Schema Design

schema, n. — a representation of a plan or theory in the form of an outline or model.

Schemas

- **Relation Schema (i.e. table schema)**
  - Logical definition of the relation (table)
- **Database Schema**
  - Blueprint of the collection of relation schemas for an entire database

Data Modeling

- **How do we represent real world relationships and properties in our program?**
  - …in a way that makes writing the program easy
  - …while remaining flexible for future changes
  - …oh, it also has to be fast (enough).

Designing a Schema

- **Analysis**
  - What does my program need to output?
  - What data will I need to produce that output?
- **Conceptual Design**
  - Conceptual entities and their relationships
- **Logical Design**
  - In a SQL database: What are my tables, attributes, and relationships?
  - In a program: What are my functions and data structures?
- **Physical Design**
  - JavaScript code, CREATE TABLE statements

Designing a Schema

What we'll focus on today

- **Analysis**
  - What does my program need to output?
  - What data will I need to produce that output?
- **Conceptual Design**
  - Conceptual entities and their relationships
- **Logical Design**
  - In a SQL database: What are my tables, attributes, and relationships?
  - In a program: What are my functions and data structures?
- **Physical Design**
  - JavaScript code, CREATE TABLE statements

Designing a Schema

- **Analysis**
  - What does my program need to output?
  - What data will I need to produce that output?
- **Conceptual Design**
  - Conceptual entities and their relationships
- **Logical Design**
  - In a SQL database: What are my tables, attributes, and relationships?
  - In a program: What are my functions and data structures?
- **Physical Design**
  - JavaScript code, CREATE TABLE statements

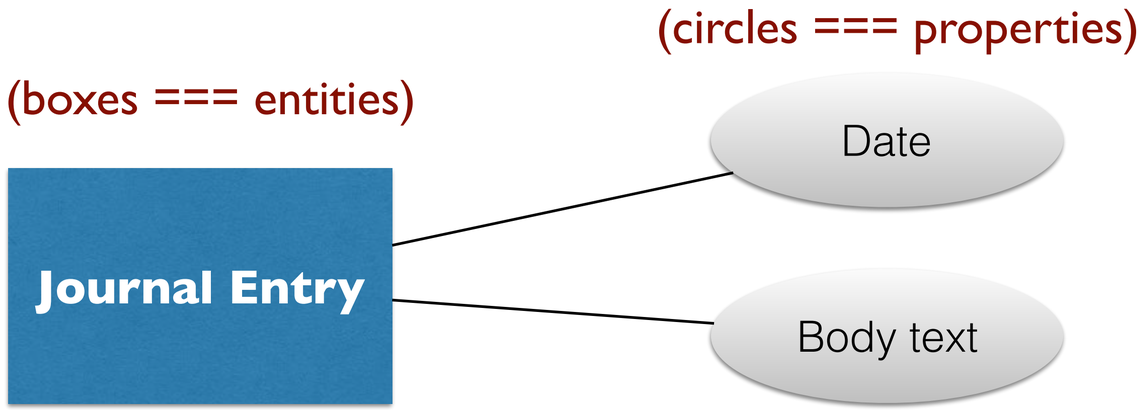
Example: A Journal
Analysis

- I want a program to keep my journal in.
- I want to be able to enter the **text** of each journal entry.
- I want to be able to see journal entries **chronologically**.

Designing a Schema

- **Analysis**
  - What does my program need to output?
  - What data will I need to produce that output?
- **Conceptual Design**
  - Conceptual entities and their relationships
- **Logical Design**
  - In a SQL database: What are my tables, attributes, and relationships?
  - In a program: What are my functions and data structures?
- **Physical Design**
  - JavaScript code, CREATE TABLE statements

Entity Relationship Diagram (ERD)
Conceptual Design

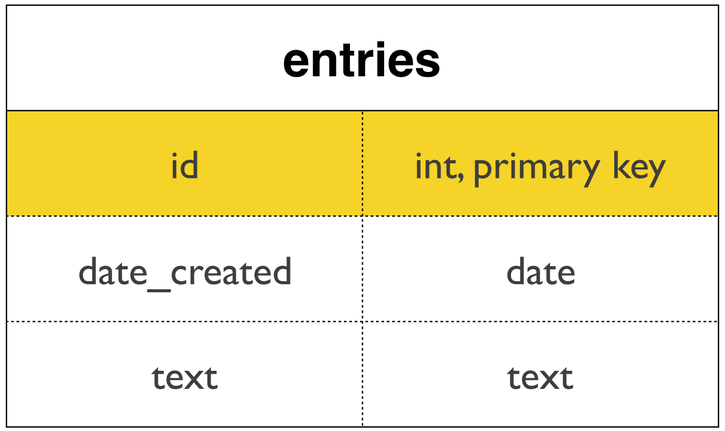


Designing a Schema

- **Analysis**
  - What does my program need to output?
  - What data will I need to produce that output?
- **Conceptual Design**
  - Conceptual entities and their relationships
- **Logical Design**
  - In a SQL database: What are my tables, attributes, and relationships?
  - In a program: What are my functions and data structures?
- **Physical Design**
  - JavaScript code, CREATE TABLE statements

Entity Relationship Diagram (ERD)

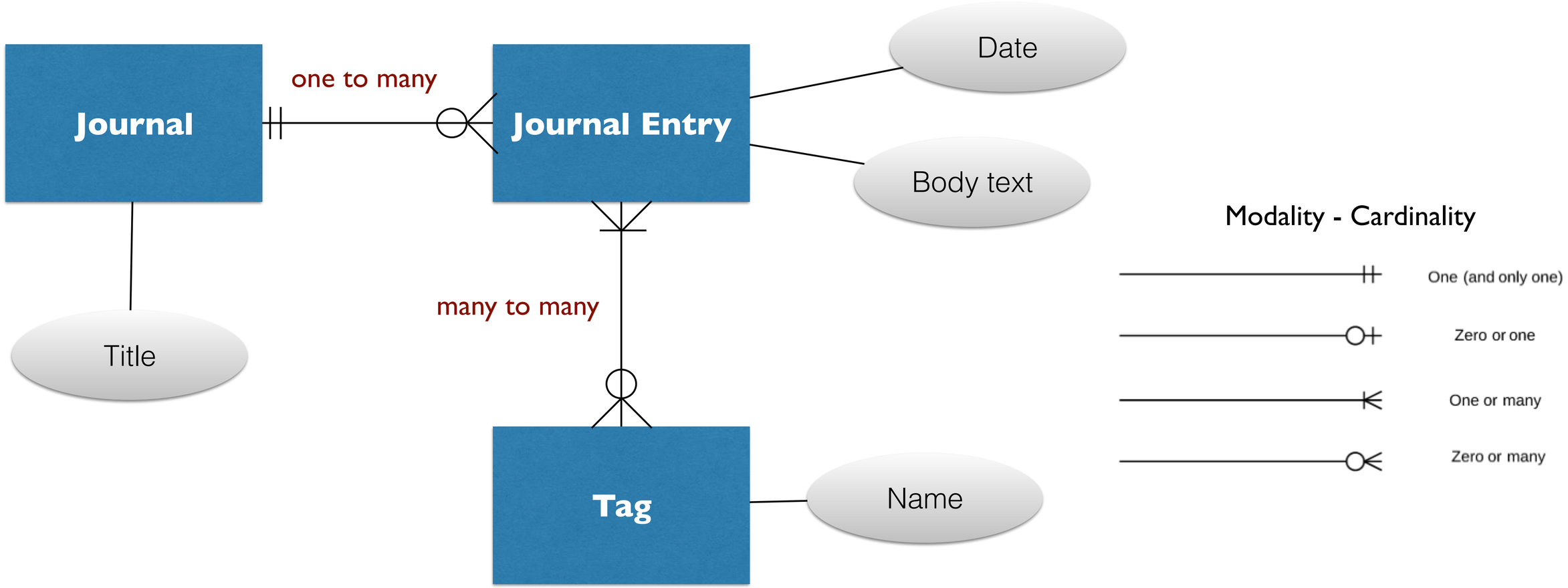
Logical Design



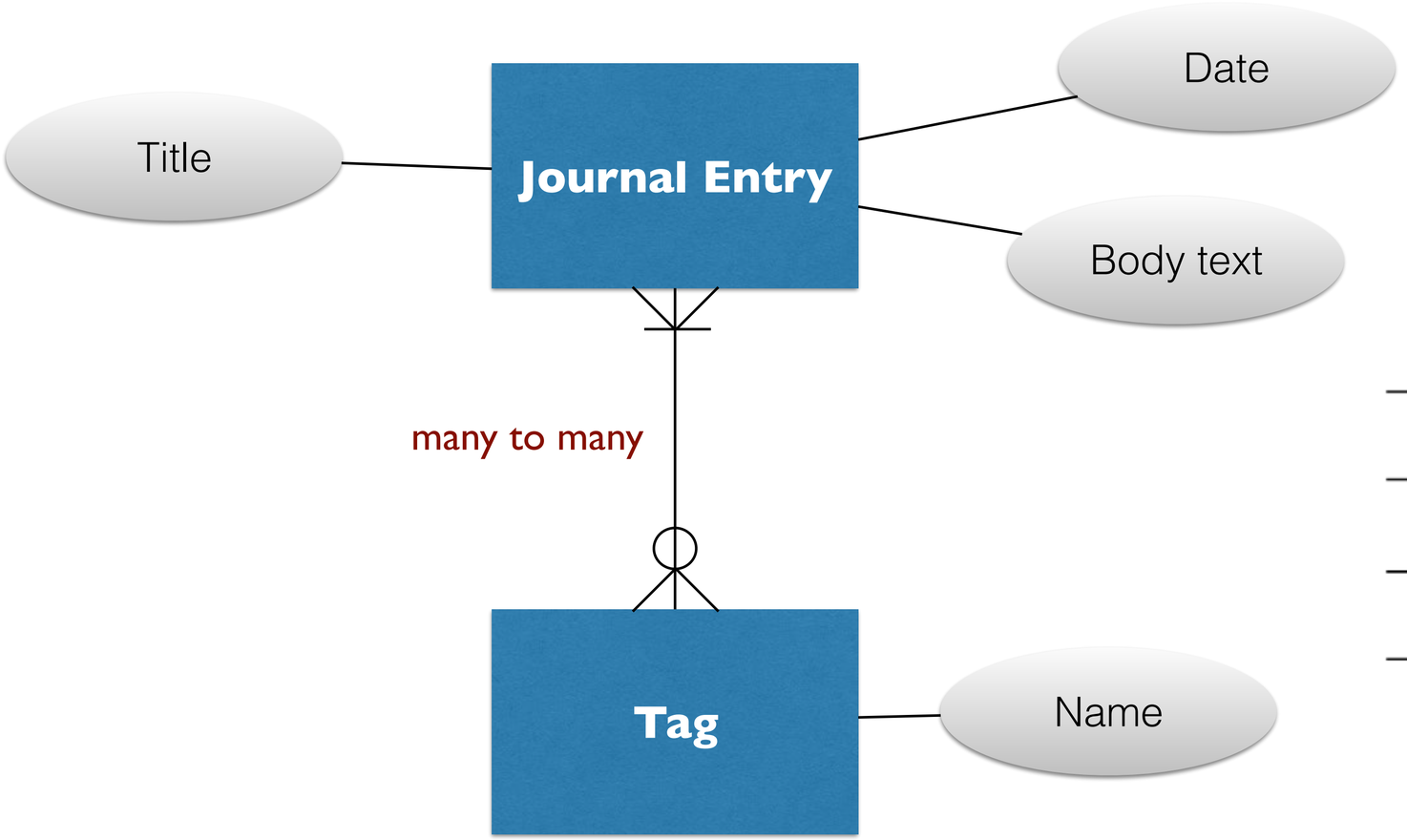
All done!

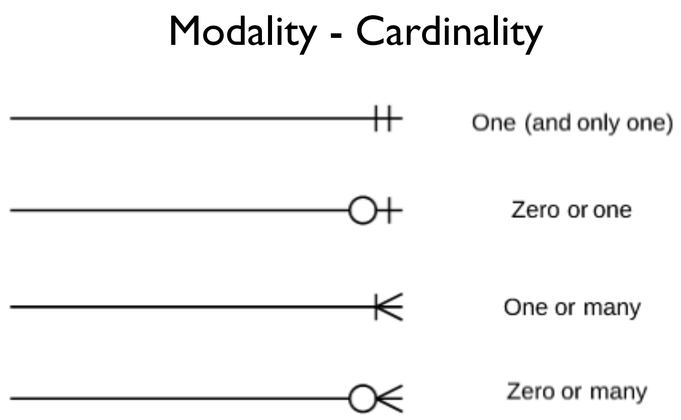
- **Oh wait, I forgot a couple of things**
  - I want to be able to have multiple journals
  - I want to be able to #tag entries and find all entries with a particular #tag
- **Take 2…**

Example: A Journal Conceptual Design, Take 2



...Or maybe like this?



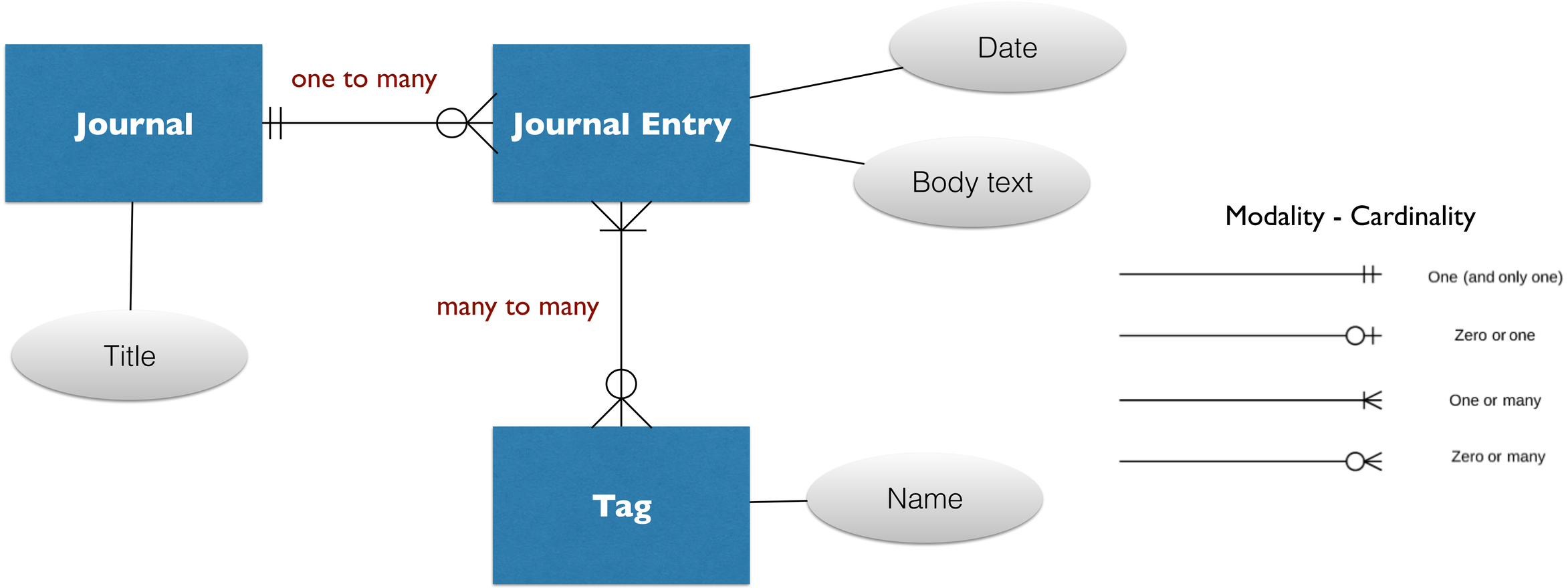


- How do I change the name of “happy times” to “sadness”?

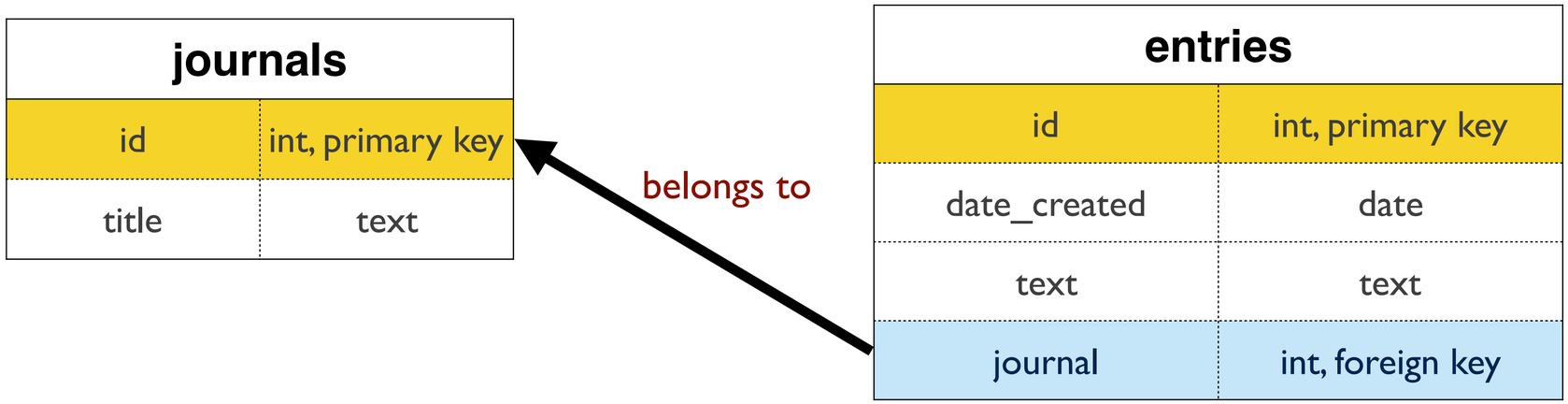
select * from entries;

id	date_created	text	journal_title
0	2016-04-01	I am happy	happy times
1	2016-04-02	I am very happy	happy times
2	2016-04-03	Despair fills me	happy times
3	2016-04-03	Sadness is my life	an anatomy of pain

Example: A Journal Conceptual Design, Take 2

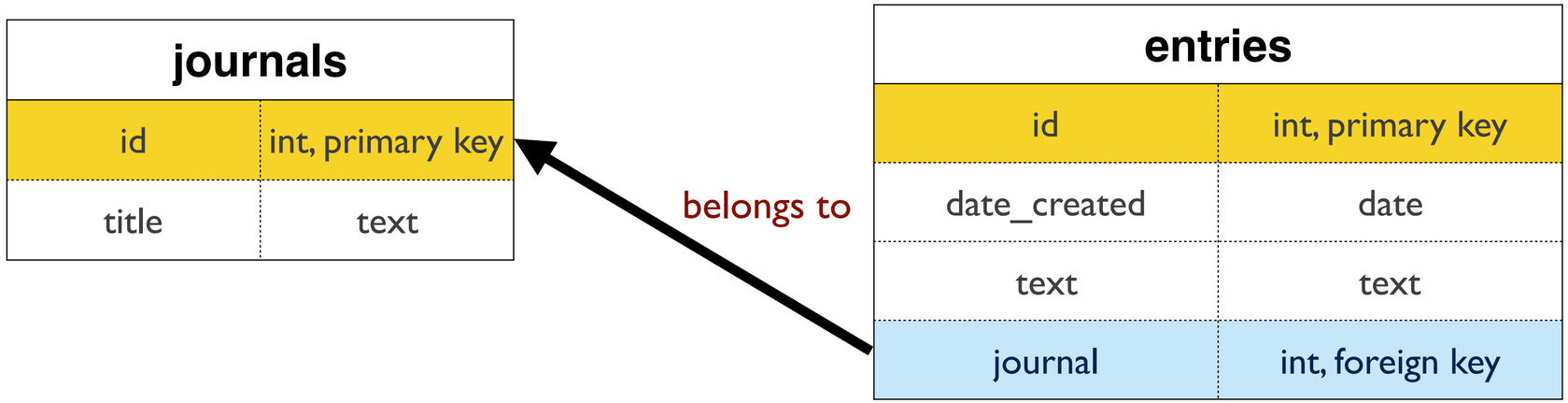


Example: A Journal
Logical Design: Take 2?

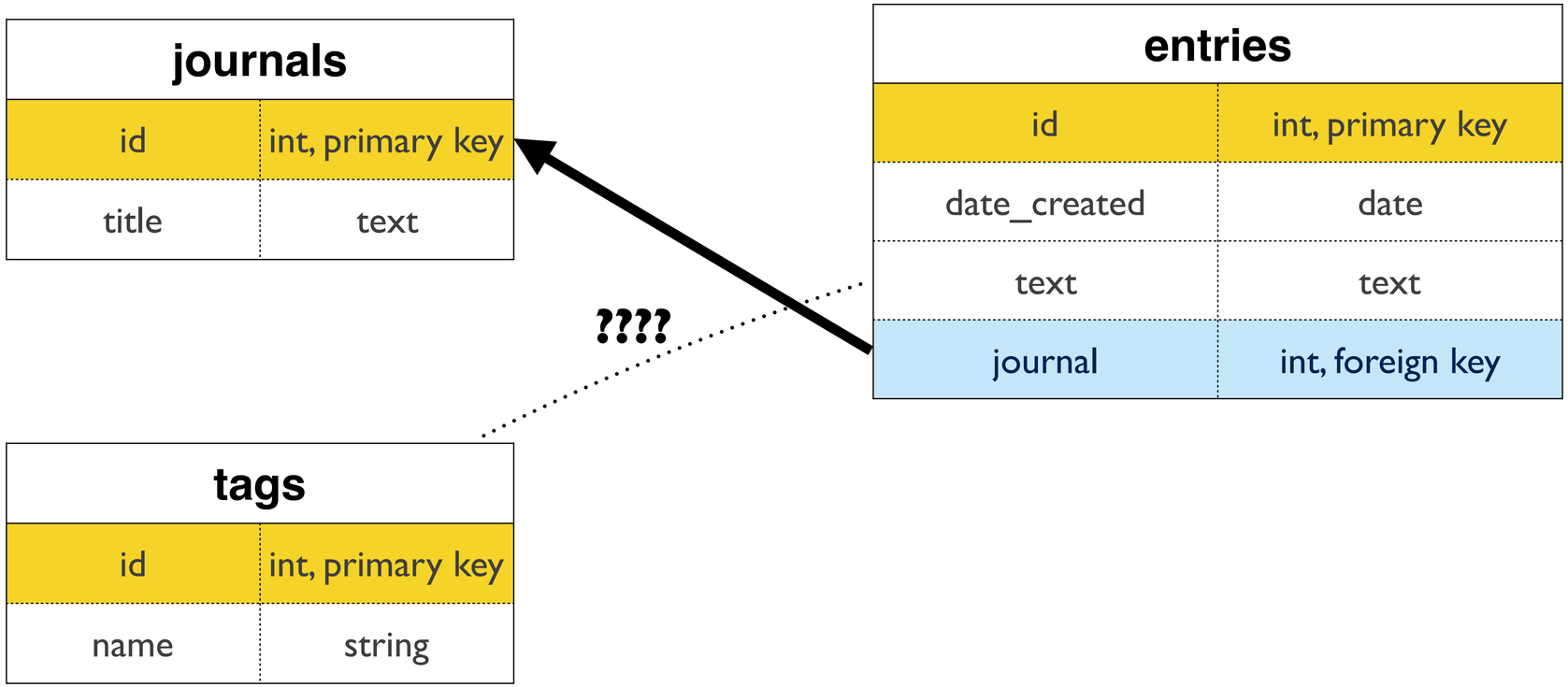


Normalization

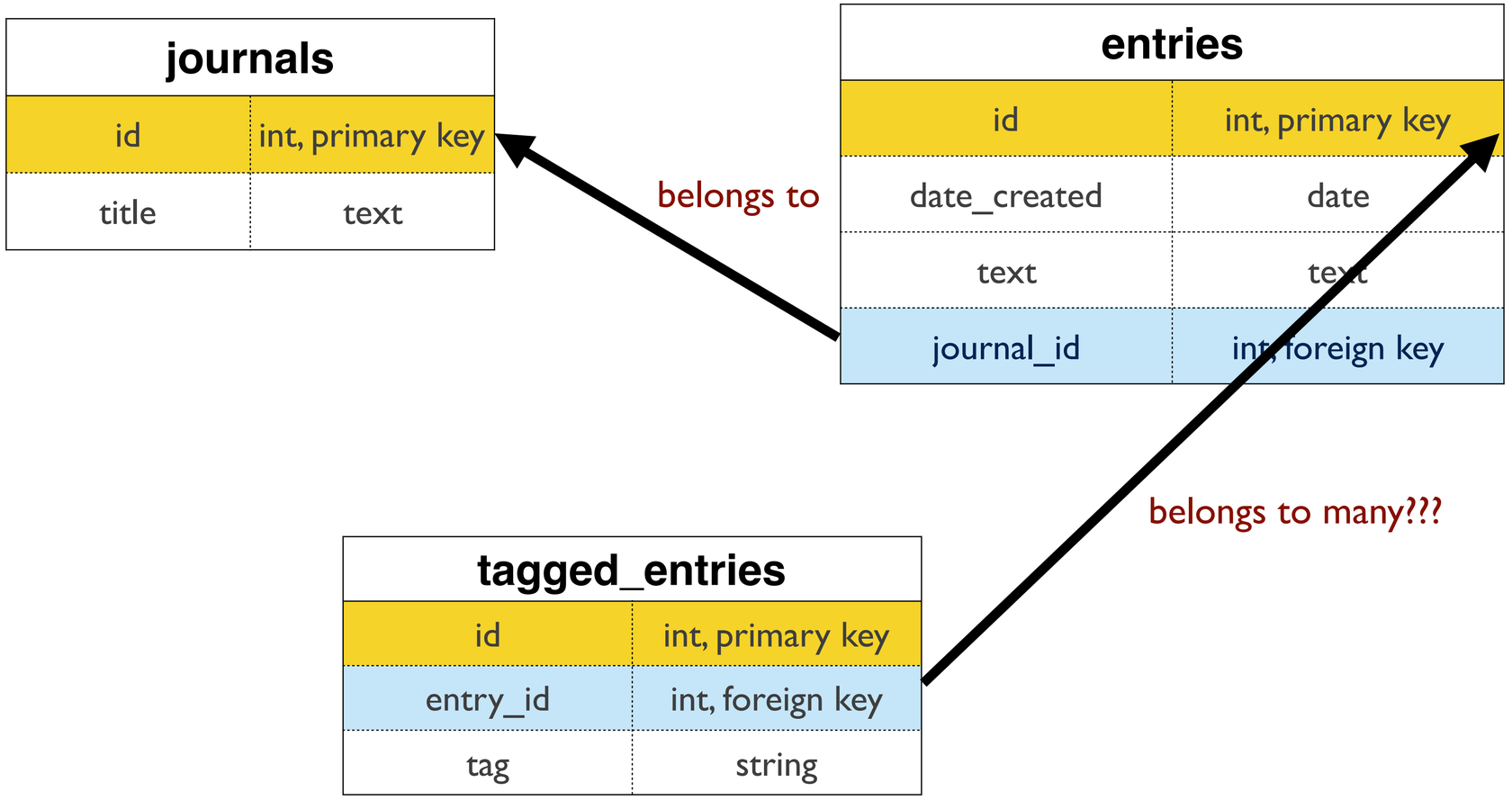
- Eliminate repeating groups in individual tables
- Create a separate table for each set of related data
- Identify each set of related data with a primary key



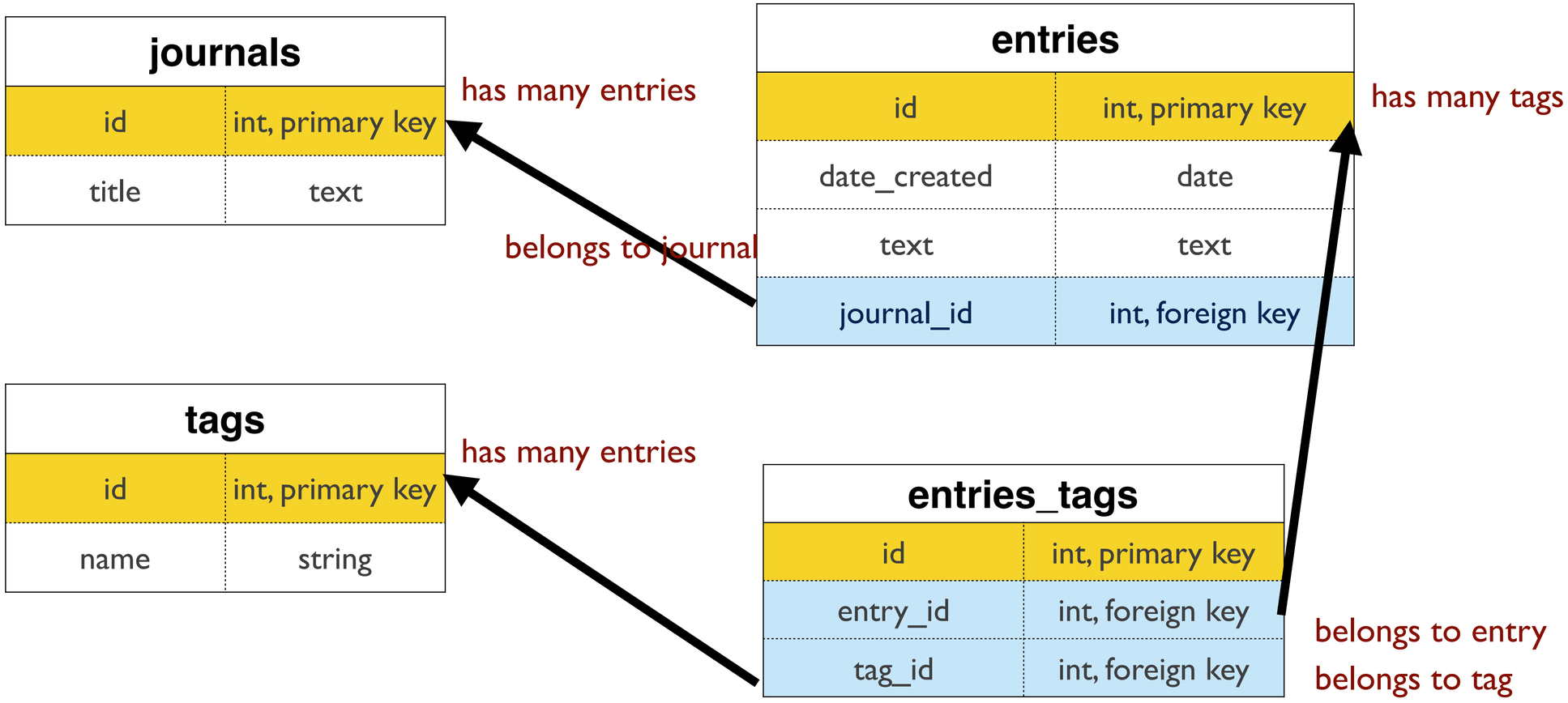
Example: A Journal
...But what about tags?!



Example: A Journal
...What about now?



Example: A Journal Normalized!



Example: A Journal Normalized!

journals

id	text
0	Fullstack journey
1	Work
2	Dank Memes
3	Despair is everywhere

entries

id	date_created	text	journal_id
0	2016-04-01	I am happy	3
1	2016-04-02	I am very happy	2
2	2016-04-03	Great weekend	1
3	2016-04-03	Fun times	3

tags

id	text
0	#NoFilterNoMakeup
1	#YOLO
2	#selfie
3	#SadnessIsMyLife

entries_tags

id	tag_id	entry_id
0	2	1
1	3	1
2	1	1
3	3	2

Rule of Thumb

- **For a 1:M relationship:**
  - One of the tables should have a foreign key column. Think about which thing 'belongs to' the other?
  - E.g. dogs belong to owners. Each dog has a collar saying which owner it belongs to. The owner doesn't have a list of the dogs he/she owns.
  - E.g. gloves belong to students. Every glove you have has your name in it. the glove table has the student foreign key in it.
- **For a M:M relationship:**
  - You will need a *join table* which contains the foreign keys of the entities you are joining.

Normalized Databases

- **Focus on optimal storage - often at odds with retrieval speed due to complex queries using complicated joins**
- **Work best when the application is write-intensive and write-load is more than read-load**
  - Tables are usually smaller as data is divided vertically (fast reads on single tables)
  - Updates and Inserts are fast because there are no duplicates to update
  - Data is not duplicated so there is less of a need for process intensive group by or distinct queries
- **Normalized tables mean join tables, which mean read operations on multiple tables suffer (indexing strategies don't work as well with joins)**

Denormalized

- **Works best when the application is read-intensive**
  - The data is present in the same table (no need for joins)
  - A single table with all required data allows for efficient index usage
- **Data is duplicated which means that updates and inserts become complex and costly**

What Do I Do?!

- Real world applications will most likely have both read-loads and write-loads
- Utilize both approaches depending on the situation!
- Also, let your DBA handle most of this…

Design one!

- Twitter
- Gmail
- Facebook
- Instagram

- Wordpress
- Wikipedia
- AirBnB
- Google (search)

Steps for Developing your ERD

1. Identify Entities
2. Define Relationships
3. Draw Rough-Draft ERD
4. Fill in Cardinality/Modality (arrows with relationship type)
5. Define Primary Keys
6. Draw Key-Based ERD (labeling Primary and Foreign Keys)
7. Identify Attributes
8. Map Attributes
9. Draw fully attributed ERD