NYU | Center for Data Science

Week 02.1: Relations

DS-GA 1004: Big Data

This week

- **Relational databases**
- SQL
- Transactions

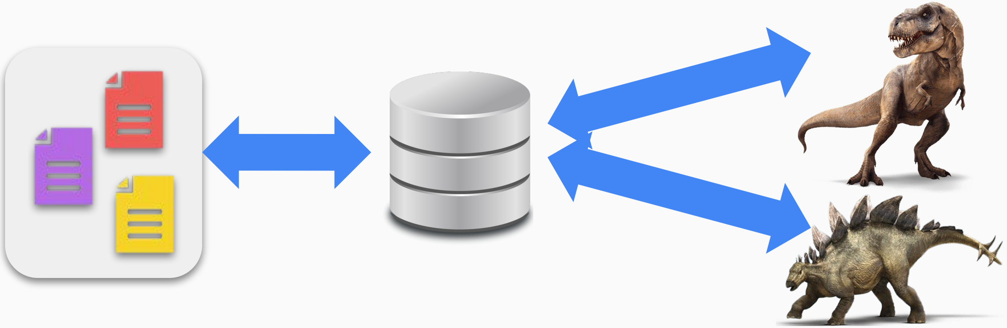
Database research is a huge topic!

This intro will be brief and cursory.

Database management systems (DBMS)

- DBMS's job is to provide
  - Data integrity / consistency
  - Concurrent access
  - Efficient storage and access
  - Standardized format / administration
  - Standardized query interface (language)

- DBMS come in many flavors
  - **Relational (RDBMS)**
  - Semi-structured (e.g., XML)
  - Object-oriented
  - Object-relational
  - ...



The relational model

- **High-level**: tables of data that you're probably used to
  - Spreadsheets, dataframes, numerical arrays, etc.

- Each column represents a **set** of possible values (numbers, strings, etc)

The relational model

- **High-level**: tables of data that you're probably used to
  - Spreadsheets, dataframes, numerical arrays, etc.

- Each column represents a **set** of possible values (numbers, strings, etc)

- A ***relation*** over sets $A_1, A_2, A_3, \ldots, A_n$ is a **subset** of their cartesian product
  - $R \subseteq A_1 \times A_2 \times A_3 \times \ldots \times A_n$
  - The **rows** of the table are elements of *R*, also known as ***tuples***
  - $(a_1, a_2, \ldots, a_n) \in R \Rightarrow a_1 \in A_1, a_2 \in A_2, \ldots, a_n \in A_n$

Example: dinosaurs

- A_1 = {s | s is a string}
  A_2 = {“Jurassic”, “Cretaceous”, “Devonian”, “Triassic”, …}
  A_3 = {“Carnivore”, “Herbivore”, “Omnivore”, …}
  A_4 = {False, True}

- Any A_i could be finite or infinite

- $R \subseteq A_1 \times A_2 \times A_3 \times A_4$ need not contain all combinations!

Species	Era	Diet	Awesome
T. Rex	Cretaceous	Carnivore	True
Stegosaurus	Jurassic	Herbivore	True
Ankylosaurus	Cretaceous	Herbivore	False

Aside: why “relations” and not “tables”?

- Relations are the abstract model of data
- **Table** refers to an **explicitly** constructed relation
  - I.E., records you’ve observed / collected
- Other relations in a DB:
  - **view**: a relation defined **implicitly**, and constructed **dynamically** at run-time
  - **temporary table**: the output of a **query**

Properties of relations

- $R \subseteq A_1 \times A_2 \times A_3 \times \dots \times A_n$ is a set
  - The tuples (rows) of R are **unordered**
  - Tuples are **unique** ⇒ no duplicates!
  - Relations over common domains (columns) can be combined by set operations

Species	**Era**	**Diet**	**Awesome**
T. Rex	Cretaceous	Carnivore	True
Stegosaurus	Jurassic	Herbivore	True
Ankylosaurus	Cretaceous	Herbivore	False

Properties of relations

- $R \subseteq A_1 \times A_2 \times A_3 \times \dots \times A_n$ is a set
  - The tuples (rows) of *R* are **unordered**
  - Tuples are **unique** ⇒ no duplicates!
  - Relations over common domains (columns) can be combined by set operations

- In practice, add a column (e.g., A_0) with **identifiers** to force uniqueness
  - This is not (usually) part of the data, but is generated automatically by the DBMS
  - ID fields are often used as **primary keys**, and give a default order to rows

id	Species	Era	Diet	Awesome
1	T. Rex	Cretaceous	Carnivore	True
2	Stegosaurus	Jurassic	Herbivore	True
3	Ankylosaurus	Cretaceous	Herbivore	False

Schemas

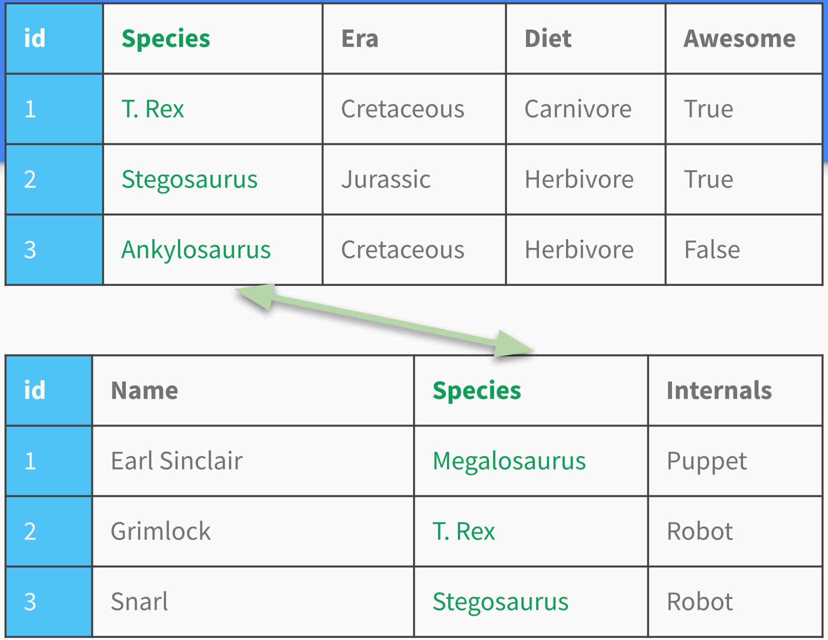
- A relation is defined by a **schema:**

id	Species	Era	Diet	Awesome
1	T. Rex	Cretaceous	Carnivore	True
2	Stegosaurus	Jurassic	Herbivore	True
3	Ankylosaurus	Cretaceous	Herbivore	False
4	Homer	Boomer	Donuts	False

Dinosaur(id: int, Species: string, Era: string, Diet: string, Awesome: boolean)

- **Any** tuple (int, string, string, string, boolean) is valid under this schema
    - ⇒ Schemas enforce type (syntax), but not semantics!

Relational databases



id	Species	Era	Diet	Awesome
1	T. Rex	Cretaceous	Carnivore	True
2	Stegosaurus	Jurassic	Herbivore	True
3	Ankylosaurus	Cretaceous	Herbivore	False

id	Name	Species	Internals
1	Earl Sinclair	Megalosaurus	Puppet
2	Grimlock	T. Rex	Robot
3	Snarl	Stegosaurus	Robot

- A relational database consists of one or more relational schemas

- Structured data can be encoded by **joining** on **shared attributes**

- The collection of schemas defines your **data model**

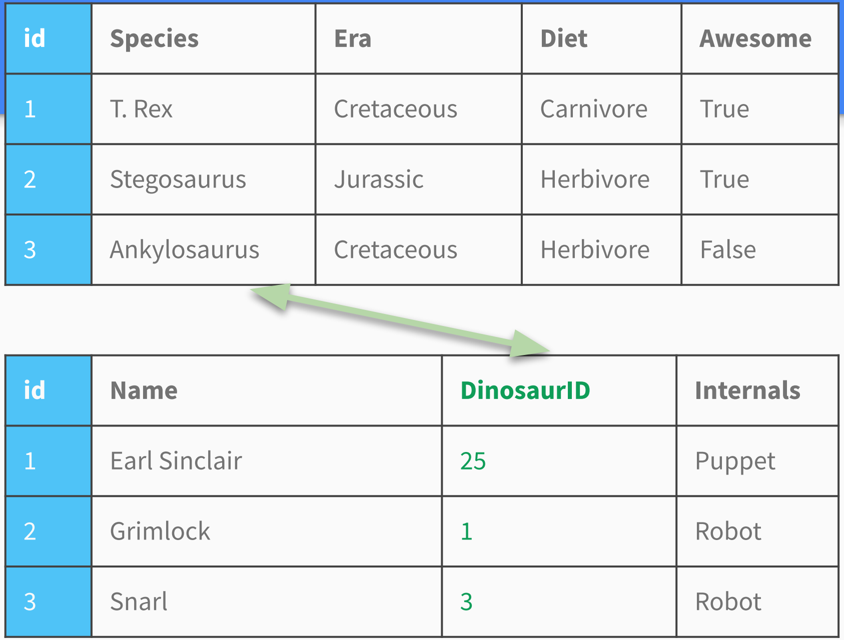
Keys

id	First Name	Last Name	Age
1	Homer	Simpson	39
2	Marge	Simpson	39
3	Bart	Simpson	10
4	Homer	Thompson	39
5	Homer	Simpson	28

- Keys are what determine the **identity** of a row
- Keys can be simple (single column) or **compound** (two or more columns)
  - Example: (First Name, Last Name)
  - This prevents two rows with the same combination of first and last name
- You can have **primary** and **alternate keys**
  - Usually a good idea to keep a primary numeric key as well as others you may want...

Foreign Keys

- A **key** from one relation can be a **column** in another
  - This is called a **FOREIGN KEY** constraint
- This can be used to ensure reference consistency **between** tables/relations



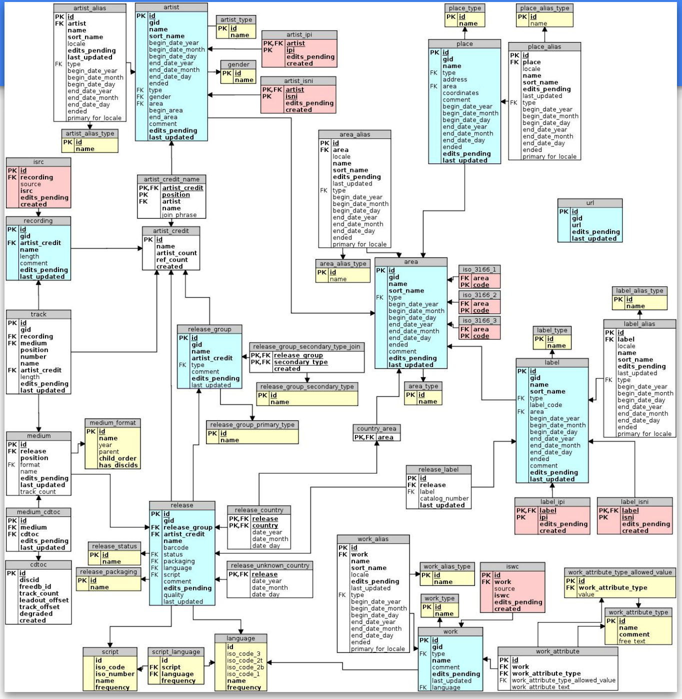
id	Species	Era	Diet	Awesome
1	T. Rex	Cretaceous	Carnivore	True
2	Stegosaurus	Jurassic	Herbivore	True
3	Ankylosaurus	Cretaceous	Herbivore	False

id	Name	DinosaurID	Internals
1	Earl Sinclair	25	Puppet
2	Grimlock	1	Robot
3	Snarl	3	Robot

- **This is not automatic**: must be included in the **schema** definition!

Normalization

- A database schema is **normal** if data is not redundantly stored
    - Use **identifiers**, not **values**, to link between relations
- Modifying a record is easy if it exists in exactly one place
- But it can also be difficult
    - Reading complex data can be cumbersome
    - Multiple levels of indirection



https://musicbrainz.org/doc/MusicBrainz_Database/Schema

Summary

Relations are cool!

- We use relational data every day without thinking of it
- Databases consist of one or more relations
- Putting data into a relational model can make it easier to work with
- Schemas provide some degree of safety and validation