Integrating SQL into R analytical workflows using *duckDB & dbplyr*

SORTEE

Society for Open, Reliable, and Transparent Ecology and Evolutionary Biology

Julien Brun & Greg Janée

Research Data Services, UCSB Library

rds@library.ucsb.edu

UC SANTA BARBARA

Why considering relational databases ?

- Enhance the management of your data
  (field stations, observation network, …)
- Centralize your data
- Ease the QA/QC process (data types, triggers, …)
- Facilitate your collaboration around data
- Handle large data more easily

Relational databases

Pros

- 50 years in the making
- Flexible query system
- Allow (well, require) schema definition
- Correctness/consistency preserving
- Fault tolerant, even replicated
- Support concurrency out of the box
- Extensions for spatial objects, large corpus text, XML, JSON, etc.

Cons

- Relatively inflexible
- Require more up-front investment
- Require relatively heavyweight installation; but the cloud sure makes things easy
- Data locked in
  - Multiple RDBMS vendors
  - beyond SQL, mutually incompatible
- No standard on-disk storage or backup/dump formats
- If database corrupted (very rare), you’re hosed

Server vs Personal databases

There are two main categories of databases:

- **Personal**: one file on your computer

DuckDB SQLite

- **Server**: run on a server, allow multiple users

MongoDB

MySQL

Data modeling importance

- “Even though the data modelling phase represents only a relatively small share of the total development effort of data systems, its impact on the final result is probably greater than that of any other phase.” (Wohner 2022)

- “Best Practice #1: Always create a data model before you start attempting to manage the data.” (Burnett 2022)

Basic unit of information: table (== entity == relation)

- Represents a type of entity
  - Person, place, thing, event, transaction, observation
- Rows *(== Record == Tuple <> Observation)* represent instances of the entity
  - Usually unique; conceptually unordered
- Columns represent attributes/properties of the entities

When you add data, you should be adding rows!!

STUDENT

Name	House	Blood status	Birthdate	Wand length (in)
Harry Potter	Gryffindor	Half-blood	1980-07-31	11
Hermione Granger	Gryffindor	Muggle-born	1979-09-19	10.75
Draco Malfoy	Slytherin	Pure-blood	1980-06-05	10
Ginny Weasley	Gryffindor	Pure-blood	1981-08-11	NULL

All data from harrypotter.fandom.com

Columns (== Variable == Attribute == Characteristic)

- Strictly typed
  - Good range of types, but annoyingly, vary by system
- Best practice: one atomic quantity

Text	Controlled vocabulary	Controlled vocabulary	Date	Real
Name	**House**	**Blood status**	**Birthdate**	**Wand length (in)**
Harry Potter	Gryffindor	Half-blood	1980-07-31	11
Hermione Granger	Gryffindor	Muggle-born	1979-09-19	10.75
Draco Malfoy	Slytherin	Pure-blood	1980-06-05	10
Ginny Weasley	Gryffindor	Pure-blood	1981-08-11	NULL

Primary keys

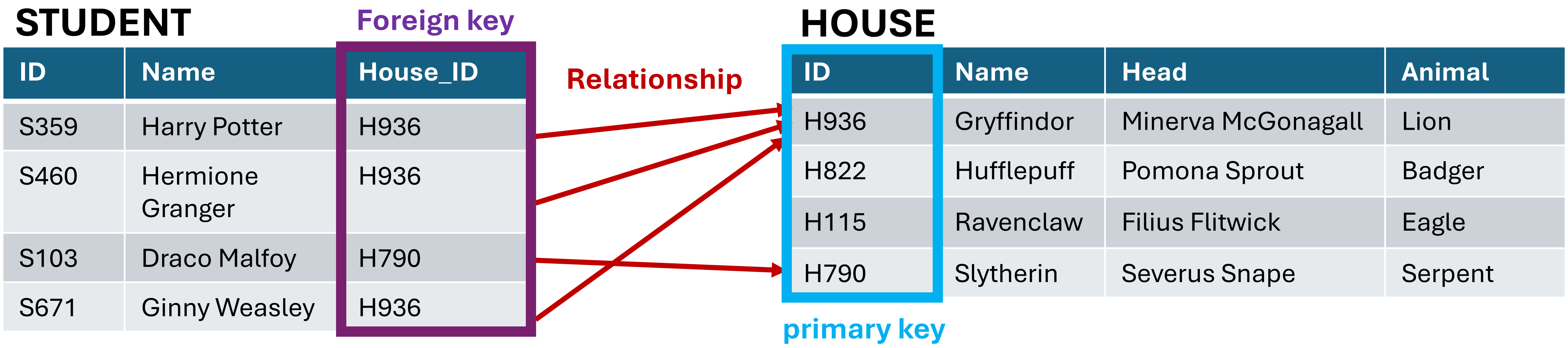
- Column(s) that uniquely identify the entities, i.e., the rows
- Not required by SQL, but usually conceptually necessary/applicable

STUDENT

ID	Name	House	Blood status	Birthdate	Wand length (in)
S359	Harry Potter	Gryffindor	Half-blood	1980-07-31	11
S460	Hermione Granger	Gryffindor	Muggle-born	1979-09-19	10.75
S103	Draco Malfoy	Slytherin	Pure-blood	1980-06-05	10
S671	Ginny Weasley	Gryffindor	Pure-blood	1981-08-11	NULL

Foreign keys

- Column(s) that reference the primary key column(s) in another table
- Basis for relationships
- Relationship cardinalities: one-to-one, ***many-to-one***, many-to-many



STUDENT

Foreign key

ID	Name	House_ID
S359	Harry Potter	H936
S460	Hermione Granger	H936
S103	Draco Malfoy	H790
S671	Ginny Weasley	H936

Relationship

HOUSE

ID	Name	Head	Animal
H936	Gryffindor	Minerva McGonagall	Lion
H822	Hufflepuff	Pomona Sprout	Badger
H115	Ravenclaw	Filius Flitwick	Eagle
H790	Slytherin	Severus Snape	Serpent

primary key

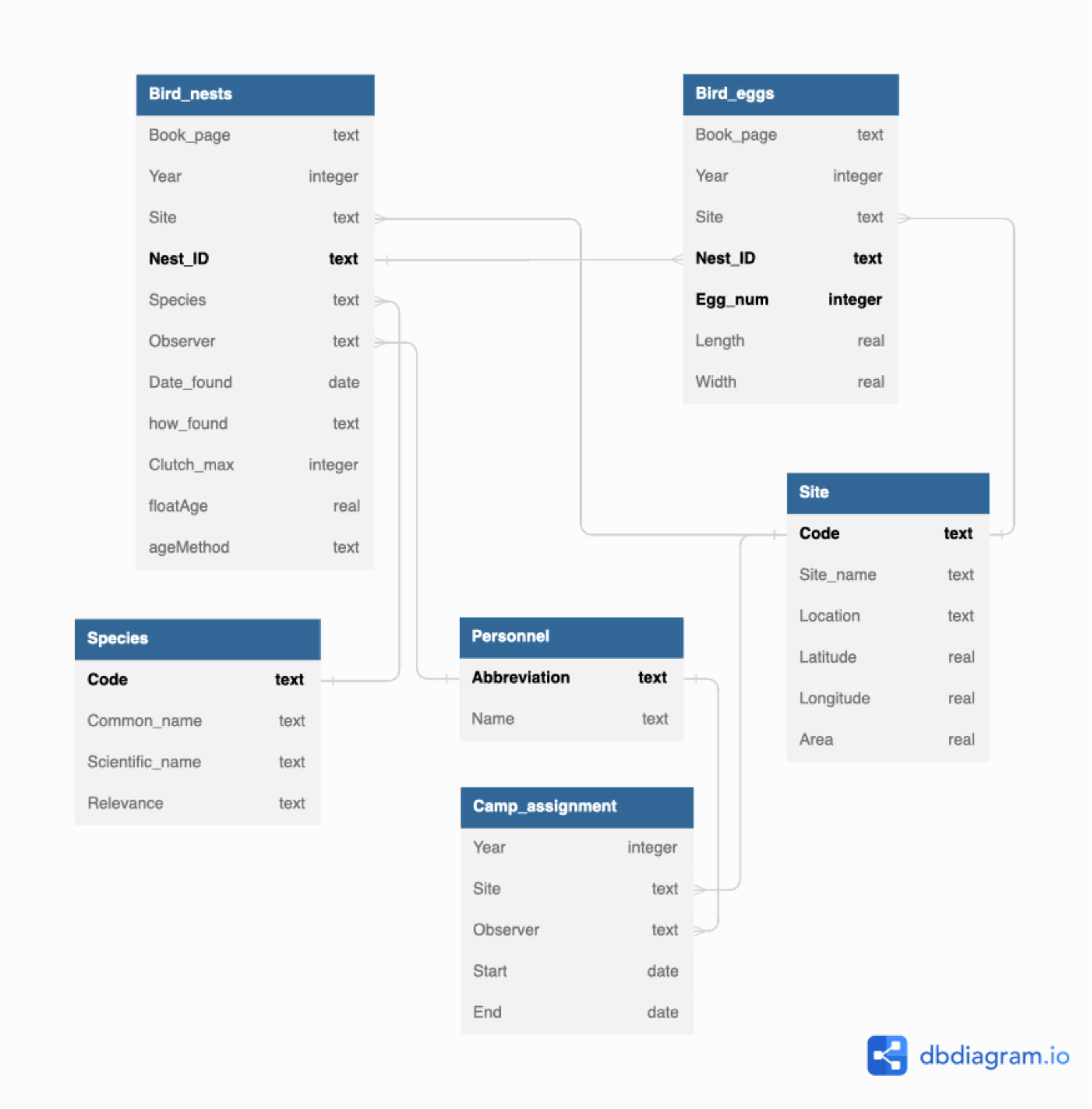
Normalization

- Data is “normalized” if it follows the principles we’ve been discussing
    - One table per type per entity
    - Rows are instances of the entity
    - Columns are attributes of the entity (and not of anything else)
    - Primary and foreign keys

- Benefits
    - Data is one and only one place
    - Maximizes flexibility and independence

- Disadvantages
    - More tables; complexity
    - Data sometimes “denormalized” for simplicity, performance

Entity Relationship model



Bird_nests	
Book_page	text
Year	integer
Site	text
Nest_ID	**text**
Species	text
Observer	text
Date_found	date
how_found	text
Clutch_max	integer
floatAge	real
ageMethod	text

Bird_eggs	
Book_page	text
Year	integer
Site	text
Nest_ID	**text**
Egg_num	**integer**
Length	real
Width	real

Site	
Code	**text**
Site_name	text
Location	text
Latitude	real
Longitude	real
Area	real

Species	
Code	**text**
Common_name	text
Scientific_name	text
Relevance	text

Personnel	
Abbreviation	**text**
Name	text

Camp_assignment	
Year	integer
Site	text
Observer	text
Start	date
End	date