Data Management for Data Science

Lecture 3: Principles of Data Management

Prof. Asoc. Endri Raço

Today’s Lecture

1. Data Management

2. Data Models

3. RDBMs and the Relational Data Model

1. Data Management

Data Management

- Data represents the **traces** of real-world processes.

- Data is valuable **but** hard and costly to manage
  - Storage, representation complexity, collection

- Data management seeks to answer two questions:
  - What operations do we want to perform on this data?
  - What functionality do we need to manage this data?

Required Functionality

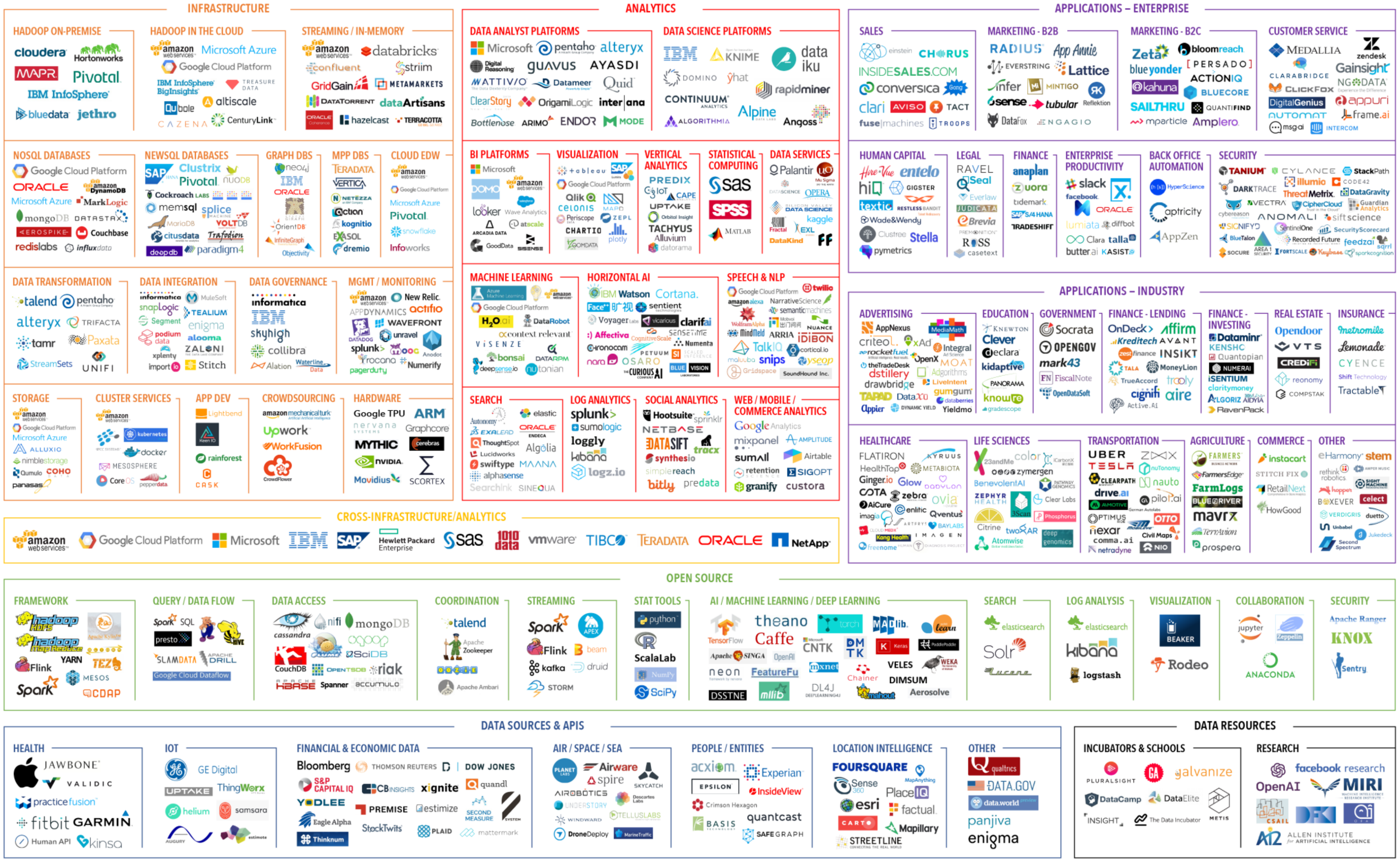
- Describe real-world entities in terms of stored data
- Create & persistently store large datasets
- Efficiently query & update
  - Must handle complex questions about the data
  - Must handle sophisticated updates
  - Performance matters
- Change structure (e.g., add attributes)
- Concurrency control: enable simultaneous queries, updates etc
- Crash recovery
- Access control, security, integrity

It is difficult and costly to implement all these features!

Systems providing data management features

- Relational database management systems
- HDFS-based systems (e.g., hadoop)
- Stream management systems: Apache Kafka
- Others?

BIG DATA LANDSCAPE 2017



INFRASTRUCTURE

HADOOP ON-PREMISE · HADOOP IN THE CLOUD · STREAMING / IN-MEMORY

NOSQL DATABASES · NEWSQL DATABASES · GRAPH DBS · MPP DBS · CLOUD EDW

DATA TRANSFORMATION · DATA INTEGRATION · DATA GOVERNANCE · MGMT / MONITORING

STORAGE · CLUSTER SERVICES · APP DEV · CROWDSOURCING · HARDWARE

ANALYTICS

DATA ANALYST PLATFORMS · DATA SCIENCE PLATFORMS

BI PLATFORMS · VISUALIZATION · VERTICAL ANALYTICS · STATISTICAL COMPUTING · DATA SERVICES

MACHINE LEARNING · HORIZONTAL AI · SPEECH & NLP

SEARCH · LOG ANALYTICS · SOCIAL ANALYTICS · WEB / MOBILE / COMMERCE ANALYTICS

APPLICATIONS – ENTERPRISE

SALES · MARKETING - B2B · MARKETING - B2C · CUSTOMER SERVICE

HUMAN CAPITAL · LEGAL · FINANCE · ENTERPRISE PRODUCTIVITY · BACK OFFICE AUTOMATION · SECURITY

APPLICATIONS – INDUSTRY

ADVERTISING · EDUCATION · GOVERNMENT · FINANCE - LENDING · FINANCE - INVESTING · REAL ESTATE · INSURANCE

HEALTHCARE · LIFE SCIENCES · TRANSPORTATION · AGRICULTURE · COMMERCE · OTHER

CROSS-INFRASTRUCTURE/ANALYTICS

OPEN SOURCE

FRAMEWORK · QUERY / DATA FLOW · DATA ACCESS · COORDINATION · STREAMING · STAT TOOLS · AI / MACHINE LEARNING / DEEP LEARNING · SEARCH · LOG ANALYSIS · VISUALIZATION · COLLABORATION · SECURITY

DATA SOURCES & APIS

HEALTH · IOT · FINANCIAL & ECONOMIC DATA · AIR / SPACE / SEA · PEOPLE / ENTITIES · LOCATION INTELLIGENCE · OTHER

DATA RESOURCES

INCUBATORS & SCHOOLS · RESEARCH

V2 – Last updated 5/3/2017

© Matt Turck (@mattturck), Jim Hao (@jimrhao), & FirstMark (@firstmarkcap) mattturck.com/bigdata2017

FIRSTMARK
EARLY STAGE VENTURE CAPITAL

2. Data Models

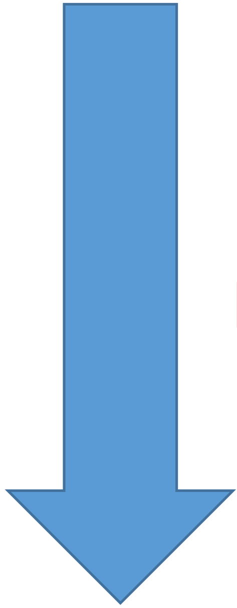
What you will learn about in this section

1. Types of Data

2. Data Models

Data is highly heterogeneous

- Structured data
- Semi-structured data
- Unstructured data



Increasing amounts of data

Structured data

- Information with a high degree of organization
- All data conforms to a schema. Ex: business data
- Easy to query, search over, aggregate
- Example: tables in a database, tables in excel, etc.

Semi-structured data

- Some structure in the data but implicit and irregular
- It contains tags or other markers to separate semantic elements and enforce hierarchies of records and fields within the data
- Example: JSON, HTML, XML

Unstructured data

- Information that either does not have a pre-defined structure or is not organized in a pre-defined manner.

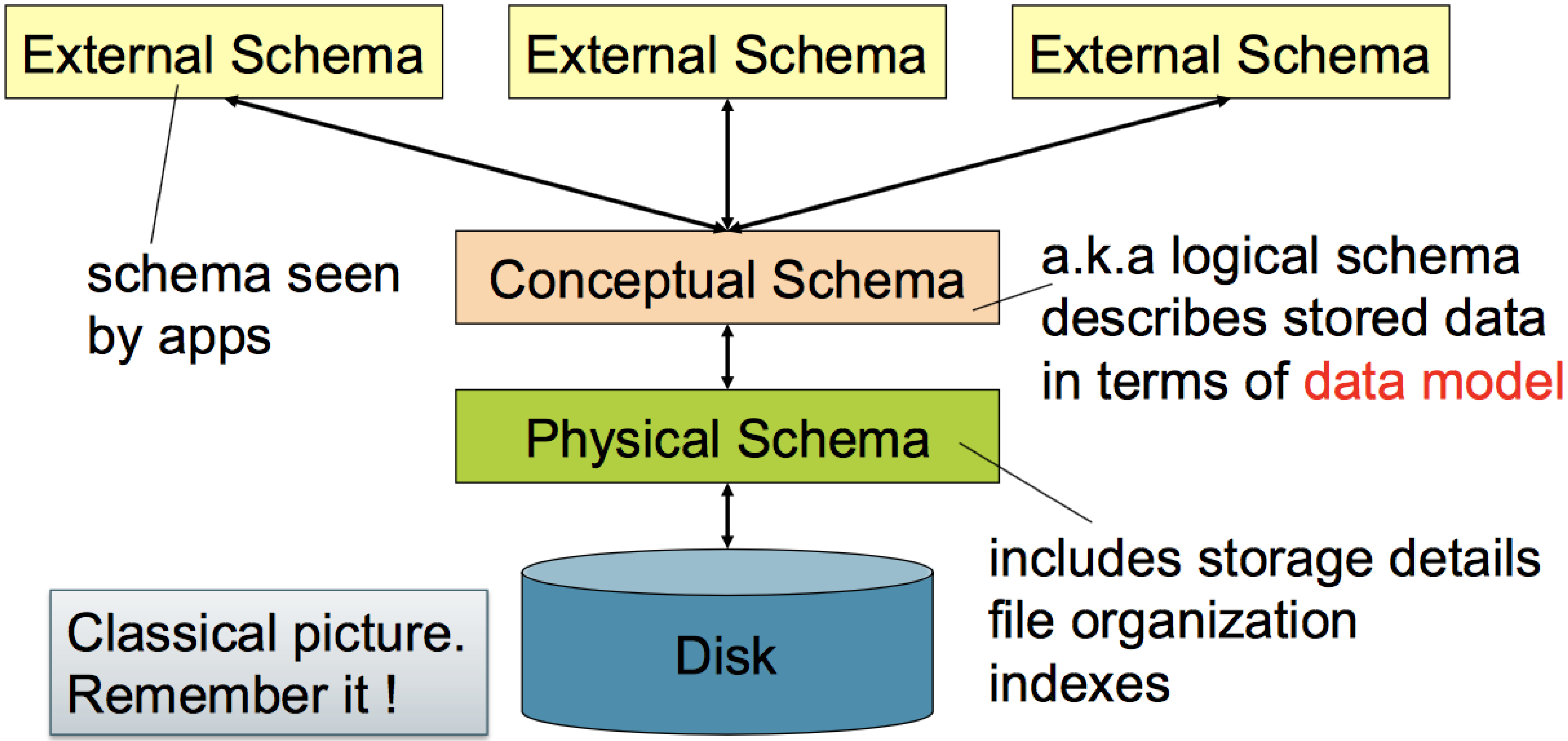
- Text, video, images, etc.

- Abundant and extremely valuable. Hard to query, aggregate, analyze, search.

Data Model

- A **data model** is a collection of concepts for describing data
- A **schema** is a description of a particular collection of data, **using the given data model**
- A **data model** enables users to define the data using high-level constructs without worrying about many low-level details of how data will be stored on disk.

Levels of abstraction



External Schema

External Schema

External Schema

schema seen
by apps

Conceptual Schema

a.k.a logical schema
describes stored data
in terms of data model

Physical Schema

includes storage details
file organization
indexes

Disk

Classical picture.
Remember it !

Data models

- Relational

Most database management systems

- Key/Value
- Graph
- Document
- Column-family
- Array/Matrix
- Hierarchical
- Network

Data models

- Relational
- Key/Value
- Graph
- Document
- Column-family
- Array/Matrix
- Hierarchical
- Network

No SQL

Data models

- Relational
- Key/Value
- Graph
- Document
- Column-family
- Array/Matrix

Machine learning, Scientific applications

- Hierarchical
- Network

Data models

- Relational
- Key/Value
- Graph
- Document
- Column-family
- Array/Matrix
- Hierarchical
- Network

Obsolete / Rare

3. RDBMs and the Relational Data Model

What you will learn about in this section

1. Definition of DBMS

2. Data models & the relational data model

3. Schemas & data independence

What is a DBMS?

- A large, integrated collection of data

- Models a real-world *enterprise*
  - *Entities* (e.g., Students, Courses)
  - *Relationships* (e.g., Alice is enrolled in CS564)

A Database Management System (DBMS) is a piece of software designed to store and manage databases

A Motivating, Running Example

- Consider building a course management system (**CMS**):
  - Students
  - Courses
  - Professors

Entities

  - Who takes what
  - Who teaches what

Relationships

Data models

- A **data model** is a collection of concepts for describing data
    - The <u>relational model of data</u> is the most widely used model today
        - Main Concept: the *relation*- essentially, a table

- A **schema** is a description of a particular collection of data, **using the given data model**
    - E.g. every *relation* in a relational data model has a *schema* describing types, etc.

Modeling the Course Management System

- *Logical Schema*
  - Students(sid: *string*, name: *string*, gpa: *float*)
  - Courses(cid: *string*, cname: *string*, credits: *int*)
  - Enrolled(sid: *string,* cid*: string,* grade*: string*)

sid	**Name**	**Gpa**
101	Bob	3.2
123	Mary	3.8

Students

Relations

cid	**cname**	**credits**
564	564-2	4
308	417	2

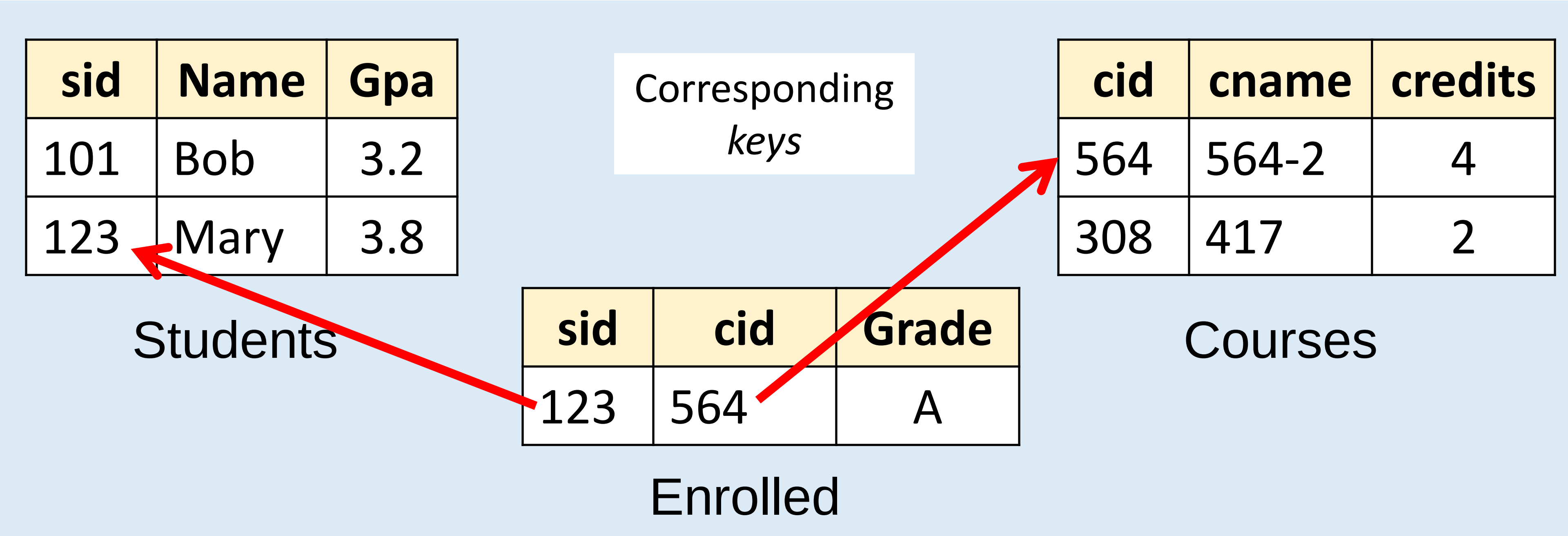
Courses

sid	**cid**	**Grade**
123	564	A

Enrolled

Modeling the Course Management System

- *Logical Schema*
  - Students(sid: *string*, name: *string*, gpa: *float*)
  - Courses(cid: *string*, cname: *string*, credits: *int*)
  - Enrolled(sid: *string,* cid*: string,* grade*: string*)



sid	Name	Gpa
101	Bob	3.2
123	Mary	3.8

Students

Corresponding *keys*

cid	cname	credits
564	564-2	4
308	417	2

Courses

sid	cid	Grade
123	564	A

Enrolled

Other Schemata…

- *Physical Schema*: describes data layout
  - Relations as unordered files
  - Some data in sorted order (index)

↑ Administrators

- *Logical Schema:* Previous slide

↓ Applications

- *External Schema*: (Views)
  - Course_info(cid: *string*, enrollment: *integer*)
  - Derived from other tables

Data independence

Concept: Applications do not need to worry about *how the data is structured and stored*

Logical data independence: protection from changes in the *logical structure of the data*

I.e. should not need to ask: can we add a new entity or attribute without rewriting the application?

Physical data independence: protection from *physical layout changes*

I.e. should not need to ask: which disks are the data stored on? Is the data indexed?

One of the most important reasons to use a DBMS

Relational Model

- **Structure:** The definition of relations and their contents.
- **Integrity:** Ensure the database's contents satisfy constraints.
- **Manipulation:** How to access and modify a database's contents.

Tables in the Relational Model

Product

PName	Price	Manufacturer
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

A **relation** or **table** is a multiset of tuples having the attributes specified by the schema

Let's break this definition down

Tables in the Relational Model

Product

PName	Price	Manufacturer
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

A **multiset** is an unordered list (or: a set with multiple duplicate instances allowed)

List:	[1, 1, 2, 3]
Set:	{1, 2, 3}
Multiset:	{1, 1, 2, 3}

i.e. no *next()*, etc. methods!

Tables in the Relational Model

Product

PName	Price	Manufacturer
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

An **attribute** (or **column**) is a typed data entry present in each tuple in the relation

Attributes must have an ***atomic*** *type, i.e. not a list, set, etc.*

Tables in the Relational Model

Product

PName	Price	Manufacturer
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

A **tuple** or **row** is a single entry in the table having the attributes specified by the schema

Also referred to sometimes as a ***record***

Tables in the Relational Model

Product

PName	Price	Manufacturer
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

The number of tuples is the **cardinality** of the relation

The number of attributes is the **arity** of the relation

***n*-ary Relation**

=

Table with *n* columns

Data Types in Relational Model

- Atomic types:
    - Characters: CHAR(20), VARCHAR(50)
    - Numbers: INT, BIGINT, SMALLINT, FLOAT
    - Others: MONEY, DATETIME, …
- Every attribute must have an atomic type
    - Hence tables are flat

Table Schemas

- The **schema** of a table is the table name, its attributes, and their types:

Product(Pname: *string*, Price: *float*, Category: *string*, Manufacturer: *string*)

- A **key** is an attribute whose values are unique; we underline a key

Product(Pname: *string*, Price: *float*, Category: *string*, Manufacturer: *string*)

Key constraints

A key is a **minimal subset of attributes** that acts as a unique identifier for tuples in a relation

- A key is an implicit constraint on which tuples can be in the relation
    - i.e. if two tuples agree on the values of the key, then they must be the same tuple!

Students(sid:string, name:string, gpa: float)

1. Which would you select as a key?
2. Is a key always guaranteed to exist?
3. Can we have more than one key?

NULL and NOT NULL

- To say "don't know the value" we use NULL
  - NULL has (sometimes painful) semantics, more details later

Students(sid:string, name:string, gpa: float)

sid	name	gpa
123	Bob	3.9
143	Jim	NULL

Say, Jim just enrolled in his first class.

We may constrain a column to be NOT NULL, e.g., "name" in this table

Foreign Key constraints

- A foreign key specifies that an attribute from one relation has to map to a tuple in another relation.

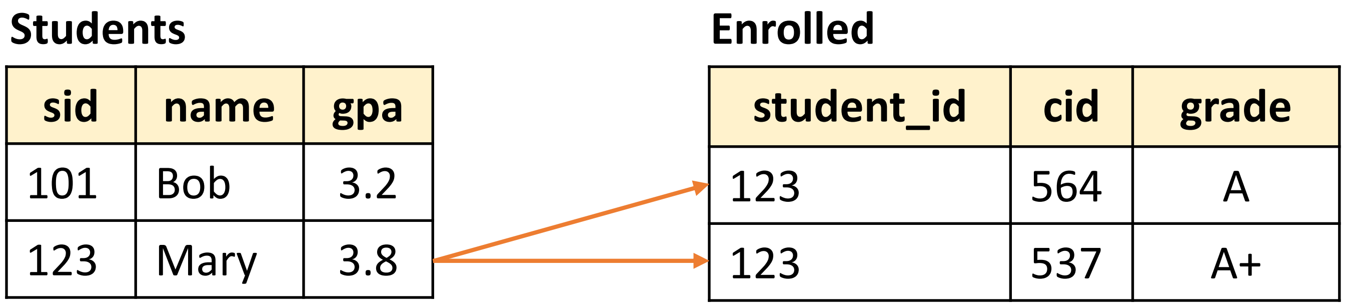
Foreign Key constraints

- Suppose we have the following schema:

Students(sid*: string,* name: *string*, gpa: *float*)

Enrolled(student_id: *string,* cid: *string*, grade: *string*)

- And we want to impose the following constraint:
  - ‘Only real students may enroll in courses’ i.e. a student must appear in the Students table to enroll in a class



Students

sid	name	gpa
101	Bob	3.2
123	Mary	3.8

Enrolled

student_id	cid	grade
123	564	A
123	537	A+

student_id alone is not a key- what is?

We say that student_id is a **foreign key** that refers to Students

Summary of Schema Information

- Schema and Constraints are how databases understand the semantics (meaning) of data

- They are also useful for optimization

DATA MANIPULATION LANGUAGES (DML)

- How to store and retrieve information from a database.
- Procedural: The query specifies the (high-level) strategy the DBMS should use to find the desired result.
- We will see SQL and Relational Algebra