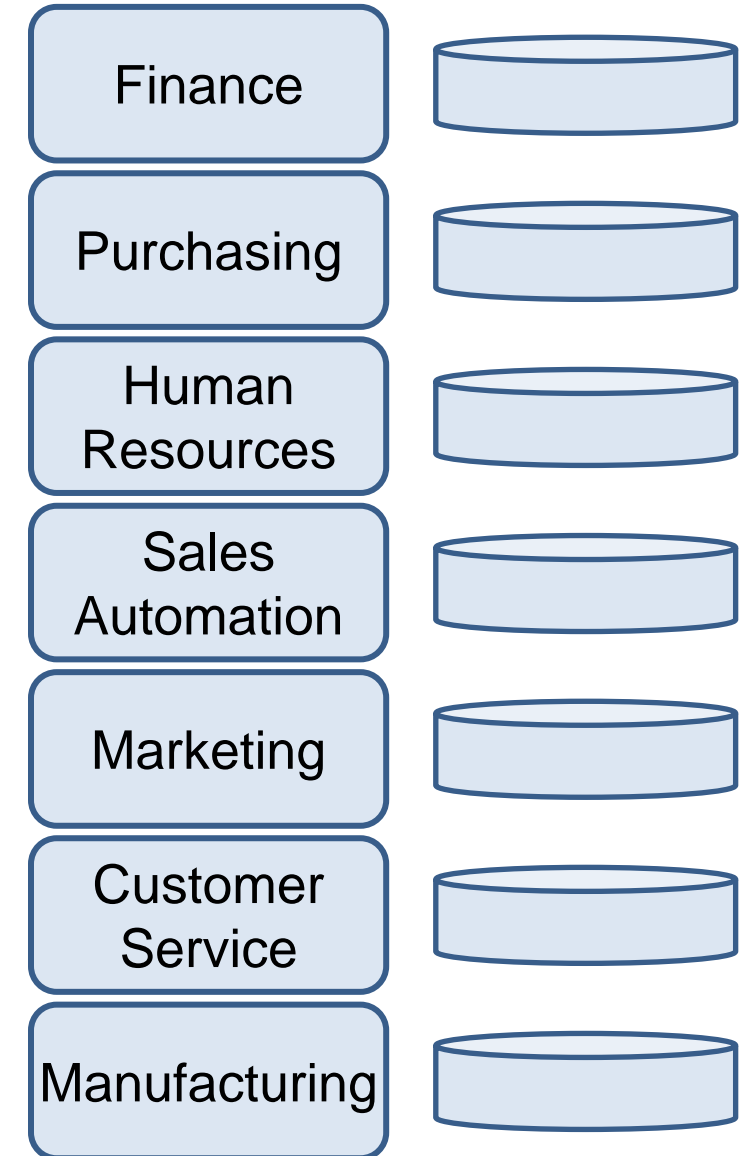
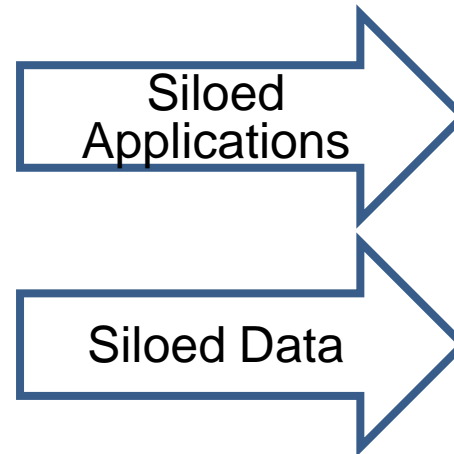
OLTP (Online transaction processing) systems are sometime called “operational systems” or “source systems”



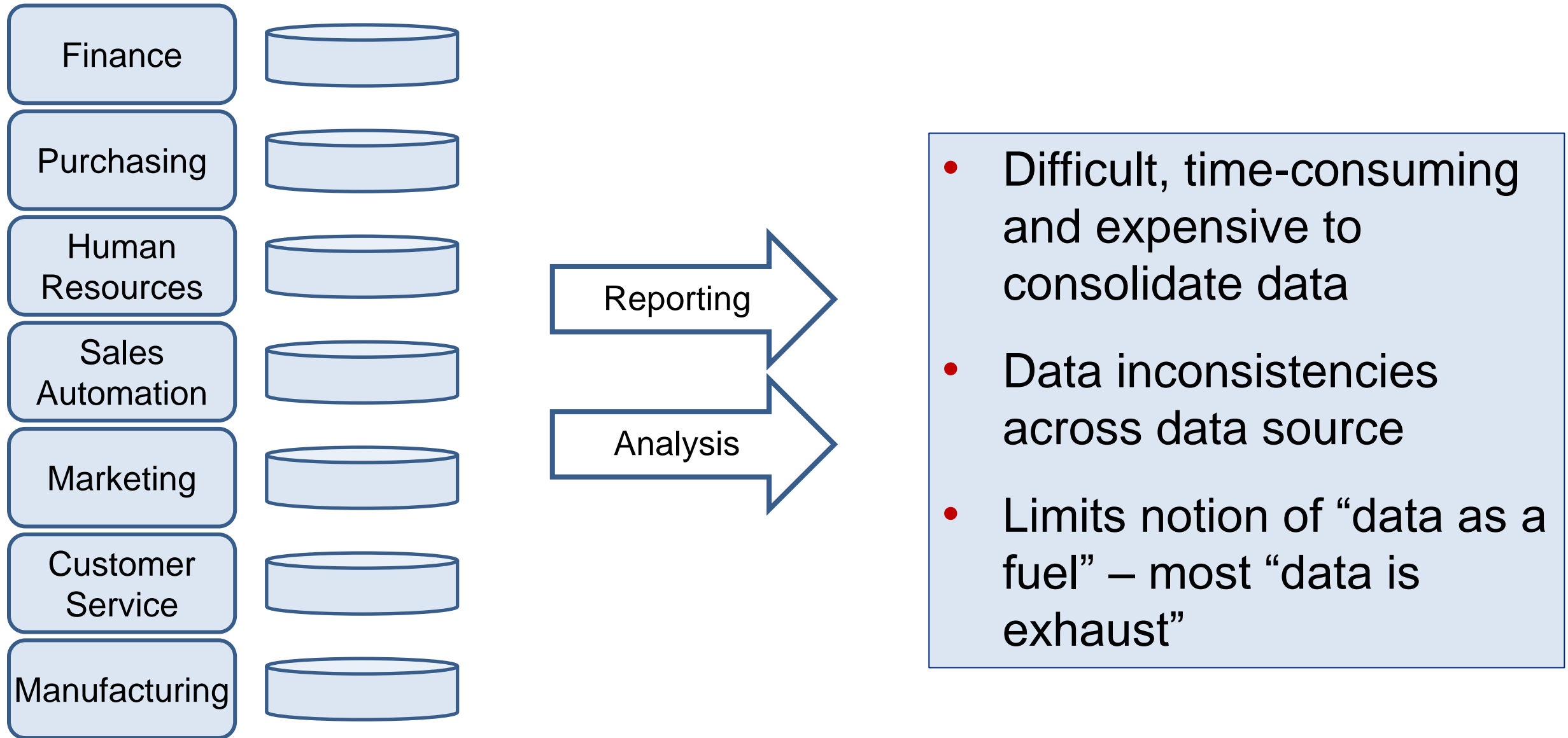
Needs normalization

OLTP or Transactional Processing results in...

- High performances
- Large number of concurrent users
- Queries are known
- Data domains are limited
- Create, read, update and delete data
- More interested in current data rather than historical data



OLTP or Transactional Processing results in...



What is the common dilemma with data?

- Data is in too many places
 - Data non maintained consistently
 - Data is “dirty” or missing values
 - Data hard to retrieve from the “legacy system”
 - There’s too much to make sense of it all
-
- Examples
 - Dell
 - NCUA
 - UT Austin



→ Legacy Perot



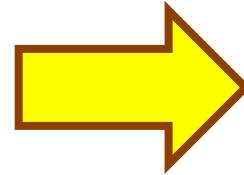
QUESTION

How many systems have your name stored at
The University of Texas?

Let's list them...

ANALYTICS NEEDS...

- Data source needs to be “company wide”
- Queries are not known, flexibility is needed
- Performance is not that important
- Read data only
- Current and historical data are required

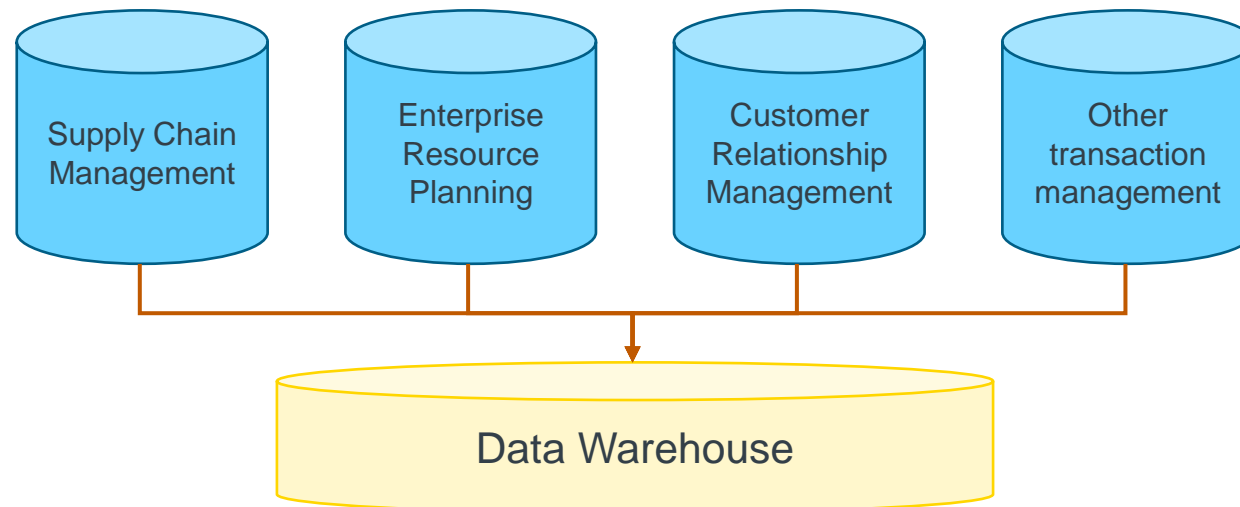


Needs
Denormalization



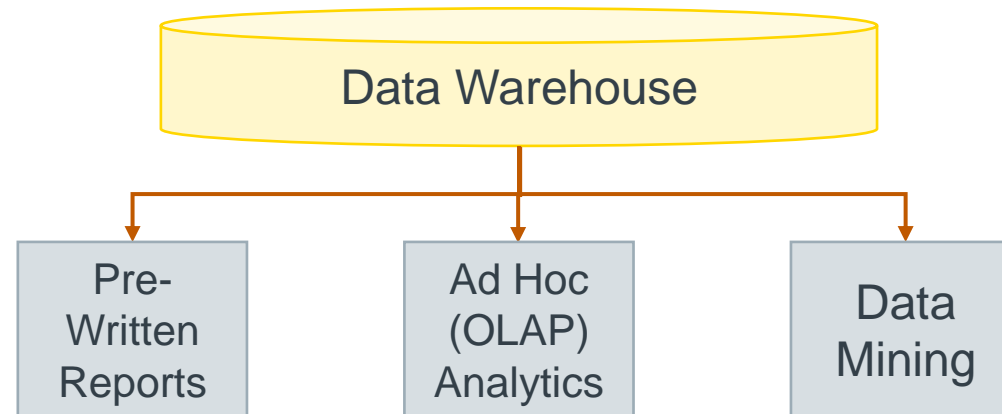
CONVERTING DATA INTO INFORMATION

- Unfortunately, transaction databases are fragmented and not useful for analytics, so we consolidate data from multiple transactional systems to build a data warehouse for analytics



BUSINESS INTELLIGENCE

- Business intelligence (BI) tools provide reporting, sophisticated data modeling, and analysis for organizational decision making from data in the data warehouse.



OLAP tools organize data in dimensions or cubes to facilitate analytics

WHAT IS A DATA WAREHOUSE?

- A data warehouse is a
- All-encompassing vs limited to one domain
 - integrated
 - historical (i.e. not real-time)
 - non-volatile (i.e. data is static)

...collection of data in support of management's decision-making process

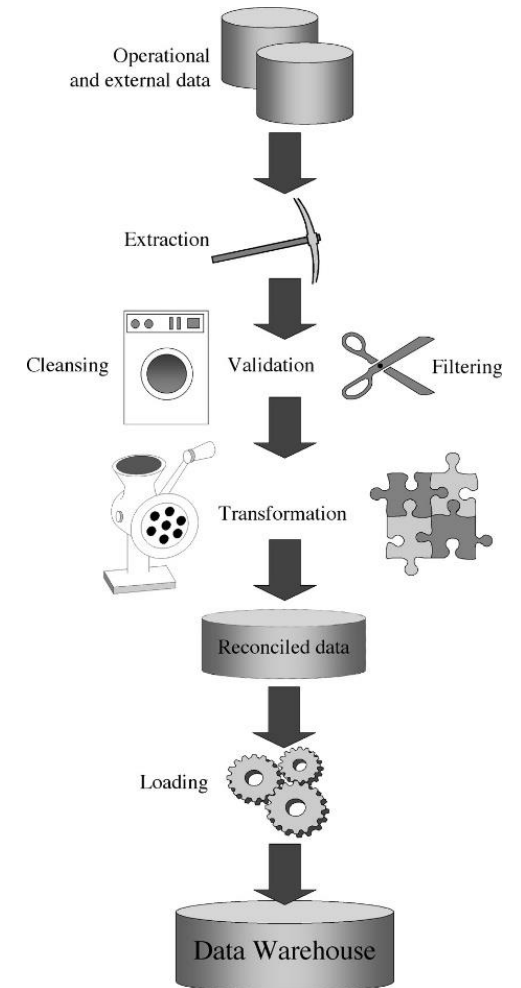
- The process of building the data warehouse is as important as the data warehouse itself



WHAT IS EXTRACT TRANSFORM AND LOAD (ETL)?

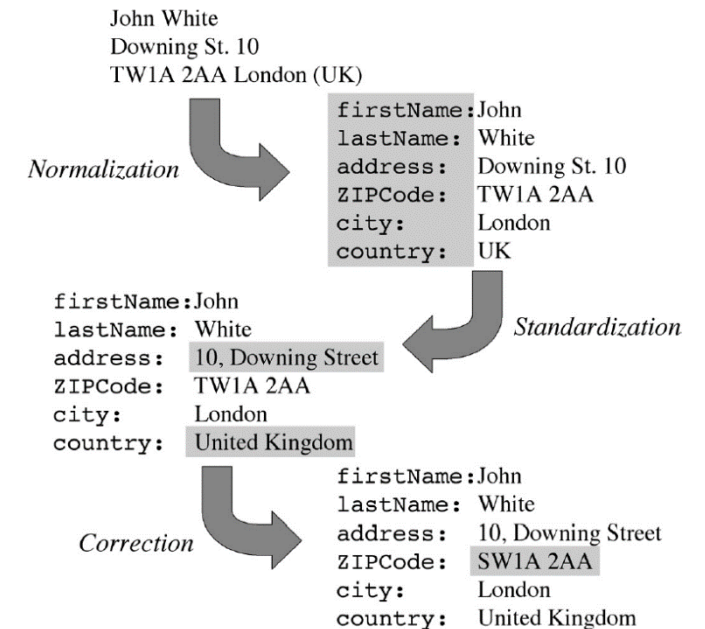
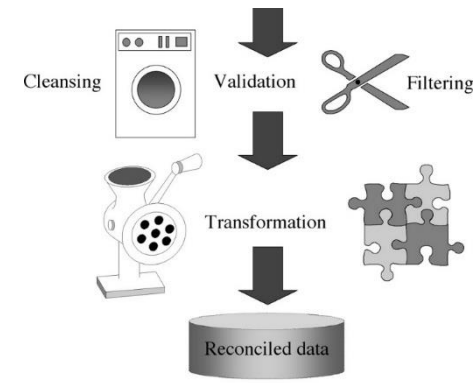
The process of

- Extracting data from multiple operational sources
- Validating, selecting, cleaning, and aligning the data to the standardized Data Warehouse data model (Transformation)
- Designing & implementing the process for Loading the data into the data warehouse on a regular schedule



WHAT DOES TRANSFORM MEAN?

- Cleansing – Duplicate data, missing values, incorrect values, inconsistent values, etc.
- Aligning data to standardized DW data model Changing column names, data types, creating new fields from existing fields, “fixing” keys, adding dates to maintain history, etc.



COMMON USES OF ETL

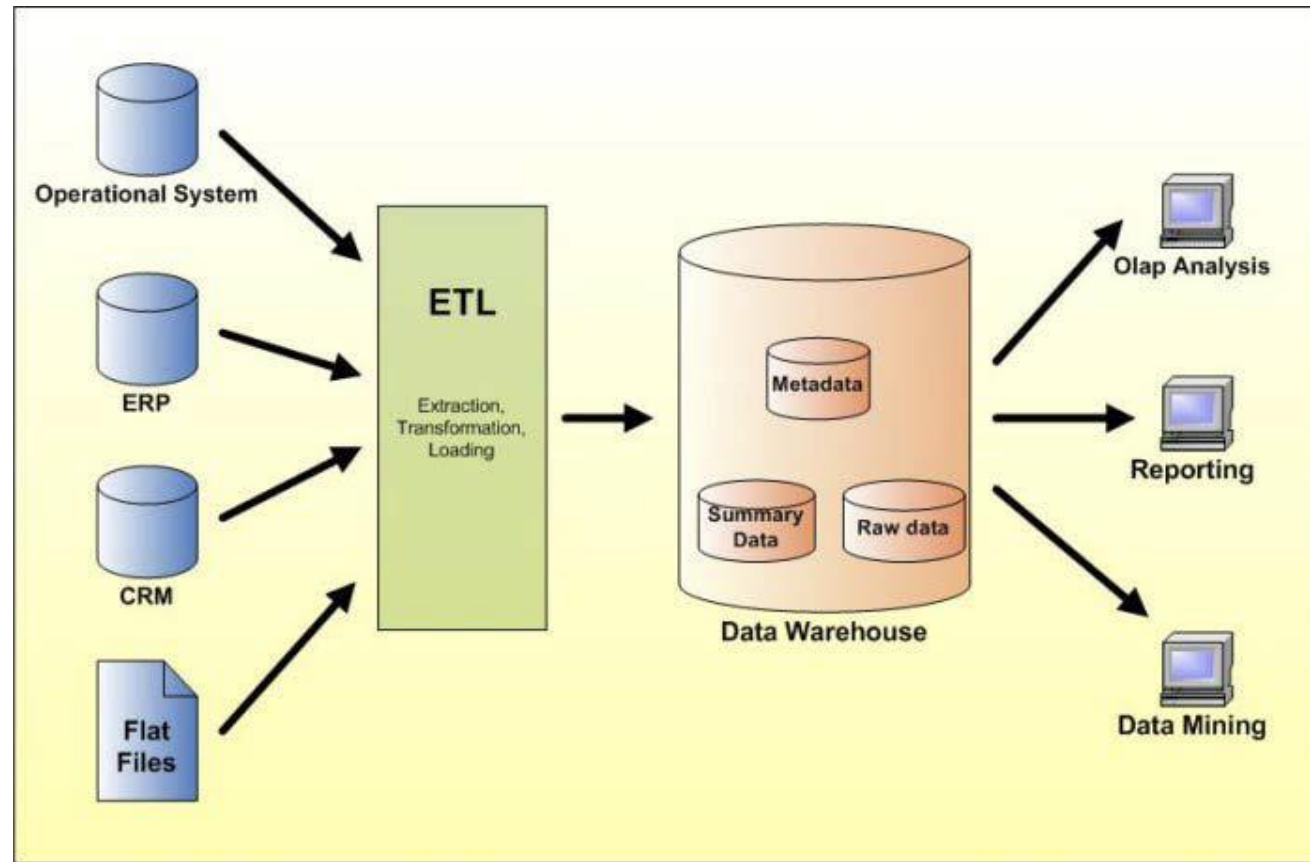
- Consolidate data for use in a Data Mart or Data Warehouse
 - Data Warehouse - A set of databases designed to consolidate data that supports decision making in an organization.
 - Data Mart - A database or databases focused on addressing the concerns of a specific problem (e.g., increasing customer retention, improving product quality) or business unit (e.g., marketing, engineering)
- Migrating Data from older system to a new one
- Merging acquired company data or pulling out divested data
- Pulling 3rd party data into your system

ETL TOOLS CAN BE CONFIGURED

- Allows for easier data mapping between source and targets that are structured differently or on different platforms
- Includes data integrity checks and transformation (e.g. Our home has different phone number formats and column names)
- Setup to pull and load ALL data or partial data based on dates or filter criteria. Do one-time or continued updates.
- Setup to run based on a trigger or schedule



EXAMPLE OF DATA WAREHOUSE ARCHITECTURE



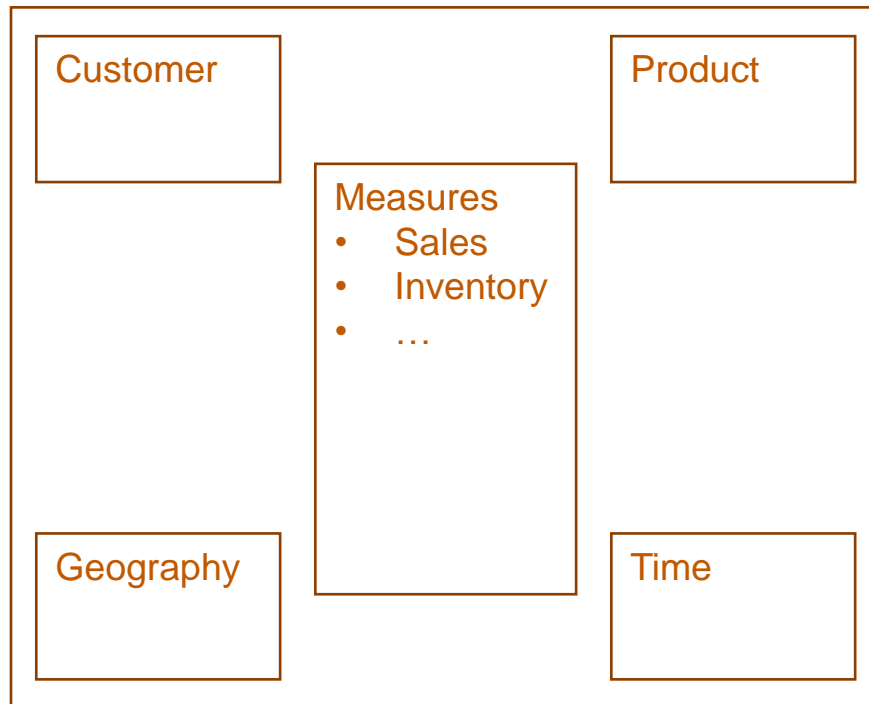


DO YOU KNOW?

Many contemporary data science concepts build on previous work with relational databases.

Have you ever heard the term Enterprise Data Warehouse? In which context? What is it?

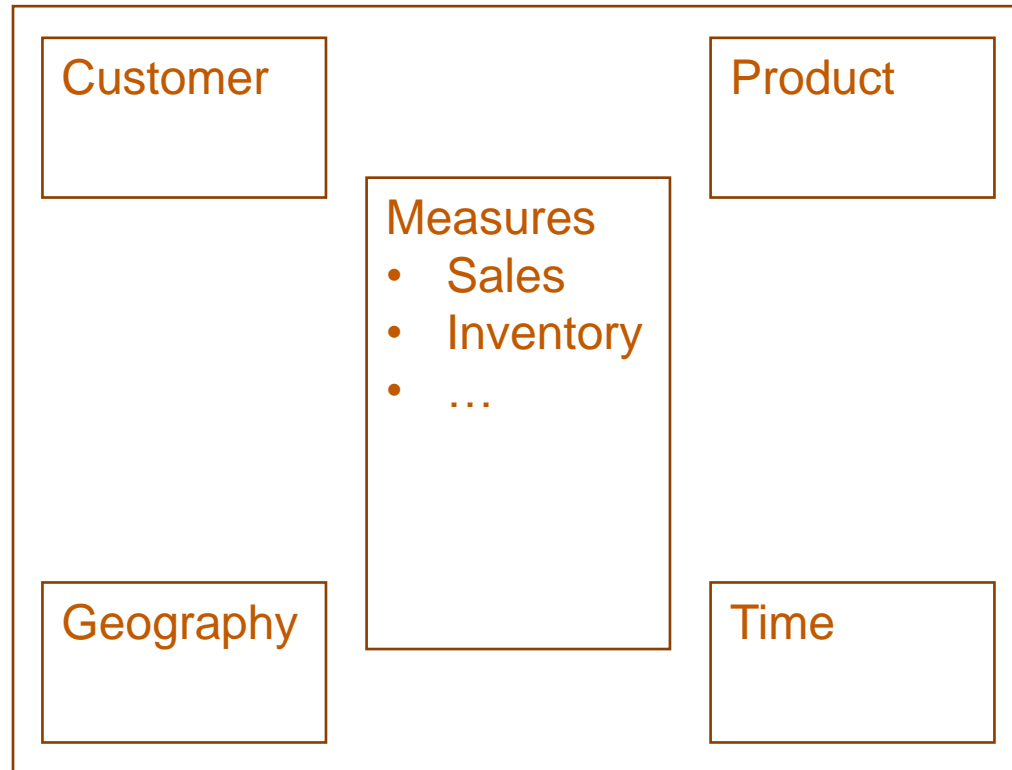
OLAP (ONLINE ANALYTICAL PROCESSING)



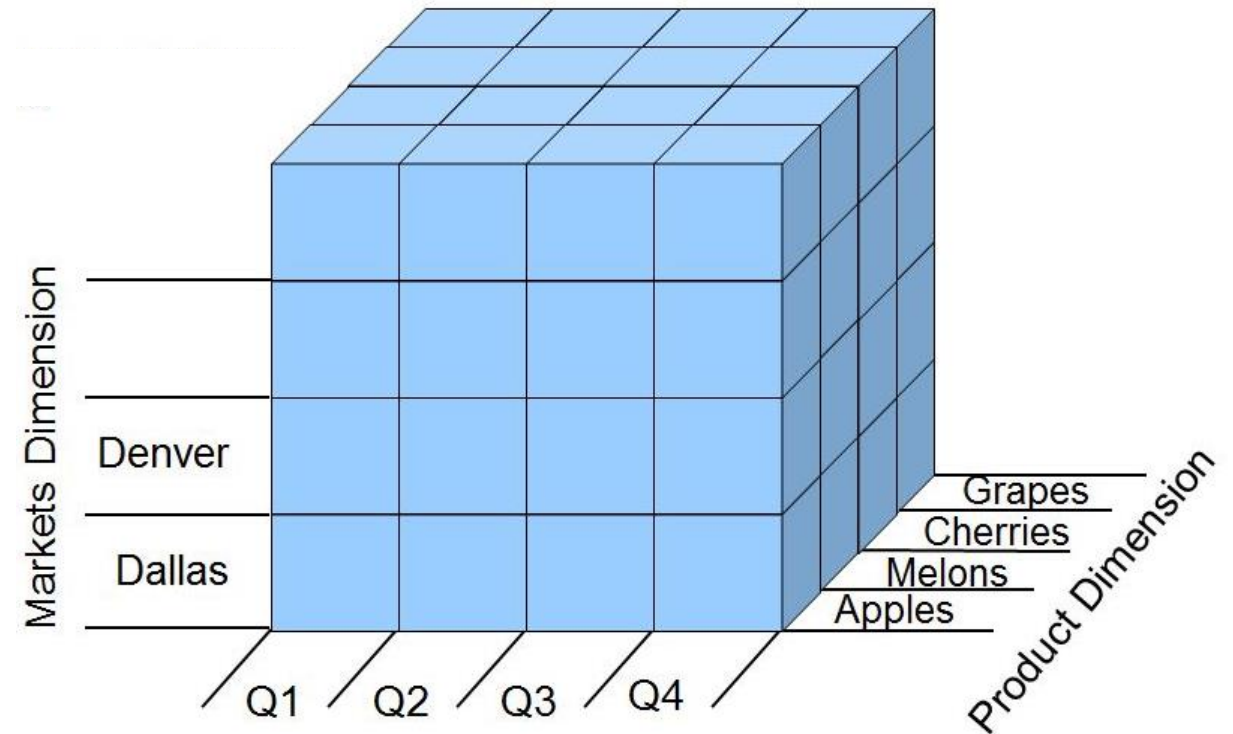
- OLAP data is multidimensional – organized in dimensions such as Customer, Product, Geography, and Time
- OLAP data represents metrics that need to be analyzed as measures at the intersection of the dimensions. For example, sales can be analyzed by Customer, Product, Geography and Time.
- Dimensions usually have associated with them hierarchies that specify aggregation levels and hence granularity of viewing data. Some of these hierarchies are Time organized by Year, Quarter, Month, Week and Day or Geography organized by State, County, and City or Product organized by Category, Sub-Category, etc.



OLAP DATA MODEL - RETAIL



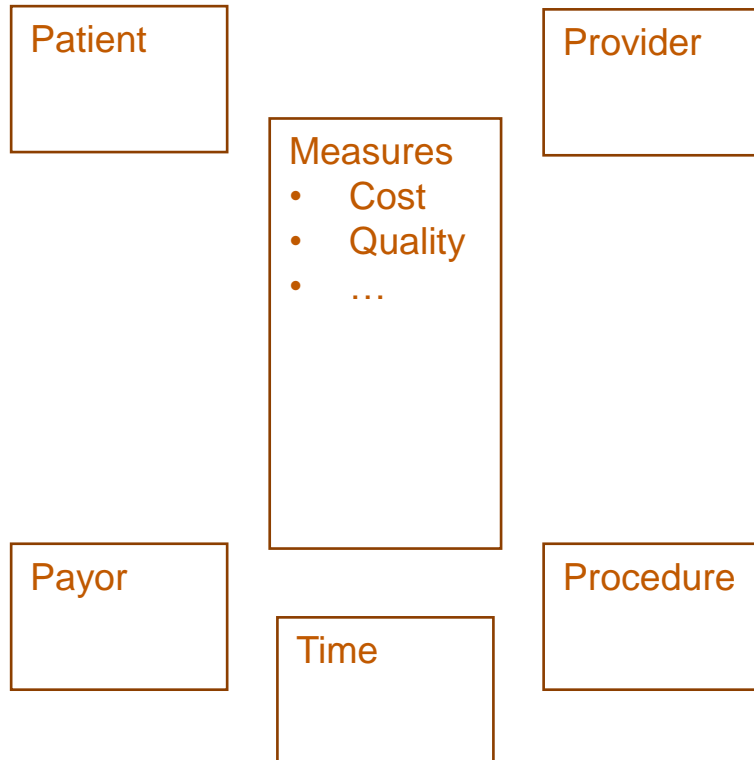
Star Schema



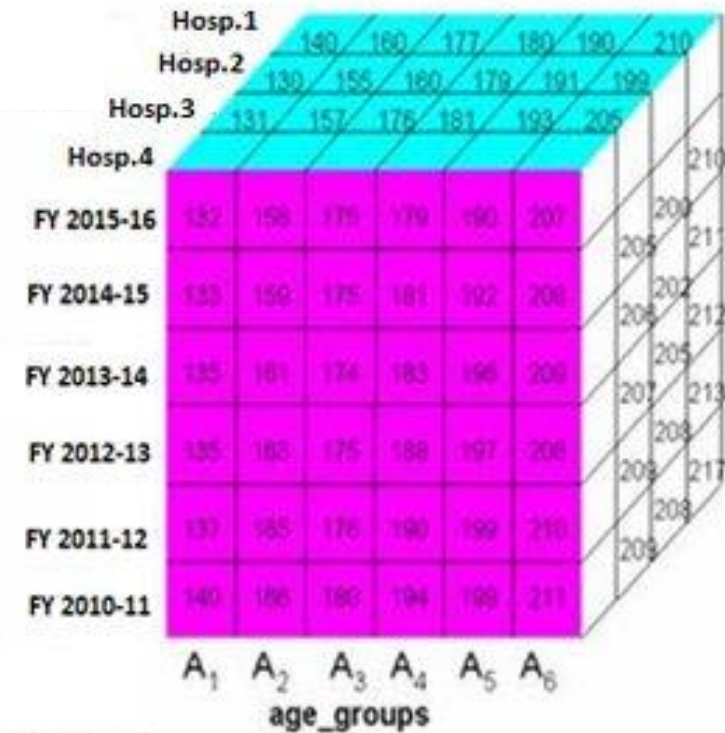
Cube



OLAP DATA MODEL – HEALTHCARE



A data cube for analyzing Hip-Fracture measure displays Count of patient visit



OLAP OPERATIONS

- **Roll up:** Summarize data (Total sales volume last year by product category by region.)
- **Drill down, Roll down:** Go from higher level summary to lower-level summary or detailed data (For a particular product category, find detailed sales data for each office by date.)
- **Slice and Dice:** Select and project (Sales of beverages in the west over the last 6 months.)
- **Pivot:** rotate the cube to show a particular face (Sales of beverages in the west over the last 6 months.)



TYPES OF OLAP

a. MOLAP – Multi-dimensional OLAP

- a. Dimension and aggregate hierarchies are pre-computed and stored
- b. Underlying database is optimized for MOLAP operations and deviates from relational database standards

b. ROLAP – Relational OLAP

- a. Dimensions and aggregate hierarchies are specific in metadata and computed on demand
- b. Underlying database is relational, and indexes are optimized for ROLAP

c. ROLAP vs. MOLAP

- a. Performance vs. flexibility
- b. Storage
- c. Implementation complexity



ENTERPRISE DATA ARCHITECTURE



COMPONENTS OF DATA MANAGEMENT

Dictated by first three components

Centralized and/or distributed?

Structured and/or unstructured?

Relational and/or non-relational

Type of unstructured data: text, images, audio, video, ...

Requirements for Reliability, Availability, Serviceability, Usability, Installability (RASUI)

Requirements for security, privacy and retention



Business Purpose of Data



Characteristics of Data under Management



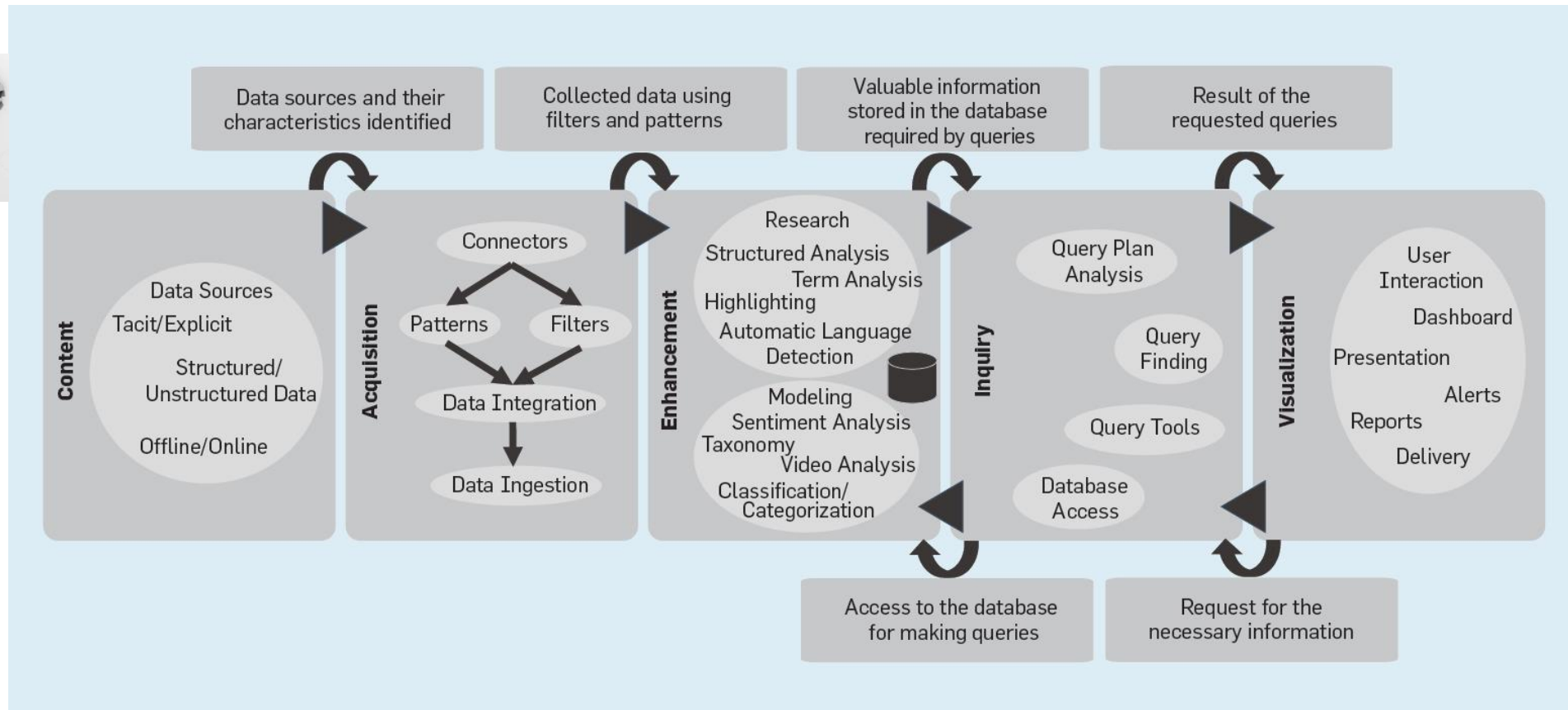
Data Management Operating Model



Data Management Technology Architecture



FRAMEWORK FOR IMPLEMENTING A BIG DATA ECOSYSTEM



Business Processes

External Data



IoT Sensors
On device file storage



The University of Texas at Austin
McCombs School of Business

CURRENT DATA ARCHITECTURE IMPEDES ...

- Digitization
- Competing with disruptive digital business models
- Agile response to changing customer needs
- Business value from technology investments
- Recruiting dynamic talent with modern skill sets
- Enterprise-wide innovation





QUESTION

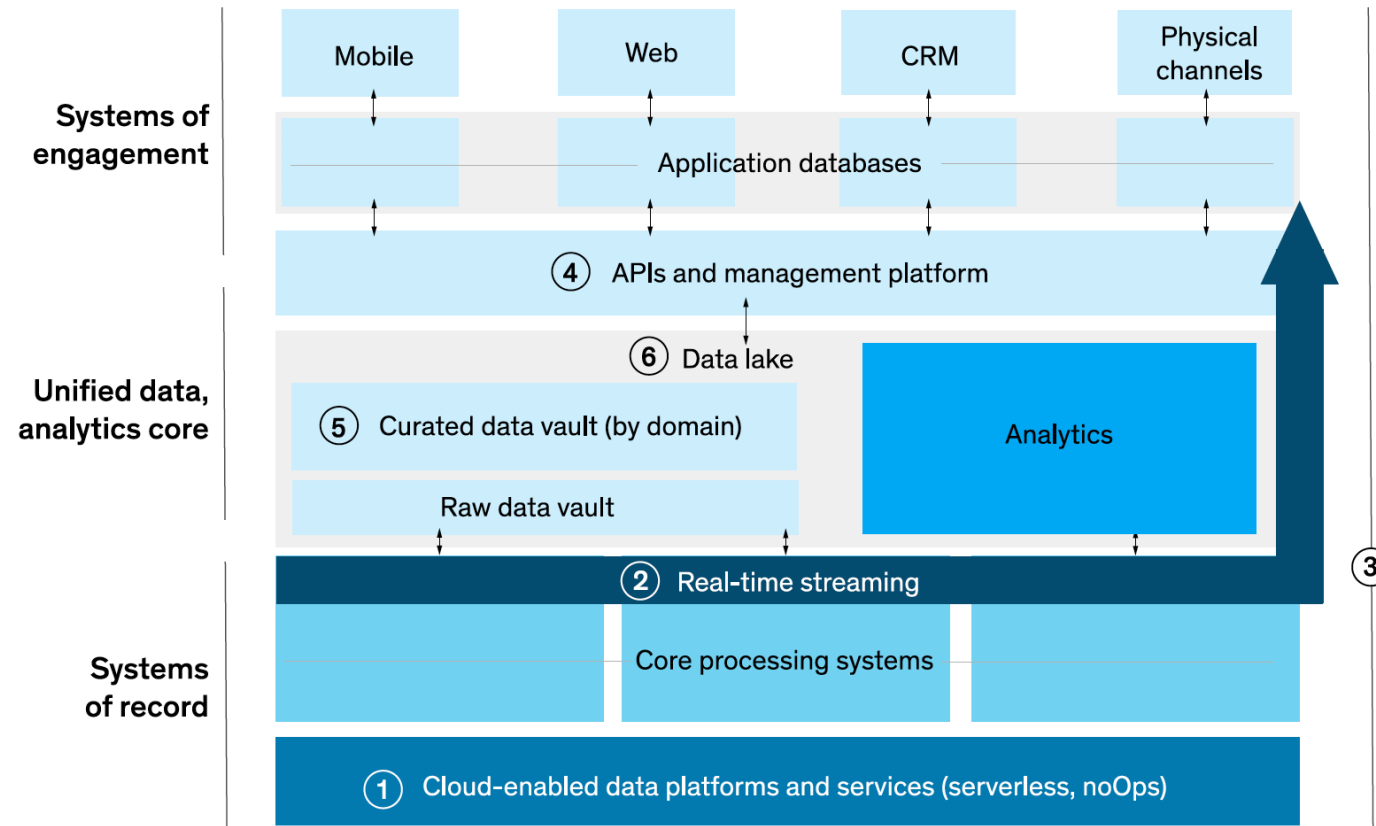
How should companies respond to these changes
in the shift of data storage and processing?

SIX TECHNOLOGY SHIFTS

1. From on-premise to cloud-based data systems
2. From batch to real-time data processing
3. From monolithic commercial solutions to modular, best-of-breed platforms
4. From point-to-point to decoupled data access
5. From an enterprise warehouse to domain-based architecture
6. From rigid, fixed data models to flexible, extensible data schemas



UPGRADED DATA ARCHITECTURE





QUESTION

What is different about the upgraded architecture?

WHAT IS DIFFERENT ABOUT THE UPGRADED ARCHITECTURE?

- Analytical data is updated more frequently
- Analytical data supports channels of engagement – so had to deal with transaction type workloads
- Streaming data bypasses analytical data stores and feeds channels of engagement
- "Models" need to be updated, deployed and monitored for performance continuously
- Systems of record are gradually modularized
- Data pipelines are intelligent and more complex



SIX TECHNOLOGY SHIFTS

1

From on-premise to cloud-based data systems

- Serverless data platforms such as Amazon S3 – data centric applications that can scale – can use SQL like interfaces against data in files
- Containerized data solutions such as those using Kubernetes – modularize applications, decouple software environment from hardware



SIX TECHNOLOGY SHIFTS

2

From batch to real-time
data processing

- Messaging platforms such as Kafka – scalable, durable, fault-tolerant publish/subscribe services
- Streaming processing and analytics solutions such as from Apache – Kafka Streaming, Flume, Storm, and Spark Streaming – direct analysis of messages in real time
- Alerting platforms such as Graphite or Splunk



SIX TECHNOLOGY SHIFTS

3

From monolithic commercial solutions to modular, best-of-breed platforms

- Data pipeline and API-based interfaces simplify integration between disparate tools and platforms – data abstraction
- Analytics workbenches such as Amazon Sagemaker and Kubeflow simplify building end-to-end solutions in a highly modular architecture



SIX TECHNOLOGY SHIFTS

4

From point-to-point to
decoupled data access

- An API management platform –API gateway –create and publish data-centric APIs including usage policies, control access, and metering for usage and performance
- A data platform to “buffer” transactions outside of core systems



SIX TECHNOLOGY SHIFTS

5

From an enterprise warehouse to domain-based architecture

- Data infrastructure as a platform – common tools and capabilities for storage and management – data producers can build data-asset platforms
- Data virtualization techniques to organize access and integrate distributed data assets.
- Data cataloging tools – search and exploration without access



SIX TECHNOLOGY SHIFTS

6

From rigid, fixed data models to flexible, extensible data schemas

- Data vault 2.0 techniques, such as data-point modeling result in extensibility
- Graph and other NoSQL databases result in flexibility
- Query file-based data akin to relational databases (e.g., Azure Synapse Analytics, Amazon S3 Glacier Select)
- Using JavaScript Object Notation (JSON) to store information



LOOKING FORWARD

- **Read:** Why Big Data Isn't Enough (see HBSP package)
- **Homework 5**
- **Exam 2**
- **NoSQL, Big Data**
- **Final Project**



THANK YOU

PART 2

Case analysis, NoSQL

CASE ANALYSIS

How to prepare a case write-up

CASE METHOD

- **A case is written to be discussed**
 - Read it, analyze, form opinions, a plan of action you can defend
 - There are no single “right answer”
- **Effective listening is a rare art**
 - “Discussion” means building on the comments of others
 - You might disagree; say so (respectfully)
- **To fulfill your part, you need to be prepared for class**
 - Do the reading before you come to class
 - Don’t just read, analyze the data
- **Participate early: the longer you wait, the harder it will be to join**



WHAT SHOULD BE IN A CASE WRITE-UP?

- **Synopsis (first paragraph, max 2)**
 - Show me that you read and understood the case
- **Answers to the case questions (up to 3 pages)**
 - Make sure you answered them all
 - Stay out of the vague zone (avoid “one would say...”)
 - Be direct and clear, use the given data
 - Analyze the problem from different perspectives
- **Conclusion (last paragraph)**
 - Give me your opinions / view



WHAT SHOULD NOT BE IN A CASE WRITE-UP?

- **Writing and grammar issues**
 - You can lose up to 50% of case grade upon writing issues
- **Format problems**
 - There is a reason for all format requirements
- **Repetition**
 - You have only three valuable pages
 - Your biggest problem should be squeezing all into three
- **Speculation (especially, any information conflicting with the case)**





OKAY, LET'S RETURN TO DBs

In database design, can you think of a solution better than the complex architecture of tables?

DEFINITION: NoSQL

- NoSQL databases (aka "not only SQL") are non-tabular databases and store data differently than relational tables
- NoSQL databases come in a variety of types based on their data model
- The main types are key-value, wide-column, document, and graph
- They provide flexible schemas and scale easily with large amounts of data and high user loads



STORING DATA WITH NoSQL

- NoSQL storage systems are used for files or semi-structured tables
- NoSQL does not use relational tables or other DB schemas
- NoSQL storage is specific to the NoSQL DB type
- Data type often drives the selection of NoSQL database













WARNING

- **NoSQL doesn't mean it's easier**
- It just means you don't use:
 - Relational tables, SQL syntax, and joins
 - STAR schemas, transactions, aggregations to optimize
- There might be a steep learning curve, especially if you're well adjusted to ANSI syntax and SQL (success takes time)



NoSQL DATABASES

Representation	Illustrative Products		
Key-value pair	 redis	 riak	
Column family	 cassandra		 amazon DynamoDB
Document	 CouchDB relax	 mongoDB	 IBM Cloudant®
Graph	 neo4j	 MarkLogic™	



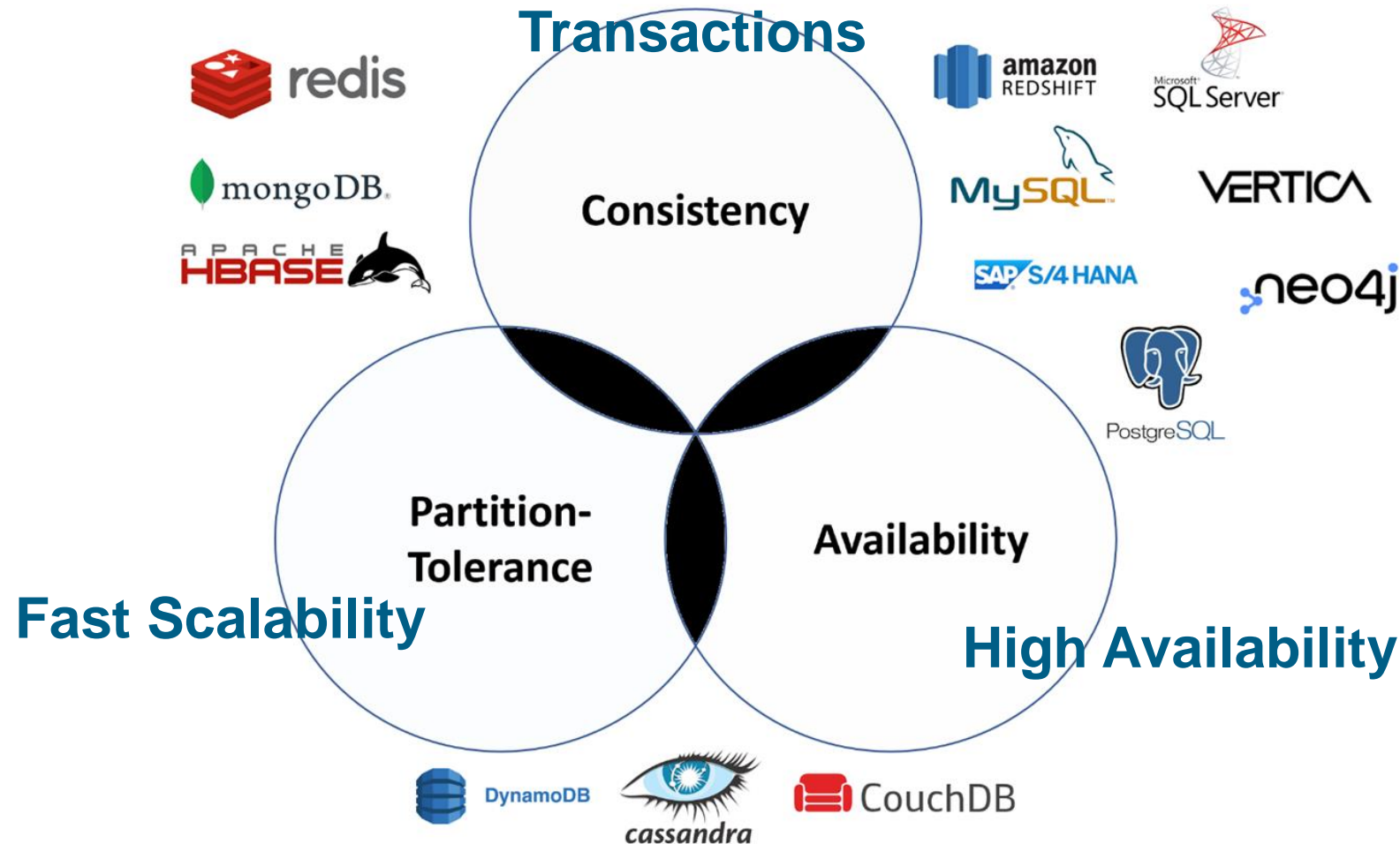
QUESTION

What would be a dream database?

- Fast?
- Consistent?
- Available?
- Something else?

CAP THEOREM

Exception: Google Cloud Spanner
<https://research.google/pubs/pub45855/>





REVIEW QUESTION

What is the difference between
SQL and NoSQL databases?



QUESTION

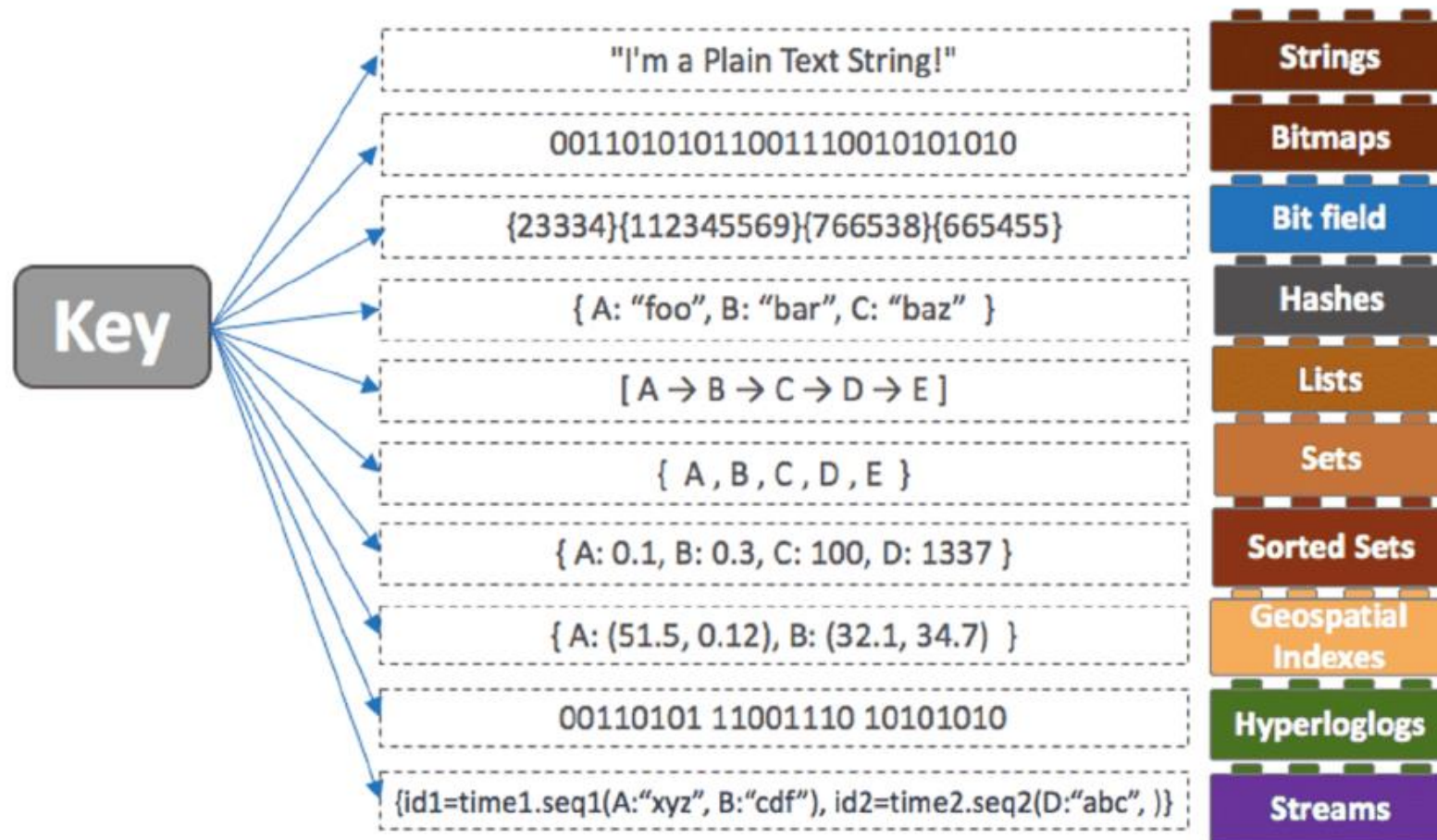
What is the difference between
data warehouse and data lake?

KEY-VALUE

- Store huge lists with NoSQL (redis)
- Lists, or dictionaries, are stored in memory
- Used to store HOT datasets (frequently accessed) often used for caching
- Extremely fast, and scalable
 - But not transactionally consistent, by definition



K/V: REDIS DATA STRUCTURES



**Very different
than RDBMS
data types!**

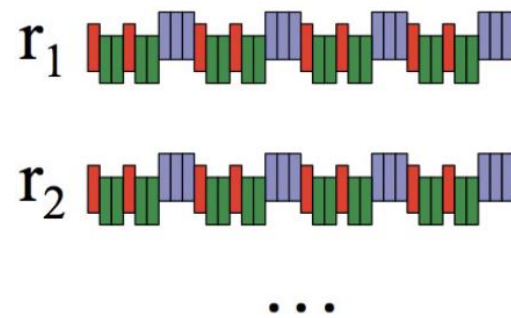


COLUMN FAMILY

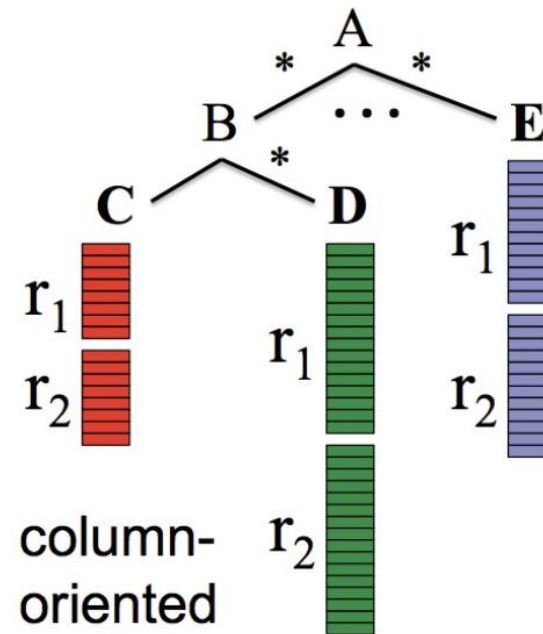
- Behavioral events are stored in irregular tables
- Used to store ragged datasets, often used for aggregation
- Extremely fast for aggregate queries and easily scalable
 - They're not going to be transactionally consistent by default, or even sometimes it's not even possible



COLUMNAR STORAGE: GCP - GOOGLE CLOUD PLATFORM (DREMEL)



record-
oriented



column-
oriented



“It all started in 2004 when Amazon was running Oracle's enterprise edition with clustering and replication. We had an advanced team of database administrators and access to top experts within Oracle. We were pushing the limits of what was a leading commercial database at the time and were unable to sustain the availability, scalability and performance needs that our growing Amazon business demanded.”

- Verner Vogels / Amazon CTO

<https://www.techrepublic.com/article/how-amazon-helped-bring-nosql-to-the-enterprise-mainstream/>

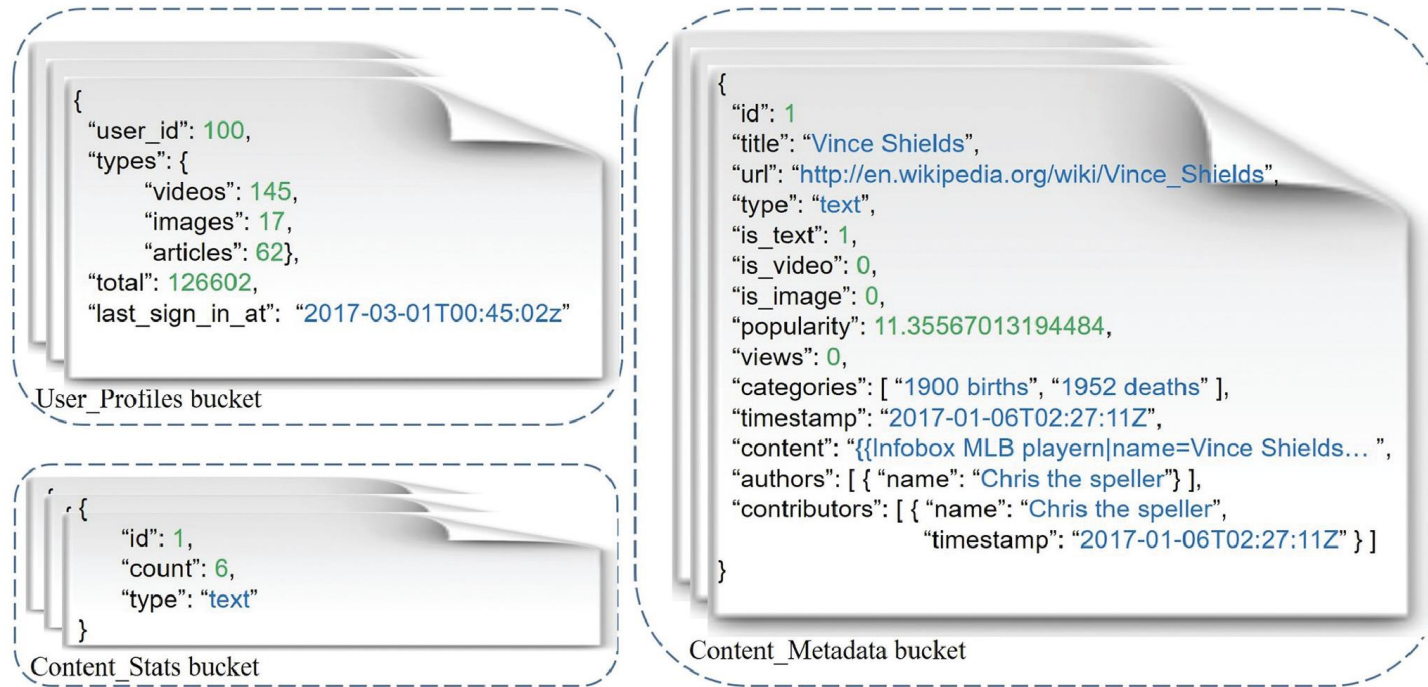


DOCUMENT

- Document stores, such as MongoDB, store customer activity
- These stores hold behavioral events and documents in irregular tables (a.k.a. rugged datasets)
- Extremely fast for queries and easily scalable
 - Not transactionally consistent by default



DOCUMENT



- Extension of key-value model where each value is a document
- Document can be formats such as XML, JSON or BSON
- Attributes in a document can be added/removed during runtime
- High fidelity to modern object-oriented programming paradigm



GRAPH

- Store data and relationships between the data items
- Nodes and edges are stored in graphs
- Extremely fast for relationship queries and easily scalable
 - Guess what? Not transactionally consistent



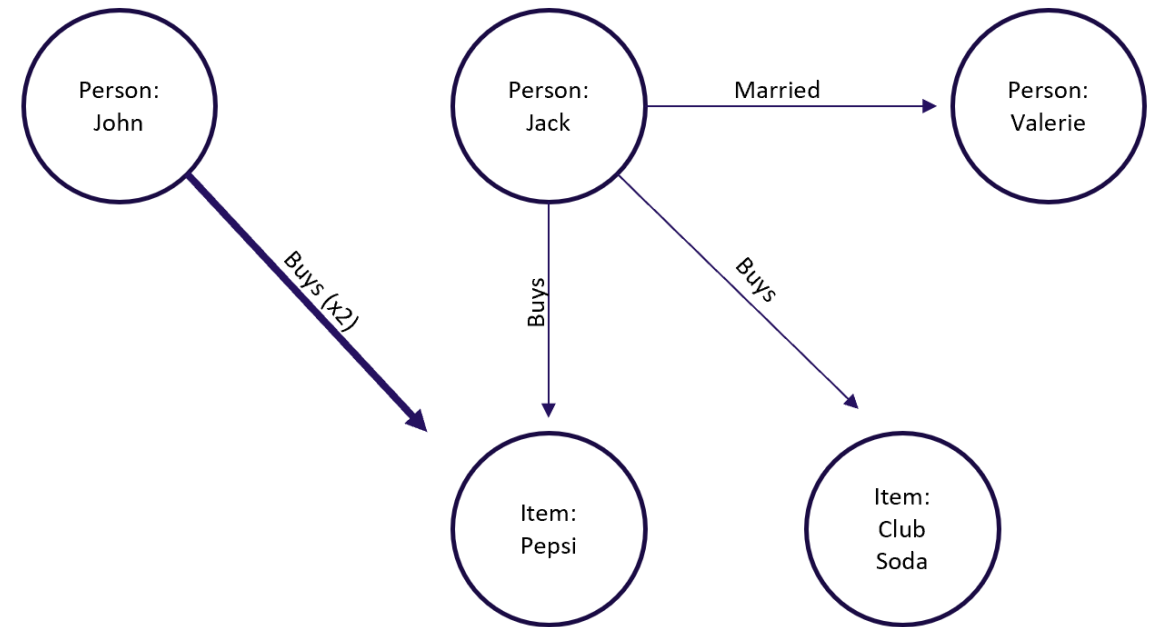
GRAPH

Sales		
Customer	Item	Time
0001	1A	20:34
0001	1A	21:15
0003	2A	21:16
0002	1A	21:16
0002	5C	

Inventory	
Description	SKU
Pepsi	1A
Club Soda	2A
.	.
.	.
Diet Coke	5C

Customer	
Name	CustID
John	0001
Jack	0002
Ted	0003
Ken	0004
Valerie	0005

Traditional database store data to efficiently store facts, but relationships must be rebuilt with JOINS and other inexact techniques.



Graph databases store both facts and the relationships between the facts, making certain types of analysis more intuitive.



“Every company has big data in its future, and every company will eventually be in the data business.”

- Thomas H. Davenport

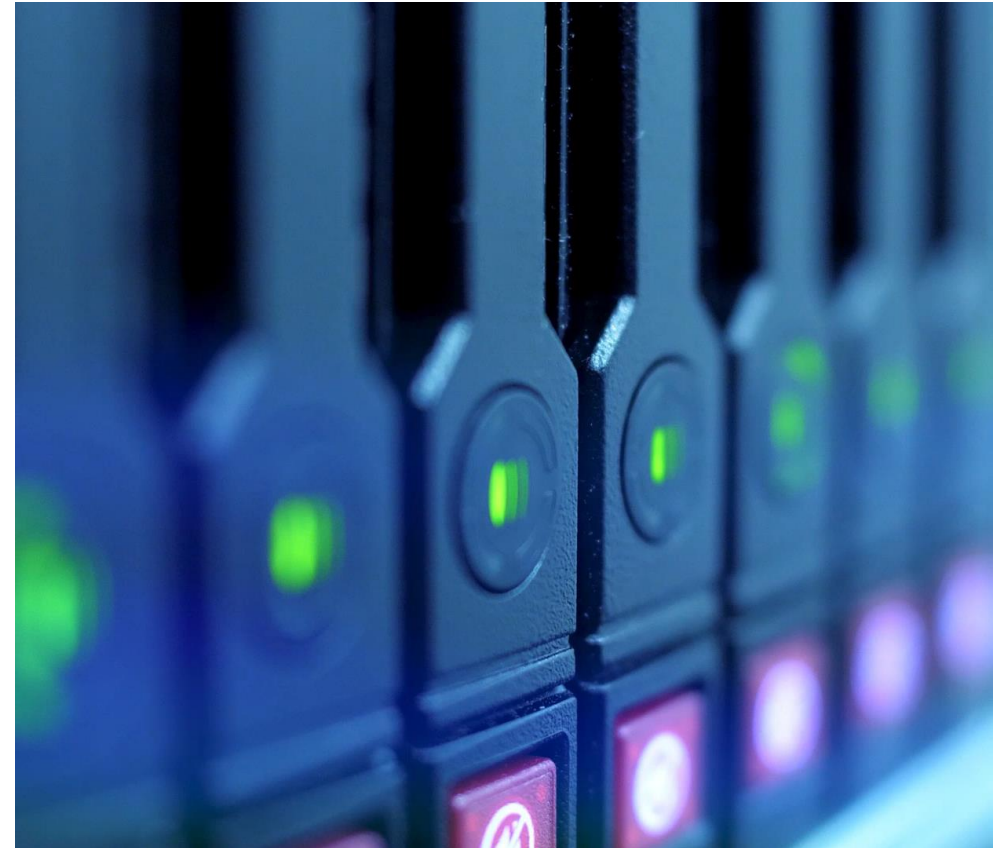
“Information is the oil of the 21st century, and analytics is the combustion engine.”

- Peter Sondergaard, Gartner



SPECIALTY NoSQL

- Top level trend: time-based events
- Real-time event databases: data is stored and made available in (near) real time
- Used to store event-based data
- Extremely fast for time-based queries and easily scalable



FUTURE TRENDS

- Consolidation of NoSQL market
 - Declining open source, mergers, adding features, and support need
- Multifunctionality: layers of features
 - Adding enterprise features (backup, monitoring, in-memory option)
- Eventually: one ecosystem for all (AWS? GCP? idk)



OPTIONAL

If you're interested in technical hands-on practice...

... or a new tool MongoDB



LINKS

TO LEARN: Learning MongoDB

- <https://hr.utexas.edu/learning-development/resources>

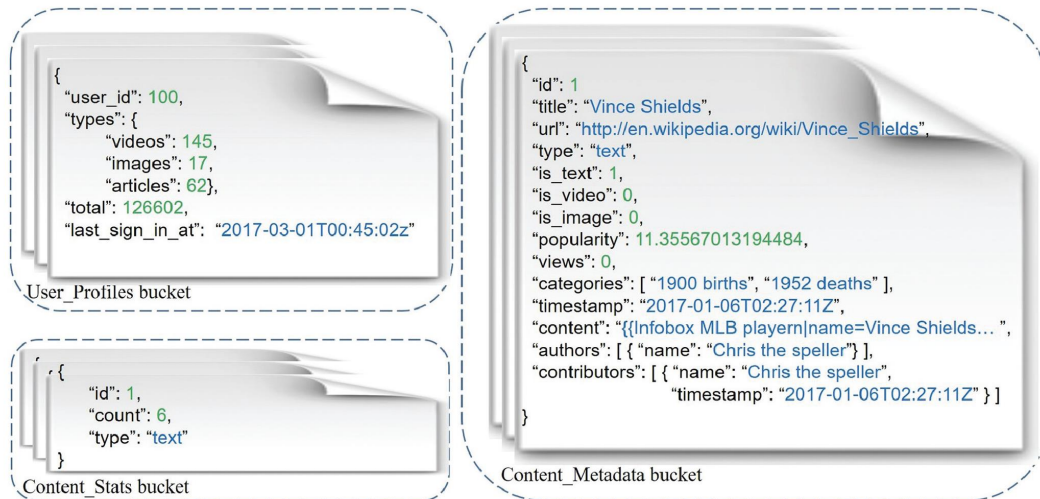
TO DOWNLOAD:

- <https://www.mongodb.com/try/download/community>
- Get the free (community) version



DOCUMENT

- Extension of key-value model where each value is a document
- Document can be formats such as XML, JSON or BSON
- Attributes in a document can be added/removed during runtime
- High fidelity to modern object-oriented programming paradigm
- Example applications: Content Management Systems, Blogging platforms
- Products: MongoDB, Amazon DynamoDB, Couchbase, Apache CouchDB, ArrangoDB



MONGODB VOCABULARY

Relational Database	MongoDB Equivalent
Database	Database
Table	Collection
Row	Document
Column	Field
Join	Embedded or Referenced Document
Primary Key	Default key_id provided by MongoDB



DOCUMENT EXAMPLE

```
{
  "business_id": "rncjoVoEFUJGCUoC1JgnUA",
  "open": true,
  "categories": ["Accountants", "Professional Services", "Tax Services",],
  "address": {
    "street": "123 Main Street",
    "city": "Peoria",
    "state": "IL",
    "zip": 14567 }
  "reviews": ["ab123xxyy", "ab124xxyy"],
  "name": "Peoria Income Tax Service",
  "neighborhoods": [],
  "type": "business",
  "phone": {
    { "type": "mobile",
      "number": 6305551212 }
    { "type": "fax",
      "number": 6305551213 }
  }
}
```

- Maximum size of a single document is 16MB
- Data stored as BSON (Binary JSON)
- For larger documents use GridFS



DOCUMENT EXAMPLE

```
{
  "business_id": "rncjoVoEFUJGCUoC1JgnUA",
  "open": true,
  "categories": ["Accountants", "Professional Services", "Tax S
  " address": {
    "street": "123 Main Street",
    "city": "Peoria",
    "state": "IL",
    "zip": 14567 }
  "reviews": ["ab123xxyy", "ab124xxyy"],
  "name": "Peoria Income Tax Service",
  "neighborhoods": [],
  "type": "business",
  "phone": {
    { "type": "mobile",
      "number": 6305551212 }
    { "type": "fax",
      "number": 6305551213 }
  }
}
```

- `_id` is a special column in each document
- Unique within each collection
- Equivalent to a Primary Key in RDBMS
- `_id` is 12 Bytes, you can set it yourself
Or:
1st 4 bytes → timestamp
Next 3 bytes → machine id
Next 2 bytes → Process id
Last 3 bytes → incremental values



DOCUMENT EXAMPLE

```
{
  "business_id": "rncjoVoEFUJGCUoC1JgnUA",
  "open": true,
  "categories": ["Accountants", "Professional Services", "Tax Se
    " address": {
      "street": "123 Main Street",
      "city": "Peoria",
      "state": "IL",
      "zip": 14567 }
  "reviews": ["ab123xxyy", "ab124xxyy"],
  "name": "Peoria Income Tax Service",
  "neighborhoods": [],
  "type": "business",
  "phone": {
    { "type": "mobile",
      "number": 6305551212 }
    { "type": "fax",
      "number": 6305551213 }
  }
}
```

- Field Name and Field Value
- Data type can be Integer, Boolean, String, etc.
- There is no defined schema (all fields do not have to be present in every document in a collection)
- The field names cannot start with the \$ character
- The field names cannot contain the . character



DOCUMENT EXAMPLE

```
{
  "business_id": "rncjoVoEFUJGCUoC1JgnUA",
  "open": true,
  "categories": ["Accountants", "Professional Services", "Tax Services".],
  "address": {
    "street": "123 Main Street",
    "city": "Peoria",
    "state": "IL",
    "zip": 14567 }
  "reviews": ["ab123xxyy", "ab124xxyy"],
  "name": "Peoria Income Tax Service",
  "neighborhoods": [],
  "type": "business",
  "phone": {
    { "type": "mobile",
      "number": 6305551212 }
    { "type": "fax",
      "number": 6305551213 }
  }
}
```

- Embedded document
- Joins not needed
- One-to-one relationship?



DOCUMENT EXAMPLE

```
{
  "business_id": "rncjoVoEFUJGCUoC1JgnUA",
  "open": true,
  "categories": ["Accountants", "Professional Services", "Tax Services",],
  "address": {
    "street": "123 Main Street",
    "city": "Peoria",
    "state": "IL",
    "zip": 14567 }
  "reviews": ["ab123xxyy", "ab124xxyy"],
  "name": "Peoria Income Tax Service",
  "neighborhoods": [],
  "type": "business",
  "phone": {
    { "type": "mobile",
      "number": 6305551212 }
    { "type": "fax",
      "number": 6305551213 }
  }
}
```

- Nested document
- Joins not needed
- One-to-many relationship?



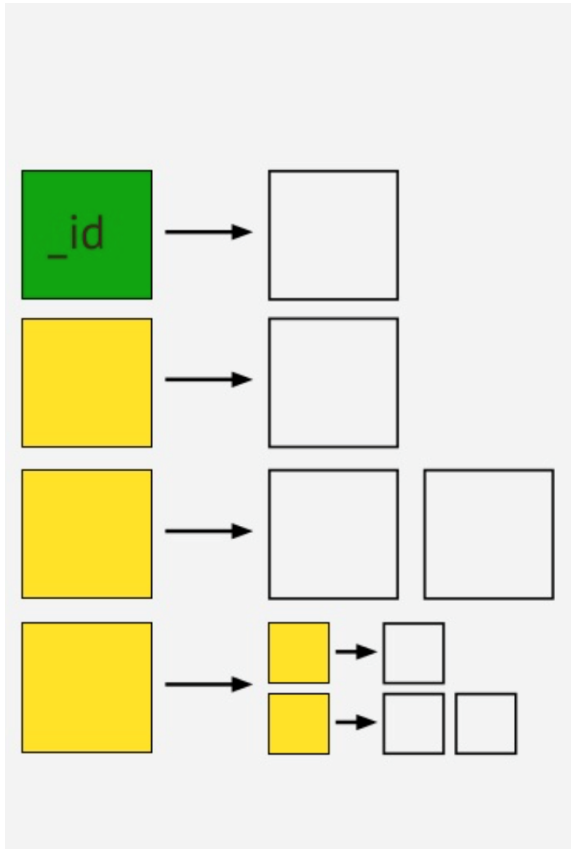
DOCUMENT EXAMPLE

```
{
  "business_id": "rncjoVoEFUJGCUoC1JgnUA",
  "open": true,
  "categories": ["Accountants", "Professional Services", "Tax Services",],
  "address": {
    "street": "123 Main Street",
    "city": "Peoria",
    "state": "IL",
    "zip": 14567 }
  "reviews": ["ab123xyy", "ab124xyy"],
  "name": "Peoria Income Tax Service",
  "neighborhoods": [],
  "type": "business",
  "phone": {
    { "type": "mobile",
      "number": 6305551212 }
    { "type": "fax",
      "number": 6305551213 }
  }
}
```

- Document referencing
- Joins not needed
- Many-to-many relationship?
- Array of JSON objects



MONGODB DATA MODEL SUMMARY



- N-dimensional storage
- Fields can contain many values, embedded values, nested values or referenced values
- Flexible schema
- Query on any field and level
- Optimal data locality requires fewer indexes and delivers better performance



RETRIEVING DATA IN MONGODB

SQL Statement	MongoDB commands
SELECT * FROM table	db.collection.find()
SELECT * FROM table WHERE artist = 'Nirvana'	db.collection.find({Artist:"Nirvana"})
SELECT* FROM table ORDER BY Title	db.collection.find().sort(Title:1)
DISTINCT	.distinct()
GROUP BY	.group()
>=, <	\$gte, \$lt



MONGODB CLOSING COMMENTS

- Data modeling skills are needed
- Decisions for document embedding, referencing, nesting are made based on an analysis and understanding of how the database will be used
- Workload analysis determines how data is distributed across nodes
- Mechanisms for some ACID requirements have been designed and implemented

LOOKING FORWARD

- Homework 5
- Exam 2
- NoSQL, Big Data
- Final Project



THANK YOU

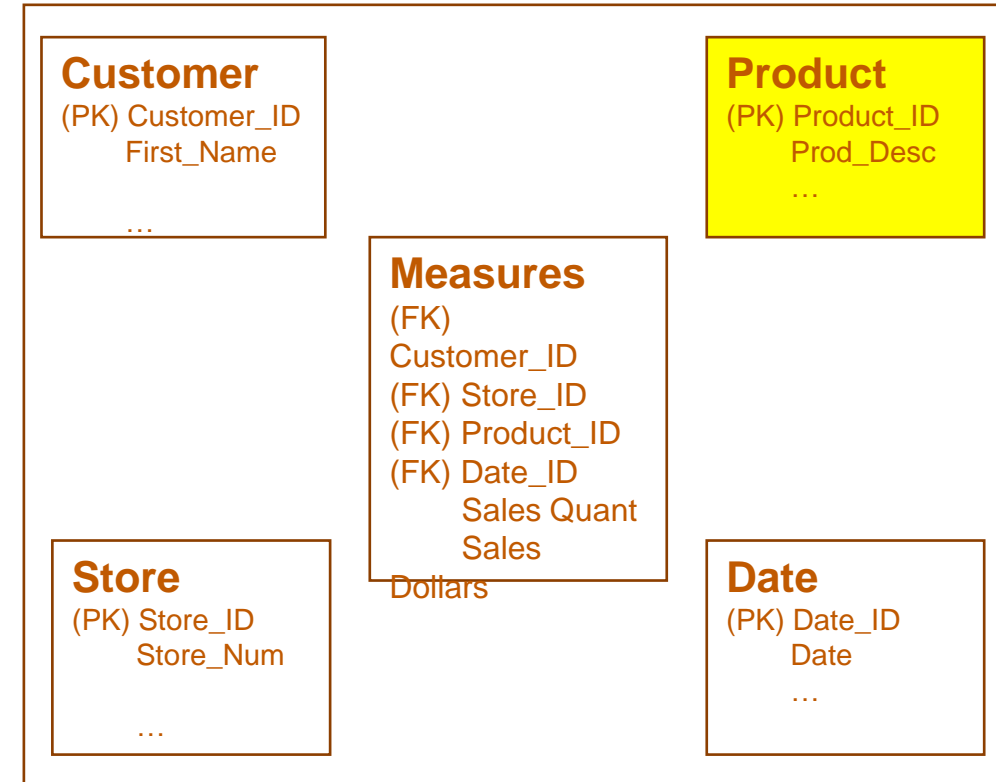
BACKUP SLIDES

PART 1

OLAP Details

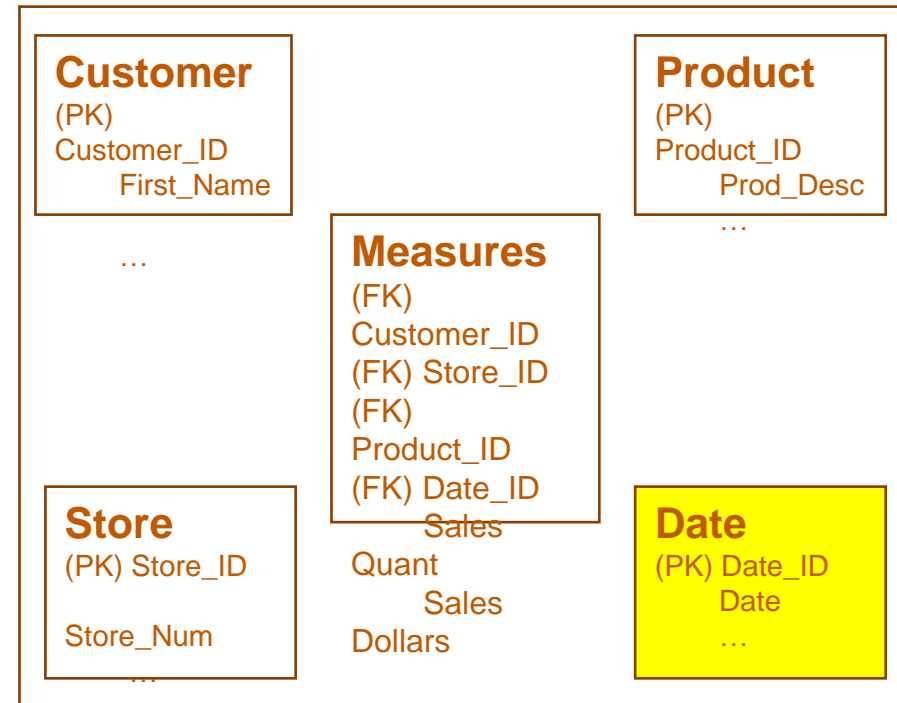
Multi-dimensional data in RDMS (Star Diagram Method)

Product_ID	Product_Description	Brand	Subcategory	Category	Department	Fat Content
1	Baked Well Light Sourdough Fresh Bread	Baked Well	Fresh	Bread	Bakery	Reduced
2	Fluffy Sliced Whole Wheat	Fluffy	Pre-Packaged	Bread	Bakery	Regular
3	Fluffy Light Sliced Whole Wheat	Fluffy	Pre-Packaged	Bread	Bakery	Reduced
4	Light Mini Cinnamon Rolls	Light	Pre-Packaged	Sweeten Bread	Bakery	Non-Fat
5	Diet Lovers Vanilla 2 Gallon	Coldpack	Ice Cream	Frozen Desserts	Frozen Foods	Non-Fat
6	Light and Creamy Butter Pecan 1 Pint	Freshlike	Ice Cream	Frozen Desserts	Frozen Foods	Reduced
7	Chocolate Lovers 1/2 Gallon	Frigid	Ice Cream	Frozen Desserts	Frozen Foods	Regular
8	Strawberry Ice Creamy 1 Pint	Icy	Ice Cream	Frozen Desserts	Frozen Foods	Regular
9	Icy Ice Cream Sandwiches	Icy	Novelties	Frozen Desserts	Frozen Foods	Regular

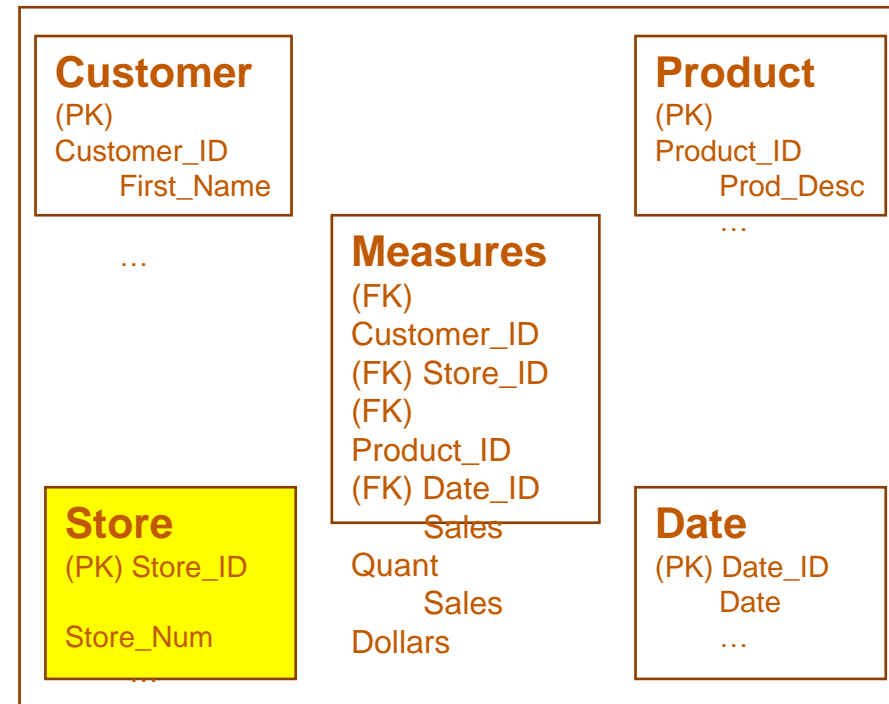


Multi-dimensional data in RDMS (Star Diagram Method)

Date_ID	Date	Full_Date	Day of Week	Cal_Month	Cal_Quarter	Cal_Year	Holiday_Flag	Weekday_Flag
20190101	1/1/2019	January 1, 2019	Tuesday	January	Q1	2019	Holiday	Weekday
20190102	1/2/2019	January 2, 2019	Wednesday	February	Q2	2020	Non-Holiday	Weekday
20190103	1/3/2019	January 3, 2019	Thursday	March	Q3	2021	Non-Holiday	Weekday
20190104	1/4/2019	January 4, 2019	Friday	April	Q4	2022	Non-Holiday	Weekday
20190105	1/5/2019	January 5, 2019	Saturday	May	Q1	2023	Non-Holiday	Weekend
20190106	1/6/2019	January 6, 2019	Sunday	June	Q2	2024	Non-Holiday	Weekend
20190107	1/7/2019	January 7, 2019	Monday	July	Q3	2025	Non-Holiday	Weekday
20190108	1/8/2019	January 8, 2019	Tuesday	August	Q4	2026	Non-Holiday	Weekday



Multi-dimensional data in RDMS (Star Diagram Method)

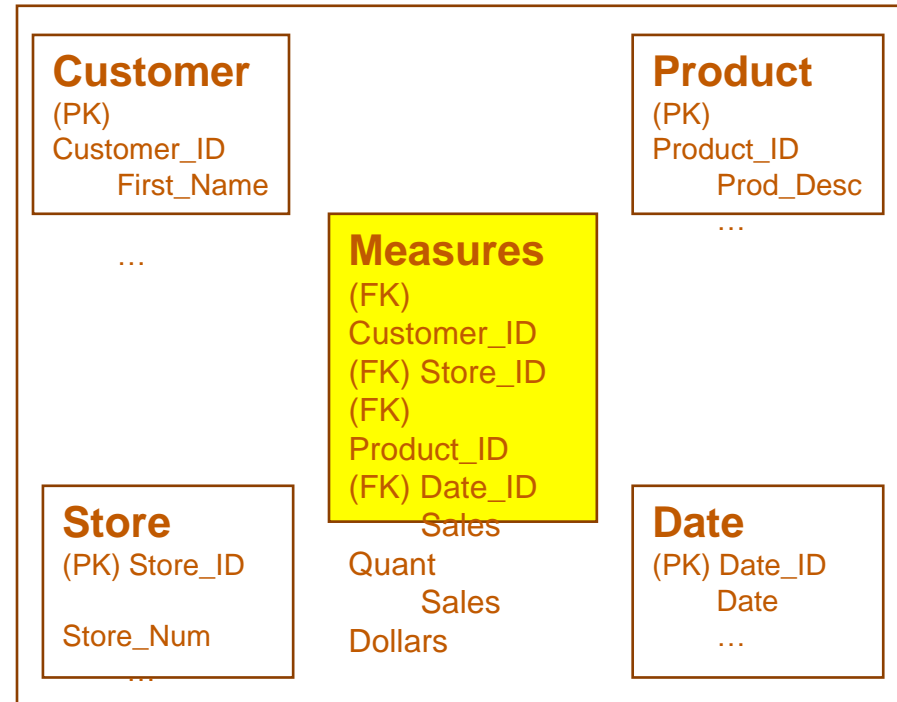
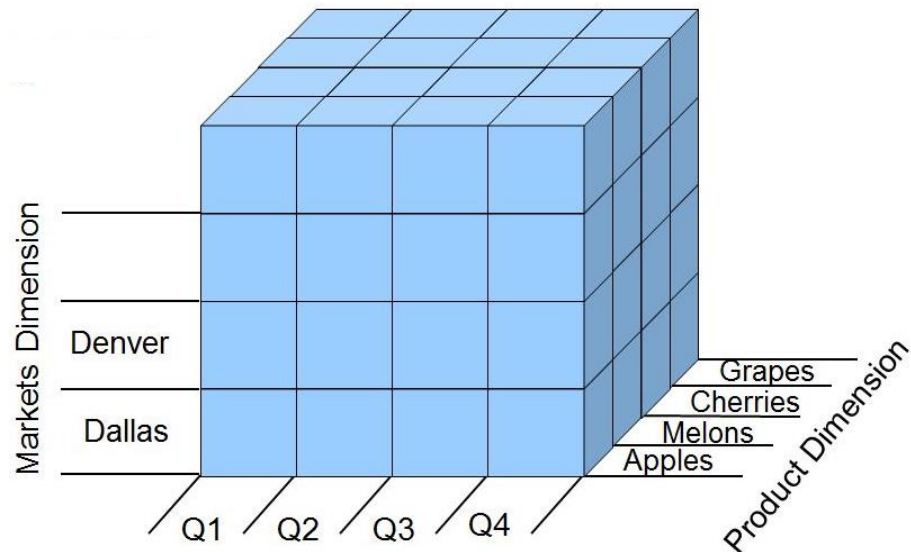


Store_ID	Store_Number	Store_Name	Address	City	State	County	Zip_Code	Manager	District	Region	Store_Type	Last_Remodel_Date
1	4950	Central Flagship	2100 N. Lamar Blvd	Austin	TX	Travis	78745	John Walters	Central Texas	South	Flagship	2008
2	3921	Austin South	830 South Congress Ave	Austin	TX	Travis	78704	Clarisse Lamar	Central Texas	South	Standard	2000
3	4923	SA South	1826 Easy St	San Antonio	TX	Bexar	78006	Tina Nguyen	South Texas	South	Standard	2015
4	5698	Financial District NYC	160 Fulton St	New York	NY	New York	10007	Zach Norton	NYC	East	Boutique	2018



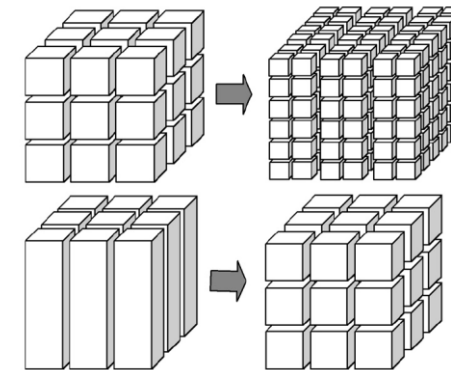
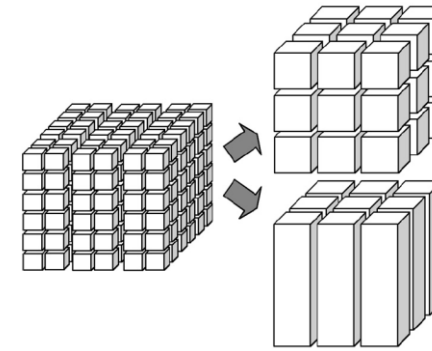
Multi-dimensional data in RDMS (Star Diagram Method)

Product_ID (FK)	Store_ID (FK)	Date_ID (FK)	Sales Quantity	Sales Dollar Value
1	1	1	2	20
1	2	1	3	15
2	1	2	2	35
2	3	3	5	50



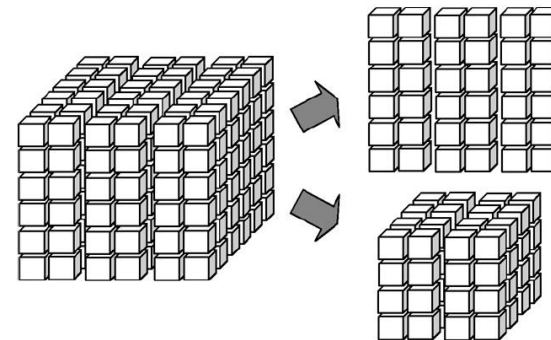
OLAP Operations

- **Roll up:** Summarize data
(e.g. Total sales by *year/quarter* by product category by *region*.)
- **Drill down, Roll down:** Go from higher level summary to lower level detailed data
(e.g. See detailed sales for a single *product* or *city*)




OLAP Operations

- **Slice and Dice:** Select a whole dimension or combo of 2
e.g. Slice - Sales of a single product across all time and geographic dimensions
e.g. Dice – Sales of single product, region, and time window



- **Pivot:** rotate the **cube** to show a particular face
(e.g Both show sales by Category/Year)

Category	Year	Metrics	Revenue
Books	2005		\$1,320,585
	2006		\$1,563,287
Electronics	2005		\$19,299,870
	2006		\$23,654,030
Movies	2005		\$1,032,391
	2006		\$1,333,126
Music	2005		\$748,966
	2006		\$940,136



Category	Year	Metrics	Revenue
		2005	2006
Books		\$1,320,585	\$1,563,287
Electronics		\$19,299,870	\$23,654,030
Movies		\$1,032,391	\$1,333,126
Music		\$748,966	\$940,136

PART 2

MongoDB Reference

Information Sources

- MongoDB manual
<https://docs.mongodb.com/master/>
- MongoDB sample datasets
<https://docs.atlas.mongodb.com/sample-data/available-sample-datasets/>



Create Read Update Delete (CRUD)

- **Create**

```
db.collection.insert( <document> )
```

```
db.collection.save( <document> )
```

```
db.collection.update( <query>, <update>, { upsert: true } )
```

- **Read**

```
db.collection.find( <query>, <projection> )
```

```
db.collection.findOne( <query>, <projection> )
```

- **Update**

```
db.collection.update( <query>, <update>, <options> )
```

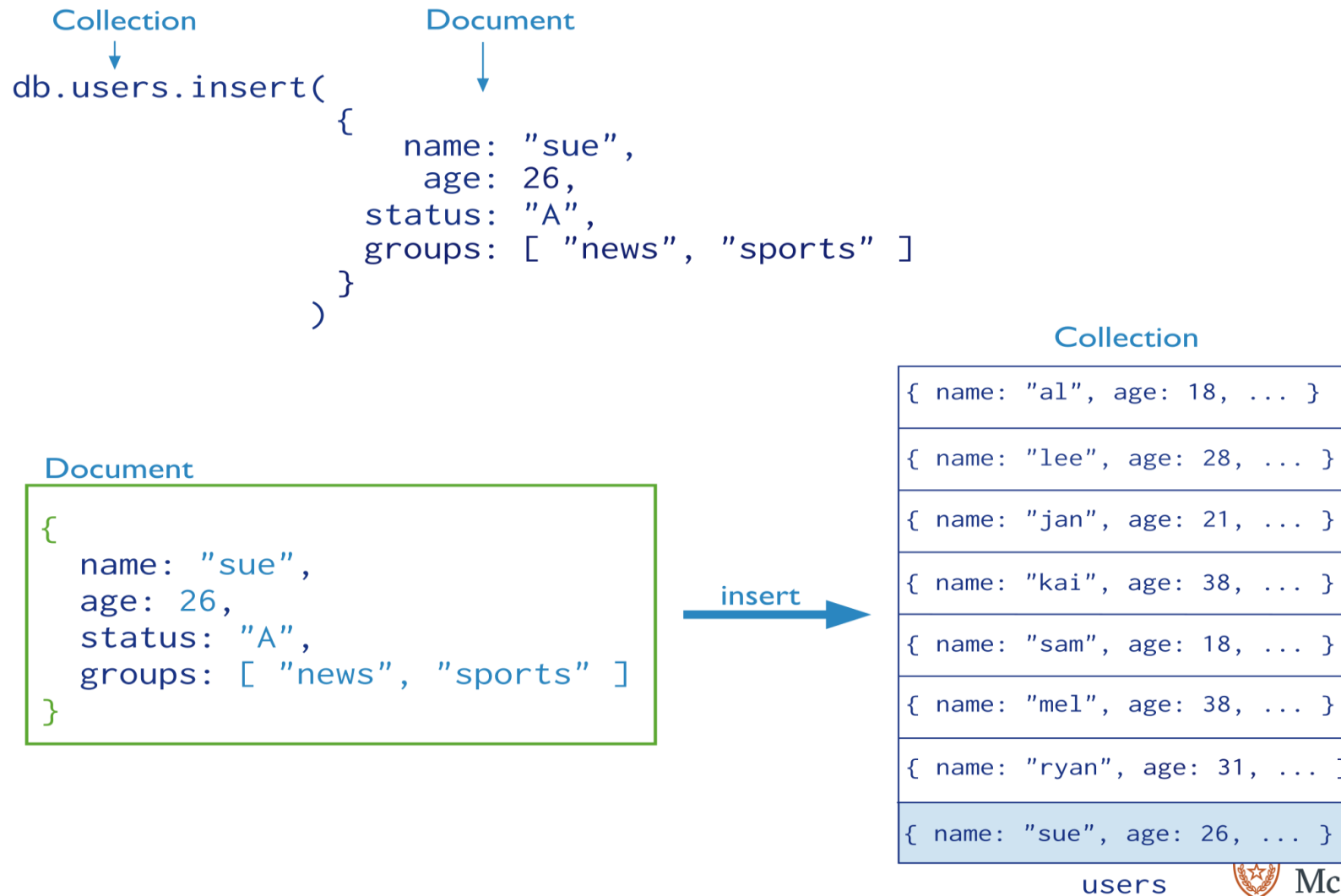
- **Delete**

```
db.collection.remove( <query>, <justOne> )
```



Insertion

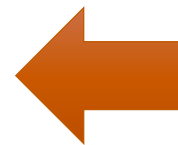
- The collection “users” is created automatically if it does not exist



Multi-Document Insertion (Use of Arrays)

```
var mydocuments =  
[  
  {  
    item: "ABC2",  
    details: { model: "14Q3", manufacturer: "M1 Corporation" },  
    stock: [ { size: "M", qty: 50 } ],  
    category: "clothing"  
  },  
  {  
    item: "MNO2",  
    details: { model: "14Q3", manufacturer: "ABC Company" },  
    stock: [ { size: "S", qty: 5 }, { size: "M", qty: 5 }, { size: "L", qty: 1 } ],  
    category: "clothing"  
  },  
  {  
    item: "IJK2",  
    details: { model: "14Q2", manufacturer: "M5 Corporation" },  
    stock: [ { size: "S", qty: 5 }, { size: "L", qty: 1 } ],  
    category: "houseware"  
  }  
];
```

```
db.inventory.insert( mydocuments );
```



All the documents are
inserted at once



Multi-Document Insertion (Bulk Operation)

- A temporary object in memory holds the insertions and uploads them at once

There is also *Bulk Ordered* object

```
var bulk = db.inventory.initializeUnorderedBulkOp();
bulk.insert(
  {
    item: "BE10",
    details: { model: "14Q2", manufacturer: "XYZ Company" },
    stock: [ { size: "L", qty: 5 } ],
    category: "clothing"
  }
);
bulk.insert(
  {
    item: "ZYT1",
    details: { model: "14Q1", manufacturer: "ABC Company" },
    stock: [ { size: "S", qty: 5 }, { size: "M", qty: 5 } ],
    category: "houseware"
  }
);
bulk.execute();
```

_id column is added automatically



Delete (Remove Operation)

- A condition can be put on any field in the document (even `_id`)

```
db.users.remove(  
  { status: "D" }  
)
```

← collection
← remove criteria

The following diagram shows the same query in SQL:

```
DELETE FROM users  
WHERE status = 'D'
```

← table
← delete criteria

`db.users.remove ()`



Removes all documents from *users* collection



update

```
db.users.update(  
  { age: { $gt: 18 } },  
  { $set: { status: "A" } },  
  { multi: true }  
)
```

← collection
← update criteria
← update action
← update option

Otherwise, only the 1st matching document will be updated



Update

Two
operators

```
db.inventory.update(  
  { item: "MNO2" },  
  {  
    $set: {  
      category: "apparel",  
      details: { model: "14Q3", manufacturer: "XYZ Company" }  
    },  
    $currentDate: { lastModified: true }  
  }  
)
```

For the document with item equal to "MNO2", use the \$set operator to update the category field and the details field to the specified values and the \$currentDate operator to update the field lastModified with the current date.



Replace a document

- For the document having item = "BE10", replace it with the given document

```
db.inventory.update(  
  { item: "BE10" },  
  {  
    item: "BE05",  
    stock: [ { size: "S", qty: 20 }, { size: "M", qty: 5 } ],  
    category: "apparel"  
  }  
)
```

Query Condition

New document



Insert or Replace

- If the document having item = "TBD1" is in the DB, it will be replaced, otherwise, it will be inserted

```
db.inventory.update(  
  { item: "TBD1" },  
  {  
    item: "TBD1",  
    details: { "model" : "14Q4", "manufacturer" : "ABC Company" },  
    stock: [ { "size" : "S", "qty" : 25 } ],  
    category: "houseware"  
  },  
  { upsert: true }  
)
```

