

# Schema Design

---

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

# 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).

# Schemas

- **Table Schema (i.e. relation schema)**
  - What is the table called?
  - What columns does it have? What are their data types?
- **Database Schema**
  - What tables are in the database?
  - How are tables related?

# 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**.

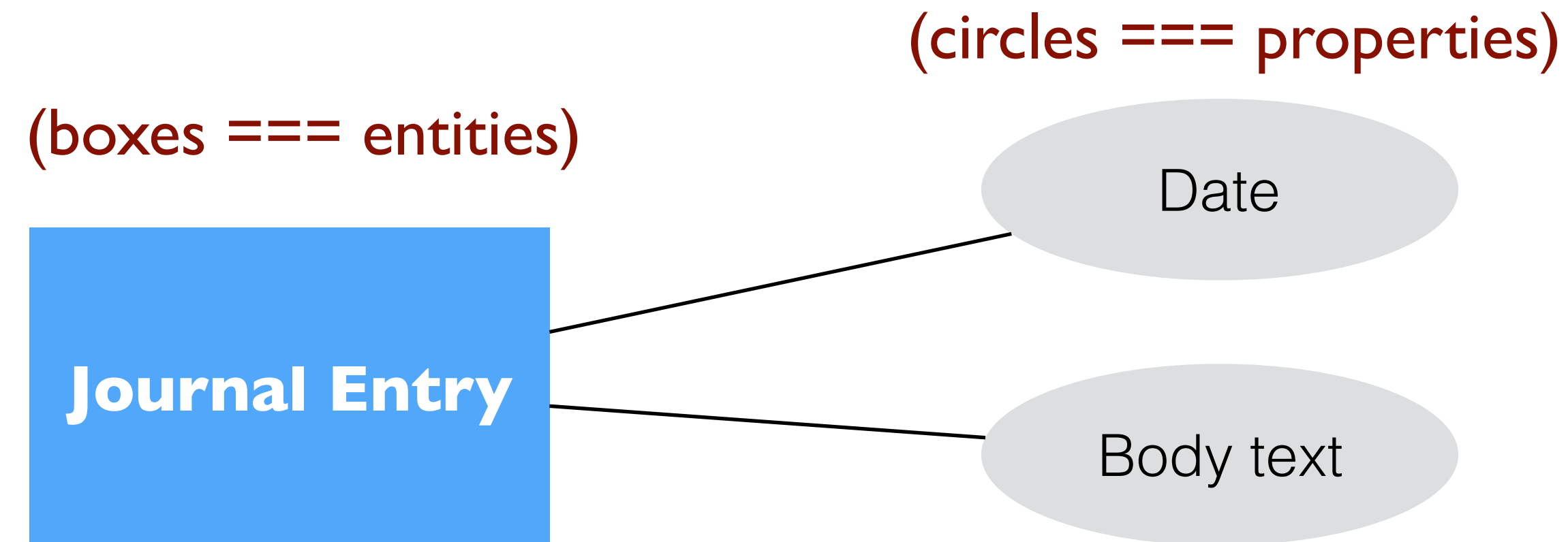
# Designing a Schema

- **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

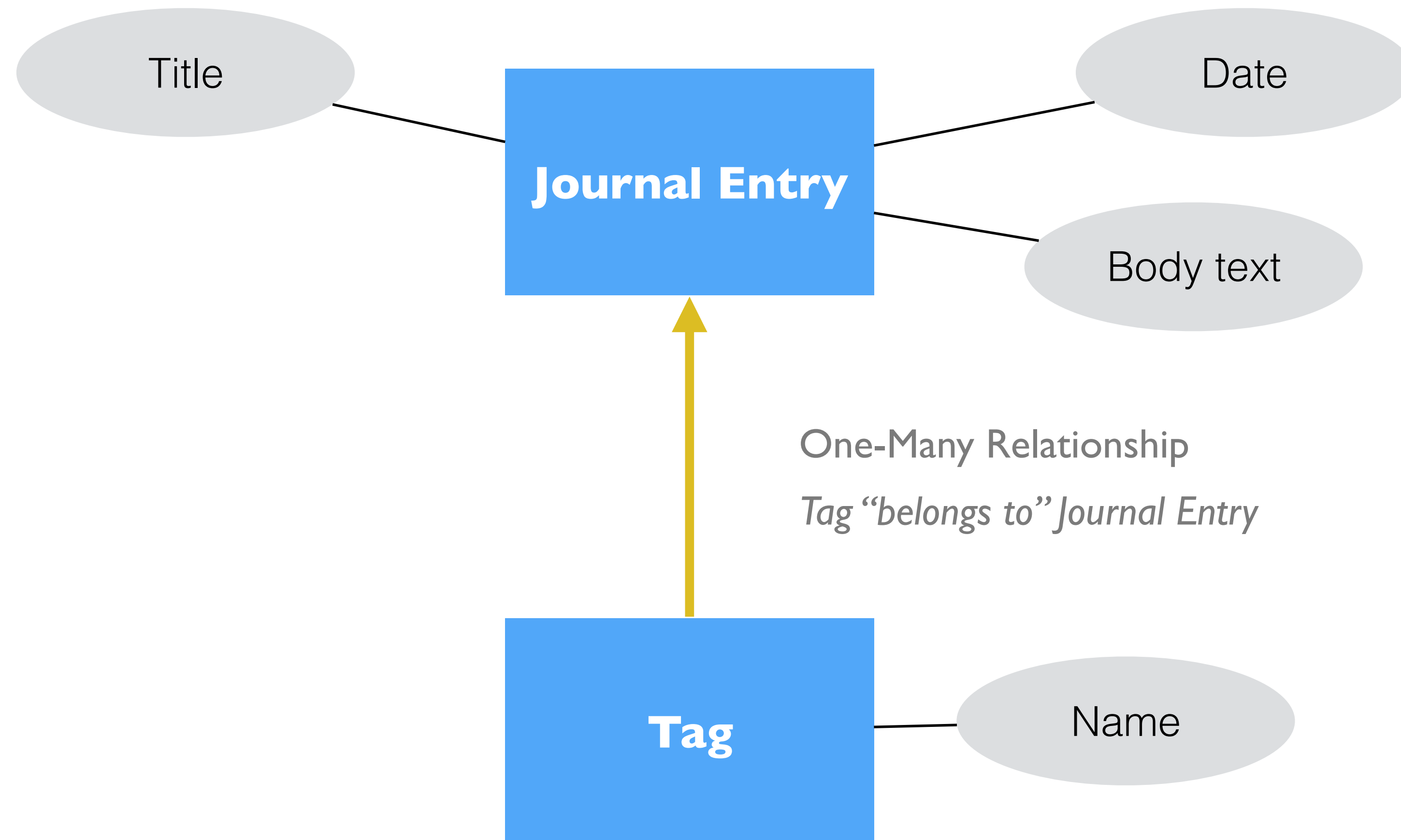
entries	
id	int, primary key
date_created	date
text	text

# 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



# Example: A Journal

## Logical Design: Take 2

tagged_entries	
id	int, primary key
entry_id	int, foreign key
tag	string



entries	
id	int, primary key
date_created	date
text	text
journal_title	text

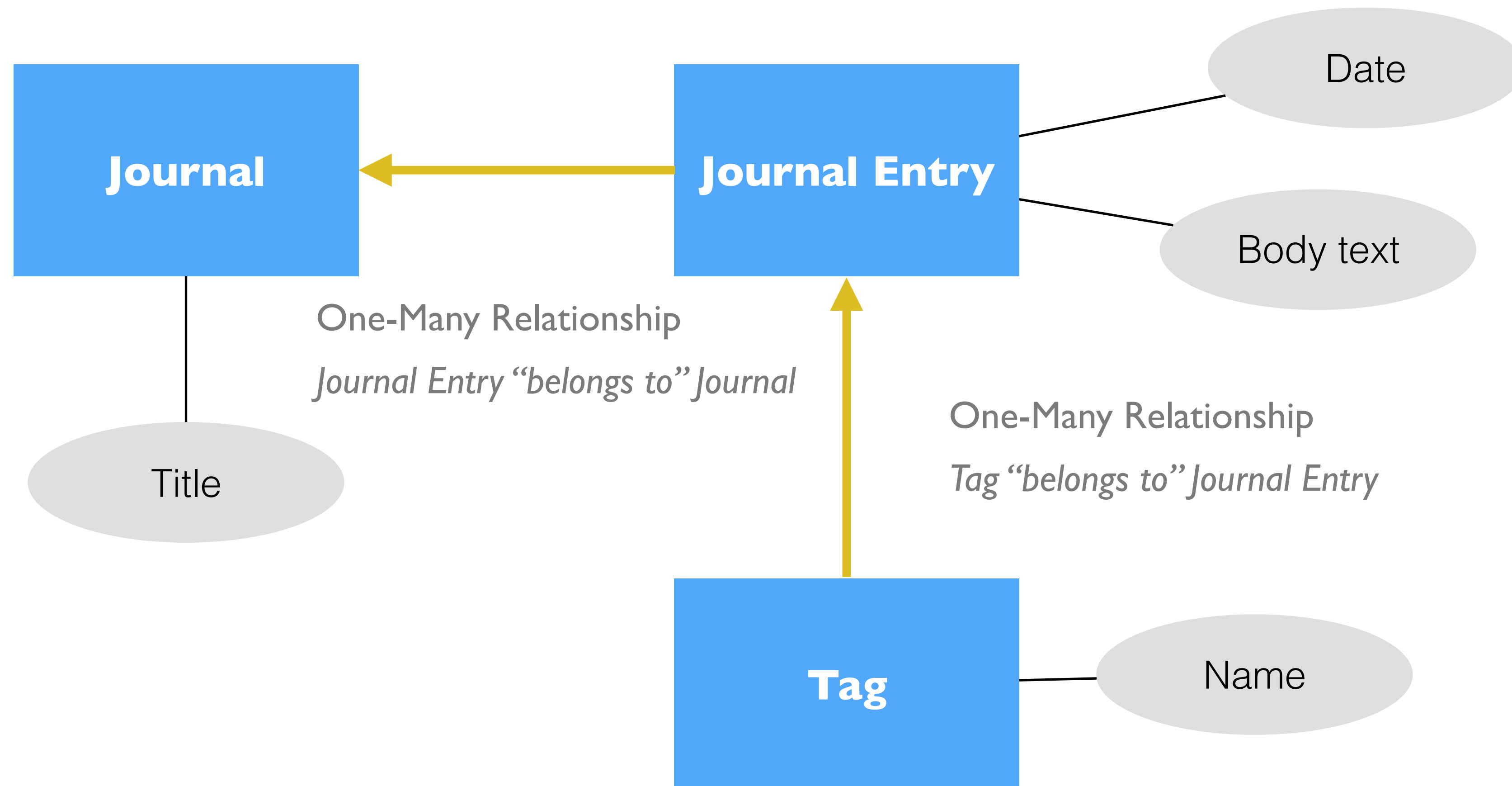
# But Wait!!!

## Normalization

- Organization that minimizes data redundancy and improves data integrity
- 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

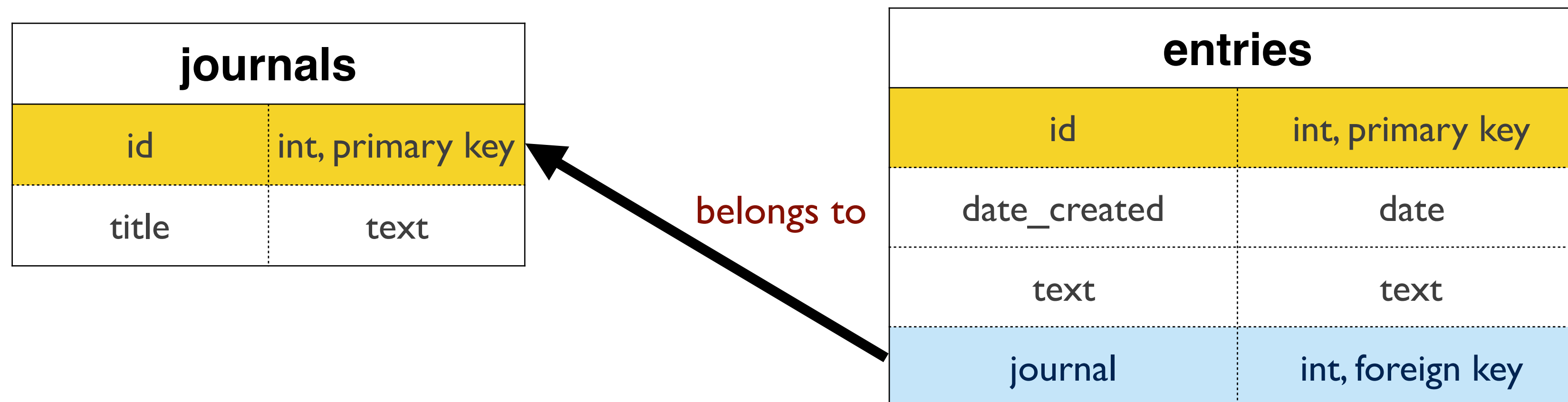
# Conceptual Design, Take 3



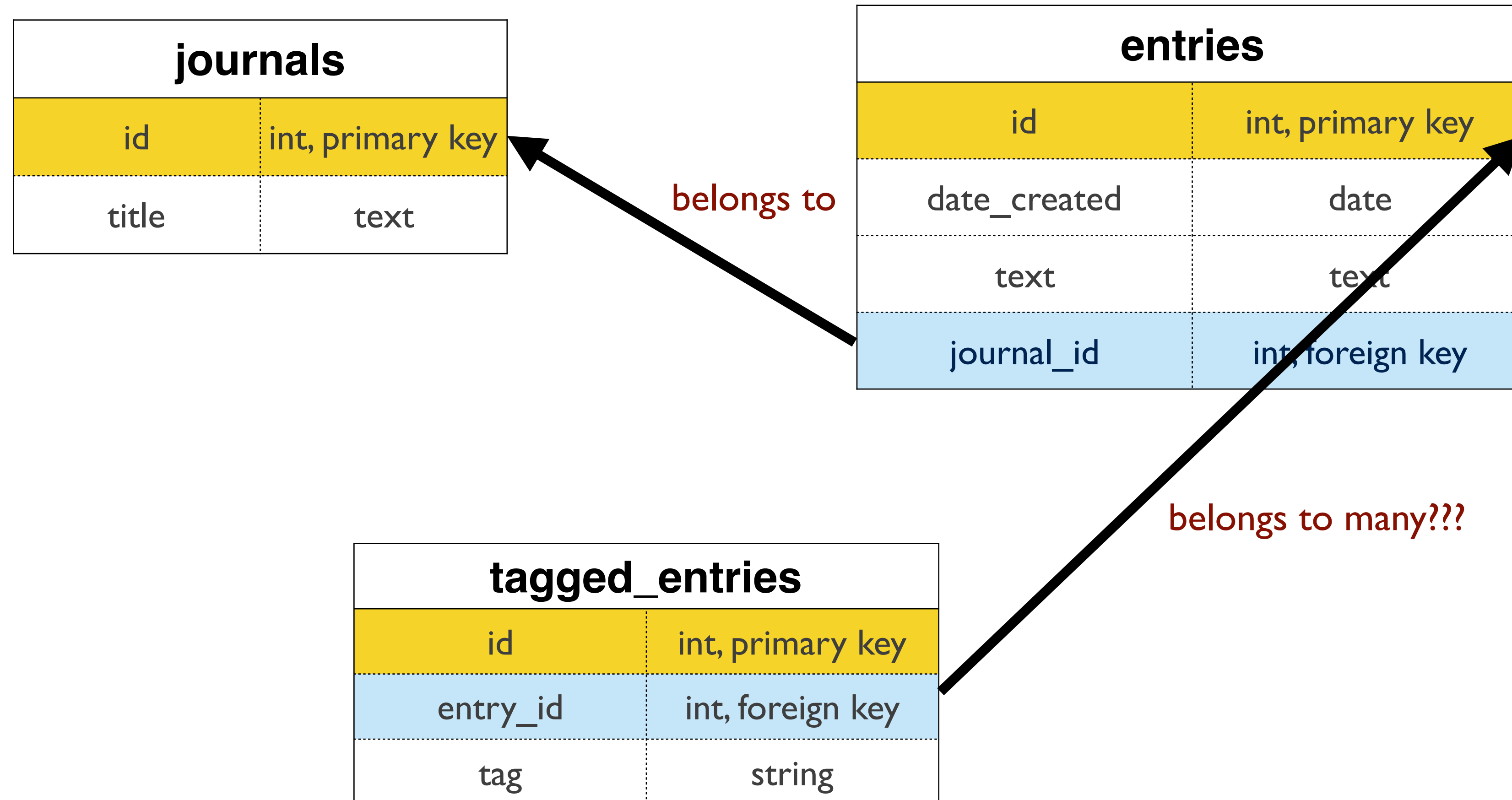


# Logical Design, Take 3

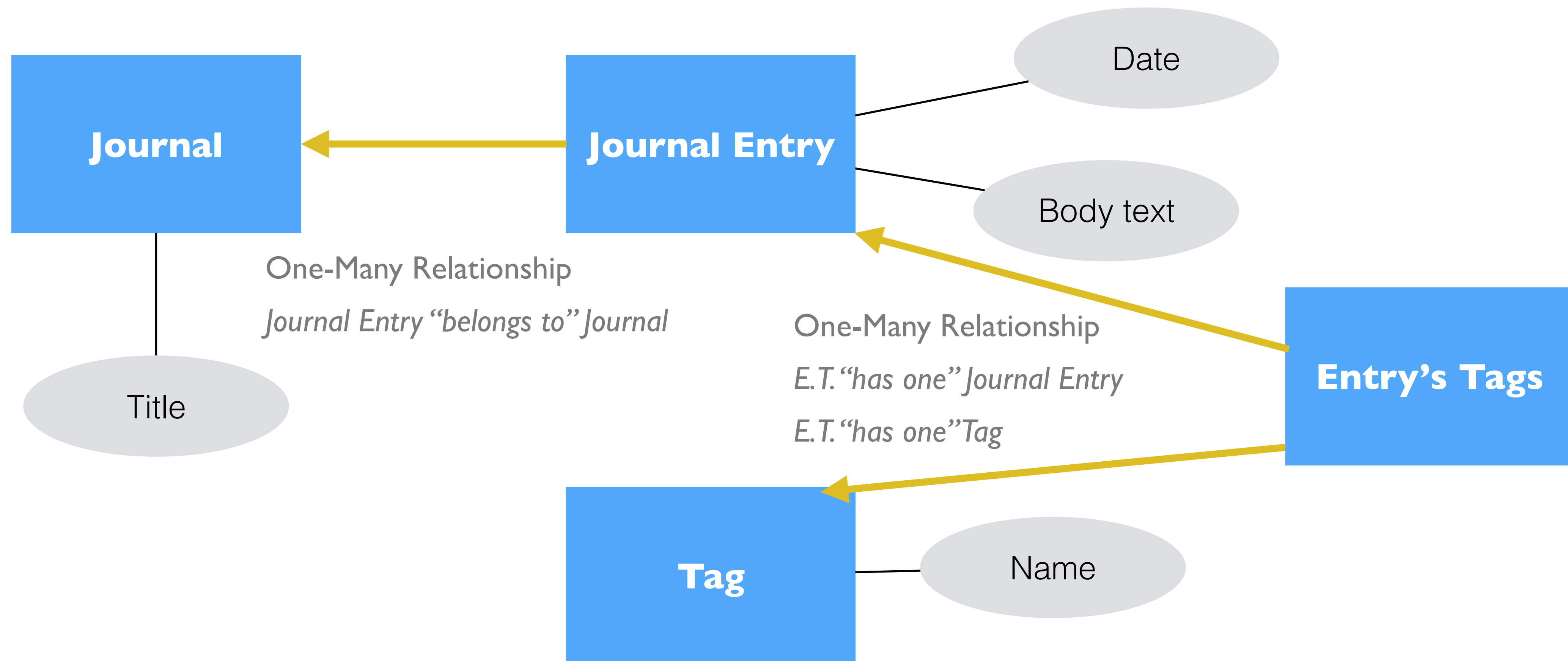
- Eliminate repeating groups in individual tables
- Create a separate table for each set of related data
- Identify each set of related data with a primary key



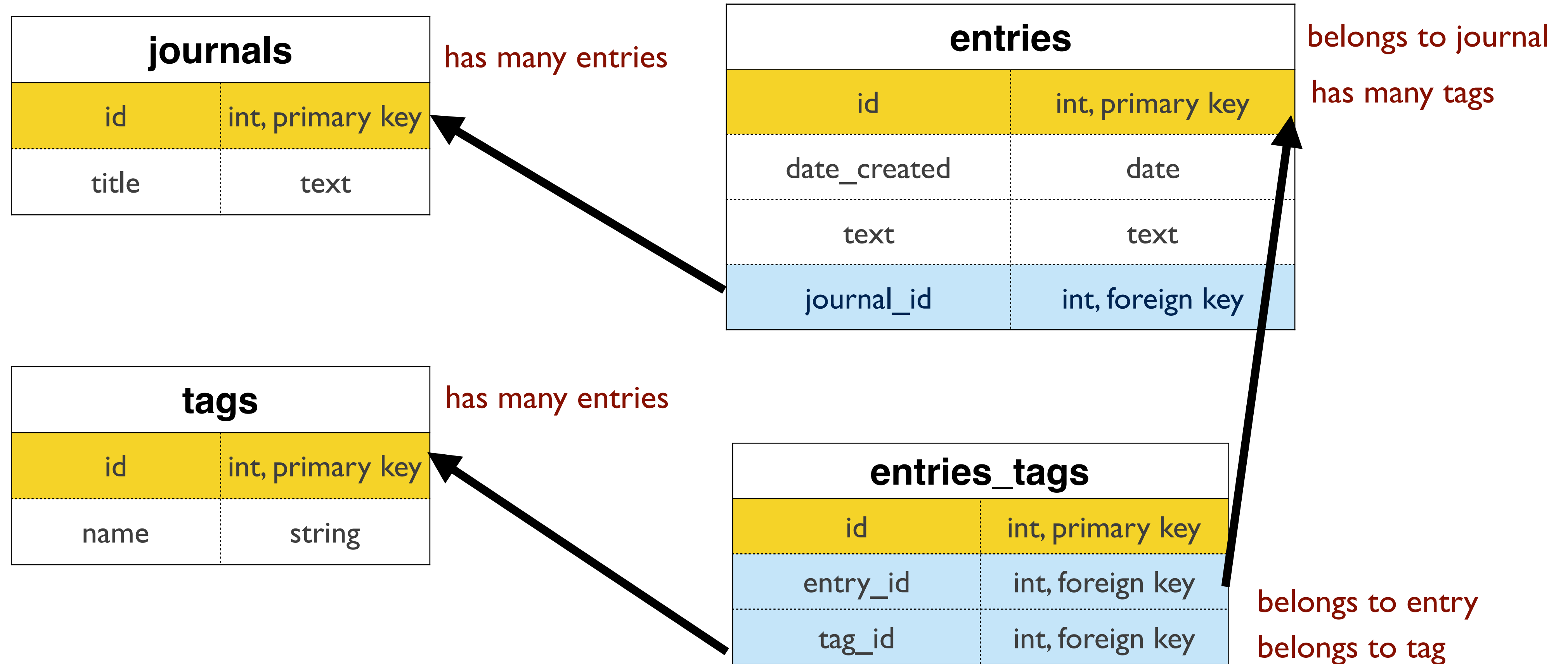
# But what about tags?!?



# Conceptual Design, Take 4



# Logical Design, Take 4



# Logical Design, Take 4

SELECT * FROM entries			
id	date_created	Text	journal_id
0	2016-04-01	I am happy	0
1	2016-04-02	I am very happy	0
2	2016-04-03	Despair fills me	0
3	2016-04-03	Sadness is my life	1

SELECT * FROM journals		
id	date_created	Title
0	2016-04-01	happy times
1	2016-04-02	an anatomy of pain

SELECT * FROM tags-entries	
tag_id	entry_id
0	0
0	1
1	0

SELECT * FROM tags		
id	date_created	Tag
0	2016-04-01	#YOLO
1	2016-04-02	#LOVELIFE

# Has vs. Belongs To

- If rows in table A “belong to” rows in table B, that means A contains a foreign key for B
- If rows in table A “have” one or many rows in table B, that means table B is responsible for keeping track of the foreign key
- Think: the “owner” has less to worry about

# Relationships

- **Has One/Belongs To**
  - Author has one Journal
  - Journal belongs to an Author
- **Has Many/Belongs To**
  - Entries belong to a Journal
  - A Journal has many Entries
- **Belongs To Many**
  - Tags and Journal Entries

# 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!
- Befriend your local DBA

# 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. Label Foreign Keys
7. Identify and Map Attributes

# Design one!

- **Twitter**
- **Gmail**
- **Facebook**
- **Instagram**
- **Wordpress**
- **Wikipedia**
- **AirBnB**
- **GitHub**
- **Youtube**
- **Spotify**
- **Slack**
- **Google (search)**