Data Representation and Intro SQL

January 28, 2020
Data Science CSCI 1951A
Brown University
Instructor: Ellie Pavlick
HTAs: Josh Levin, Diane Mutako, Sol Zitter

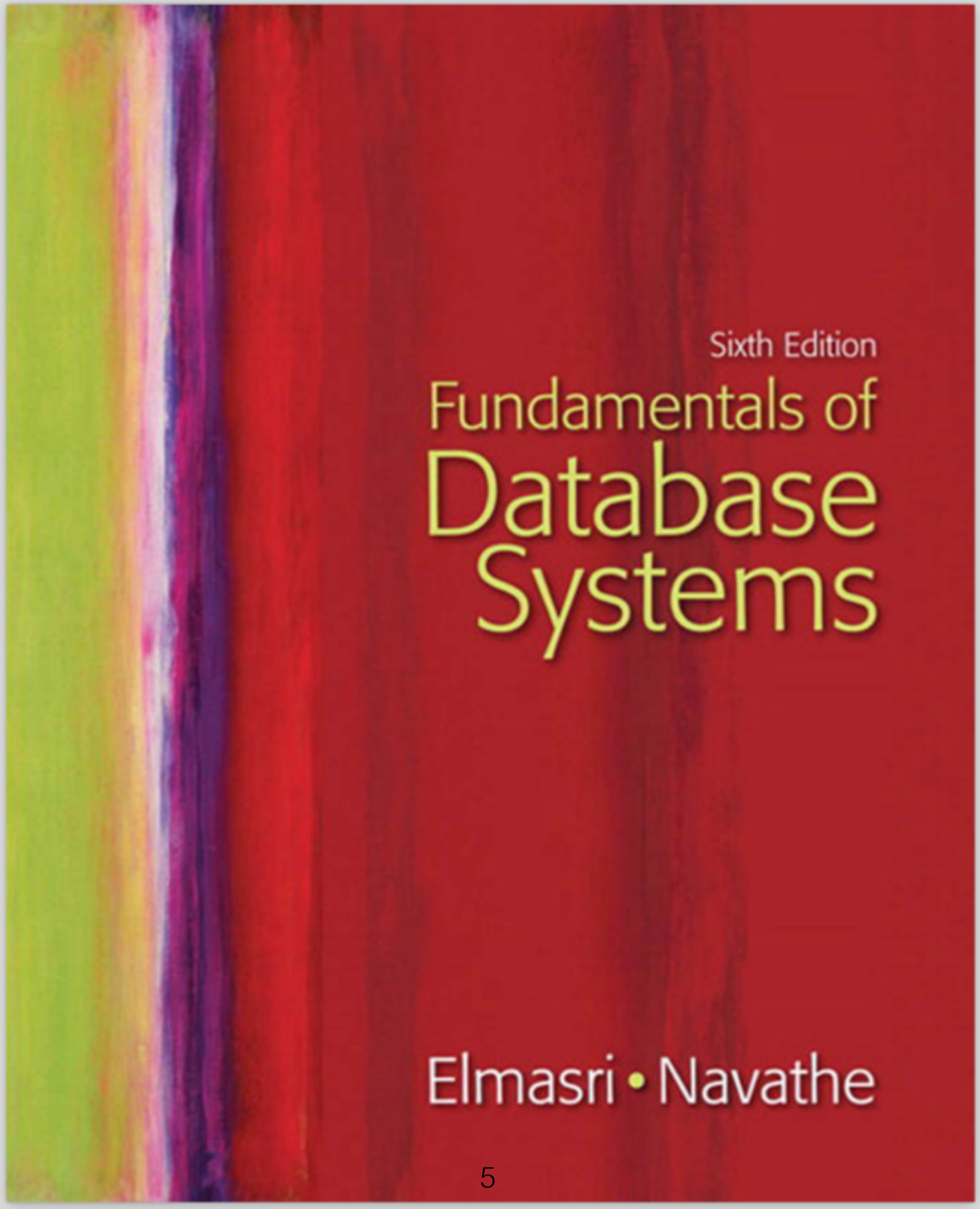
But first!

- Waitlist—we are working our way through
- Top Hat; Have pen/paper or sit by someone who does—this will help for working through longer in-class exercises
- Start/end times. Please don't leave early! 3 minutes per day = one whole lecture! 😩
- Sign the collab policy!
- SQL out, labs starting this week
- Project Mixer next week
- I have office hours today—come say hi! Talk to me about project ideas

But first!

Burning Questions?

Data Representation and Intro SQL


Sixth Edition
Fundamentals of
Database
Systems
Elmasri • Navathe

DATABASES FOR DATA SCIENTIST

Requirement Engineering	Conceptual Modeling	Logical and Physical Modeling	Asking and Answering Questions (Analysis)
"Book of Duty"	Conceptual Design (ER)	Logical Design (schema, table names, data types), Physical Design (indices, memory layout, optimizations)	Relational Algebra, SQL

https://cs.brown.edu/courses/csci1951-a/res/static/lec02_databases.pdf

DATABASES FOR DATA SCIENTIST

Requirement Engineering	Conceptual Modeling	Logical and Physical Modeling	Asking and Answering Questions (Analysis)
"Book of Duty"	Conceptual Design (ER)	Logical Design (schema, table names, data types), Physical Design (indices, memory layout, optimizations)	Relational Algebra, SQL

https://cs.brown.edu/courses/csci1951-a/res/static/lec02_databases.pdf

“Book of Duty”/“Miniworld”

- Informal description of data domain
- In natural language:
  - What are the objects you care about?
  - What properties/attributes of those objects are you measuring?
  - What are the relationships between them?
  - What assumptions are we making? (E.g. sizes, cardinalities)
  - What is the workload on the database (Read-only? Read/write?)
  - Permissions and privacy concerns?


The most distinctive jargon in U.S. job listings
through the roof
revolutioneering
strategic initiative
drives change
face time
go the extra mile
corporate values
achieve alignment
strategic communication
reach out to you
blaze a trail
eye of the tiger
herd cats
thought leaders
increase productivity
learnings
overachiever
exit strategy
message alignment
out of the park
change driver
identify opportunity
company value
off the floor
not intimidated
possess strength
build synergy
close the loop
bleeding edge
driven results
touch base
good practice
buy in
value add
hit your numbers
statement of duties
synergize
magic happens
textio

Description of “Miniworld”

- Project: We want to analyze political trends surrounding 2020 primary candidates

- Plan: Crawl Twitter for posts from or about 2020 primary candidates. We want to analyze the spread of opinions, through through following/follower relationships, share/retweet chains, and language use

Description of “Miniworld”

- Objects used:
- Domains of attributes of objects:
- Identifiers, references / relationships:
- Cardinalities:
- Distributions:
- Workload:
- Priorities and service level agreements:

Description of “Miniworld”

- Objects used: People, Tweets, Candidates
- Domains of attributes of objects:
- Identifiers, references / relationships:
- Cardinalities:
- Distributions:
- Workload:
- Priorities and service level agreements:

Description of “Miniworld”

- Objects used: People, Tweets, Candidates
- Domains of attributes of objects: Tweets have timestamps (date), authors (Person), text (max 140 characters), attachments, hashtags...
- Identifiers, references / relationships:
- Cardinalities:
- Distributions:
- Workload:
- Priorities and service level agreements:

Description of “Miniworld”

- Objects used: People, Tweets, Candidates
- Domains of attributes of objects: Tweets have timestamps (date), authors (Person), text (max 140 characters), attachments, hashtags...
- Identifiers, references / relationships: People have unique IDs, People author Tweets, People retweet Tweets, People like Tweets, People follow People, Tweets mention People...
- Cardinalities:
- Distributions:
- Workload:
- Priorities and service level agreements:

Description of “Miniworld”

- Objects used: People, Tweets, Candidates
- Domains of attributes of objects: Tweets have timestamps (date), authors (Person), text (max 140 characters), attachments, hashtags...
- Identifiers, references / relationships: People have unique IDs, People author Tweets, People retweet Tweets, People like Tweets, People follow People, Tweets mention People...
- Cardinalities: Every tweet has exactly one author, one author can have many tweets, a tweet can mention 0 to many candidates, a candidate can be mentioned in 0 to many tweets
- Distributions:
- Workload:
- Priorities and service level agreements:

Description of “Miniworld”

- Objects used: People, Tweets, Candidates
- Domains of attributes of objects: Tweets have timestamps (date), authors (Person), text (max 140 characters), attachments, hashtags...
- Identifiers, references / relationships: People have unique IDs, People author Tweets, People retweet Tweets, People like Tweets, People follow People, Tweets mention People...
- Cardinalities: Every tweet has exactly one author, one author can have many tweets, a tweet can mention 0 to many candidates, a candidate can be mentioned in 0 to many tweets
- Distributions: Some people tweet a lot, others just follow; some candidates are mentioned a lot; follower/followee asymmetries
- Workload:
- Priorities and service level agreements:

Description of “Miniworld”

- Objects used: People, Tweets, Candidates
- Domains of attributes of objects: Tweets have timestamps (date), authors (Person), text (max 140 characters), attachments, hashtags...
- Identifiers, references / relationships: People have unique IDs, People author Tweets, People retweet Tweets, People like Tweets, People follow People, Tweets mention People...
- Cardinalities: Every tweet has exactly one author, one author can have many tweets, a tweet can mention 0 to many candidates, a candidate can be mentioned in 0 to many tweets
- Distributions: Some people tweet a lot, others just follow; some candidates are mentioned a lot; follower/followee asymmetries
- Workload: scrape and populate once, read often
- Priorities and service level agreements:

Description of “Miniworld”

- Objects used: People, Tweets, Candidates
- Domains of attributes of objects: Tweets have timestamps (date), authors (Person), text (max 140 characters), attachments, hashtags...
- Identifiers, references / relationships: People have unique IDs, People author Tweets, People retweet Tweets, People like Tweets, People follow People, Tweets mention People...
- Cardinalities: Every tweet has exactly one author, one author can have many tweets, a tweet can mention 0 to many candidates, a candidate can be mentioned in 0 to many tweets
- Distributions: Some people tweet a lot, others just follow; some candidates are mentioned a lot; follower/followee asymmetries
- Workload: scrape and populate once, read often
- Priorities and service level agreements: “right to be forgotten” rules...

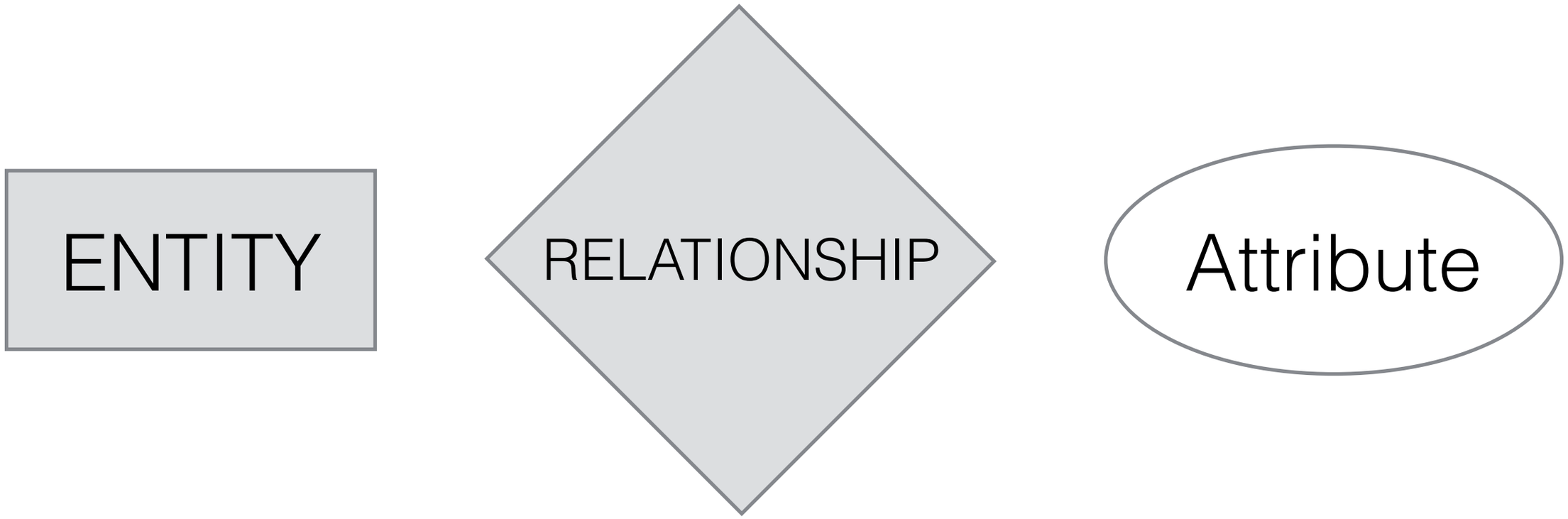
DATABASES FOR DATA SCIENTIST

Requirement Engineering	Conceptual Modeling	Logical and Physical Modeling	Asking and Answering Questions (Analysis)
"Book of Duty"	Conceptual Design (ER)	Logical Design (schema, table names, data types), Physical Design (indices, memory layout, optimizations)	Relational Algebra, SQL

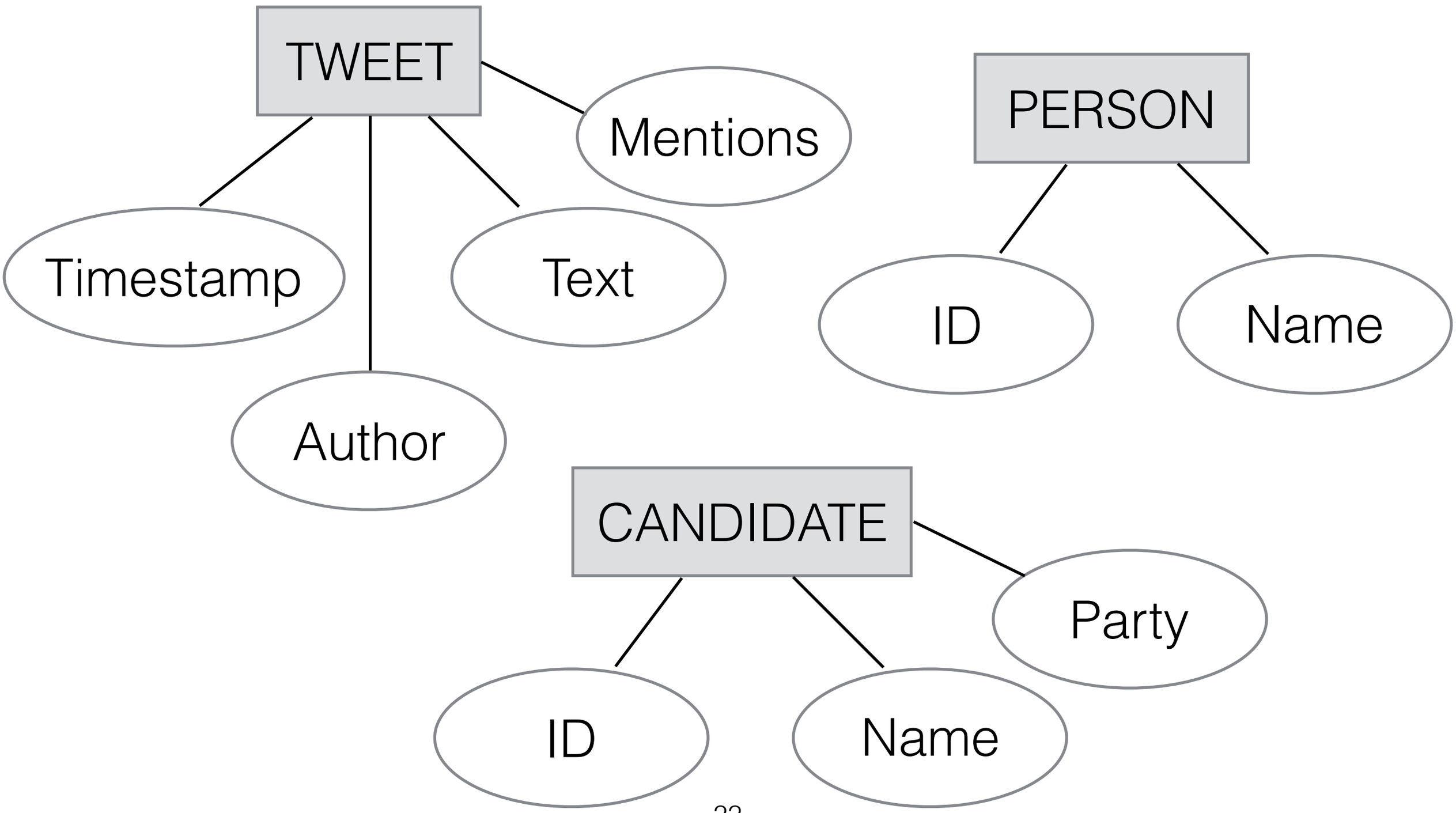
https://cs.brown.edu/courses/csci1951-a/res/static/lec02_databases.pdf

Entity-Relationship (ER) Model

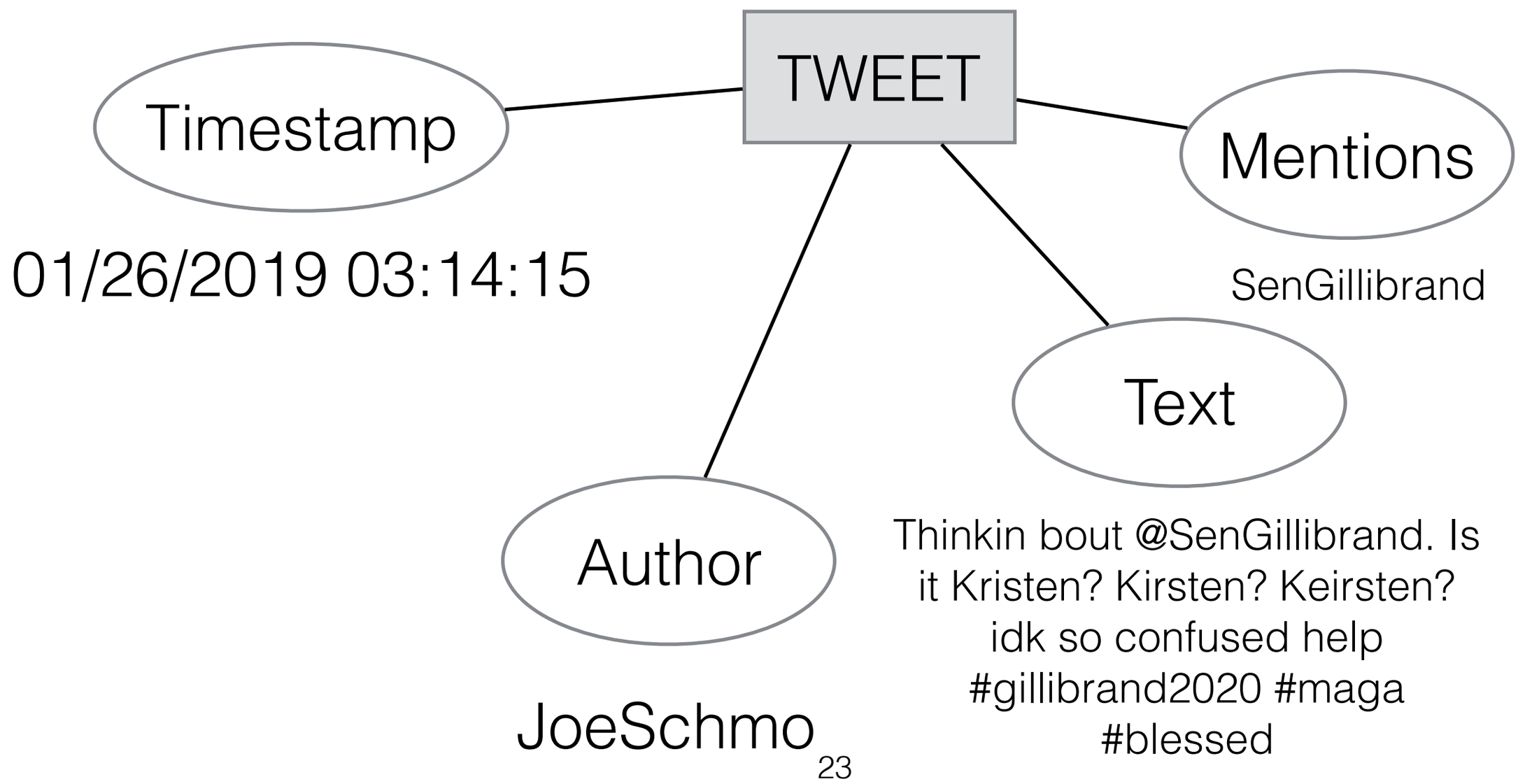
Entity-Relationship (ER) Model



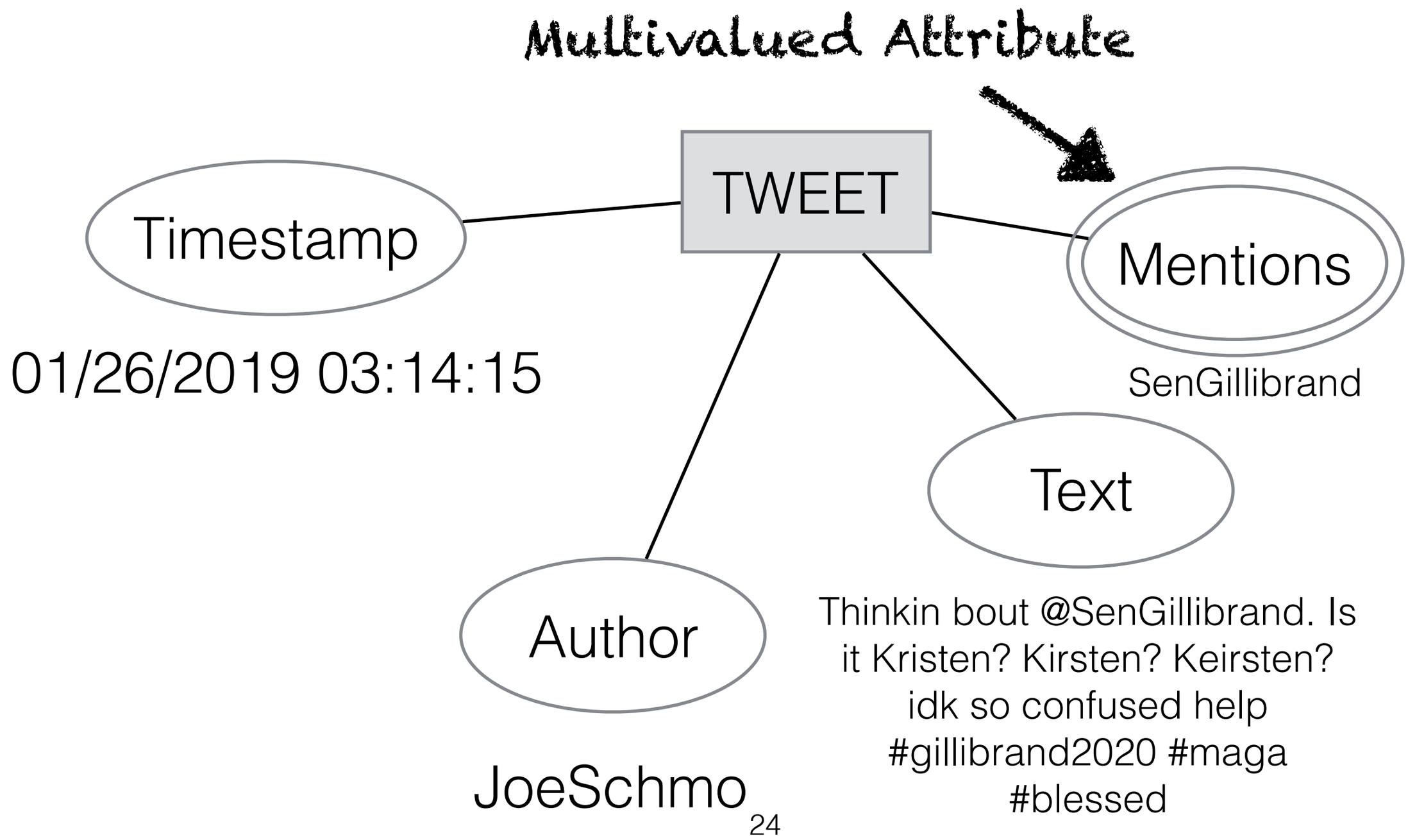
Entity-Relationship (ER) Model



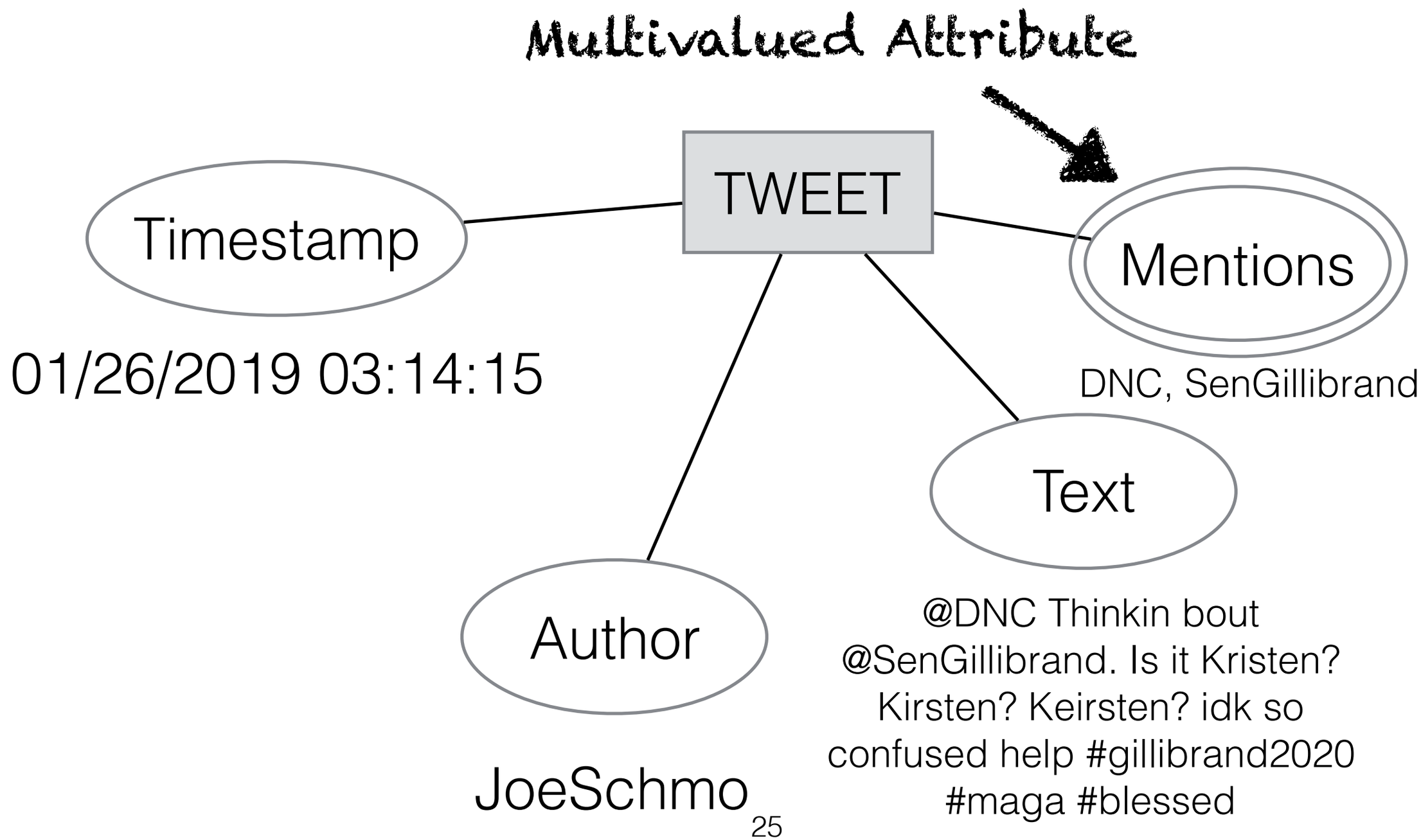
Entity-Relationship (ER) Model



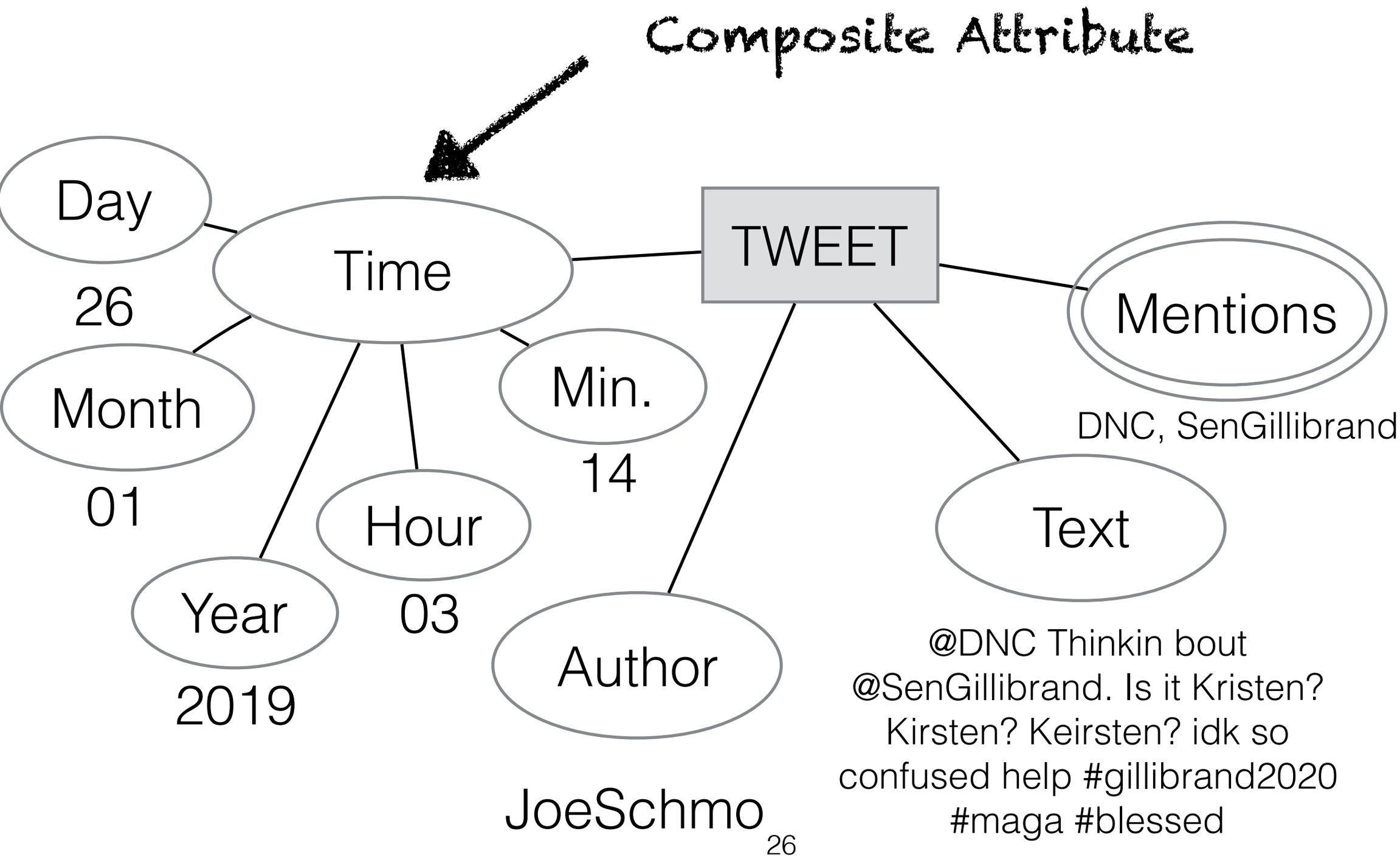
Entity-Relationship (ER) Model



Entity-Relationship (ER) Model



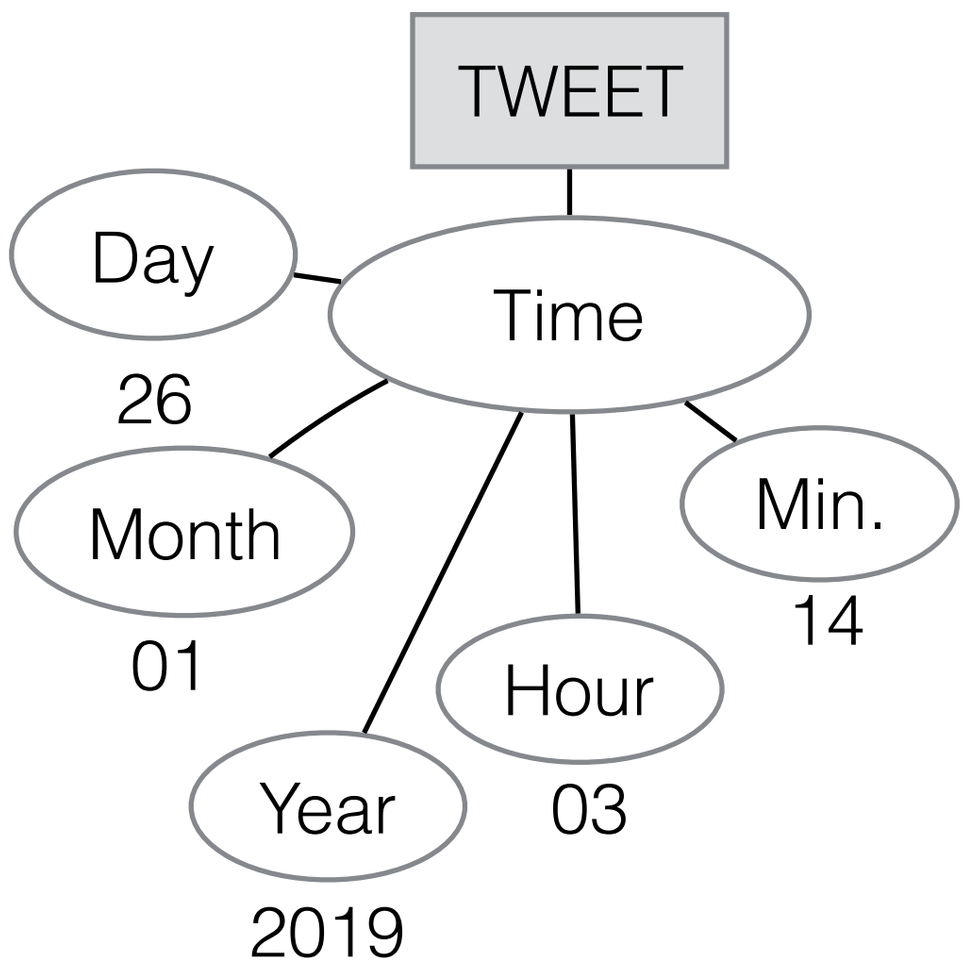
Entity-Relationship (ER) Model



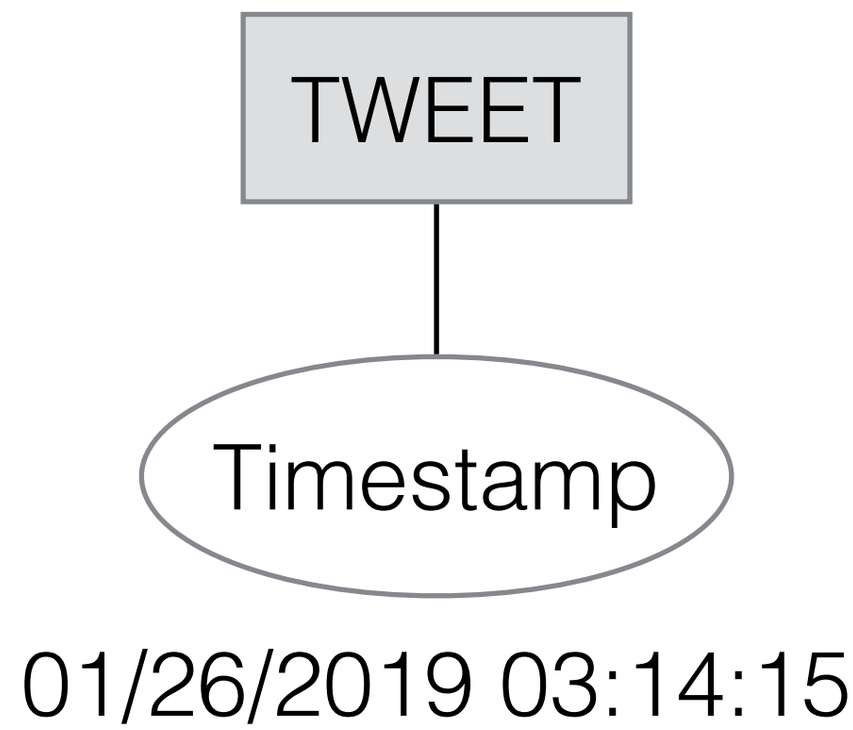
Clicker Question!

Entity-Relationship (ER) Model

Clicker Question!
Which representation is better?



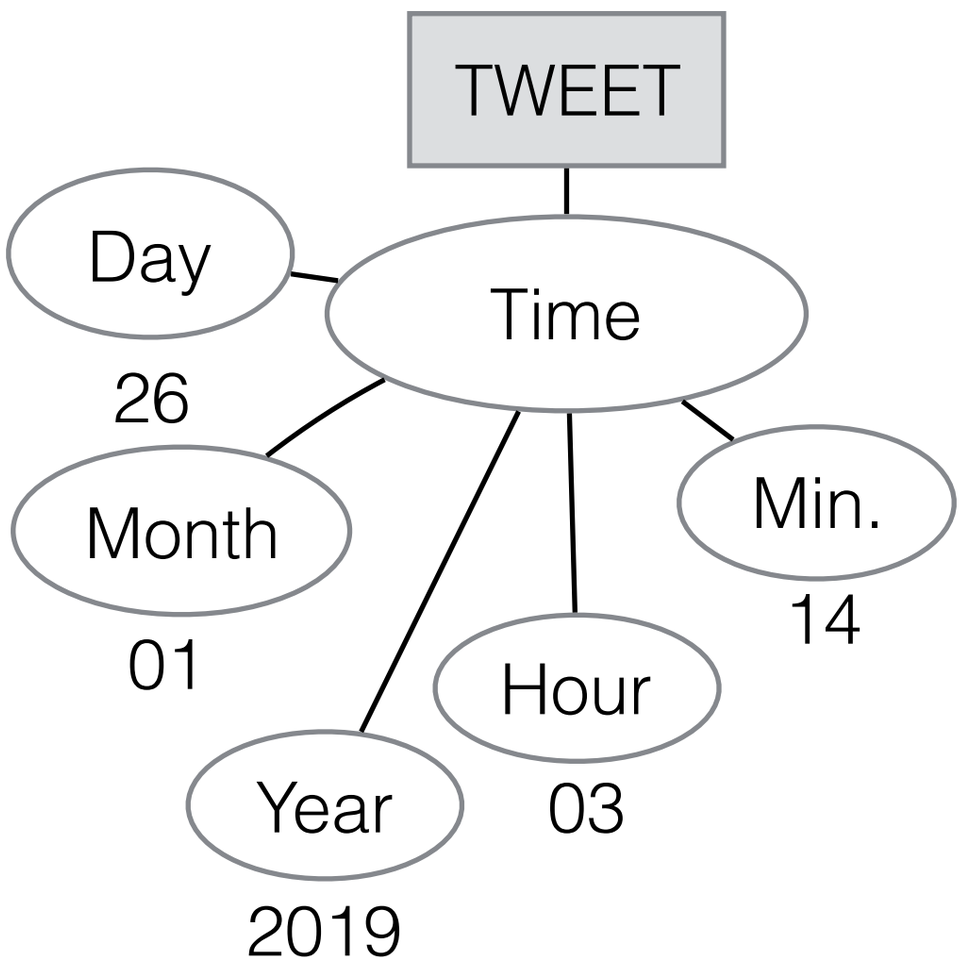
(a) Composite



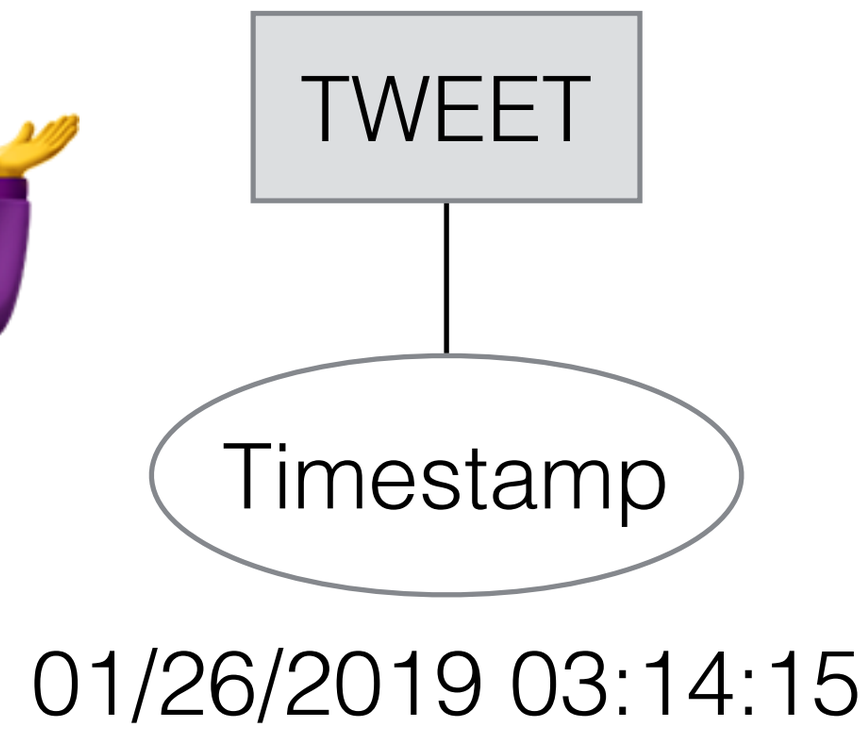
(b) Normal

Entity-Relationship (ER) Model

Clicker Question!
Which representation is better?



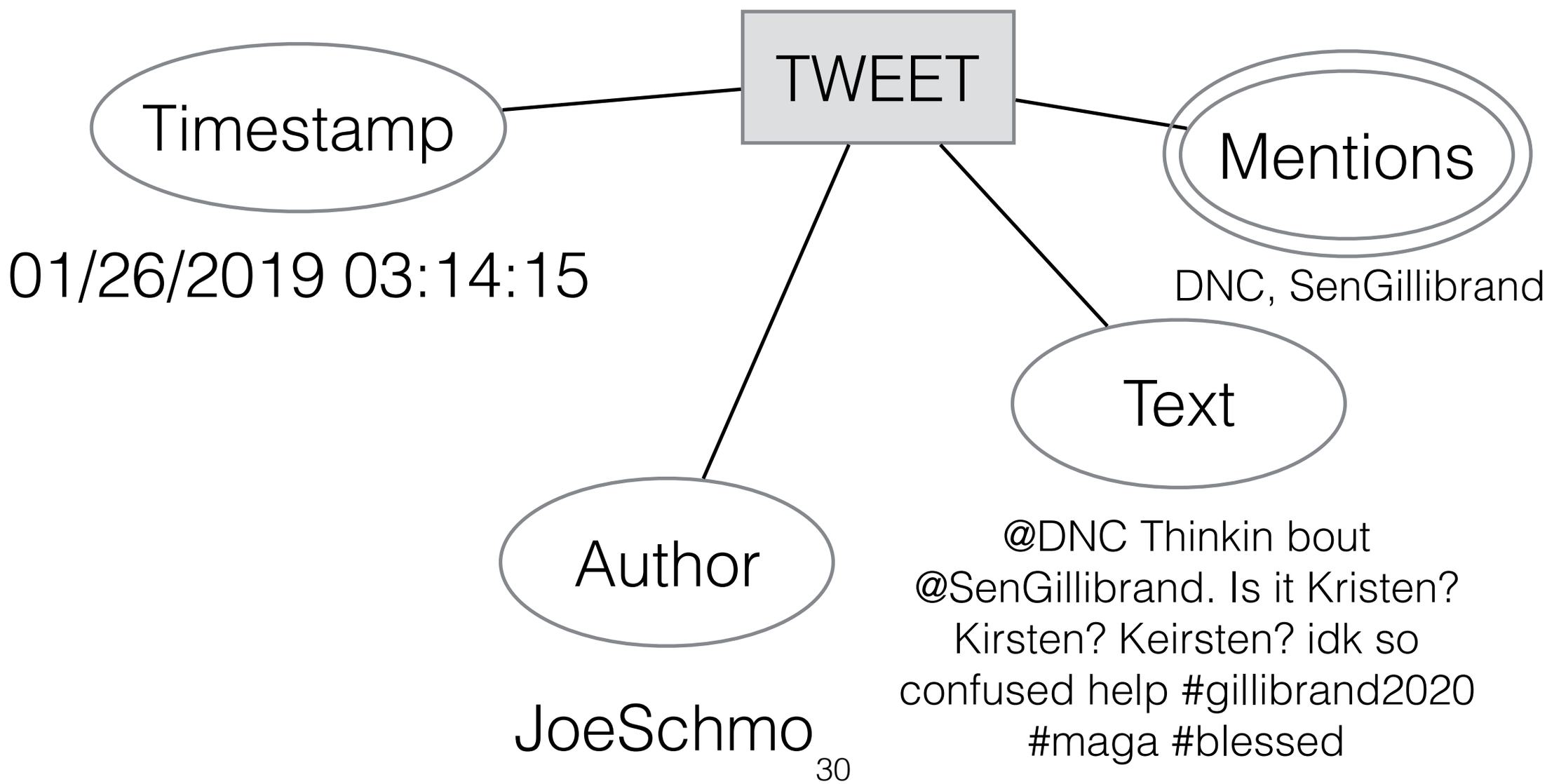
(a) Composite



(b) Normal

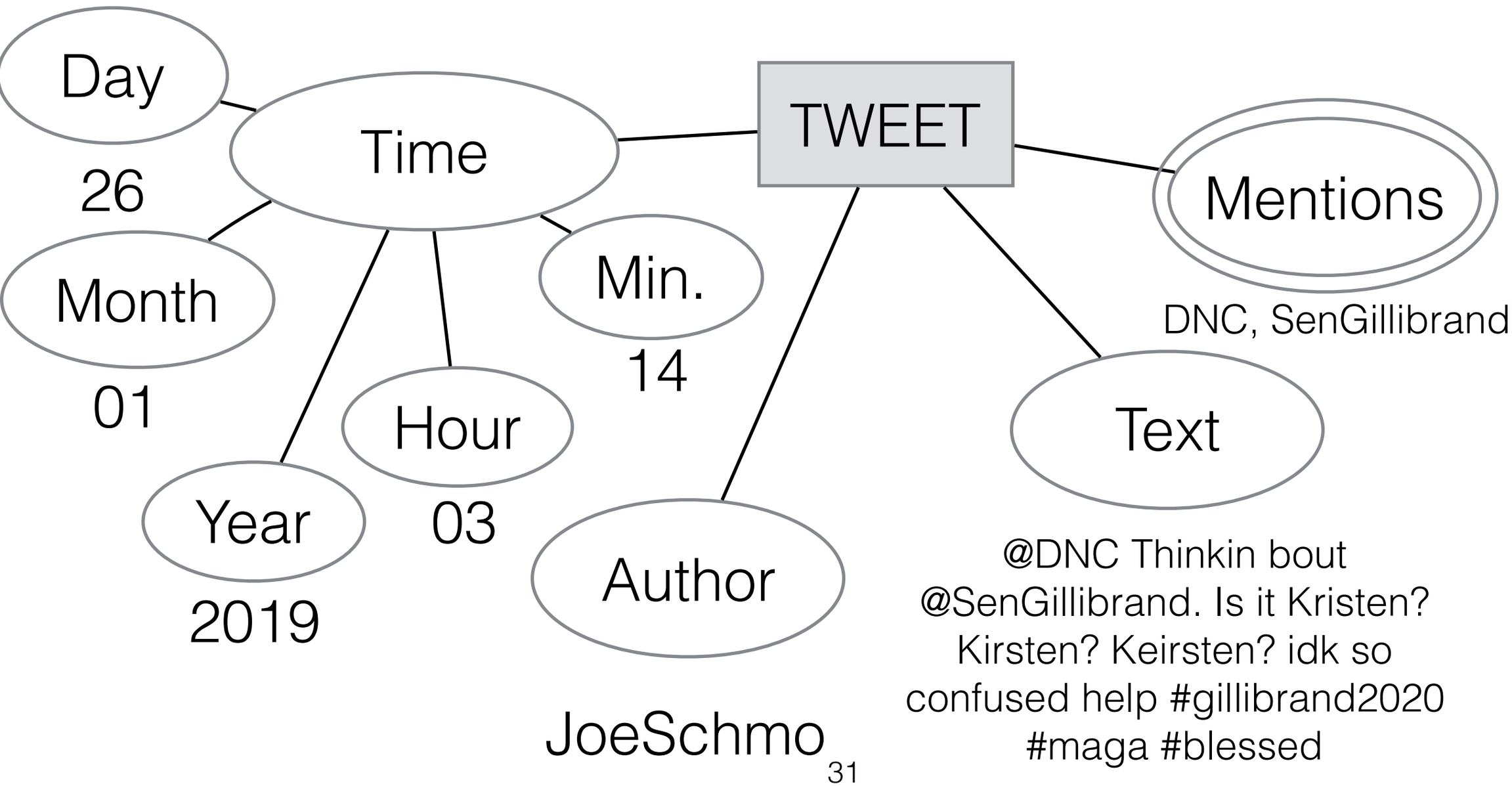
Entity-Relationship (ER) Model

Find all tweets sent between 2am and 4am that mention democratic primary candidates ❌❌❌



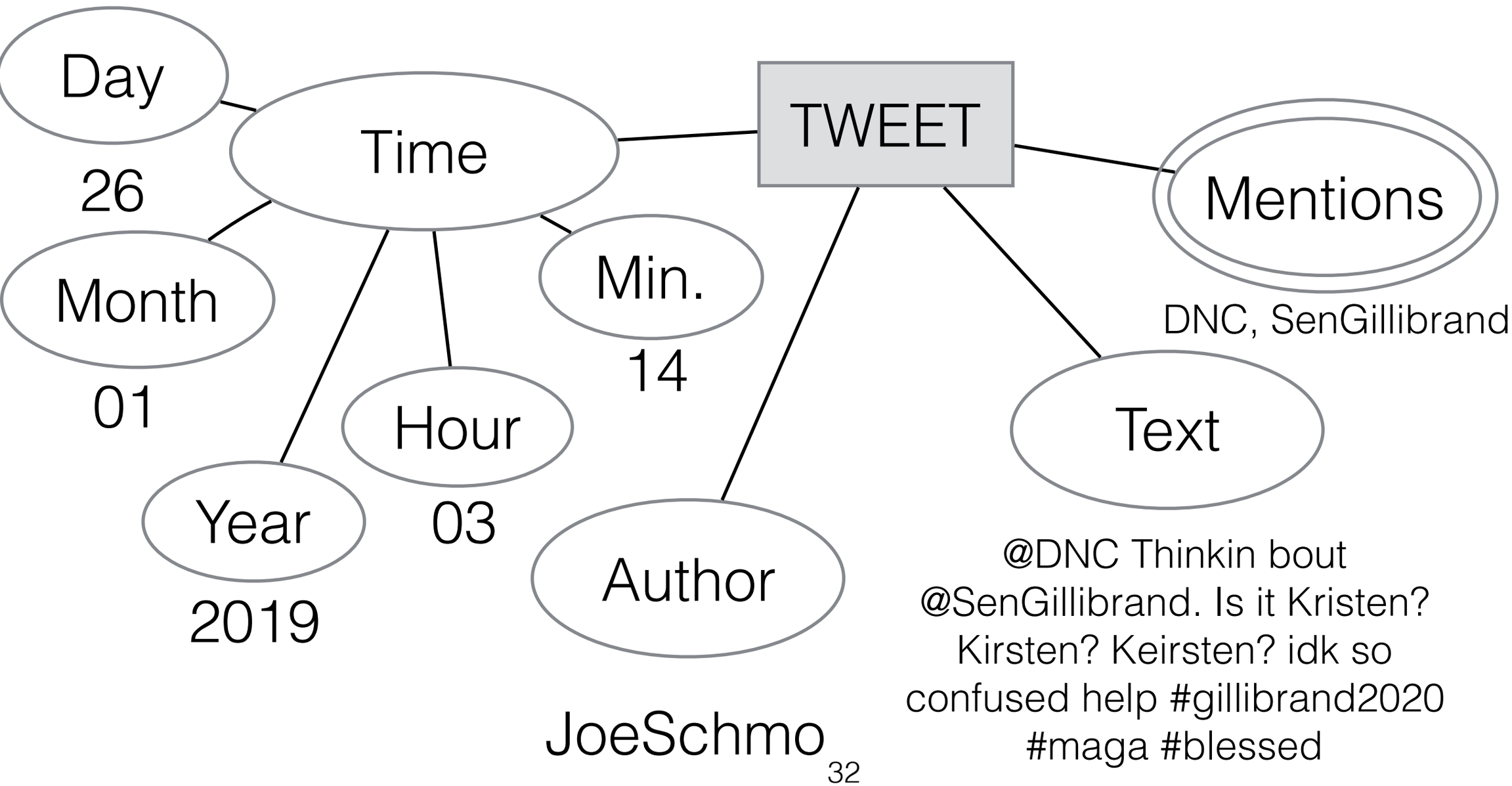
Entity-Relationship (ER) Model

Find all tweets sent between 2am and 4am that mention democratic primary candidates ✔✔✔



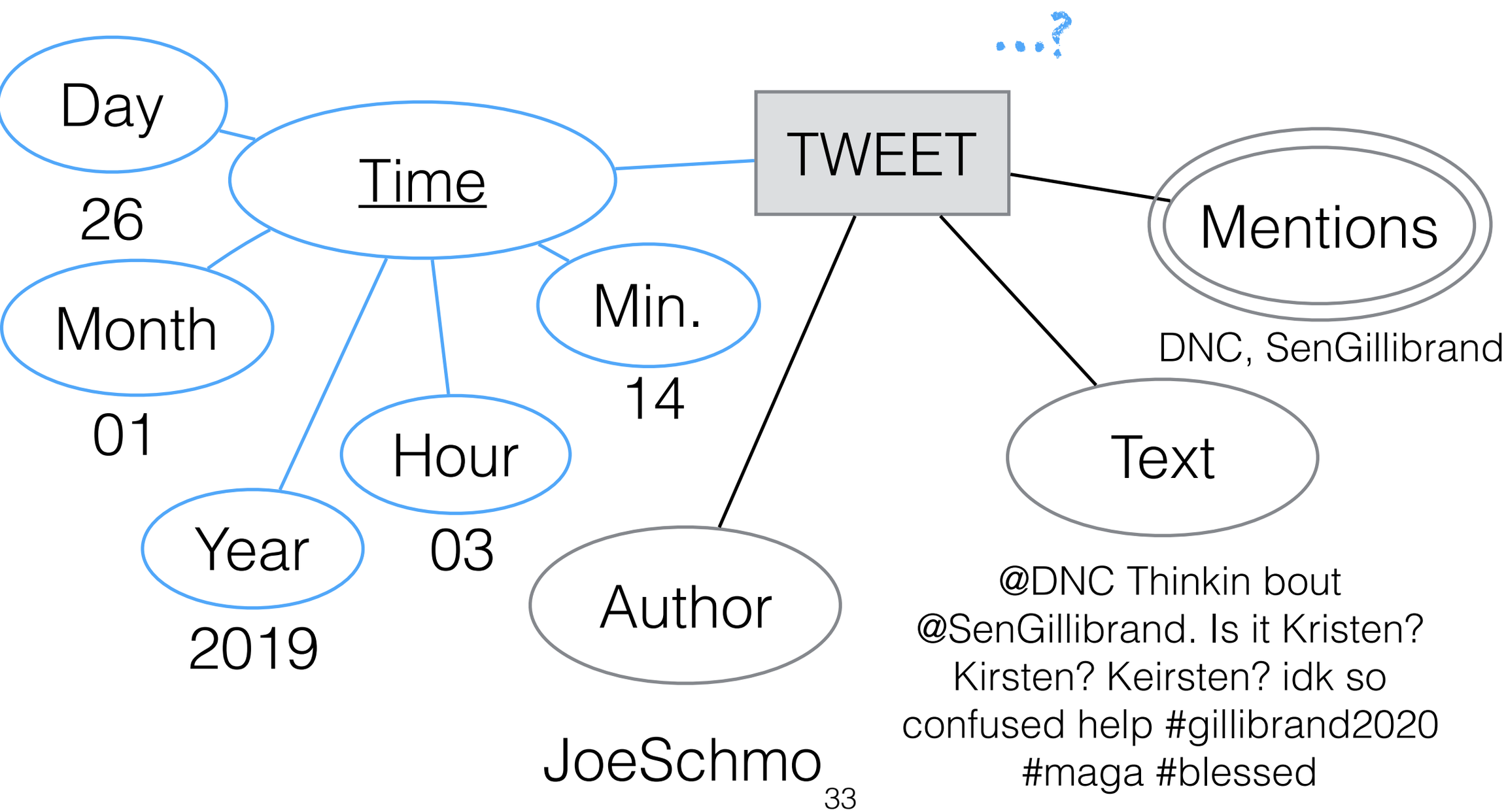
Entity-Relationship (ER) Model

Key Attribute: Designated attribute that uniquely identifies the entry



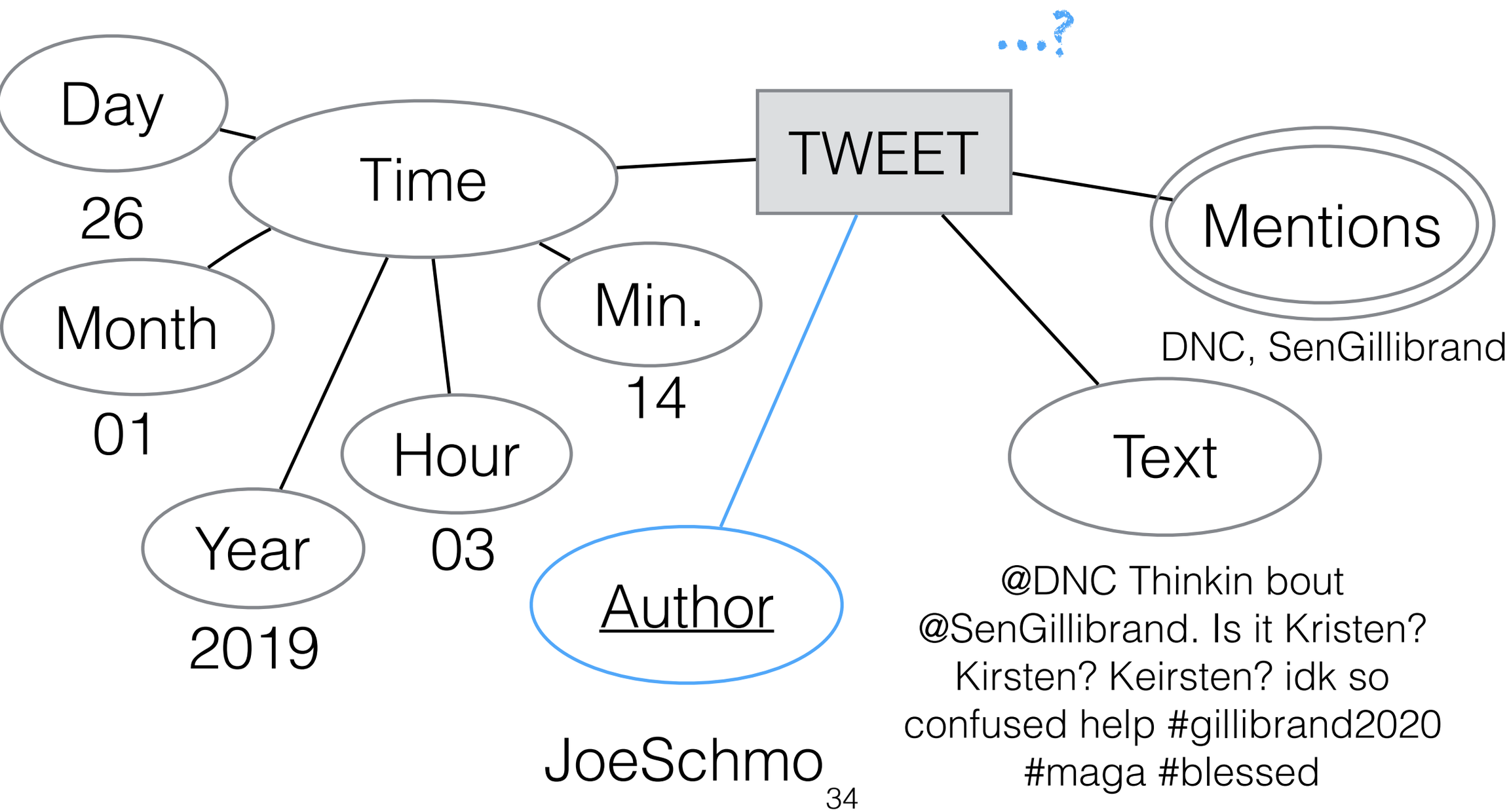
Entity-Relationship (ER) Model

Key Attribute: Designated attribute that uniquely identifies the entry



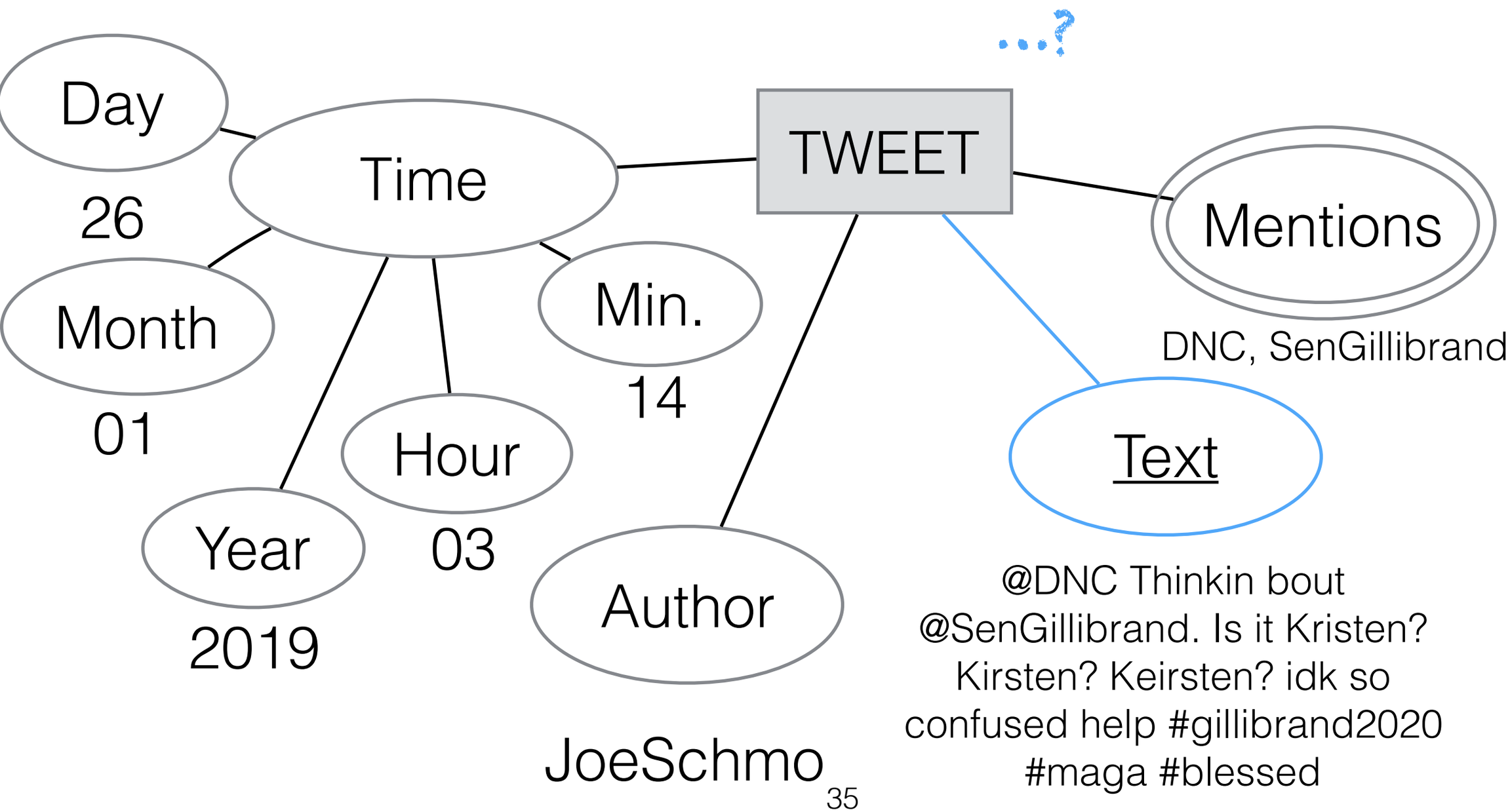
Entity-Relationship (ER) Model

Key Attribute: Designated attribute that uniquely identifies the entry



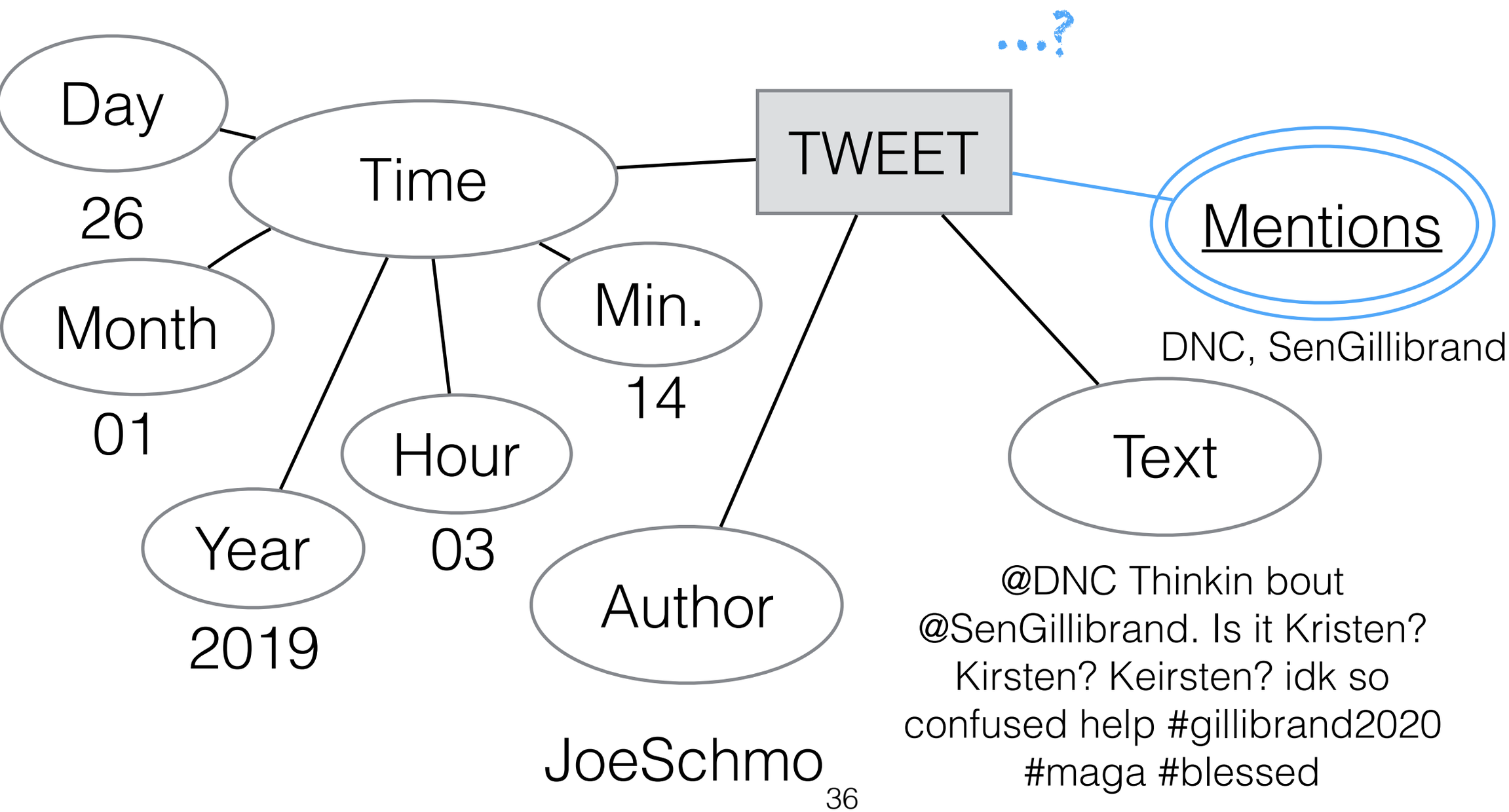
Entity-Relationship (ER) Model

Key Attribute: Designated attribute that uniquely identifies the entry



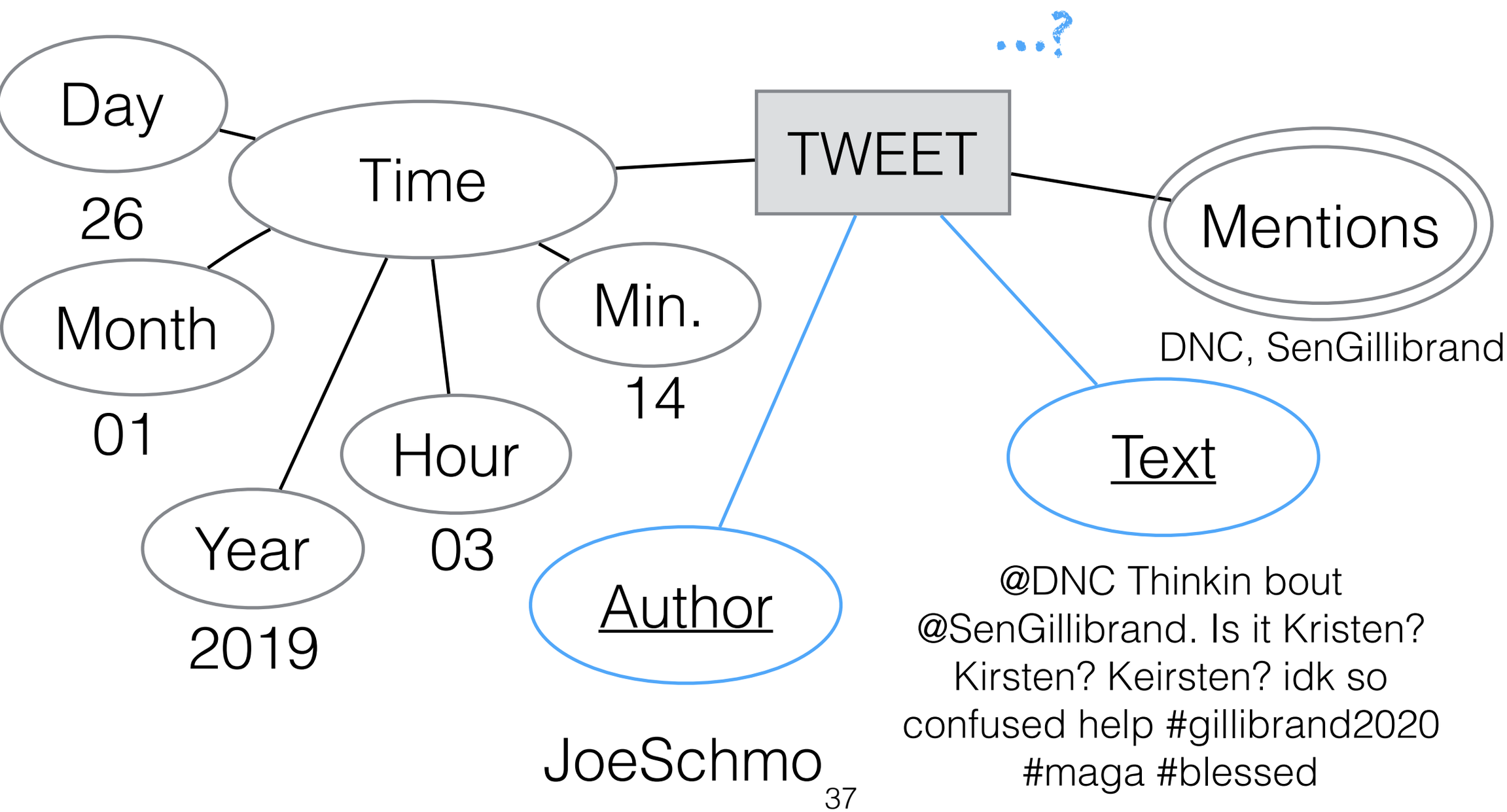
Entity-Relationship (ER) Model

Key Attribute: Designated attribute that uniquely identifies the entry



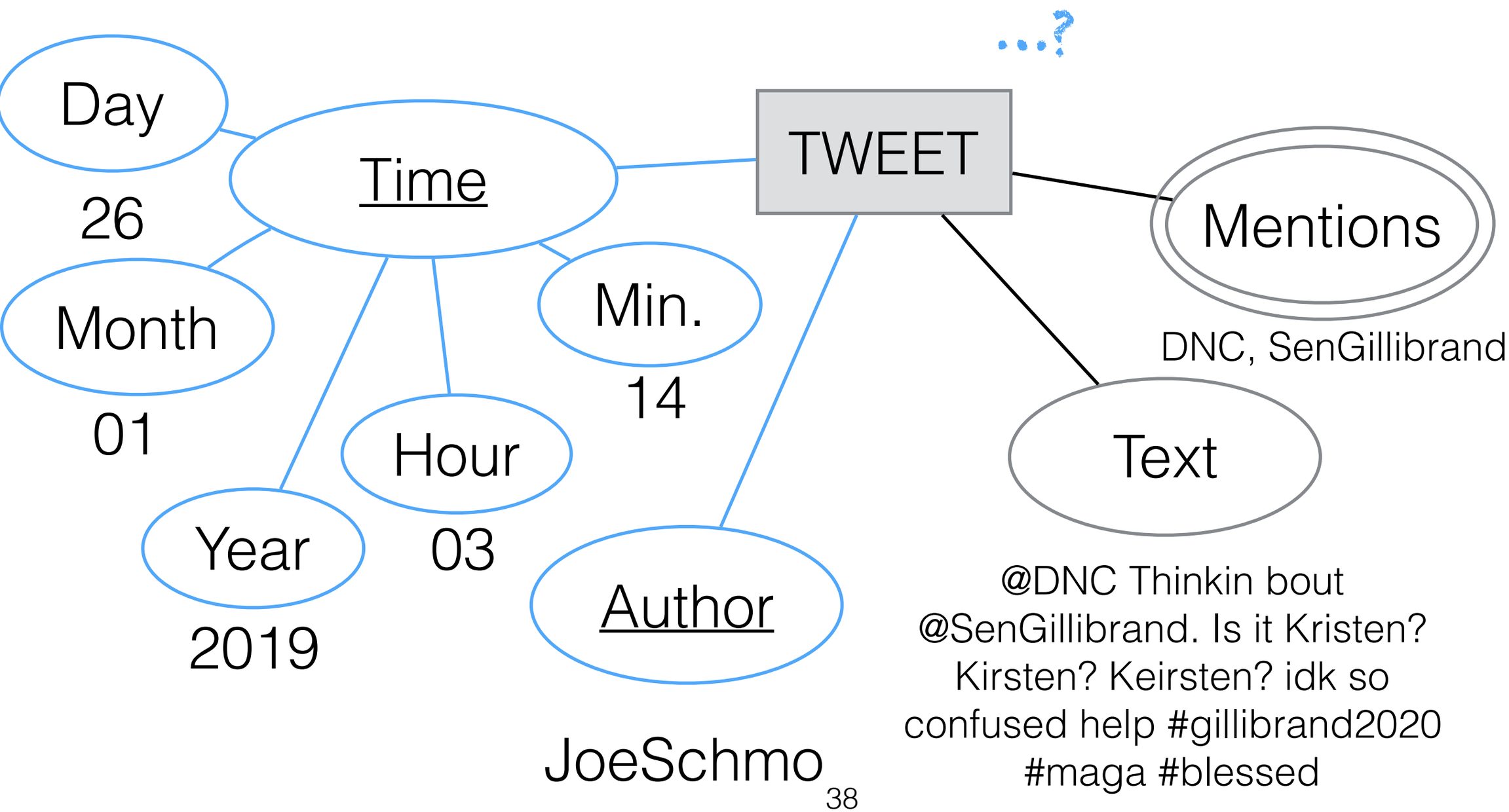
Entity-Relationship (ER) Model

Key Attribute: Designated attribute that uniquely identifies the entry



Entity-Relationship (ER) Model

Key Attribute: Designated attribute that uniquely identifies the entry

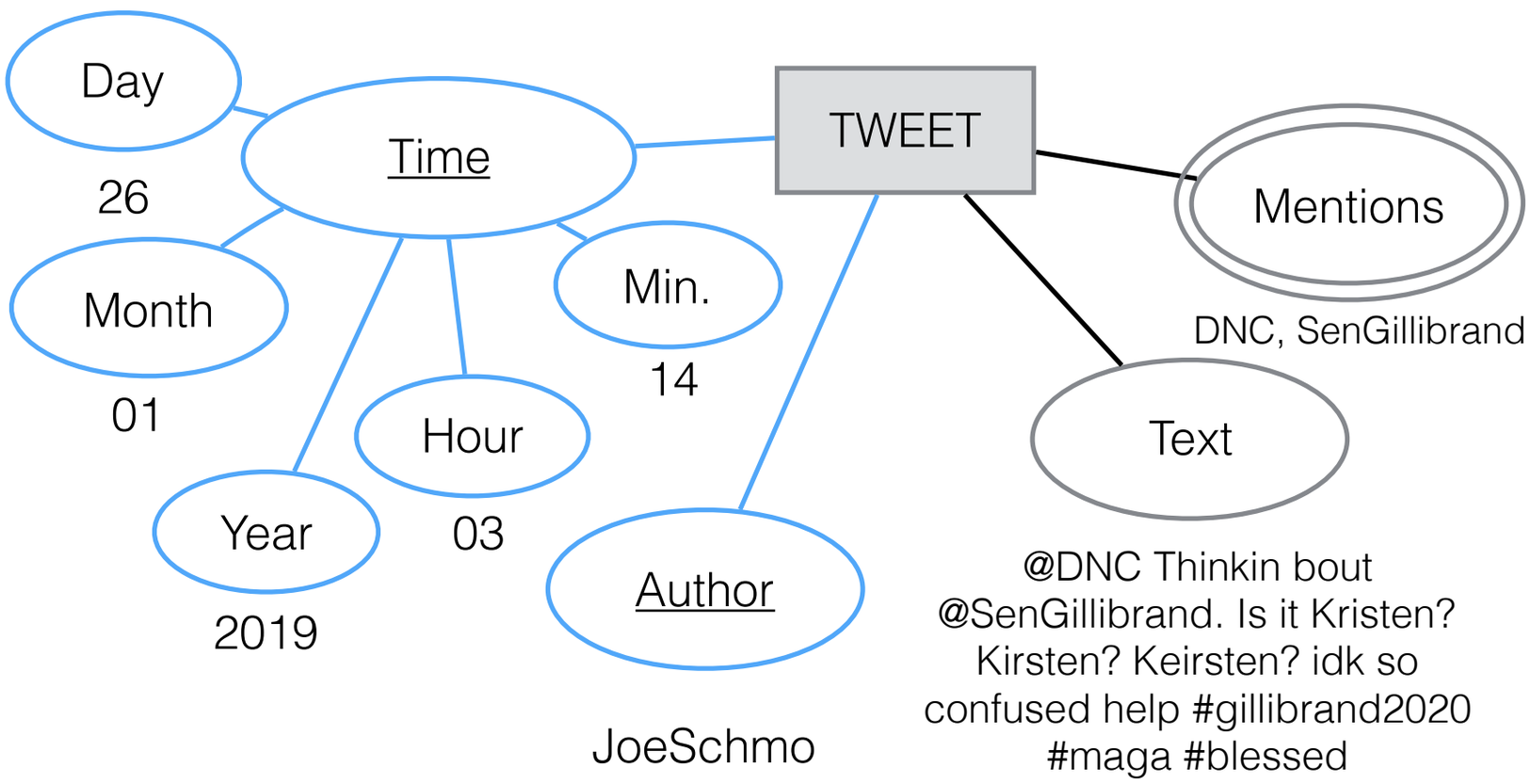


Clicker Question!

Entity-Relationship (ER) Model

Clicker Question!

Is it a good idea to use author +timestamp as a key?



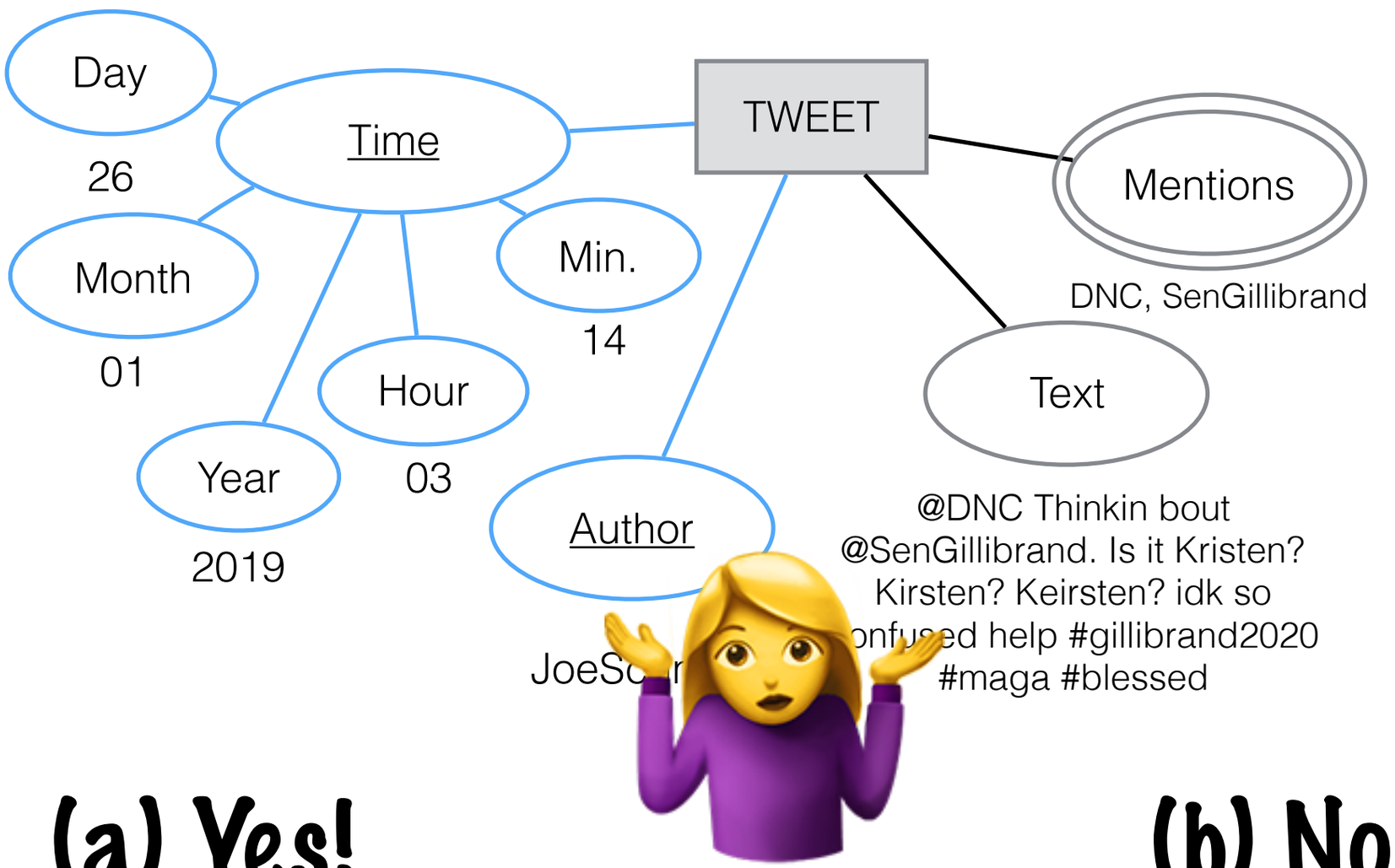
(a) Yes!

(b) No!

Entity-Relationship (ER) Model

Clicker Question!

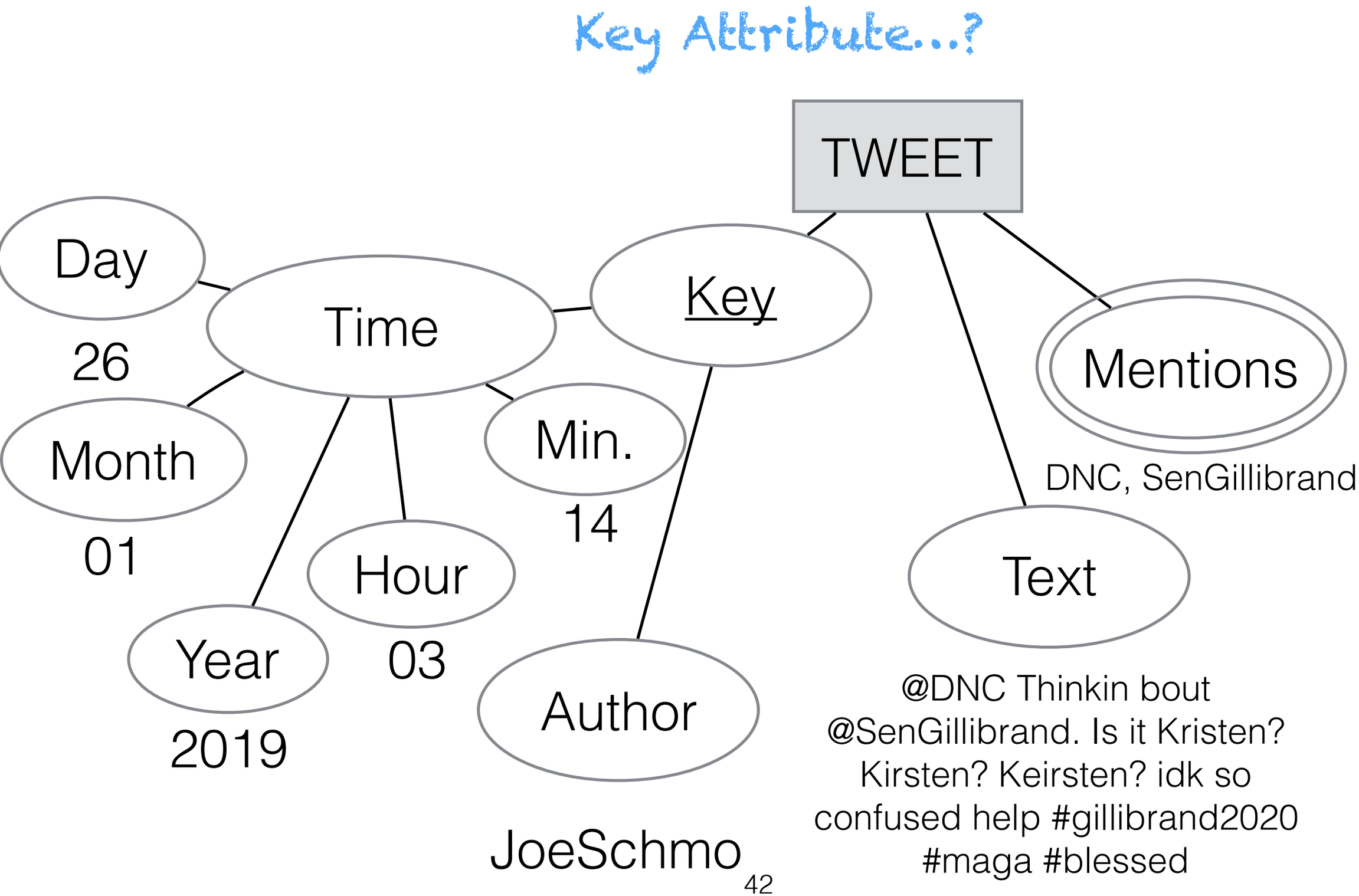
Is it a good idea to use author +timestamp as a key?



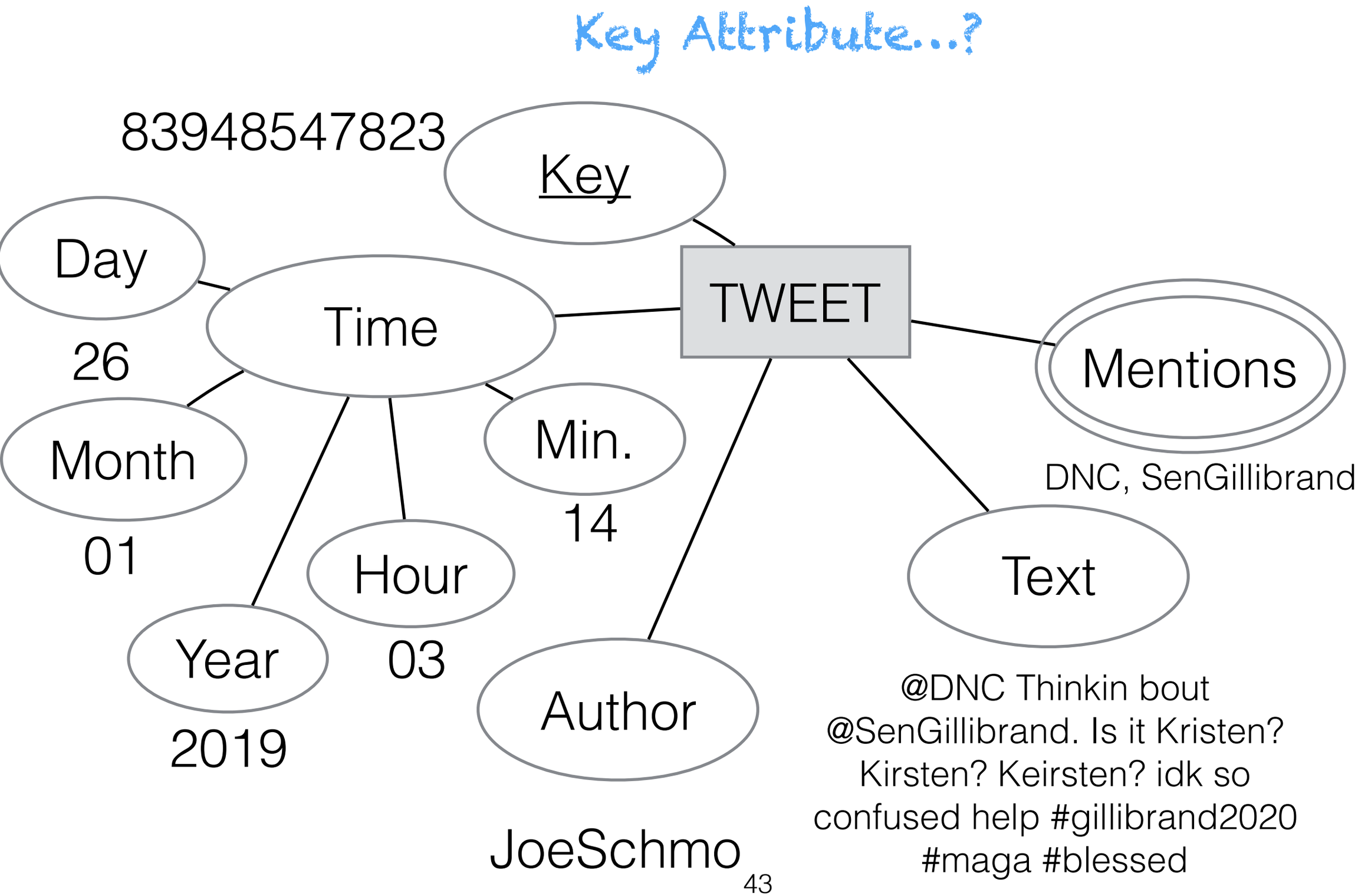
(a) Yes!

(b) No!

Entity-Relationship (ER) Model

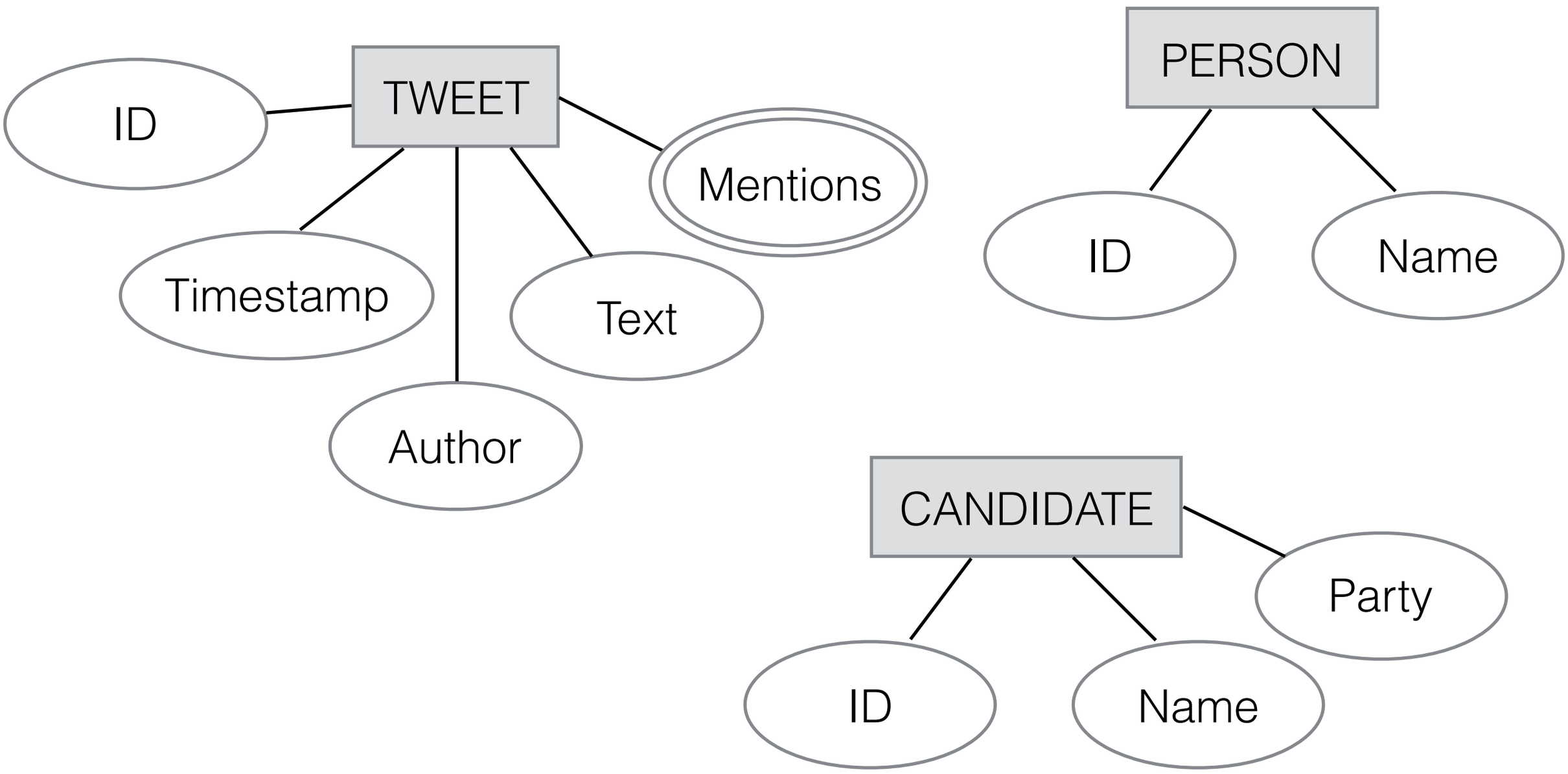


Entity-Relationship (ER) Model

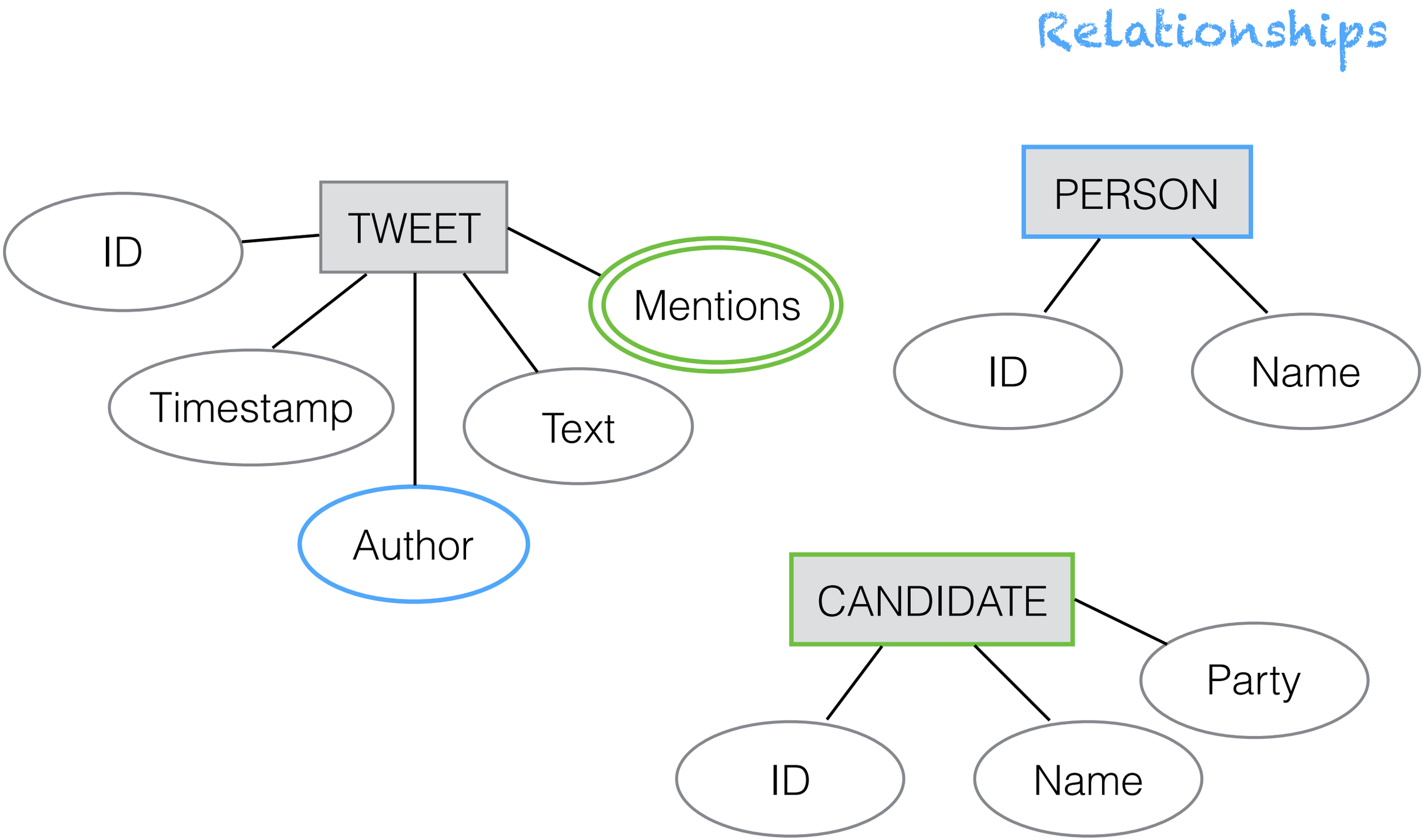


Entity-Relationship (ER) Model

Relationships

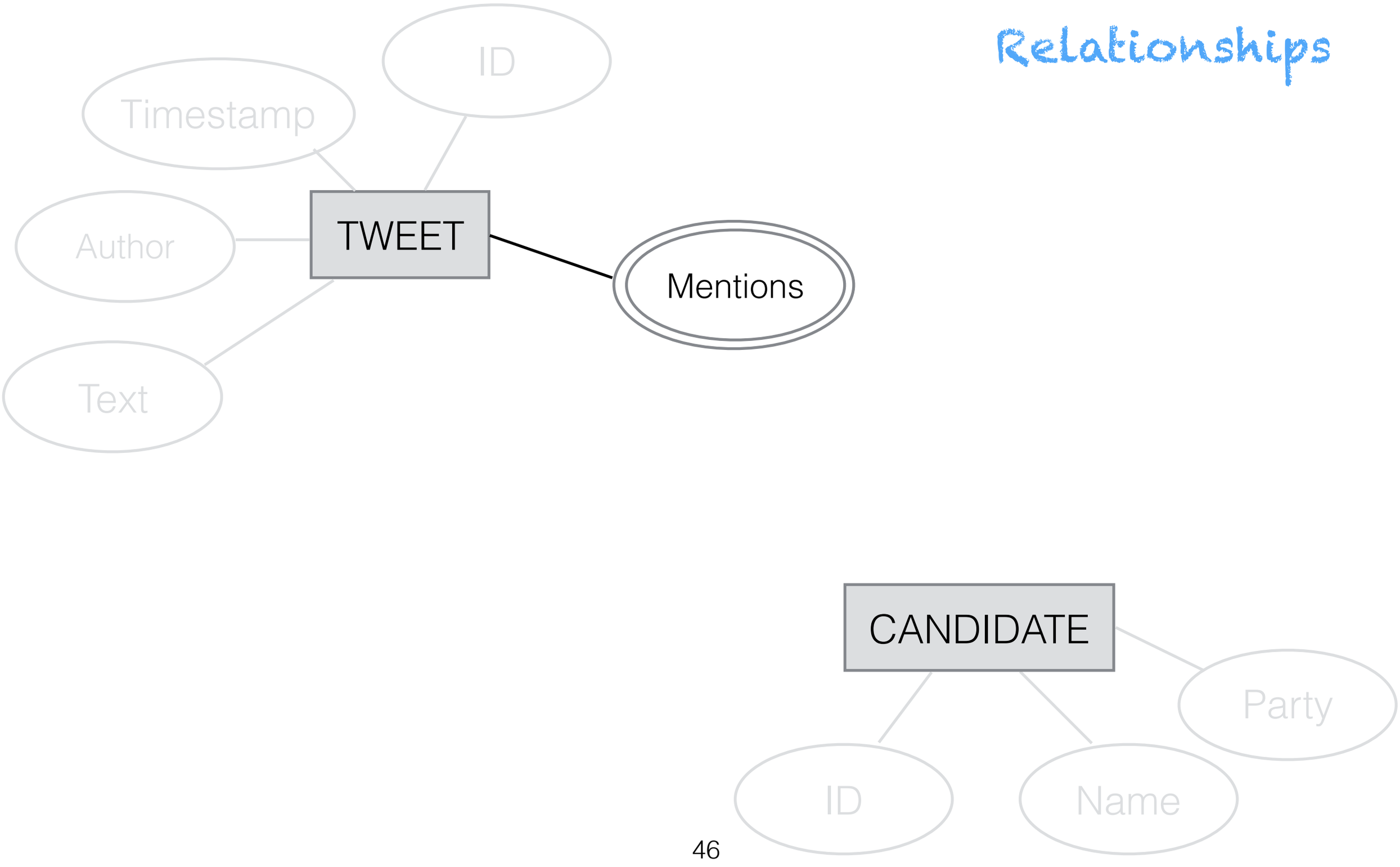


Entity-Relationship (ER) Model

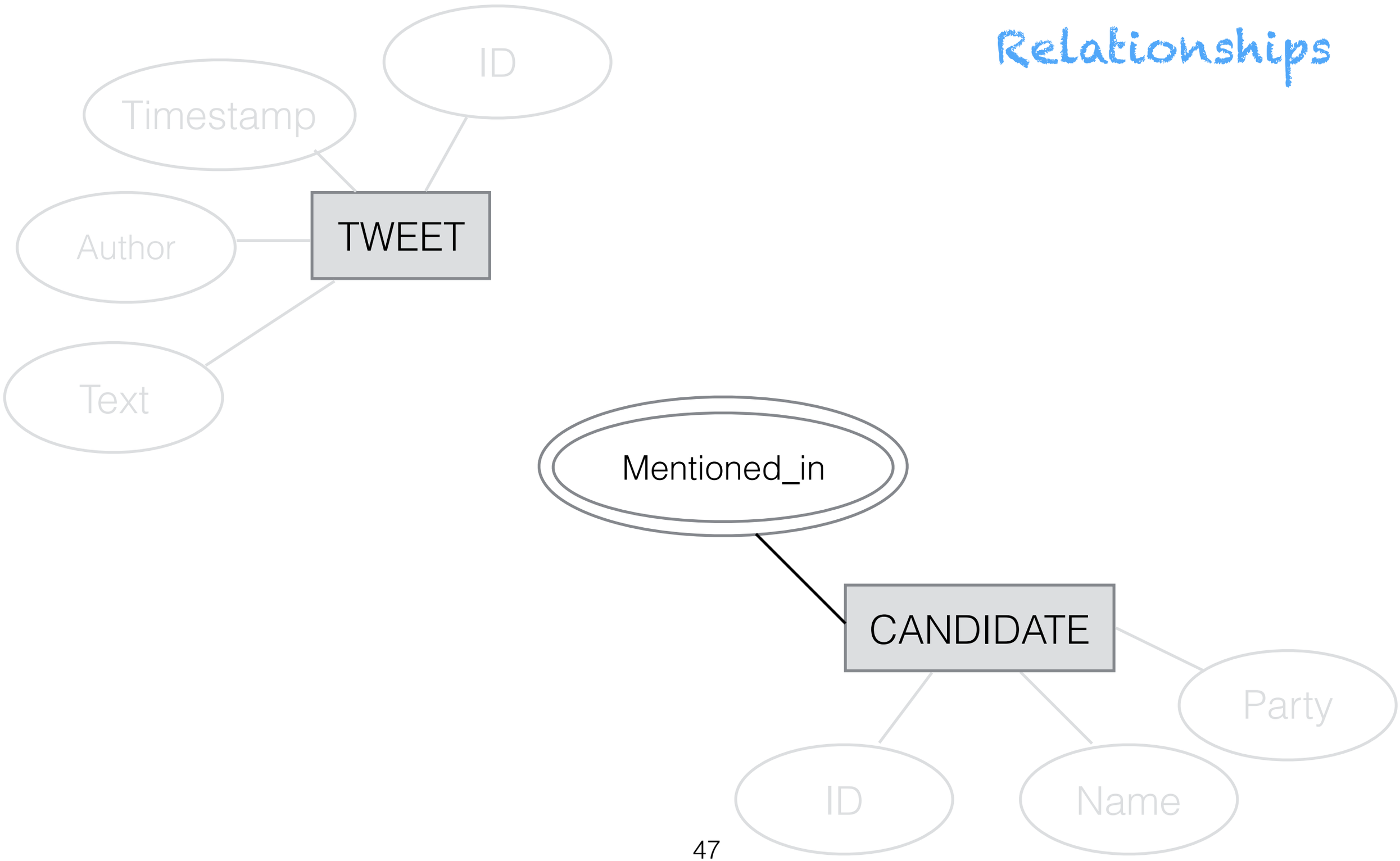


Entity-Relationship (ER) Model

Relationships

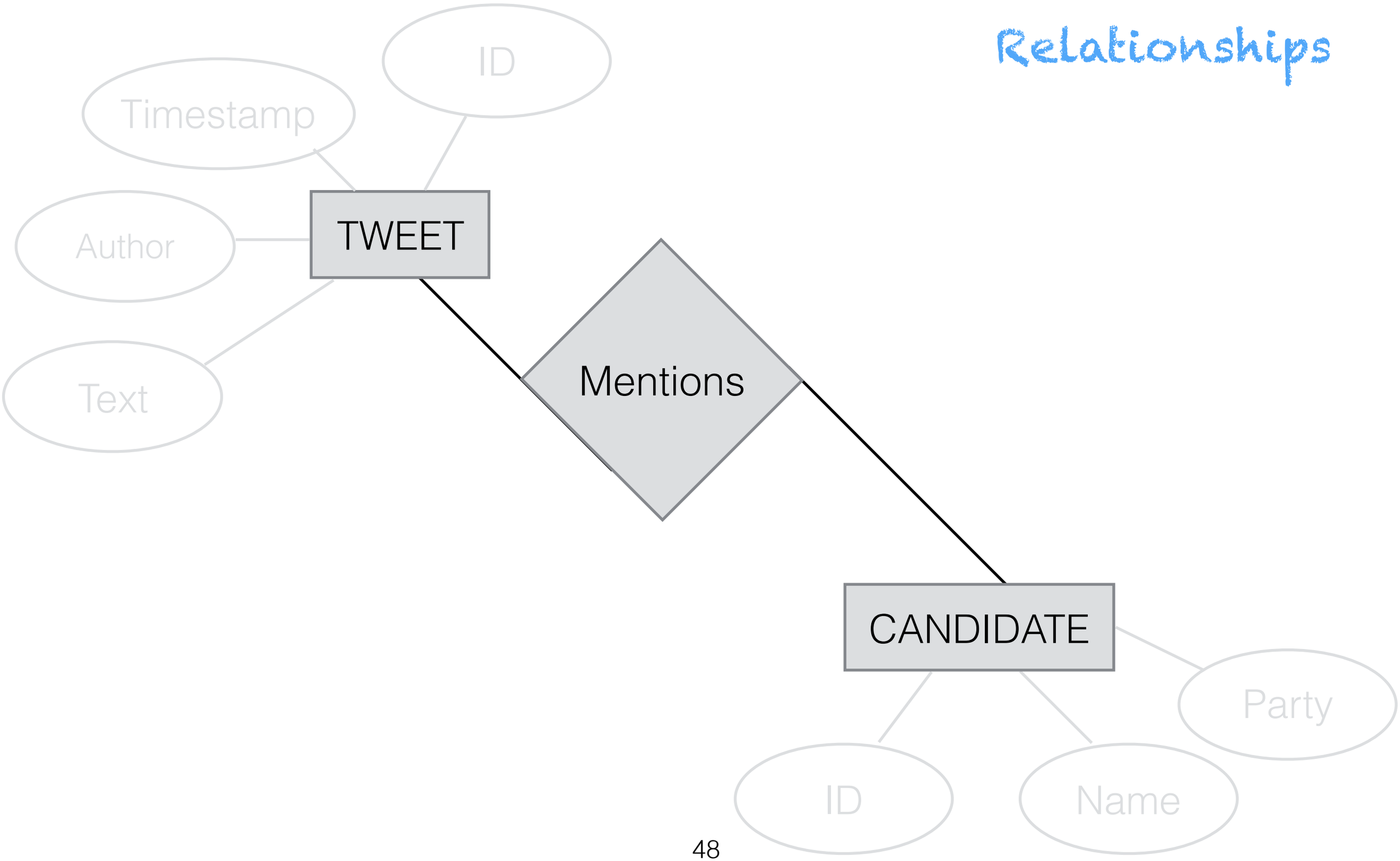


Entity-Relationship (ER) Model

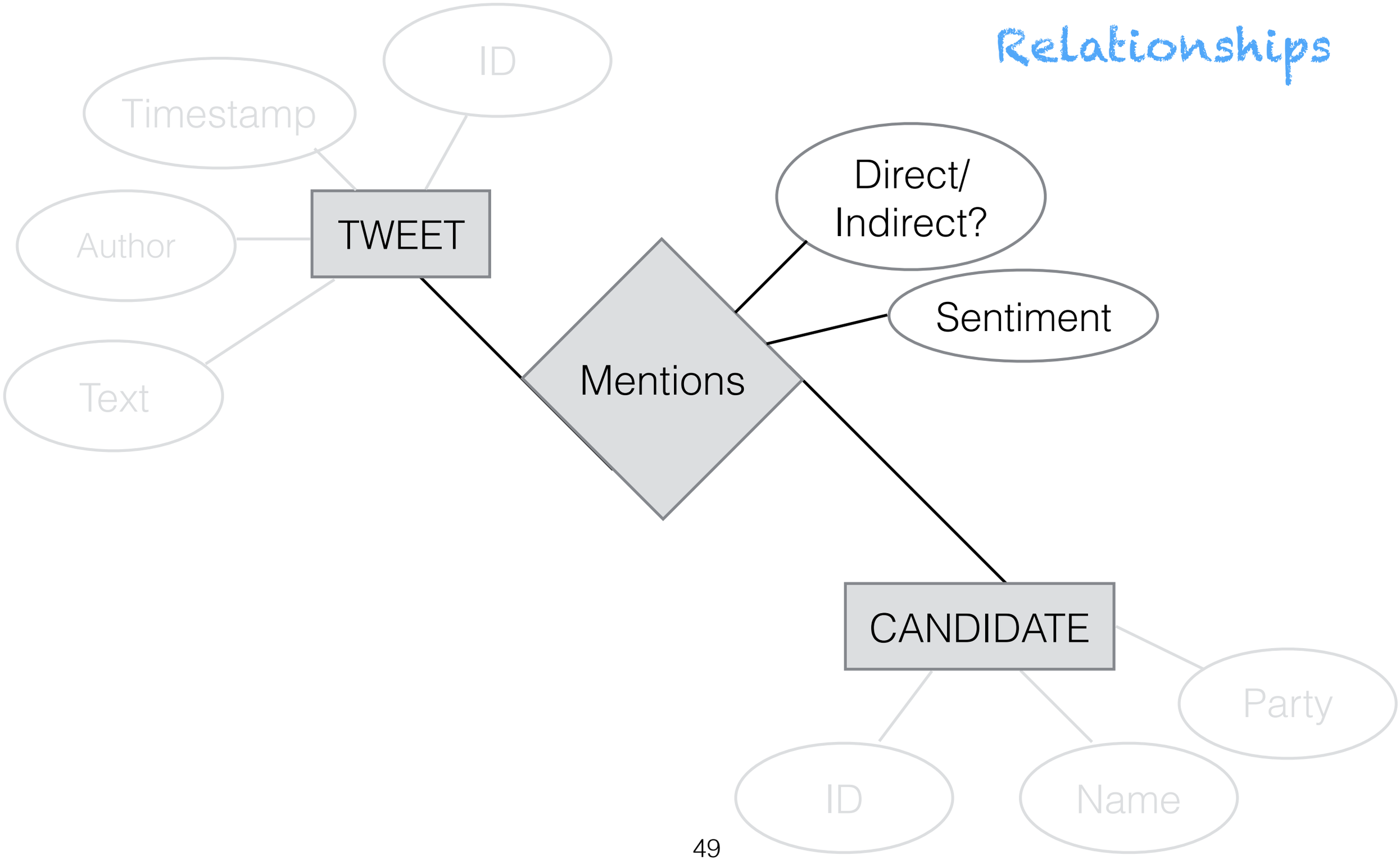


Entity-Relationship (ER) Model

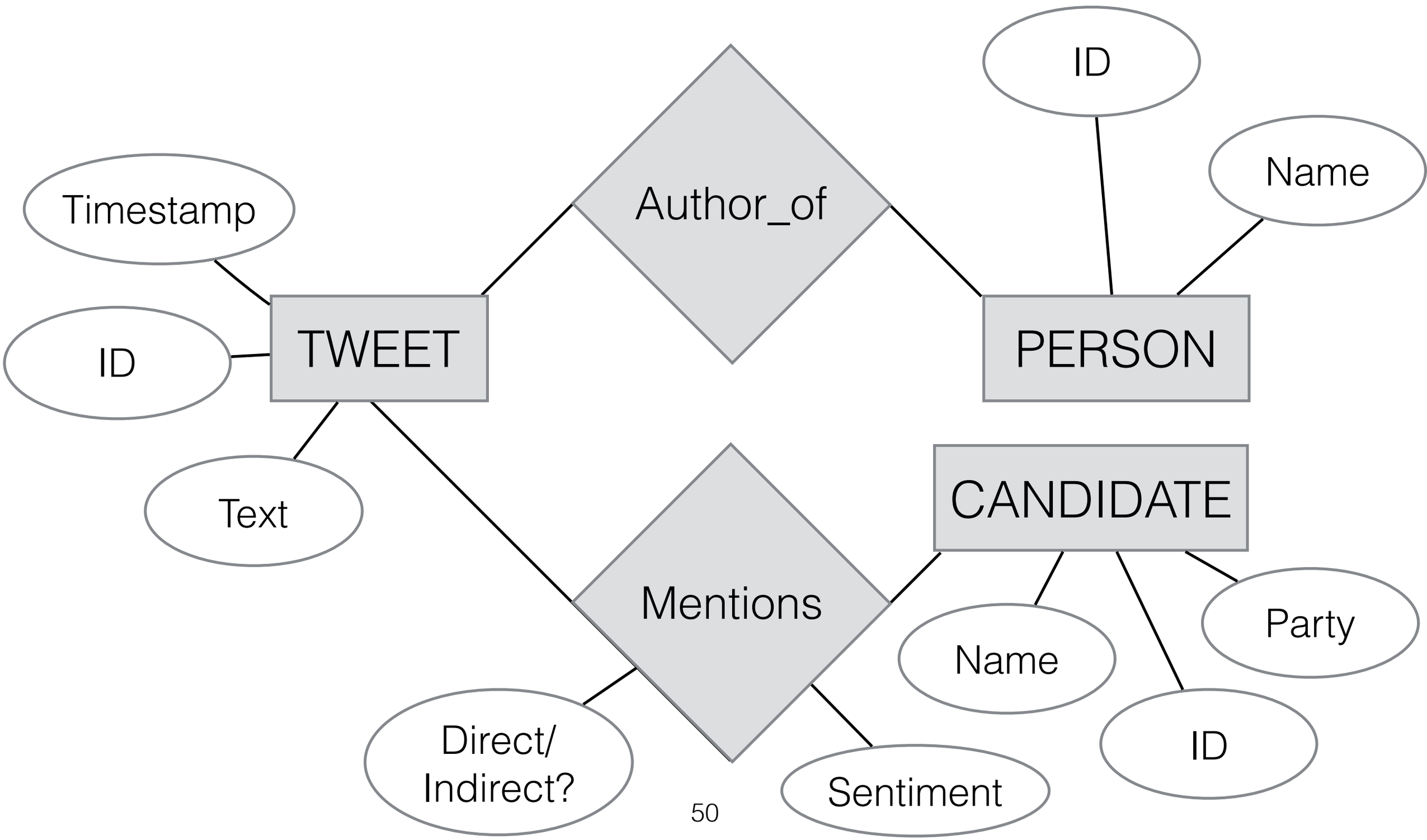
Relationships



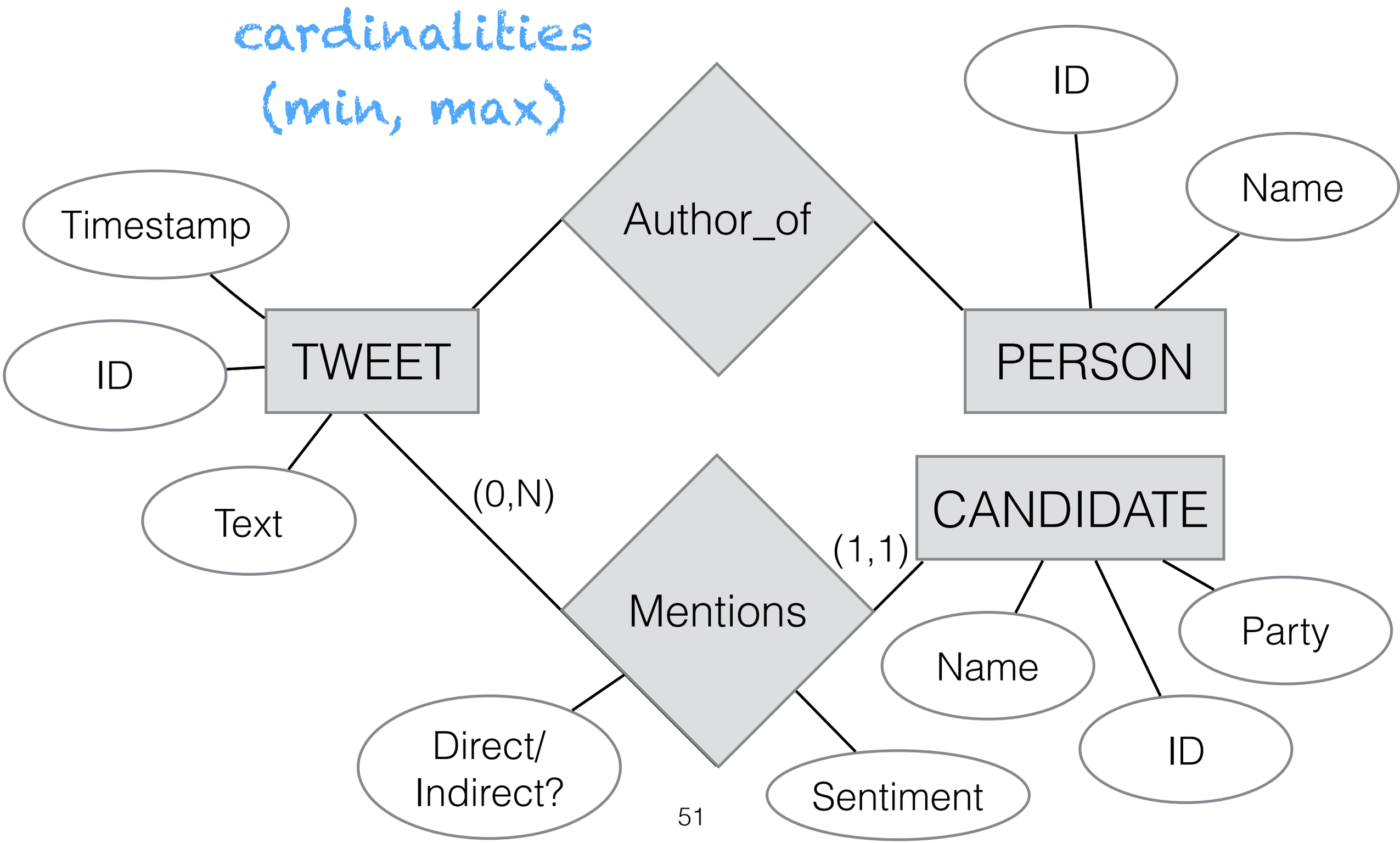
Entity-Relationship (ER) Model



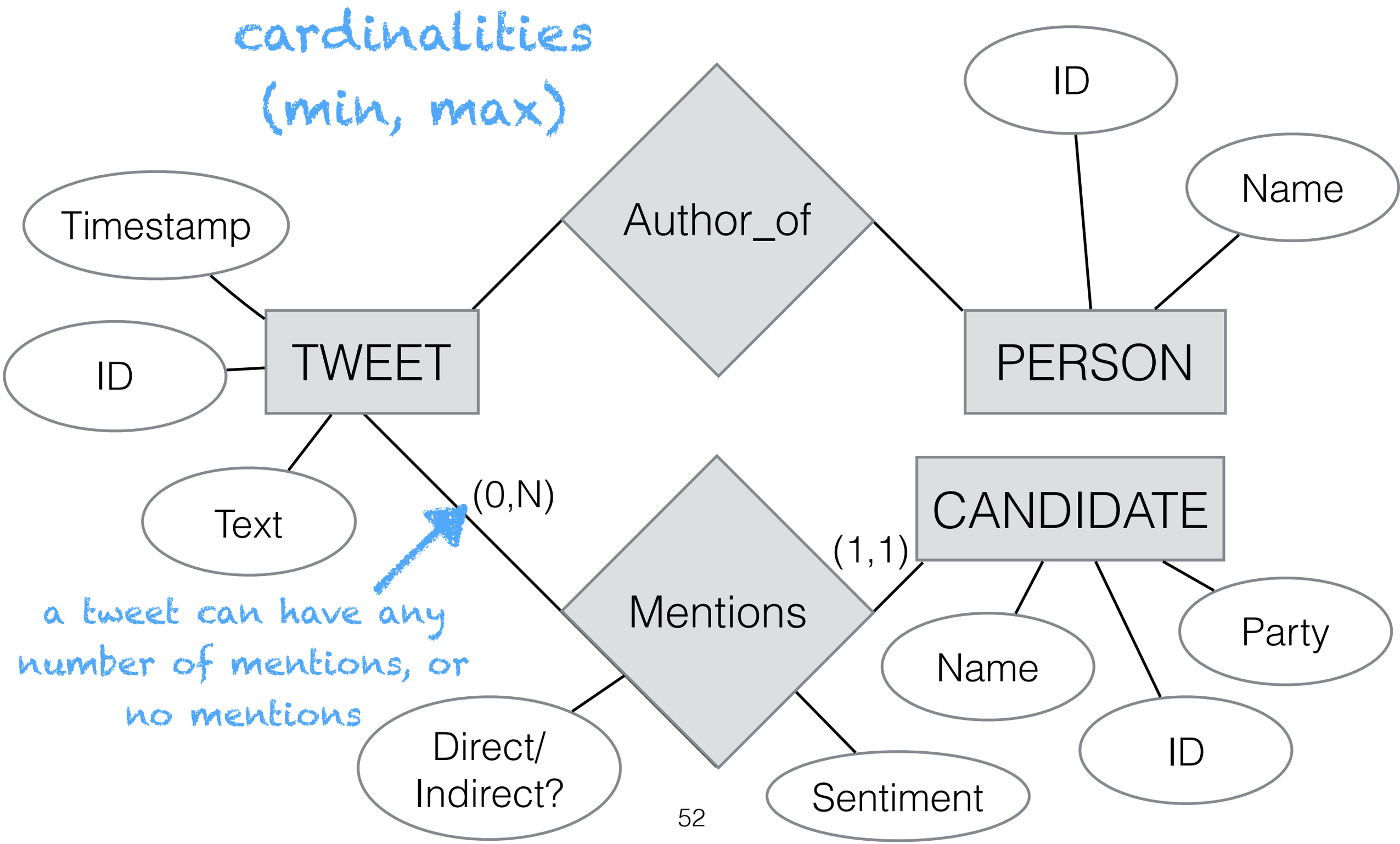
Entity-Relationship (ER) Model



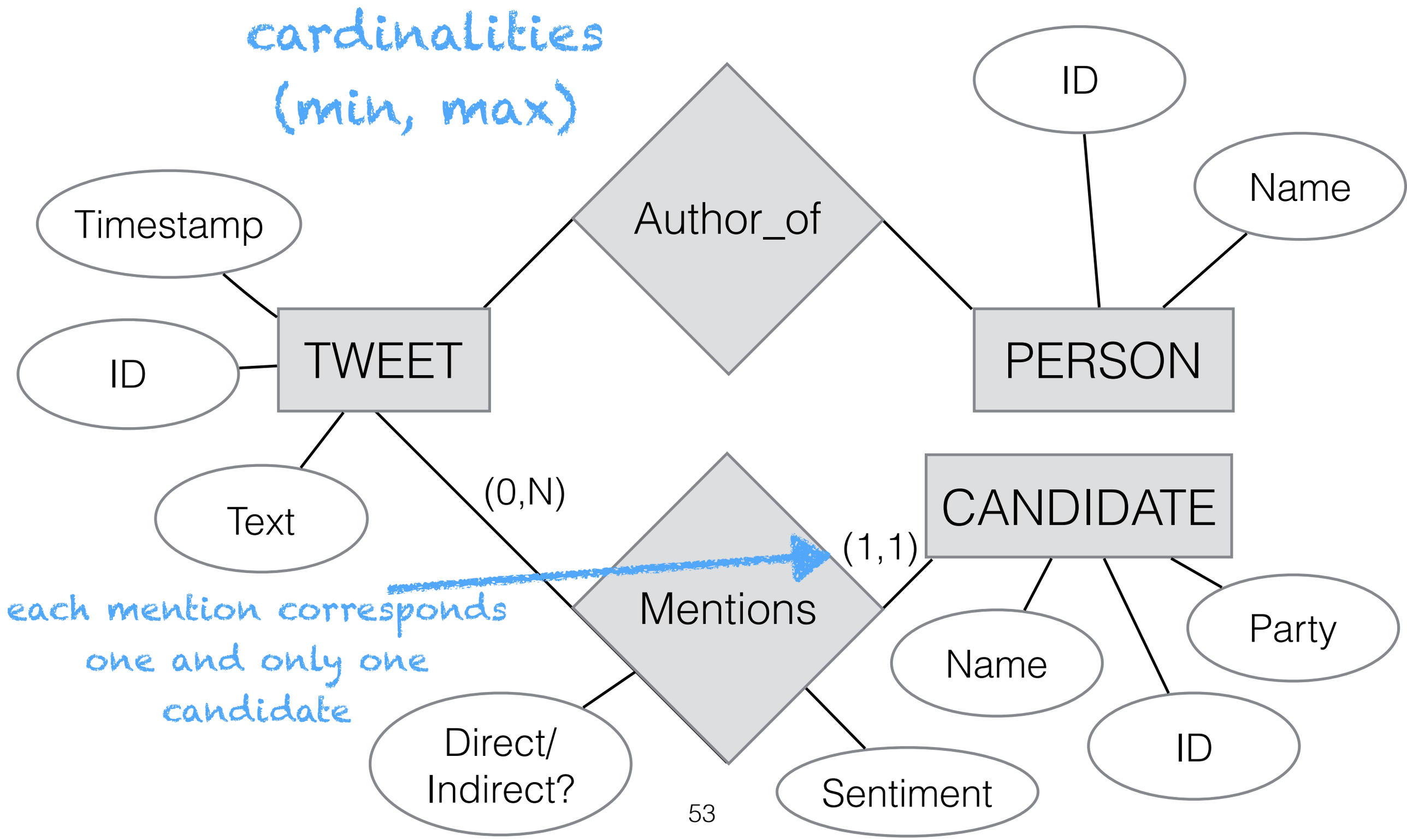
Entity-Relationship (ER) Model



Entity-Relationship (ER) Model



Entity-Relationship (ER) Model



Design Decisions

Design Decisions

- You can’t talk about things that don’t exist! Make sure the representation supports the analysis you want to do.

Design Decisions

- You can’t talk about things that don’t exist! Make sure the representation supports the analysis you want to do.
- Should this concept be an entity? Attribute? Relation?

Design Decisions

- You can’t talk about things that don’t exist! Make sure the representation supports the analysis you want to do.
- Should this concept be an entity? Attribute? Relation?
  - As with most things, there is no good answer

Design Decisions

- You can’t talk about things that don’t exist! Make sure the representation supports the analysis you want to do.
- Should this concept be an entity? Attribute? Relation?
  - As with most things, there is no good answer
  - Draft, refine, document, iterate…

Before we proceed…

Burning Questions?

DATABASES FOR DATA SCIENTIST

Requirement Engineering	Conceptual Modeling	Logical and Physical Modeling	Asking and Answering Questions (Analysis)
"Book of Duty"	Conceptual Design (ER)	Logical Design (schema, table names, data types), Physical Design (indices, memory layout, optimizations)	Relational Algebra, SQL

https://cs.brown.edu/courses/csci1951-a/res/static/lec02_databases.pdf

Relational Model

Relational Model

TWEET

ID	Timestamp	Author	Text	Mentions
389472	1/1/19 12:34	Bob	hey	NULL
123794	1/1/19 12:32	Maria	lol	{Bob}
596208	1/2/19 1:04	Yu	:-D	NULL

Relational Model

Relation

TWEET

ID	Timestamp	Author	Text	Mentions
389472	1/1/19 12:34	Bob	hey	NULL
123794	1/1/19 12:32	Maria	lol	{Bob}
596208	1/2/19 1:04	Yu	:-D	NULL

Relational Model

TWEET

Relation Name

ID	Timestamp	Author	Text	Mentions
389472	1/1/19 12:34	Bob	hey	NULL
123794	1/1/19 12:32	Maria	lol	{Bob}
596208	1/2/19 1:04	Yu	:-D	NULL

Relational Model

Attribute

TWEET

ID	Timestamp	Author	Text	Mentions
389472	1/1/19 12:34	Bob	hey	NULL
123794	1/1/19 12:32	Maria	lol	{Bob}
596208	1/2/19 1:04	Yu	:-D	NULL

Domain: D = dom(Timestamp) =

Valid time strings = ##/##/#### ##:##

Relational Model

TWEET

ID	Timestamp	Author	Text	Mentions
389472	1/1/19 12:34	Bob	hey	NULL
123794	1/1/19 12:32	Maria	lol	{Bob}
596208	1/2/19 1:04	Yu	:-D	NULL

Tuple

Relational Model

TWEET

Relation Schema (R)

ID	Timestamp	Author	Text	Mentions
389472	1/1/19 12:34	Bob	hey	NULL
123794	1/1/19 12:32	Maria	lol	{Bob}
596208	1/2/19 1:04	Yu	:-D	NULL

Relational Model

TWEET

Relation Schema (R)

ID	Timestamp	Author	Text	Mentions
389472	1/1/19 12:34	Bob	hey	NULL
123794	1/1/19 12:32	Maria	lol	{Bob}
596208	1/2/19 1:04	Yu	:-D	NULL

Relation State r(R)

Relational Model

TWEET

Intension

ID	Timestamp	Author	Text	Mentions
389472	1/1/19 12:34	Bob	hey	NULL
123794	1/1/19 12:32	Maria	lol	{Bob}
596208	1/2/19 1:04	Yu	:-D	NULL

Extension

Find all the tweets by authors named Diane.

TWEET

ID	Timestamp	Author	Text	Mentions
389472	1/1/19 12:34	Bob	hey	NULL
123794	1/1/19 12:32	Maria	lol	{Bob}
596208	1/2/19 1:04	Yu	:-D	NULL

Find all the tweets by authors named Diane.

```
SELECT * FROM TWEET WHERE Name is "Diane"
```

TWEET

ID	Timestamp	Author	Text	Mentions
389472	1/1/19 12:34	Bob	hey	NULL
123794	1/1/19 12:32	Maria	lol	{Bob}
596208	1/2/19 1:04	Yu	:-D	NULL

Find all the tweets by authors named Diane.

```
SELECT * FROM TWEET WHERE Name is "Diane"
```

∅

TWEET

ID	Timestamp	Author	Text	Mentions
389472	1/1/19 12:34	Bob	hey	NULL
123794	1/1/19 12:32	Maria	lol	{Bob}
596208	1/2/19 1:04	Yu	:-D	NULL

Find all the tweets by authors named Diane.

```
SELECT * FROM TWEET WHERE Name is "Diane"
```

∅

TWEET

ID	Timestamp	Author	Text	Mentions
389472	1/1/19 12:34	Bob	hey	NULL
123794	1/1/19 12:32	Maria	lol	{Bob}
596208	1/2/19 1:04	Yu	:-D	NULL

Find all the tweets by authors named Diane.

```
SELECT * FROM TWEET WHERE Name is "Diane"
```

∅

"Closed world assumption"

TWEET

ID	Timestamp	Author	Text	Mentions
389472	1/1/19 12:34	Bob	hey	NULL
123794	1/1/19 12:32	Maria	lol	{Bob}
596208	1/2/19 1:04	Yu	:-D	NULL

Find all the tweets which weight less than 45lbs

TWEET

ID	Timestamp	Author	Text	Mentions
389472	1/1/19 12:34	Bob	hey	NULL
123794	1/1/19 12:32	Maria	lol	{Bob}
596208	1/2/19 1:04	Yu	:-D	NULL

Find all the tweets which weight less than 45lbs

```
SELECT * FROM TWEET WHERE Weight < 45
```

TWEET

ID	Timestamp	Author	Text	Mentions
389472	1/1/19 12:34	Bob	hey	NULL
123794	1/1/19 12:32	Maria	lol	{Bob}
596208	1/2/19 1:04	Yu	:-D	NULL

Find all the tweets which weight less than 45lbs

```
SELECT * FROM TWEET WHERE Weight < 45
```

????

TWEET

ID	Timestamp	Author	Text	Mentions
389472	1/1/19 12:34	Bob	hey	NULL
123794	1/1/19 12:32	Maria	lol	{Bob}
596208	1/2/19 1:04	Yu	:-D	NULL

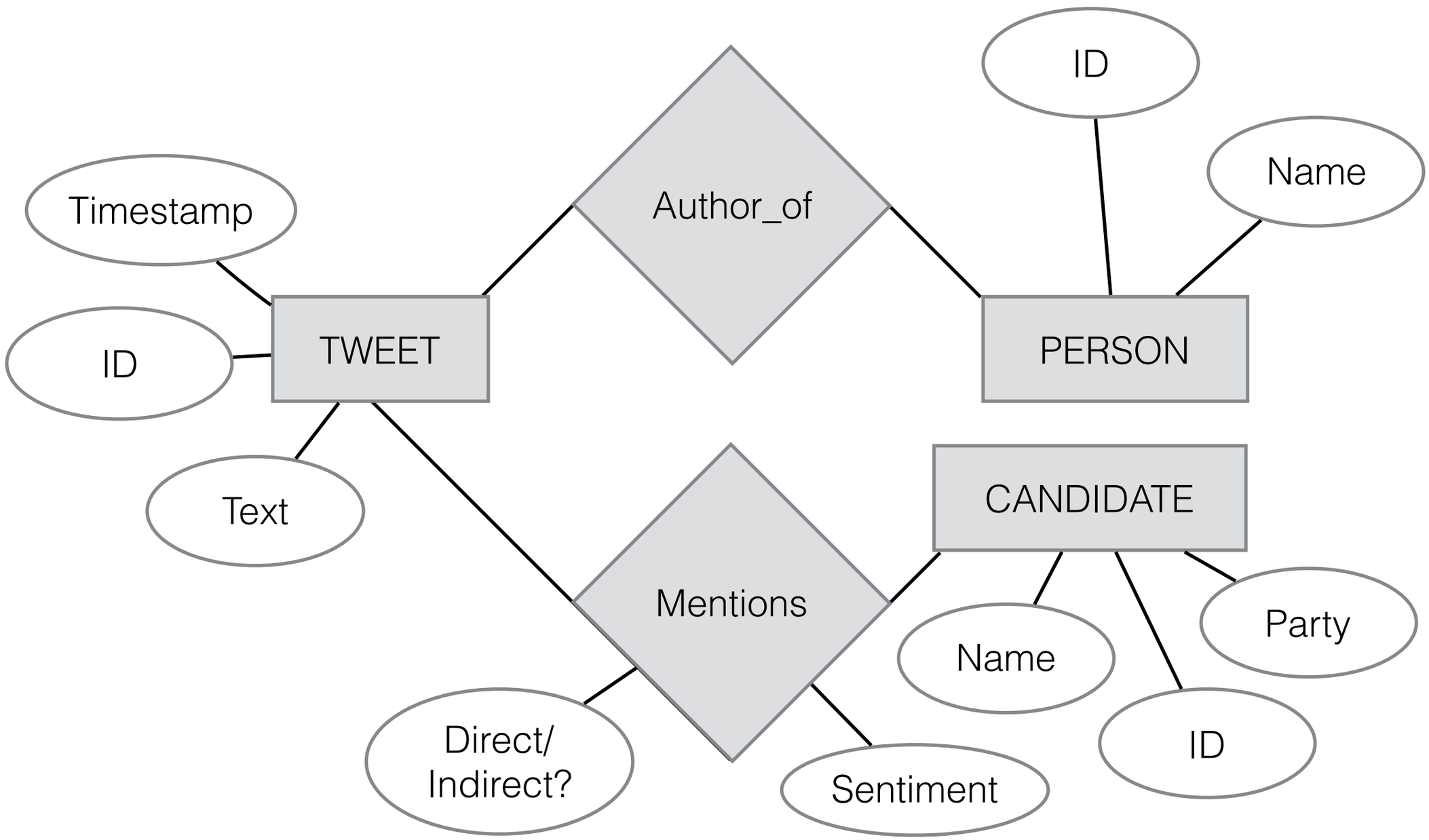
SQL

SQL

- Data Definition Language (DDL): Defining data types and Relation Schemas (intensions!)
- Data Manipulation and Query Language (DML):
  - Populating/updating data bases (extensions!)
  - Querying data bases

Creating and Manipulating Tables

Creating and Manipulating Tables



Data Types

Data Types

- Numeric: INT, FLOAT, REAL, DOUBLE

Data Types

- Numeric: INT, FLOAT, REAL, DOUBLE
- Character Strings: CHAR(n), VARCHAR(n), CLOB(size)

Data Types

- Numeric: INT, FLOAT, REAL, DOUBLE
- Character Strings: CHAR(n), VARCHAR(n), CLOB(size)
  - CLOB(2MB) for large objects e.g. documents/web pages

Data Types

- Numeric: INT, FLOAT, REAL, DOUBLE
- Character Strings: CHAR(n), VARCHAR(n), CLOB(size)
  - CLOB(2MB) for large objects e.g. documents/web pages
- Bit Strings: BIT(n), BIT VARYING(n), BLOB

Data Types

- Numeric: INT, FLOAT, REAL, DOUBLE
- Character Strings: CHAR(n), VARCHAR(n), CLOB(size)
  - CLOB(2MB) for large objects e.g. documents/web pages
- Bit Strings: BIT(n), BIT VARYING(n), BLOB
  - BLOB(20MB) e.g. for images

Data Types

- Numeric: INT, FLOAT, REAL, DOUBLE
- Character Strings: CHAR(n), VARCHAR(n), CLOB(size)
  - CLOB(2MB) for large objects e.g. documents/web pages
- Bit Strings: BIT(n), BIT VARYING(n), BLOB
  - BLOB(20MB) e.g. for images
- Boolean

Data Types

- Numeric: INT, FLOAT, REAL, DOUBLE
- Character Strings: CHAR(n), VARCHAR(n), CLOB(size)
  - CLOB(2MB) for large objects e.g. documents/web pages
- Bit Strings: BIT(n), BIT VARYING(n), BLOB
  - BLOB(20MB) e.g. for images
- Boolean
- Dates: DATE, TIME, TIMESTAMP, TIME WITH TIME ZONE

Creating Tables

Creating Tables

TWEET: <ID, Time, Text>

Creating Tables

TWEET: <ID, Time, Text>

```
create table TWEET (
 ID INT,
 Time TIMESTAMP,
 Text ???
);
```

Creating Tables

TWEET: <ID, Time, Text>

```
create table TWEET (
 ID INT,
 Time TIMESTAMP,
 Text ???
);
```

CHAR(n), VARCHAR(n), CLOB(size) ??

Creating Tables

TWEET: <ID, Time, Text>

```
create table TWEET (
 ID INT,
 Time TIMESTAMP,
 Text VARCHAR(140)
);
```

Creating Tables

- CHAR(n): faster -> can use static memory allocation; no length checks in operations, so less overhead
- VARCHAR(n): uses less space on average

Text>

```
WEET (
 Time TIMESTAMP,
 Text VARCHAR(140)
);
```

Creating Tables

PERSON: <Handle, Name>

```
create table PERSON (
 Handle VARCHAR(100),
 Name VARCHAR(1000)
);
```

Creating Tables

PERSON: <Handle, Name, ProfilePic>

```
create table PERSON (
 Handle VARCHAR(100),
 Name VARCHAR(1000),
 ProfilePic ???
);
```

Creating Tables

PERSON: <Handle, Name, ProfilePic>

```
create table PERSON (
 Handle VARCHAR(100),
 Name VARCHAR(1000),
 ProfilePic BLOB(20MB),
);
```

Creating Tables

PERSON: <Handle, Name, ProfilePic, ProfilePage>

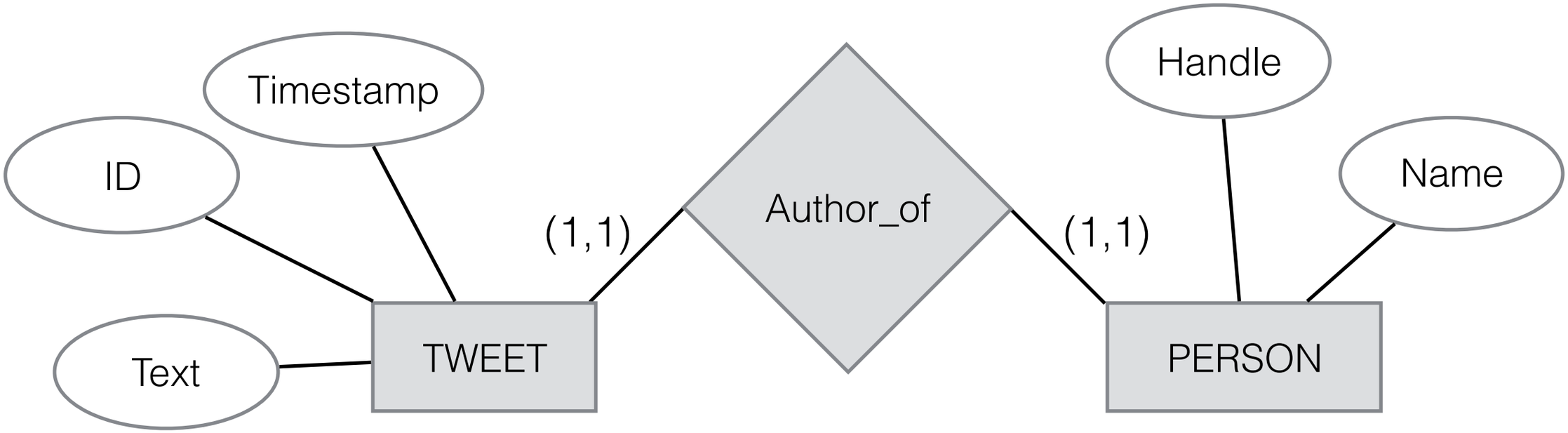
```
create table PERSON (
 Handle VARCHAR(100),
 Name VARCHAR(1000),
 ProfilePic BLOB(20MB),
 ProfilePage ???
);
```

Creating Tables

PERSON: <Handle, Name, ProfilePic, ProfilePage>

```
create table PERSON (
 Handle VARCHAR(100),
 Name VARCHAR(1000),
 ProfilePic BLOB(20MB),
 ProfilePage CLOB(20MB)
);
```

Clicker Question!

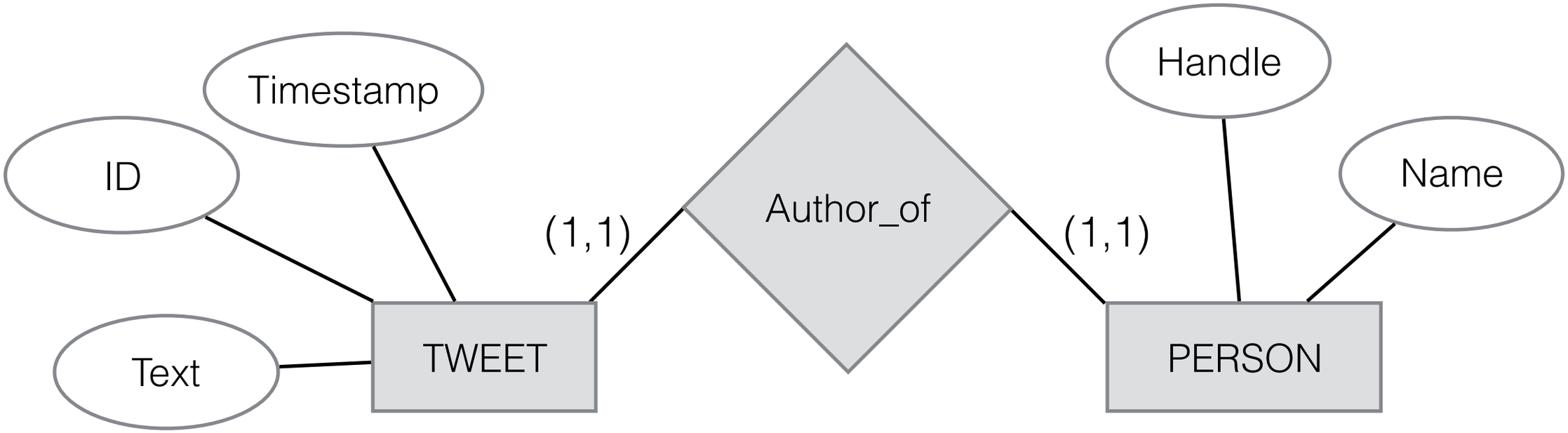


Clicker Question!

TWEET: <ID:INT, Time:TIMESTAMP, Text:VARCHAR(140)>

PERSON: <Handle:VARCHAR(100), Name:VARCHAR(1000)>

```
create table AUTHOR (
        ???
);
```



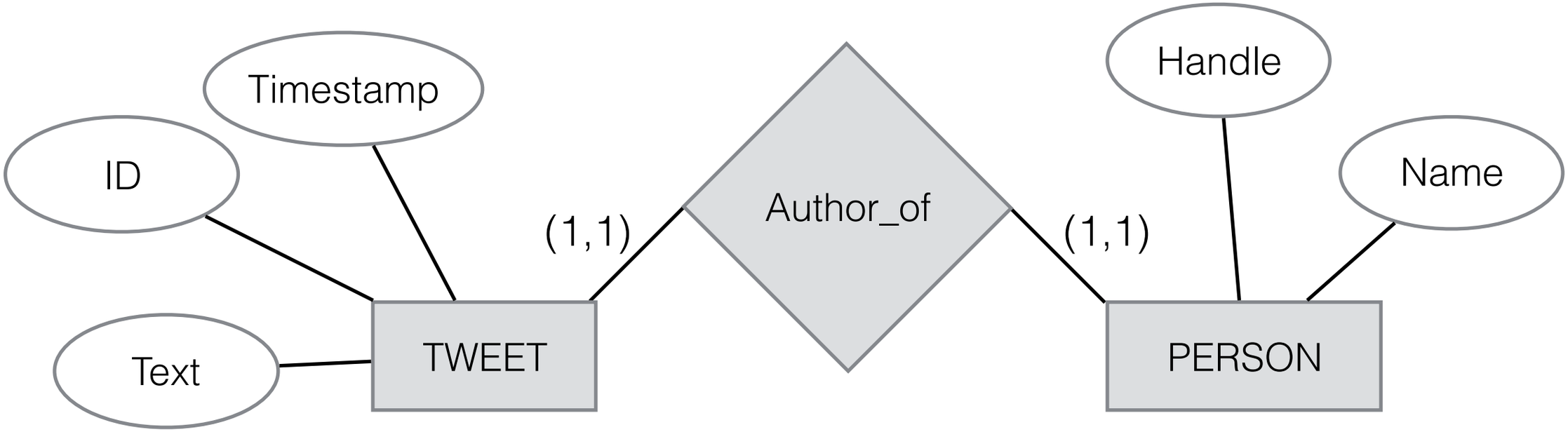
Clicker Question!

TWEET: <ID:INT, Time:TIMESTAMP, Text:VARCHAR(140)>

PERSON: <Handle:VARCHAR(100), Name:VARCHAR(1000)>

```
create table AUTHOR (
        ???
);
```

```
create table TABLE_NAME (
 Attr Data_type,
 …
);
```



TWEET: <ID:INT, Time:TIMESTAMP, Text:VARCHAR(140)>

PERSON: <Handle:VARCHAR(100), Name:VARCHAR(1000)>

```
create table AUTHOR (
 Tweet INT,
 Person VARCHAR(100),
);
```

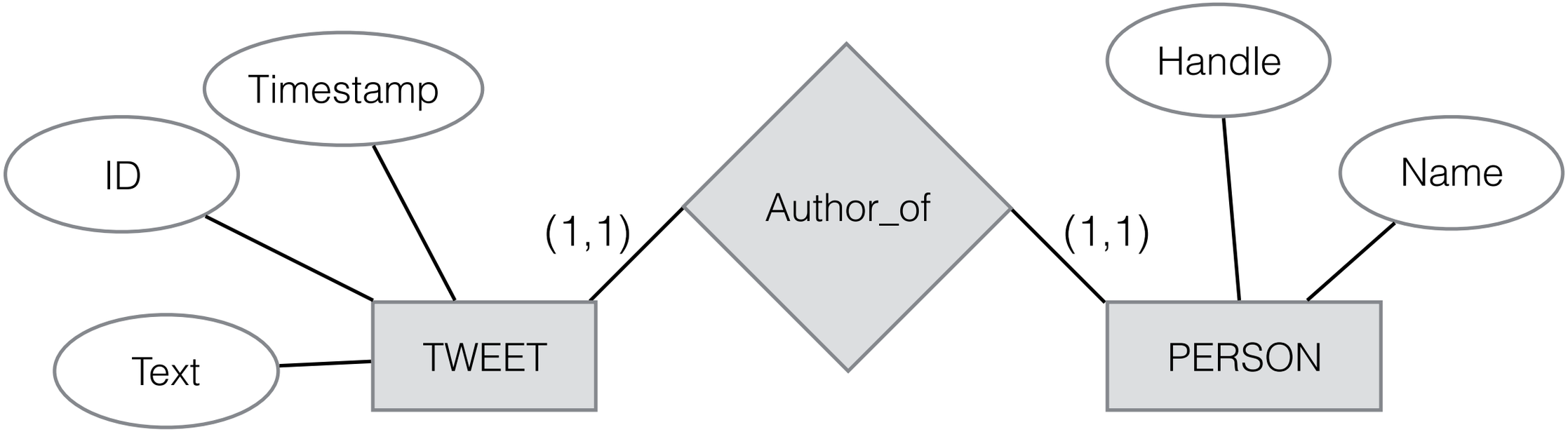
(a)

```
create table AUTHOR (
 Tweet INT,
 Person INT,
);
```

(b)

```
create table AUTHOR (
 Tweet INT,
 Person VARCHAR(1000),
);
```

(c)



TWEET: <ID:INT, Time:TIMESTAMP, Text:VARCHAR(140)>

PERSON: <Handle:VARCHAR(100), Name:VARCHAR(1000)>

```
create table AUTHOR (
 Tweet INT,
 Person VARCHAR(100),
);
```

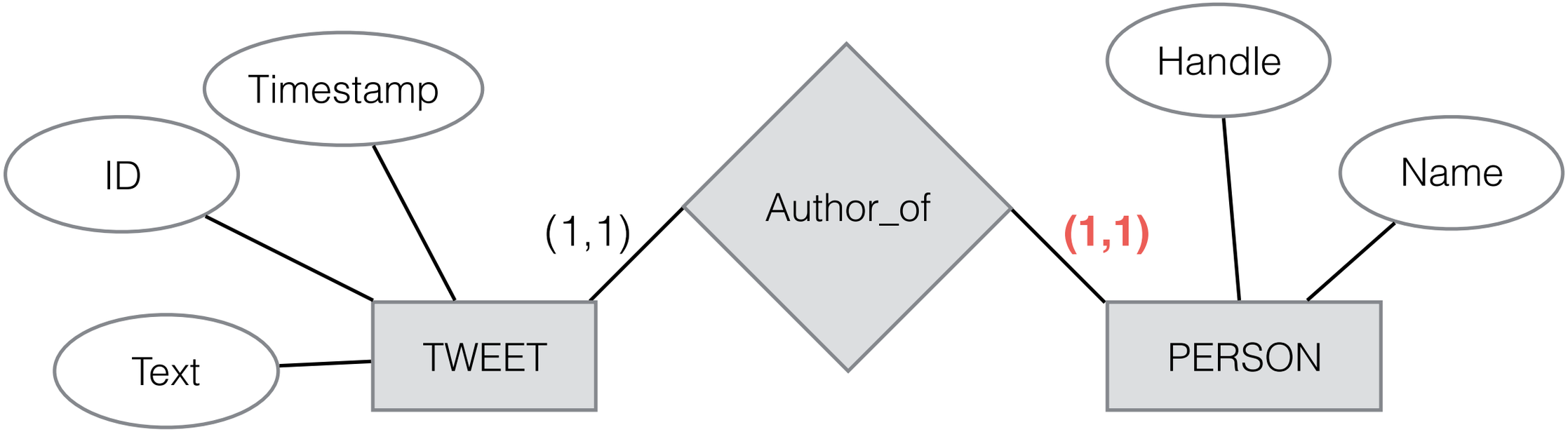
(a)

```
create table AUTHOR (
 Tweet INT,
 Person INT,
);
```

(b)

```
create table AUTHOR (
 Tweet INT,
 Person VARCHAR(1000),
);
```

(c)



TWEET: <ID:INT, Time:TIMESTAMP, Text:VARCHAR(140)>

PERSON: <Handle:VARCHAR(100), Name:VARCHAR(1000)>

```
create table AUTHOR (
 Tweet INT,
 Person VARCHAR(100),
);
```

(a)

```
create table AUTHOR (
 Tweet INT,
 Person INT,
```

Should use handle because they will be unique.

```
create table AUTHOR (
 Tweet INT,
 Person VARCHAR(1000),
);
```

(c)

Inserting Tuples

Inserting Tuples

TWEET: <ID:INT, Time:TIMESTAMP, Text:VARCHAR(140)>

ID	Timestamp	Text

Inserting Tuples

TWEET: <ID:INT, Time:TIMESTAMP, Text:VARCHAR(140)>

ID	Timestamp	Text
389472	2019-01-01 12:34:56	hey

```
insert into TWEET values(
  389472,
  2019-01-01 12:34:56,
  “hey”);
```

```
create table TWEET (
 ID INT,
 Time TIMESTAMP,
 Text VARCHAR(140)
);
```

ID	Timestamp	Text
389472	2019-01-01 12:34:56	hey
NULL	2019-01-01 12:34:57	lol

```
insert into TWEET(Timestamp, Text) values(
   2019-01-01 12:34:57,
   “lol”);
```

```
create table TWEET (
 ID INT,
 Time TIMESTAMP,
 Text VARCHAR(140)
);
```

ID	Timestamp	Text
389472	2019-01-01 12:34:56	hey
NULL	2019-01-01 12:34:57	lol

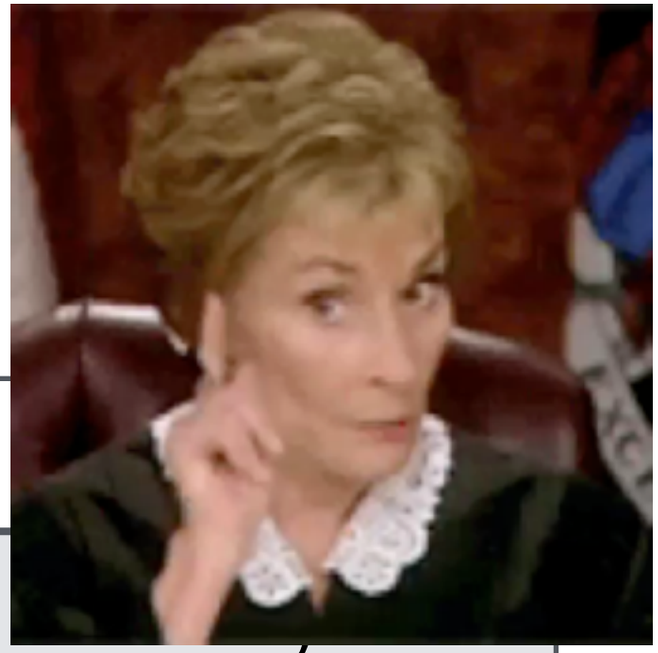
```
insert into TWEET(Timestamp, Text) values(
   2019-01-01 12:34:57,
   “lol”);
```

```
create table TWEET (
 ID INT DEFAULT 0,
 Time TIMESTAMP,
 Text VARCHAR(140)
);
```

ID	Timestamp	Text
389472	2019-01-01 12:34:56	hey
0	2019-01-01 12:34:57	lol

```
insert into TWEET(Timestamp, Text) values(
   2019-01-01 12:34:57,
   “lol”);
```

```
create table TWEET (
 ID INT NOT NULL,
 Time TIMESTAMP,
 Text VARCHAR(140)
);
```



ID	Timestamp	
389472	2019-01-01 12:34:56	

```
insert into TWEET(Timestamp, Text) values(
   2019-01-01 12:34:57,
   "lol");
```

Primary Keys

Primary Keys

TWEET: <ID, Time, Text>

```
create table TWEET (
 ID INT,
 Time TIMESTAMP,
 Text VARCHAR(140),
);
```

Primary Keys

TWEET: <ID, Time, Text>

```
create table TWEET (
 ID INT PRIMARY KEY,
 Time TIMESTAMP,
 Text VARCHAR(140),
);
```

Primary Keys

TWEET: <ID, Time, Text>

```
create table TWEET (
 ID INT PRIMARY KEY,
 Time TIMESTAMP,
 Text VARCHAR(140),
);
```

Enforces "NOT NULL"

Primary Keys

TWEET: <ID, Time, Text>

```
create table TWEET (
 ID INT PRIMARY KEY,
 Time TIMESTAMP,
 Text VARCHAR(140),
);
```

Enforces Uniqueness

Primary Keys

TWEET: <ID, Time, Text>

```
create table TWEET (
 ID INT PRIMARY KEY,
 Time TIMESTAMP,
 Text VARCHAR(140),
);
```

=

```
create table TWEET (
 ID INT,
 Time TIMESTAMP,
 Text VARCHAR(140),
 PRIMARY KEY (ID)
);
```

Primary Keys

```
create table AUTHOR (
 Tweet INT,
 Person VARCHAR(100),
 PRIMARY KEY (Tweet, Person)
);
```

Clicker Question!

Clicker Lightening Round!

(a)

(b)

TWEET

ID	Timestamp	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
596208	2019-01-02 3:14:15	:-D

Clicker Lightening Round!

(a)

(b)

TWEET

ID	Timestamp	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
596208	2019-01-02 3:14:15	:-D

```
create table TWEET (
 ID INT PRIMARY KEY,
 Time TIMESTAMP ,
 Text VARCHAR(140)
);

insert into TWEET
 values(5, “2019-01-01 12:34:57”, “lol”);
```

Clicker Lightening Round!

TWEET

ID	Timestamp	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
596208	2019-01-02 3:14:15	:-D

```
create table TWEET (
 ID INT PRIMARY KEY,
 Time TIMESTAMP ,
 Text VARCHAR(140)
);

insert into TWEET
 values(5, “2019-01-01 12:34:57”, “lol”);
```

Clicker Lightening Round!

(a)

(b)

TWEET

ID	Timestamp	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
596208	2019-01-02 3:14:15	:-D

```
create table TWEET (
ID INT PRIMARY KEY,
Time TIMESTAMP ,
Text VARCHAR(140)
);

insert into TWEET
values(E7w3WKVDB, “2019-01-01 12:34:57”, “lol”);
```

Clicker Lightening Round!

(a)

(b)

TWEET

ID	Timestamp	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
596208	2019-01-02 3:14:15	:-D

```
create table TWEET (
ID INT PRIMARY KEY,
Time TIMESTAMP ,
Text VARCHAR(140)
);

insert into TWEET
values(E7w3WKVDB, “2019-01-01 12:34:57”, “lol”);
```

data type mismatch

Clicker Lightening Round!

(a)

(b)

TWEET

ID	Timestamp	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
596208	2019-01-02 3:14:15	:-D

```
create table TWEET (
ID INT PRIMARY KEY,
Time TIMESTAMP ,
Text VARCHAR(140)
);

insert into TWEET(Timestamp, Text)
values(“2019-01-01 12:34:57”, “lol”);
```

Clicker Lightening Round!

(a)

(b)

TWEET

ID	Timestamp	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
596208	2019-01-02 3:14:15	:-D

```
create table TWEET (
ID INT PRIMARY KEY,
Time TIMESTAMP ,
Text VARCHAR(140)
);

insert into TWEET(Timestamp, Text)
values("2019-01-01 12:34:57", "lol");
```

requires NOT NULL

Clicker Lightening Round!

(a)

(b)

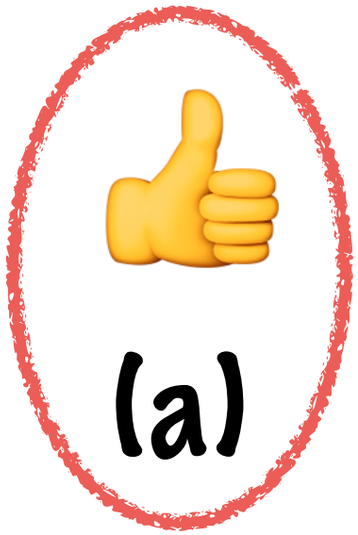
TWEET

ID	Timestamp	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
596208	2019-01-02 3:14:15	:-D

```
create table TWEET (
ID INT NOT NULL,
Time TIMESTAMP,
Text VARCHAR(140) DEFAULT “lol”
);

insert into TWEET(ID, Text)
values(389472, “2019-01-01 12:34:57”);
```

Clicker Lightening Round!



TWEET

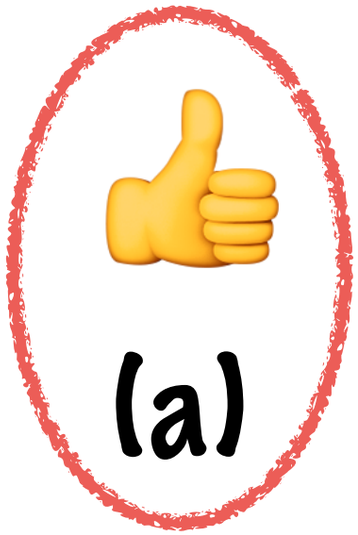
ID	Timestamp	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
596208	2019-01-02 3:14:15	:-D

```
create table TWEET (
ID INT NOT NULL,
Time TIMESTAMP,
Text VARCHAR(140) DEFAULT “lol”
);

insert into TWEET(ID, Text)
values(389472, “2019-01-01 12:34:57”);
```

weird, but technically fine

Clicker Lightening Round!



ID	Timestamp	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
596208	2019-01-02 3:14:15	:-D
389472	NULL	2019-01-01 12:34:57

```
create table TWEET (
ID INT NOT NULL,
Time TIMESTAMP,
Text VARCHAR(140) DEFAULT “lol”
);

insert into TWEET(ID, Text)
values(389472, “2019-01-01 12:34:57”);
```

weird, but technically fine

Clicker Lightening Round!

(a)

(b)

TWEET

ID	Timestamp	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
596208	2019-01-02 3:14:15	:-D

```
create table TWEET (
ID INT PRIMARY KEY,
Time TIMESTAMP ,
Text VARCHAR(140)
);

insert into TWEET
values(389472, “2019-01-04 12:14:37”, “ugh”);
```

Clicker Lightening Round!

(a)

(b)

TWEET

ID	Timestamp	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
596208	2019-01-02 3:14:15	:-D

```
create table TWEET (
ID INT PRIMARY KEY,
Time TIMESTAMP ,
Text VARCHAR(140)
);

insert into TWEET
values(389472, “2019-01-04 12:14:37”, “ugh”);
```

primary key needs to be unique

Foreign Keys

```
create table TWEET (
 ID INT,
 Time TIMESTAMP,
 Text VARCHAR(140)
);
```

```
create table PERSON (
 Handle VARCHAR(100),
 Name VARCHAR(1000),
);
```

```
create table AUTHOR (
 Tweet INT,
 Person VARCHAR(100),
 PRIMARY KEY (Tweet, Person)
);
```

Foreign Keys

```
create table TWEET (
 ID INT,
 Time TIMESTAMP,
 Text VARCHAR(140)
);
```

```
create table PERSON (
 Handle VARCHAR(100),
 Name VARCHAR(1000),
);
```

```
create table AUTHOR (
Tweet INT,
Person VARCHAR(100),
PRIMARY KEY (Tweet, Person)
FOREIGN KEY (Tweet) REFERENCES TWEET(ID),
FOREIGN KEY (Person) REFERENCES PERSON(Handle),
);
```

Foreign Keys

- Not required to be Primary Keys, but!
  - Have to be unique
  - Have to be not NULL
    - NULLs are all considered distinct, i.e. NULL != NULL
- So! Generally stick to the rule of making FK reference a PK
  - If you can’t do this, try refactoring your DB to make it possible (if you are in a position to do this)

Referential Integrity

```
create table PERSON (
 Handle VARCHAR(100),
 Name VARCHAR(1000),
 PRIMARY KEY (Handle)
);
```

```
create table TWEET (
 ID INT PRIMARY KEY,
 Time TIMESTAMP ,
 Text VARCHAR(140)
);
```

```
create table AUTHOR (
 Tweet INT, Person VARCHAR(100),
 FOREIGN KEY (Tweet) REFERENCES TWEET(ID),
 FOREIGN KEY (Person) REFERENCES PERSON(Handle),
);
```

TWEET

ID	Timestamp	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
596208	2019-01-02 3:14:15	:-D
782138	2019-01-04 12:34:57	1951A 4 lyfe

PERSON

Handle	Name
j	Josh
d	Diane
s	Sol

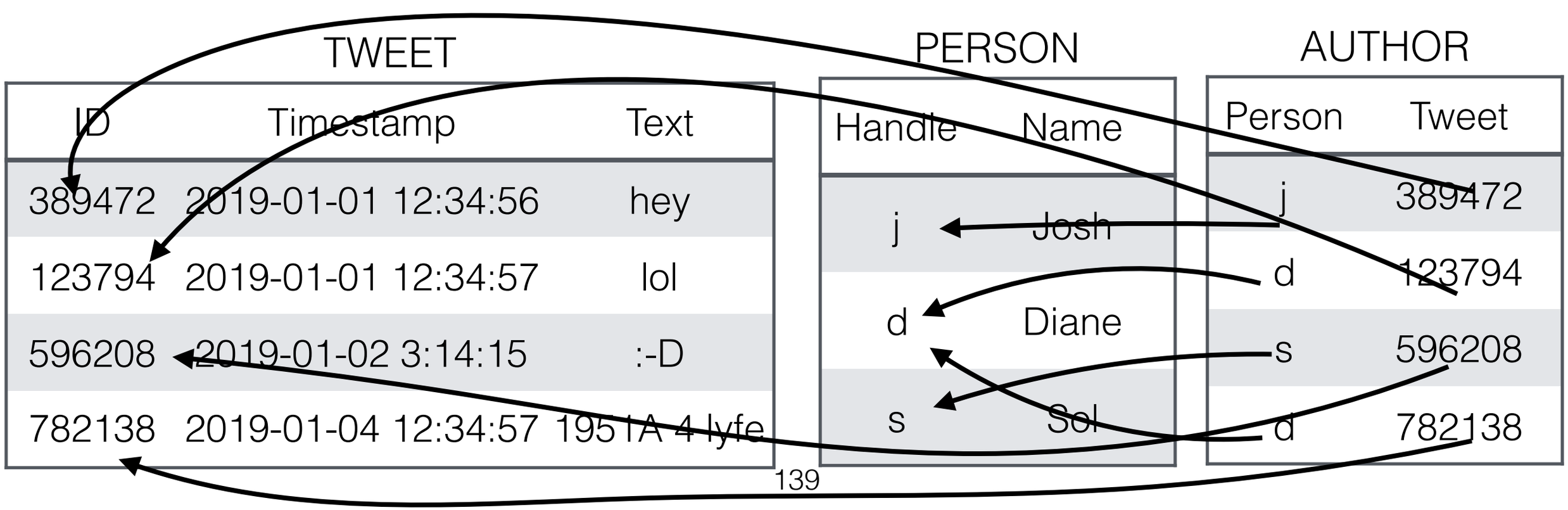
AUTHOR

Person	Tweet
j	389472
d	123794
s	596208
d	782138

```
create table PERSON (
 Handle VARCHAR(100),
 Name VARCHAR(1000),
 PRIMARY KEY (Handle)
);
```

```
create table TWEET (
 ID INT PRIMARY KEY,
 Time TIMESTAMP ,
 Text VARCHAR(140)
);
```

```
create table AUTHOR (
 Tweet INT, Person VARCHAR(100),
 FOREIGN KEY (Tweet) REFERENCES TWEET(ID),
 FOREIGN KEY (Person) REFERENCES PERSON(Handle),
);
```



```
create table PERSON (
 Handle VARCHAR(100),
 Name VARCHAR(1000),
 PRIMARY KEY (Handle)
);
```

```
create table TWEET (
 ID INT PRIMARY KEY,
 Time TIMESTAMP ,
 Text VARCHAR(140)
);
```

```
create table AUTHOR (
 Tweet INT, Person VARCHAR(100),
 FOREIGN KEY (Tweet) REFERENCES TWEET(ID),
 FOREIGN KEY (Person) REFERENCES PERSON(Handle),
);
```

```
DELETE FROM TWEET WHERE ID = "596208"
```

TWEET

ID	Timestamp	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
596208	2019-01-02 3:14:15	:-D
782138	2019-01-04 12:34:57	195IA 4 lyfe

PERSON

Handle	Name
j	Josh
d	Diane
s	Sol

AUTHOR

Person	Tweet
j	389472
d	123794
s	596208
d	782138

```
create table PERSON (
 Handle VARCHAR(100),
 Name VARCHAR(1000),
 PRIMARY KEY (Handle)
);
```

```
create table TWEET (
 ID INT PRIMARY KEY,
 Time TIMESTAMP ,
 Text VARCHAR(140)
);
```

```
create table AUTHOR (
 Tweet INT, Person VARCHAR(100),
 FOREIGN KEY (Tweet) REFERENCES TWEET(ID),
 FOREIGN KEY (Person) REFERENCES PERSON(Handle),
);
```

```
DELETE FROM TWEET WHERE ID = "596208"
```

TWEET

ID	Timestamp	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
782138	2019-01-04 12:34:57	1951A 4 lyfe

PERSON

Handle	Name
j	Josh
d	Diane
s	Sol

AUTHOR

Person	Tweet
j	389472
d	123794
s	596208
d	782138

```
create table PERSON (
 Handle VARCHAR(100),
 Name VARCHAR(1000),
 PRIMARY KEY (Handle)
);
```

```
create table TWEET (
 ID INT PRIMARY KEY,
 Time TIMESTAMP ,
 Text VARCHAR(140)
);
```

```
create table AUTHOR (
 Tweet INT, Person VARCHAR(100),
 FOREIGN KEY (Tweet) REFERENCES TWEET(ID) ON DELETE SET NULL,
 FOREIGN KEY (Person) REFERENCES PERSON(Handle),
);
```

```
DELETE FROM TWEET WHERE ID = "596208"
```

TWEET

ID	Timestamp	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
782138	2019-01-04 12:34:57	1951A 4 lyfe

PERSON

Handle	Name
j	Josh
d	Diane
s	Sol

AUTHOR

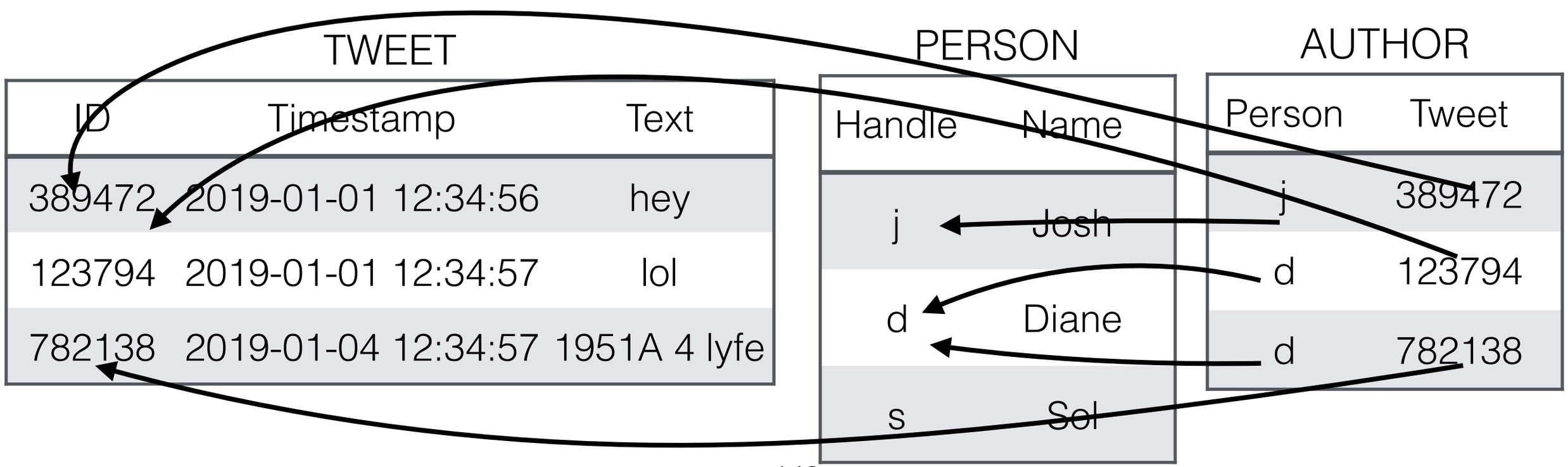
Person	Tweet
j	389472
d	123794
s	**NULL**
d	782138

```
create table PERSON (
 Handle VARCHAR(100),
 Name VARCHAR(1000),
 PRIMARY KEY (Handle)
);
```

```
create table TWEET (
 ID INT PRIMARY KEY,
 Time TIMESTAMP ,
 Text VARCHAR(140)
);
```

```
create table AUTHOR (
 Tweet INT, Person VARCHAR(100),
 FOREIGN KEY (Tweet) REFERENCES TWEET(ID) ON DELETE CASCADE,
 FOREIGN KEY (Person) REFERENCES PERSON(Handle),
);
```

```
DELETE FROM TWEET WHERE ID = "596208"
```

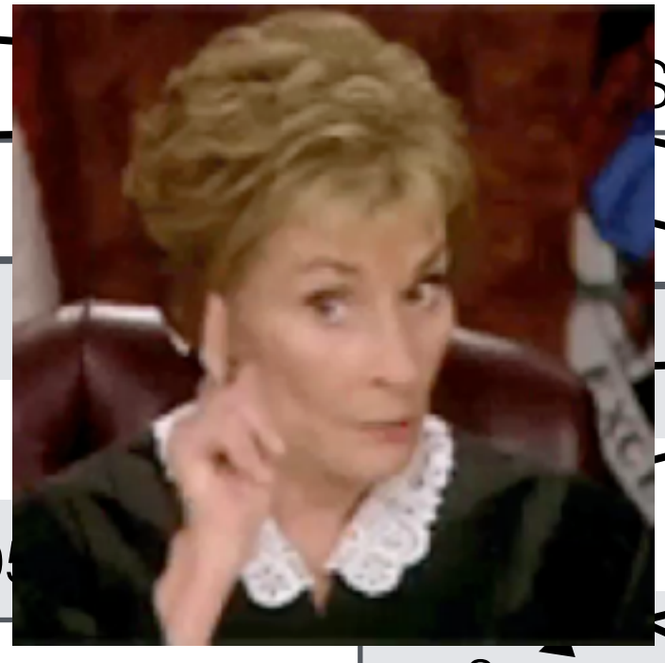


```
create table PERSON (
 Handle VARCHAR(100),
 Name VARCHAR(1000),
 PRIMARY KEY (Handle)
);
```

```
create table TWEET (
 ID INT PRIMARY KEY,
 Time TIMESTAMP ,
 Text VARCHAR(140)
);
```

```
create table AUTHOR (
 Tweet INT, Person VARCHAR(100),
 FOREIGN KEY (Tweet) REFERENCES TWEET(ID) ON DELETE RESTRICT,
 FOREIGN KEY (Person) REFERENCES PERSON(Handle),
);
```

```
DELETE FROM TWEET WHERE ID = "596208"
```



TWEET

ID	Timestamp	
389472	2019-01-01 12:34:56	
123794	2019-01-01 12:34:57	
782138	2019-01-04 12:34:57	19

SON

Name
Josh
Diane
Sol

AUTHOR

Person	Tweet
j	389472
d	123794
s	NULL
d	782138

```
create table PERSON (
 Handle VARCHAR(100),
 Name VARCHAR(1000),
 PRIMARY KEY (Handle)
);
```

```
create table TWEET (
 ID INT PRIMARY KEY,
 Time TIMESTAMP ,
 Text VARCHAR(140)
);
```

```
create table AUTHOR (
 Tweet INT, Person VARCHAR(100),
 FOREIGN KEY (Tweet) REFERENCES TWEET(ID)
 ON DELETE RESTRICT
 ON UPDATE CASCADE,
 FOREIGN KEY (Person) REFERENCES PERSON(Handle),
);
```

```
DELETE FROM TWEET WHERE ID = "596208"
```

TWEET

ID	Timestamp	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
596208	2019-01-02 3:14:15	:-D
782138	2019-01-04 12:34:57	1951A 4 lyfe

PERSON

Handle	Name
j	Josh
d	Diane
s	Sol

AUTHOR

Person	Tweet
j	389472
d	123794
s	596208
d	782138

Clicker Question!

Clicker Question!

```
create table PERSON (
 Handle VARCHAR(100), Name VARCHAR(1000),
);
create table TWEET (
 ID INT, Text VARCHAR(140), Author VARCHAR(100),
 FOREIGN KEY (Author) REFERENCES PERSON(Handle) ON DELETE CASCADE,
);
create table RETWEET (
 Person VARCHAR(100), Tweet INT,
 FOREIGN KEY (Person) REFERENCES PERSON(Handle) ON DELETE SET NULL,
 FOREIGN KEY (Tweet) REFERENCES TWEET(ID) ON DELETE SET NULL,
);
```

PERSON

Handle	Name
j	Josh
d	Diane
s	Sol

TWEET

ID	Text	Author
1	hey	s
2	lol	s
3	:-D	d

RETWEET

Person	Tweet
j	1
j	2
d	1

```
create table PERSON (
 Handle VARCHAR(100), Name VARCHAR(1000),
);
create table TWEET (
 ID INT, Text VARCHAR(140), Author VARCHAR(100),
 FOREIGN KEY (Author) REFERENCES PERSON(Handle) ON DELETE CASCADE,
);
create table RETWEET (
 Person VARCHAR(100), Tweet INT,
 FOREIGN KEY (Person) REFERENCES PERSON(Handle) ON DELETE SET NULL,
 FOREIGN KEY (Tweet) REFERENCES TWEET(ID) ON DELETE SET NULL,
);
```

PERSON

Handle	Name
j	Josh
d	Diane
s	Sol

TWEET

ID	Text	Author
1	hey	s
2	lol	s
3	:-D	d

RETWEET

Person	Tweet
j	1
j	2
d	1

```
DELETE FROM PERSON WHERE Handle = "s"
```

What happens...?

Clicker Question!

PERSON | TWEET | RETWEET

(a)

Handle	Name
j	Josh
d	Diane

ID	Text	Author
3	:-D	d

Person	Tweet

(b)

Handle	Name
j	Josh
d	Diane

ID	Text	Author
3	:-D	d

Person	Tweet
NULL	NULL
NULL	NULL
NULL	NULL

(c)

Handle	Name
j	Josh
d	Diane

ID	Text	Author
3	:-D	d

Person	Tweet
j	NULL
j	NULL
d	NULL

Clicker Question!

PERSON | TWEET | RETWEET

(a)

Handle	Name
j	Josh
d	Diane

ID	Text	Author
3	:-D	d

Person	Tweet

(b)

Handle	Name
j	Josh
d	Diane

ID	Text	Author
3	:-D	d

Person	Tweet
NULL	NULL
NULL	NULL
NULL	NULL

(c)

Handle	Name
j	Josh
d	Diane

ID	Text	Author
3	:-D	d

Person	Tweet
j	NULL
j	NULL
d	NULL

```
create table PERSON (
 Handle VARCHAR(100), Name VARCHAR(1000),
);
create table TWEET (
 ID INT, Text VARCHAR(140), Author VARCHAR(100),
 FOREIGN KEY (Author) REFERENCES PERSON(Handle) ON DELETE CASCADE,
);
create table RETWEET (
 Person VARCHAR(100), Tweet INT,
 FOREIGN KEY (Person) REFERENCES PERSON(Handle) ON DELETE SET NULL,
 FOREIGN KEY (Tweet) REFERENCES TWEET(ID) ON DELETE SET NULL,
);
```

PERSON

Handle	Name
j	Josh
d	Diane
s	Sol

TWEET

ID	Text	Author
1	hey	s
2	lol	s
3	:-D	d

RETWEET

Person	Tweet
j	1
j	2
d	1

```
DELETE FROM PERSON WHERE Handle = "s"
```

What happens...?

```
create table PERSON (
 Handle VARCHAR(100), Name VARCHAR(1000),
);
create table TWEET (
 ID INT, Text VARCHAR(140), Author VARCHAR(100),
 FOREIGN KEY (Author) REFERENCES PERSON(Handle) ON DELETE CASCADE,
);
create table RETWEET (
 Person VARCHAR(100), Tweet INT,
 FOREIGN KEY (Person) REFERENCES PERSON(Handle) ON DELETE SET NULL,
 FOREIGN KEY (Tweet) REFERENCES TWEET(ID) ON DELETE SET NULL,
);
```

PERSON

Handle	Name
j	Josh
d	Diane

TWEET

ID	Text	Author
1	hey	s
2	lol	s
3	:-D	d

RETWEET

Person	Tweet
j	1
j	2
d	1

```
DELETE FROM PERSON WHERE Handle = "s"
```

What happens...?

```
create table PERSON (
 Handle VARCHAR(100), Name VARCHAR(1000),
);
create table TWEET (
 ID INT, Text VARCHAR(140), Author VARCHAR(100),
 FOREIGN KEY (Author) REFERENCES PERSON(Handle) ON DELETE CASCADE,
);
create table RETWEET (
 Person VARCHAR(100), Tweet INT,
 FOREIGN KEY (Person) REFERENCES PERSON(Handle) ON DELETE SET NULL,
 FOREIGN KEY (Tweet) REFERENCES TWEET(ID) ON DELETE SET NULL,
);
```

PERSON

Handle	Name
j	Josh
d	Diane

TWEET

ID	Text	**Author**
1	hey	**s**
2	lol	**s**
3	:-D	d

RETWEET

Person	Tweet
j	1
j	2
d	1

```
DELETE FROM PERSON WHERE Handle = "s"
```

What happens...?

```
create table PERSON (
 Handle VARCHAR(100), Name VARCHAR(1000),
);
create table TWEET (
 ID INT, Text VARCHAR(140), Author VARCHAR(100),
 FOREIGN KEY (Author) REFERENCES PERSON(Handle) ON DELETE CASCADE,
);
create table RETWEET (
 Person VARCHAR(100), Tweet INT,
 FOREIGN KEY (Person) REFERENCES PERSON(Handle) ON DELETE SET NULL,
 FOREIGN KEY (Tweet) REFERENCES TWEET(ID) ON DELETE SET NULL,
);
```

PERSON

Handle	Name
j	Josh
d	Diane

TWEET

ID	Text	Author
3	:-D	d

RETWEET

Person	Tweet
j	1
j	2
d	1

```
DELETE FROM PERSON WHERE Handle = "s"
```

What happens...?

```
create table PERSON (
 Handle VARCHAR(100), Name VARCHAR(1000),
);
create table TWEET (
 ID INT, Text VARCHAR(140), Author VARCHAR(100),
 FOREIGN KEY (Author) REFERENCES PERSON(Handle) ON DELETE CASCADE,
);
create table RETWEET (
 Person VARCHAR(100), Tweet INT,
 FOREIGN KEY (Person) REFERENCES PERSON(Handle) ON DELETE SET NULL,
 FOREIGN KEY (Tweet) REFERENCES TWEET(ID) ON DELETE SET NULL,
);
```

PERSON

Handle	Name
j	Josh
d	Diane

TWEET

ID	Text	Author
3	:-D	d

RETWEET

Person	Tweet
j	1
j	2
d	1

```
DELETE FROM PERSON WHERE Handle = "s"
```

What happens...?

```
create table PERSON (
 Handle VARCHAR(100), Name VARCHAR(1000),
);
create table TWEET (
 ID INT, Text VARCHAR(140), Author VARCHAR(100),
 FOREIGN KEY (Author) REFERENCES PERSON(Handle) ON DELETE CASCADE,
);
create table RETWEET (
 Person VARCHAR(100), Tweet INT,
 FOREIGN KEY (Person) REFERENCES PERSON(Handle) ON DELETE SET NULL,
 FOREIGN KEY (Tweet) REFERENCES TWEET(ID) ON DELETE SET NULL,
);
```

PERSON

Handle	Name
j	Josh
d	Diane

TWEET

ID	Text	Author
3	:-D	d

RETWEET

Person	Tweet
j	NULL
j	NULL
d	NULL

```
DELETE FROM PERSON WHERE Handle = "s"
```

What happens...?

k bye