BROWN
Computer Science

CS1951A: Data Science

Lecture 3: The Entity-Relationship model

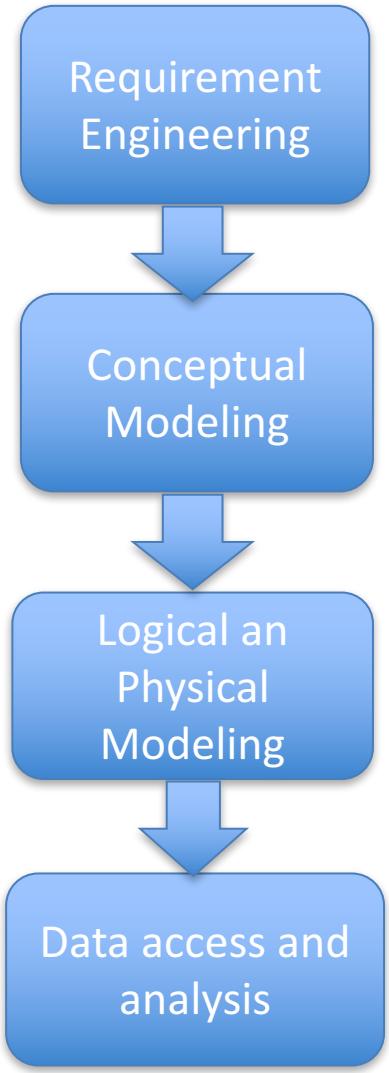
Lorenzo De Stefani

Spring 2022

Outline

- Conceptual DB design
- Entities and Relationships
- Attributes
- Relational model
- SQL

Databases for Data Scientists



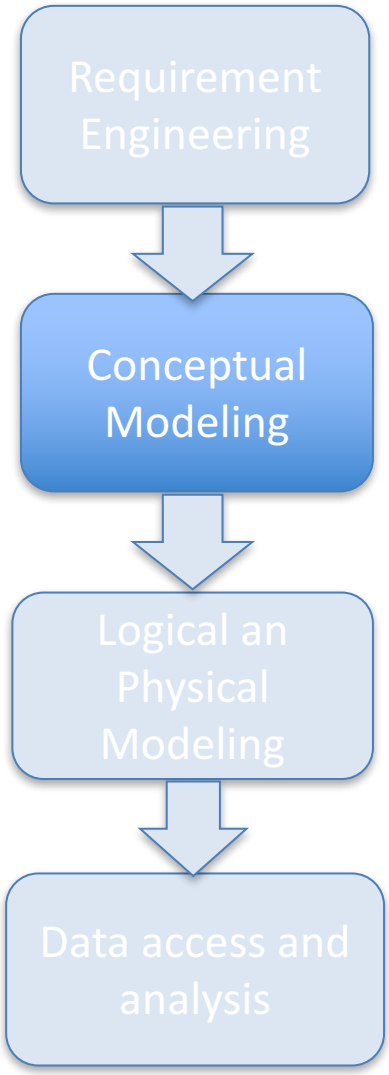
- "Book of duty"
- Understand and model the "world" of interest

- Conceptual DB design
- Entity Relations (ER) method

- Logical design (schema, table names, data types)
- Physical design (index, hints, memory organization)

- Asking and answering questions (queries)
- Extract information form the DBMS (views)
- SQL and relational algebra

Databases for Data Scientists



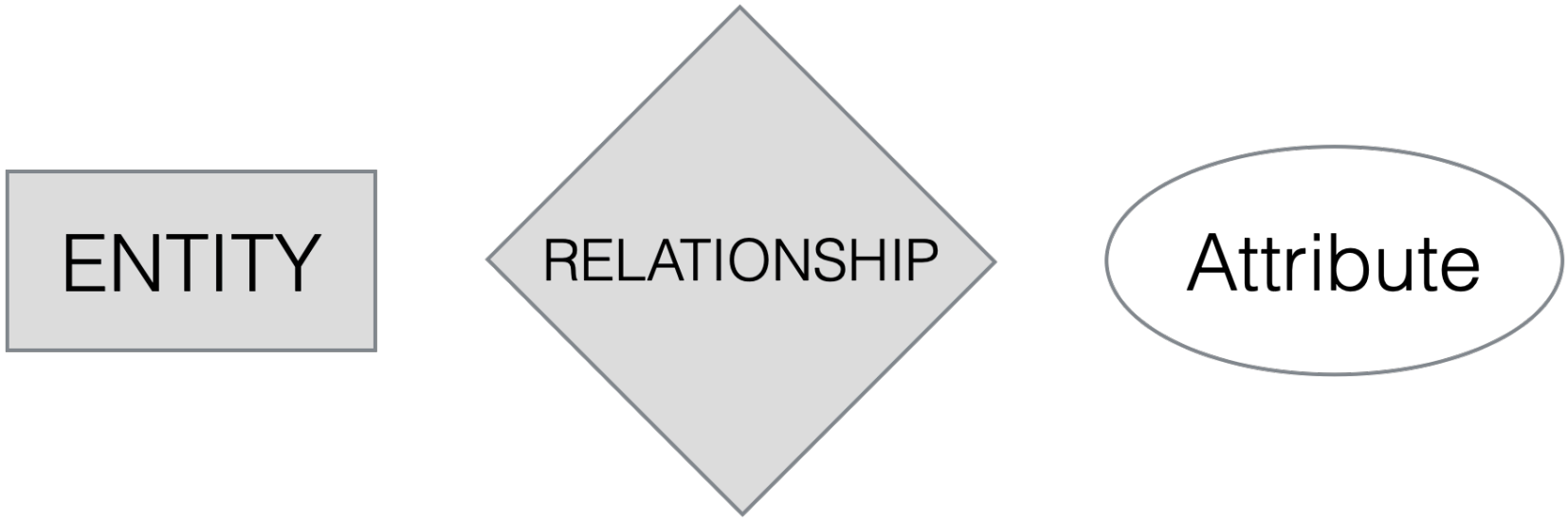
- "Book of duty"
- Understand and model the "world" of interest

- Conceptual DB design
- Entity Relations (ER) method

- Logical design (schema, table names, data types)
- Physical design (index, hints, memory organization)

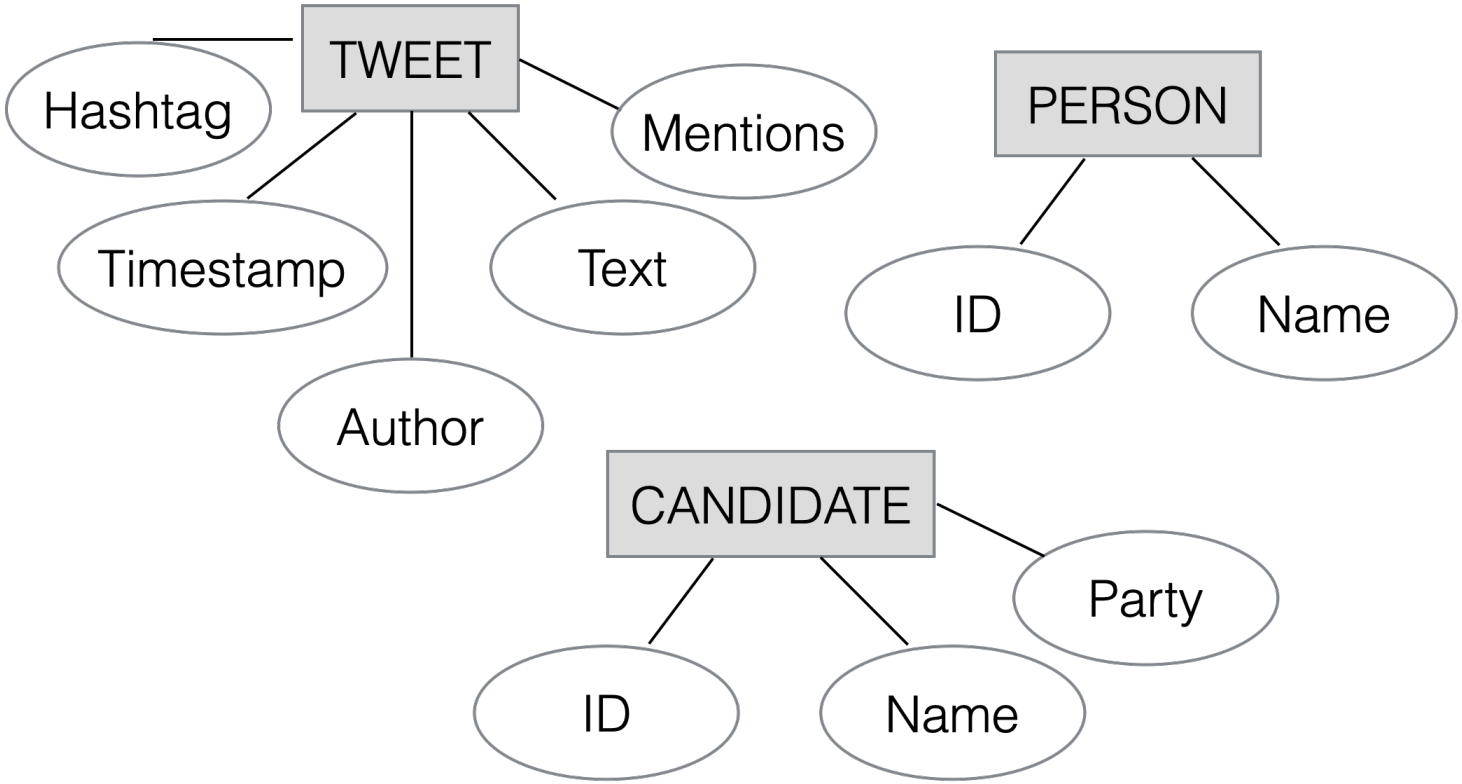
- Asking and answering questions (queries)
- Extract information form the DBMS (views)
- SQL and relational algebra

The Entity-Relationship (ER) model

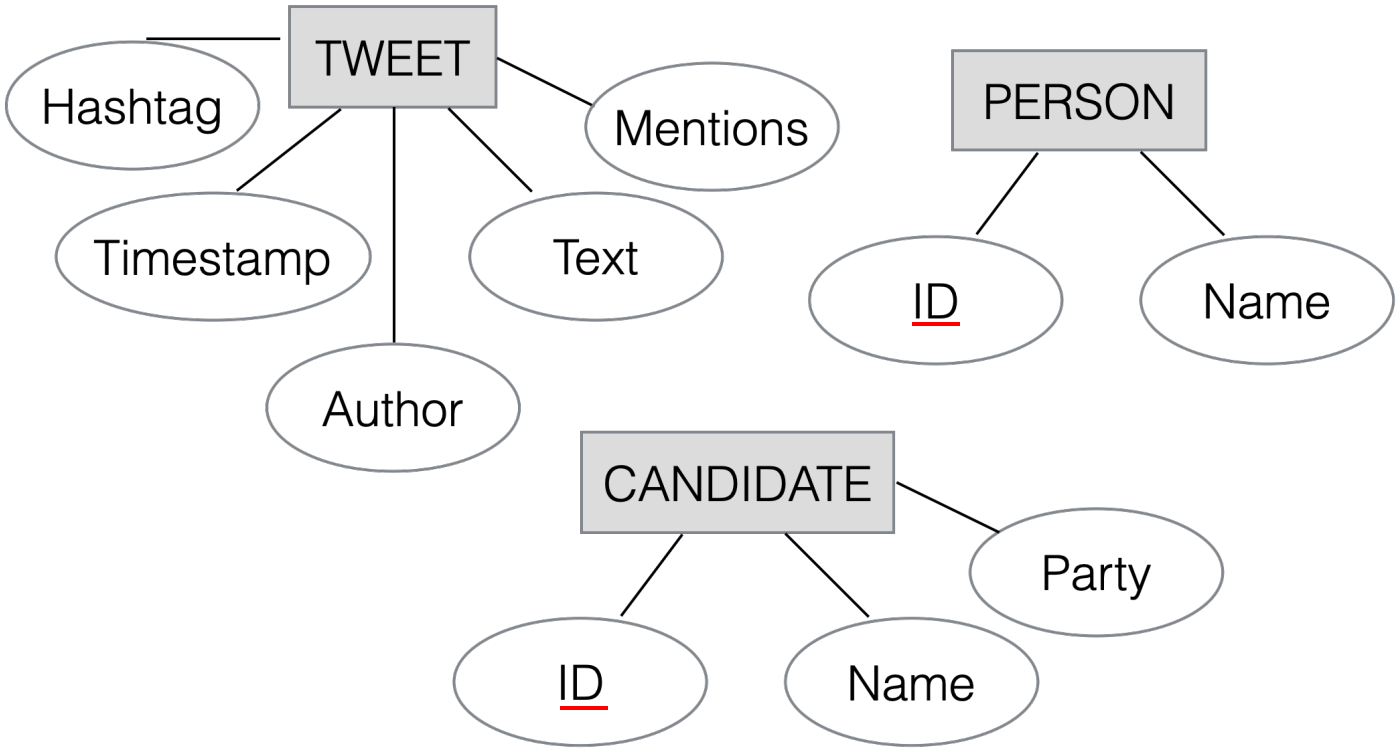


The Entity-Relationship (ER) model

- We want to populate a database of tweet related to the 2020 presidential election

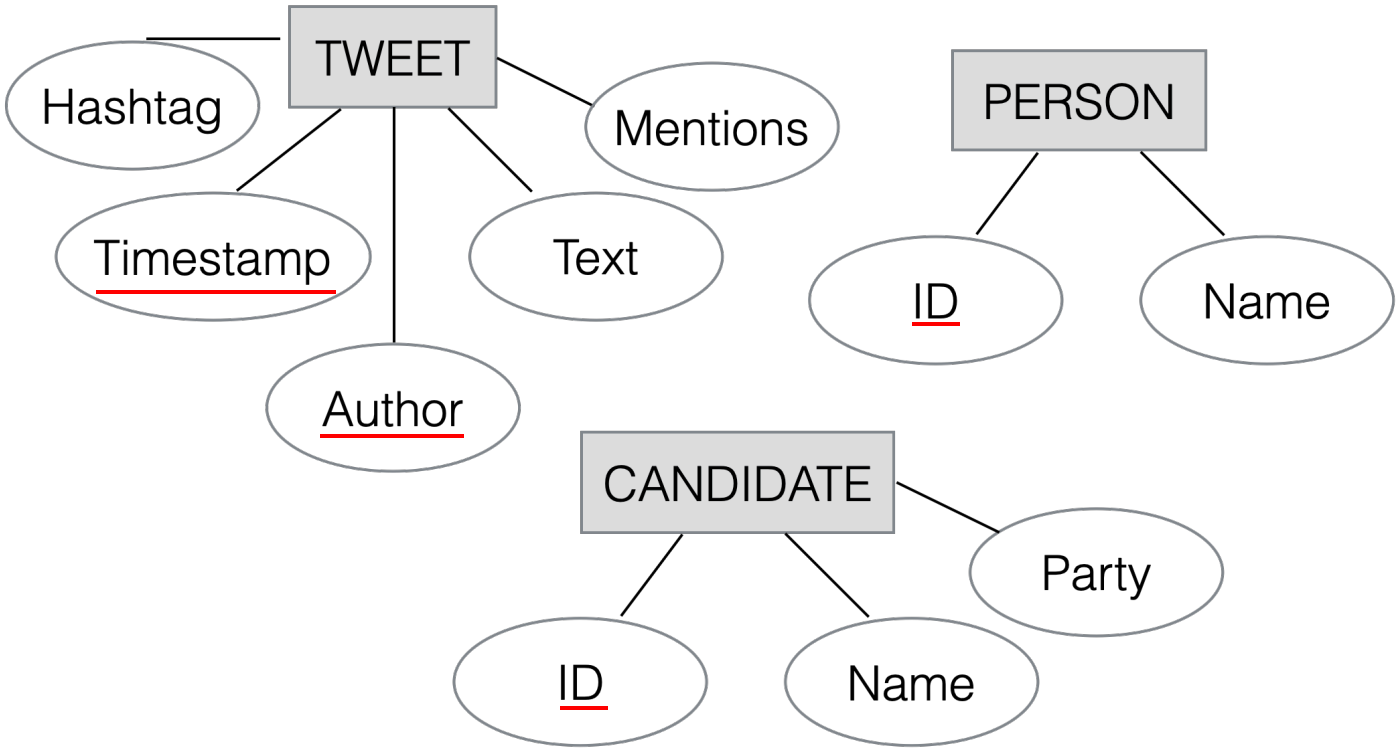


Identifier attributes



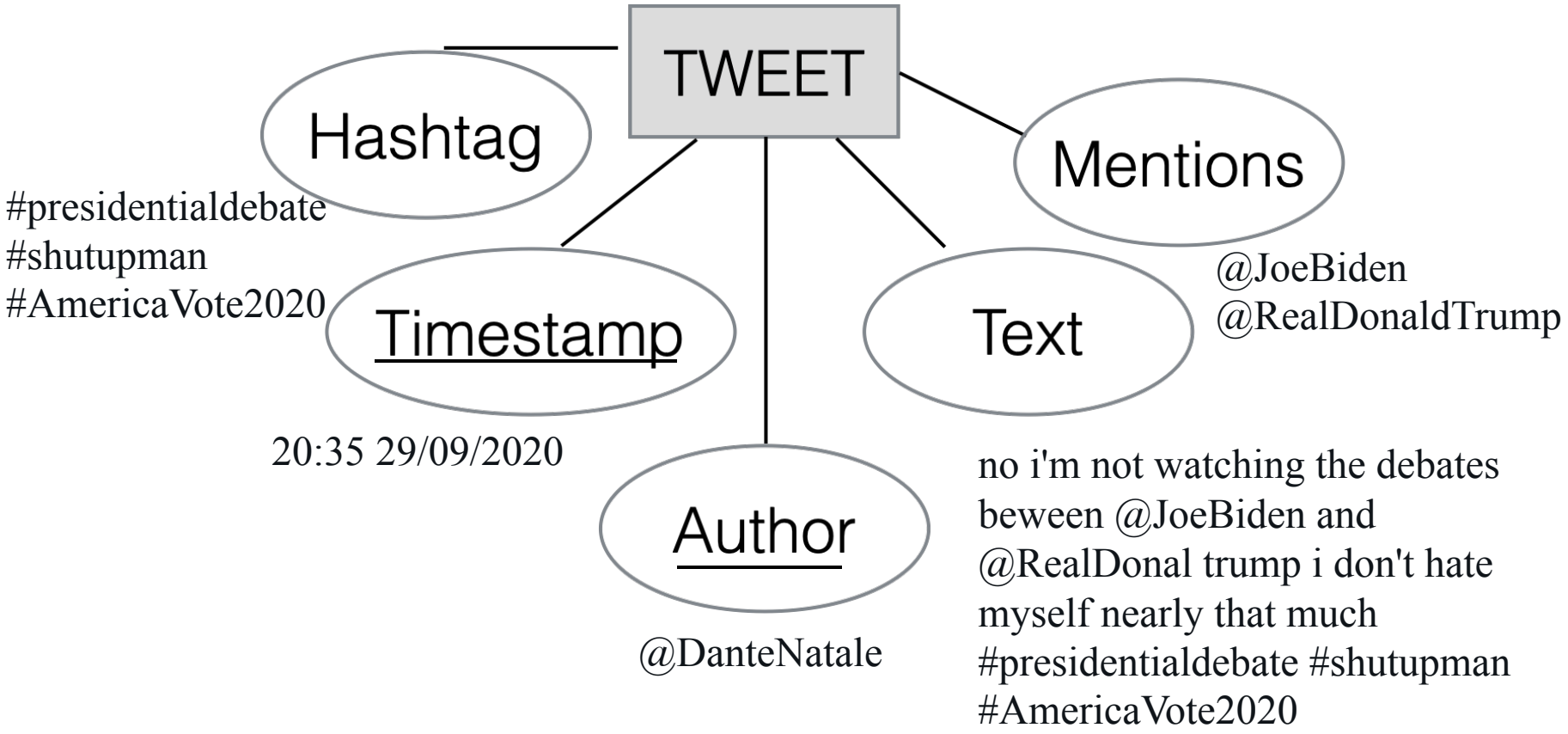
- Identifiers are attributes whose values uniquely identifies the corresponding concept

Identifier attributes

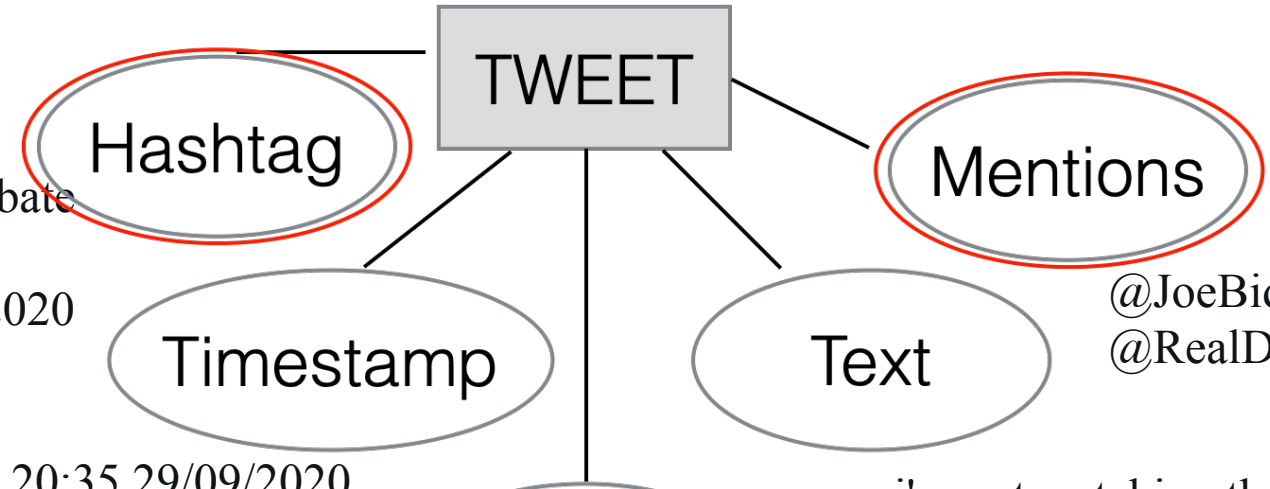


- Identifiers are attributes whose values uniquely identifies the corresponding concept
- In some cases, pairs (or tuples) of attributes can be used in combination to obtain an identifier

Multivalued attributes



Multivalued attributes



#presidentialdebate
#shutupman
#AmericaVote2020

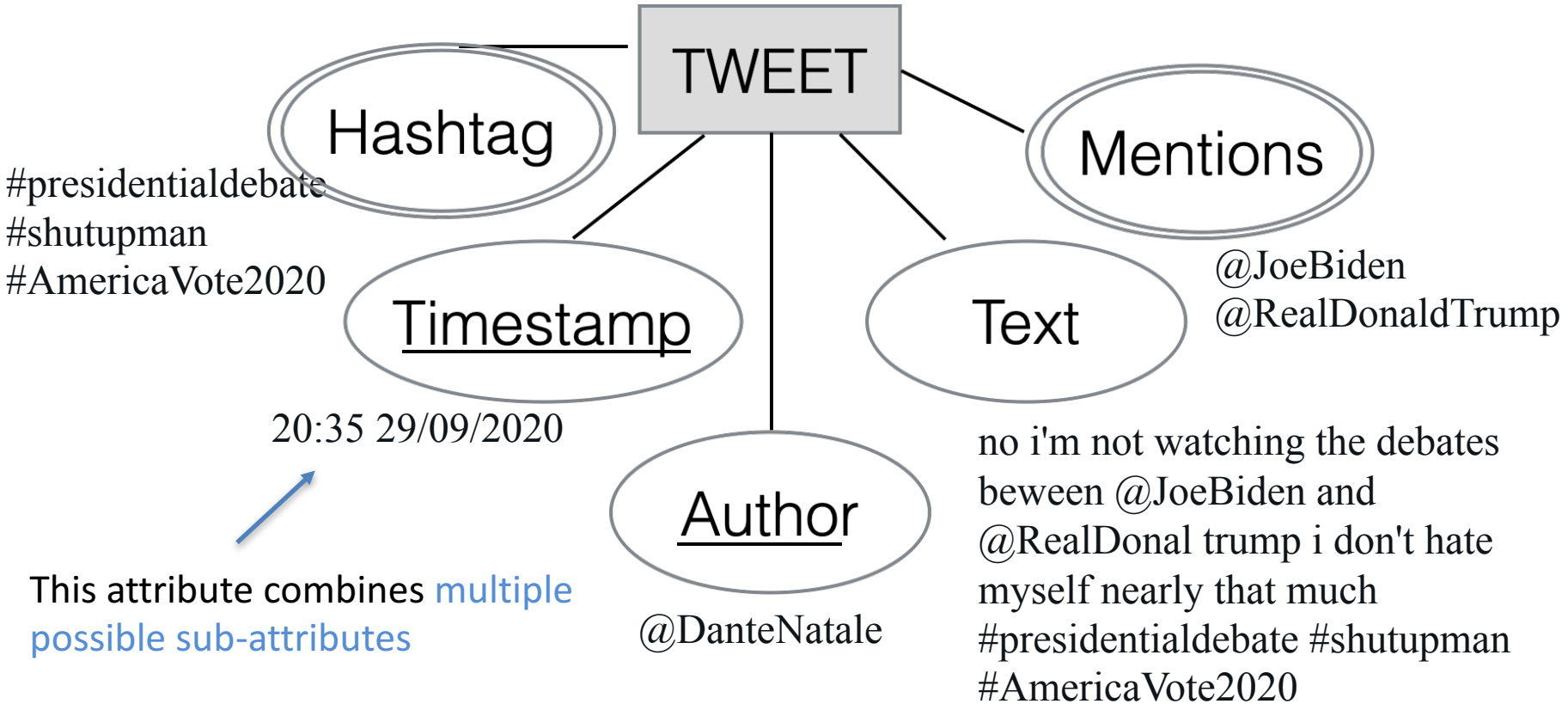
@JoeBiden
@RealDonaldTrump

20:35 29/09/2020

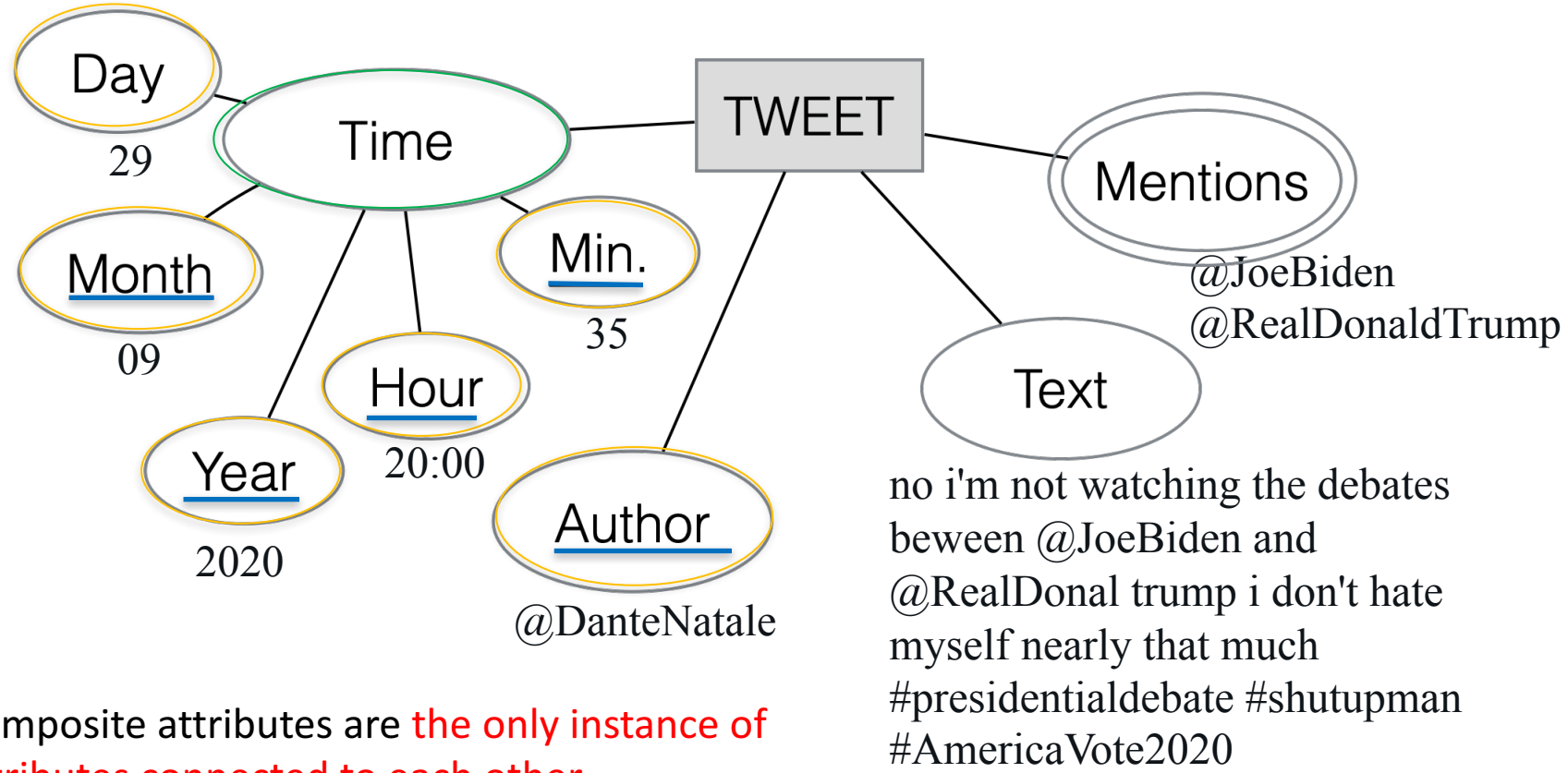
no i'm not watching the debates beween @JoeBiden and @RealDonal trump i don't hate myself nearly that much #presidentialdebate #shutupman #AmericaVote2020

@DanteNatale

Composite attributes



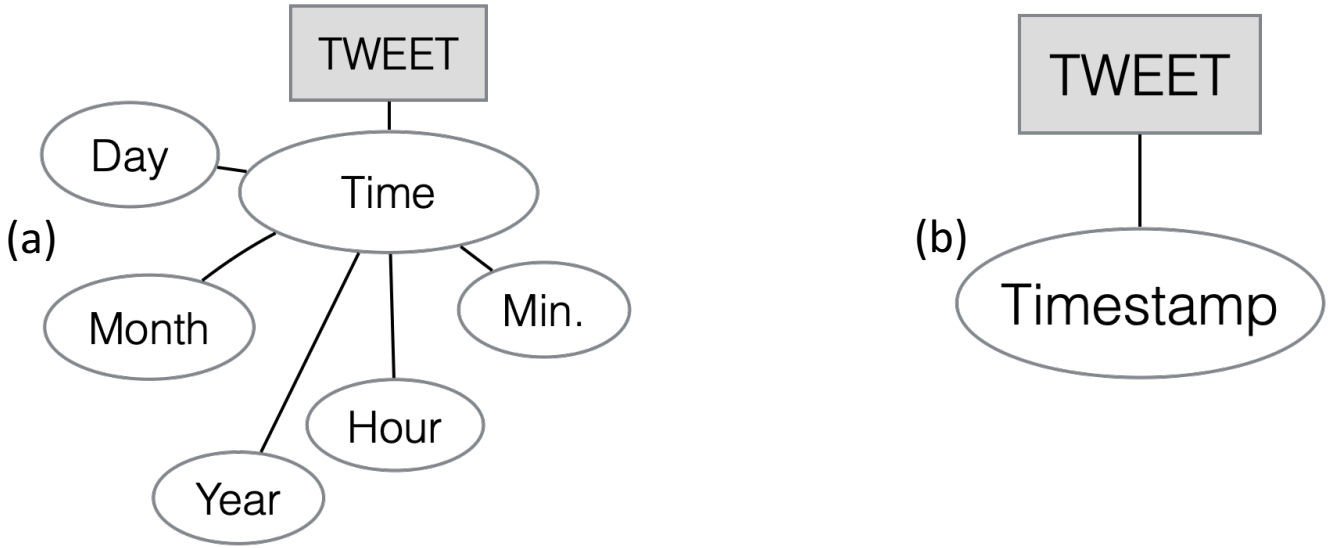
Composite attributes



- Composite attributes are the only instance of attributes connected to each other
- Central attribute not associate to any value
- Sub-attributes compose the attribute value
- If the central attribute is an identifier (or part of it), the sub-attributes become part of composite identifier

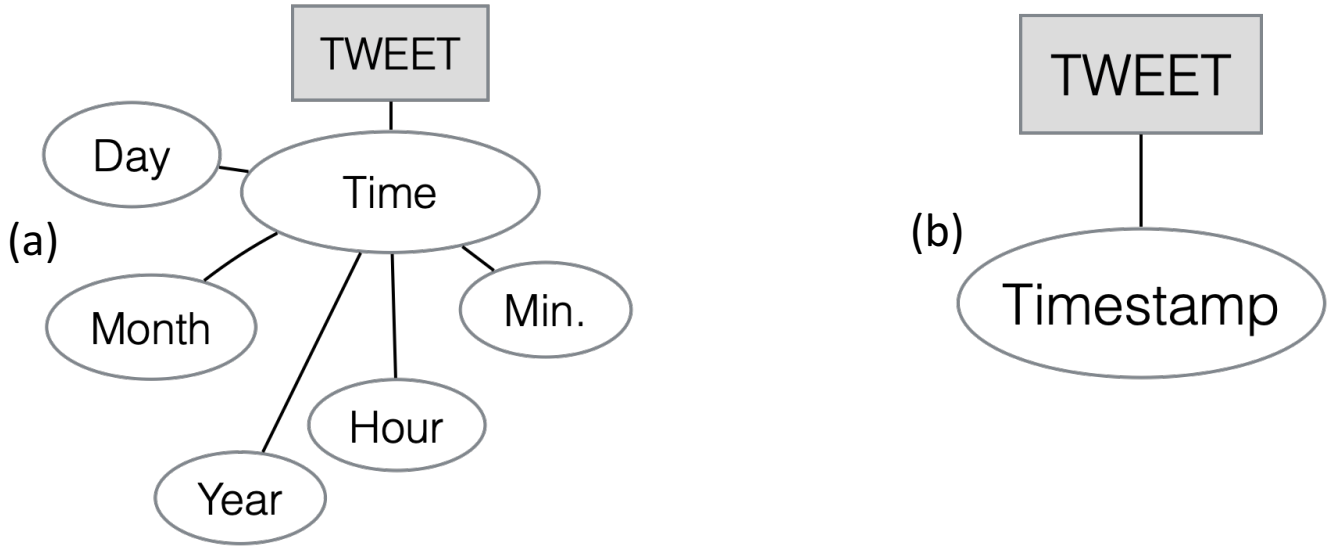
Composite attributes

- Which representation is better?

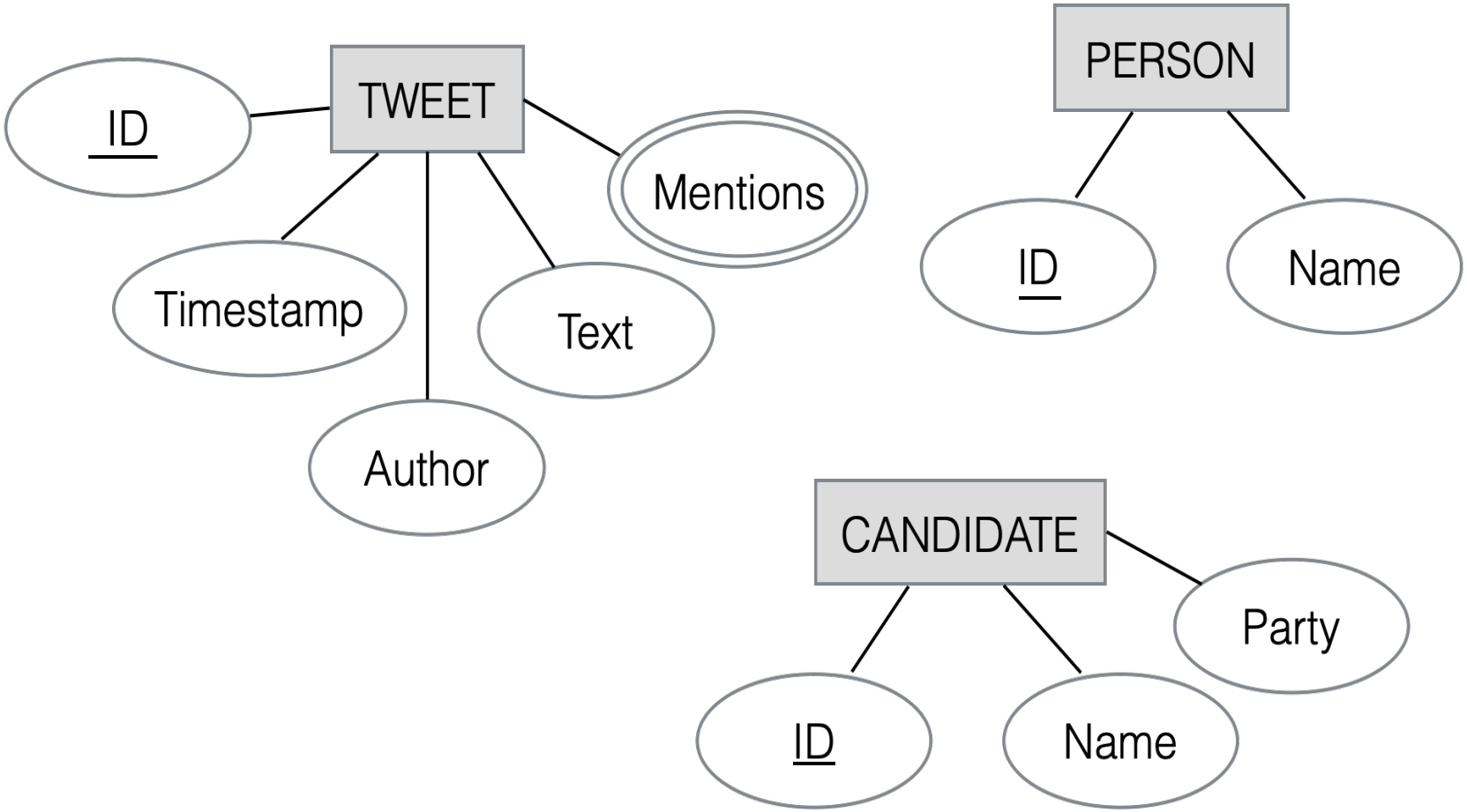


Composite attributes

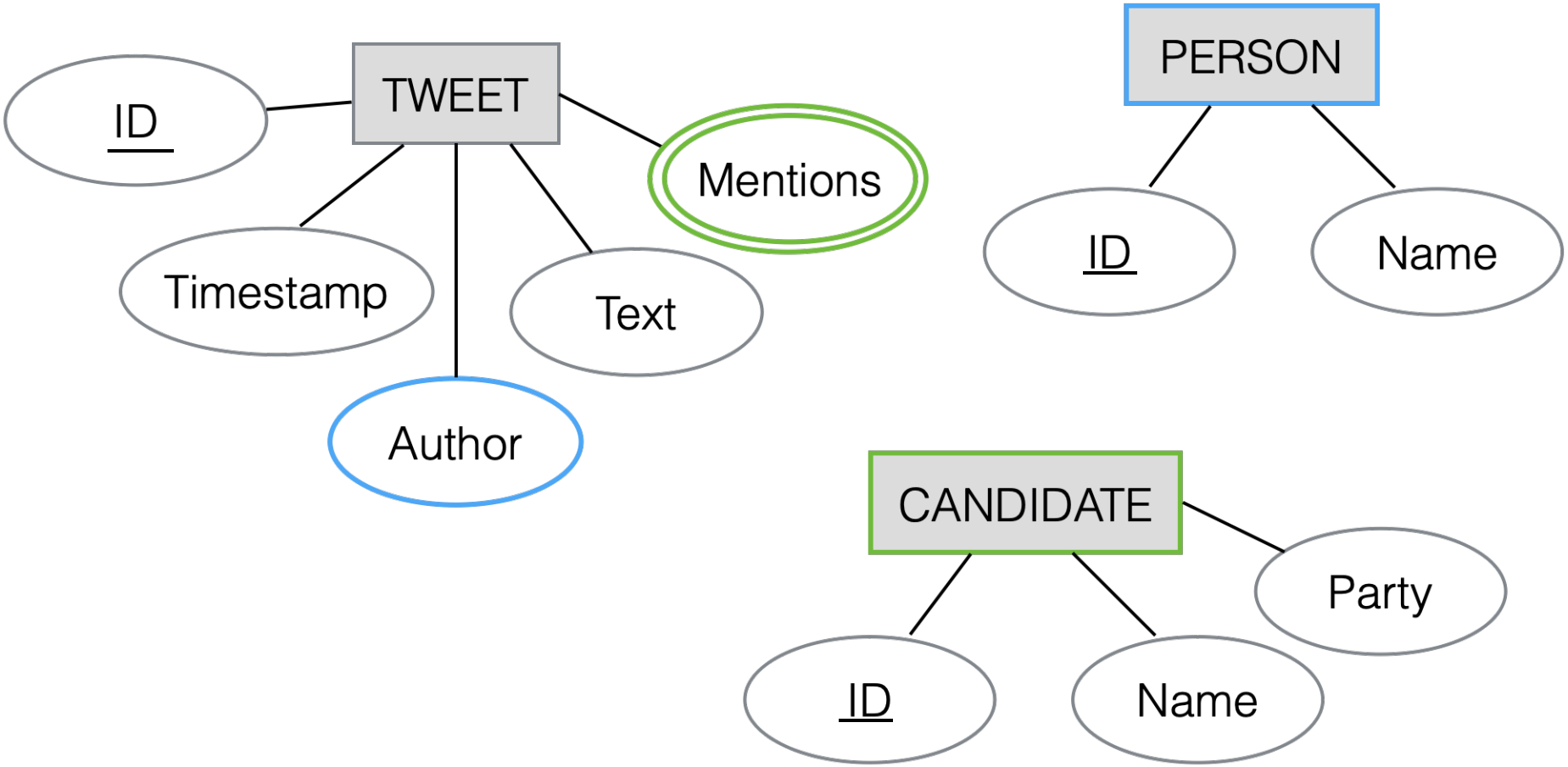
Consider the query: “Find all tweets send between 2 and 4 pm related to the debate”



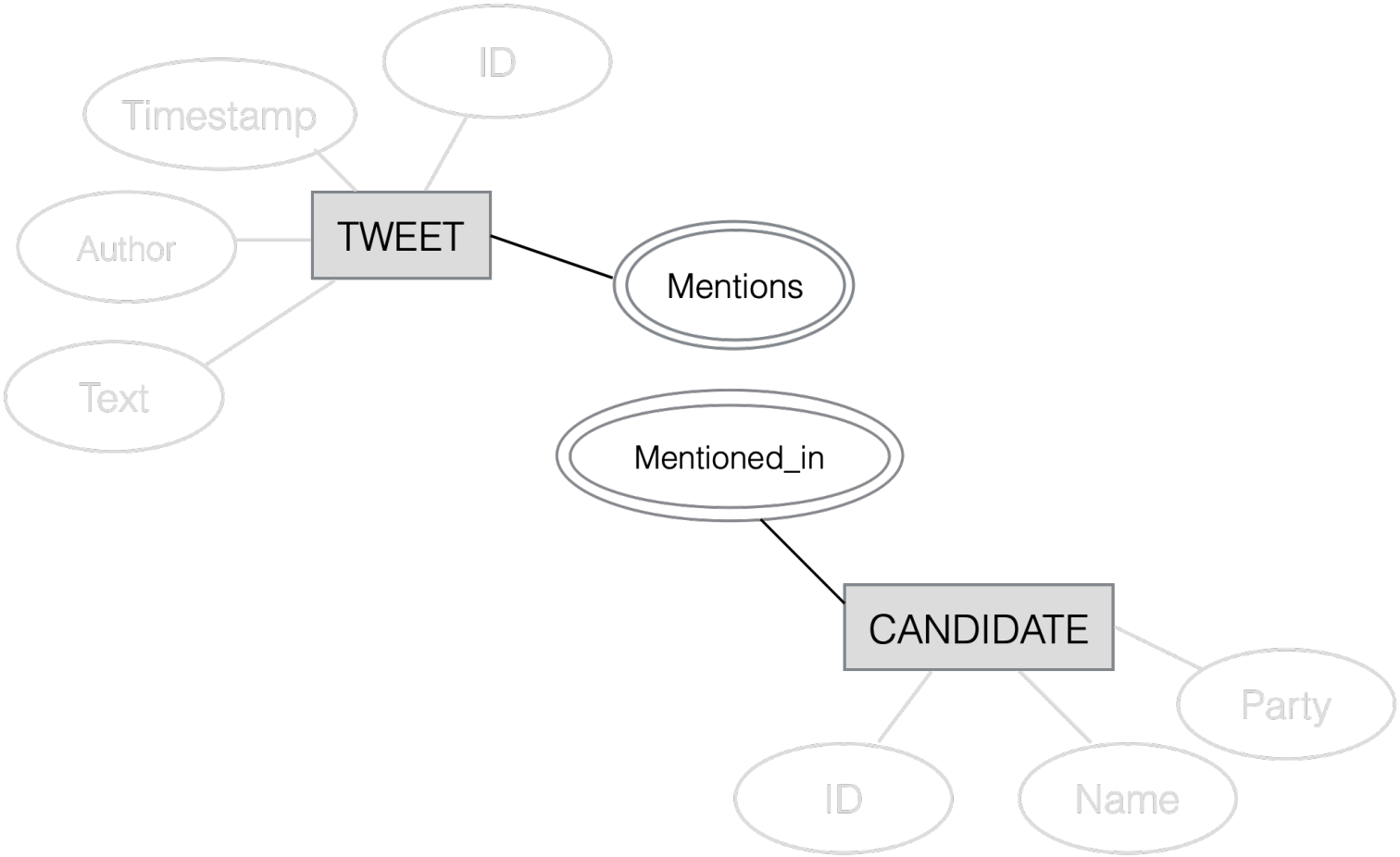
Relationships



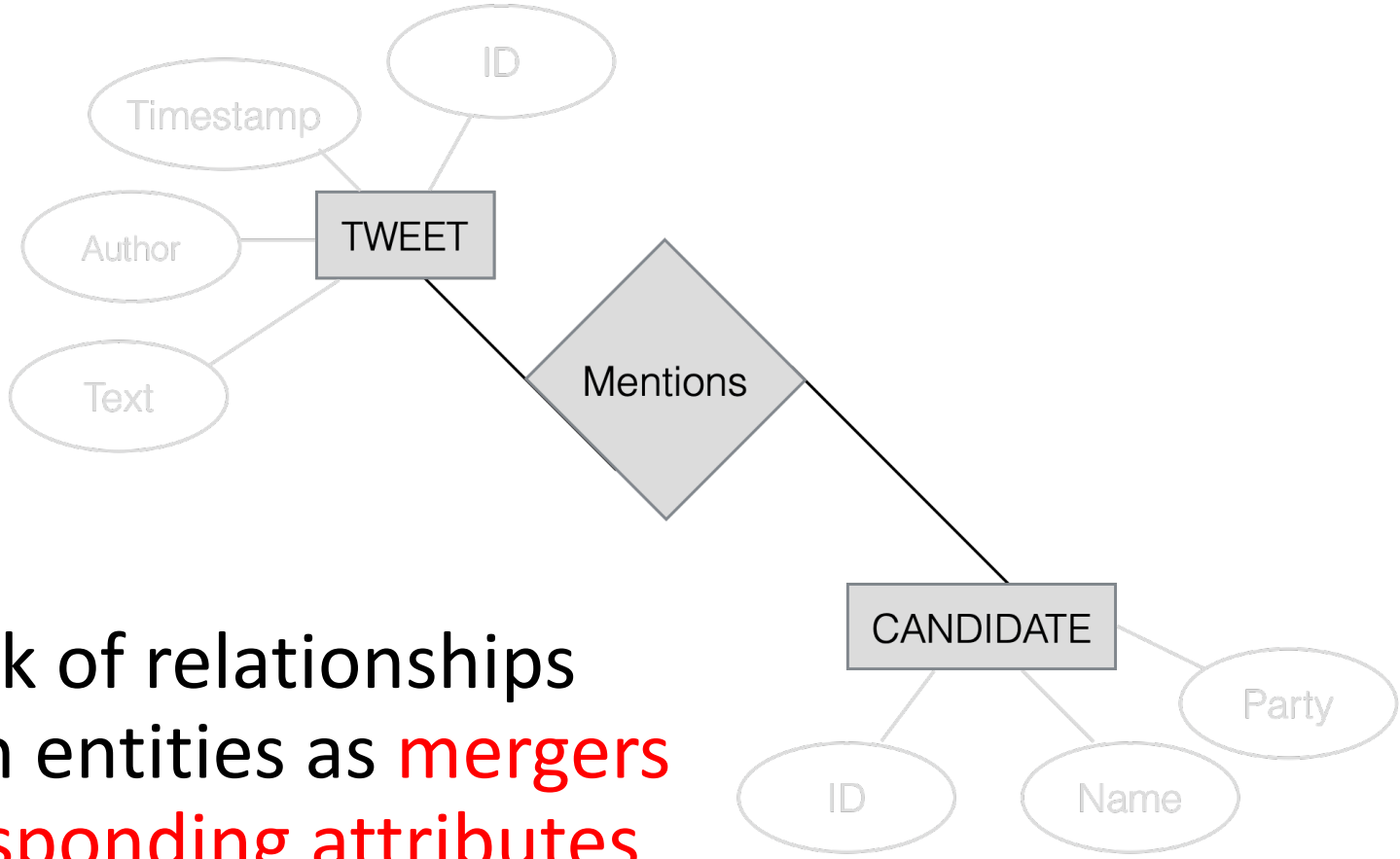
Relationships



Relationships

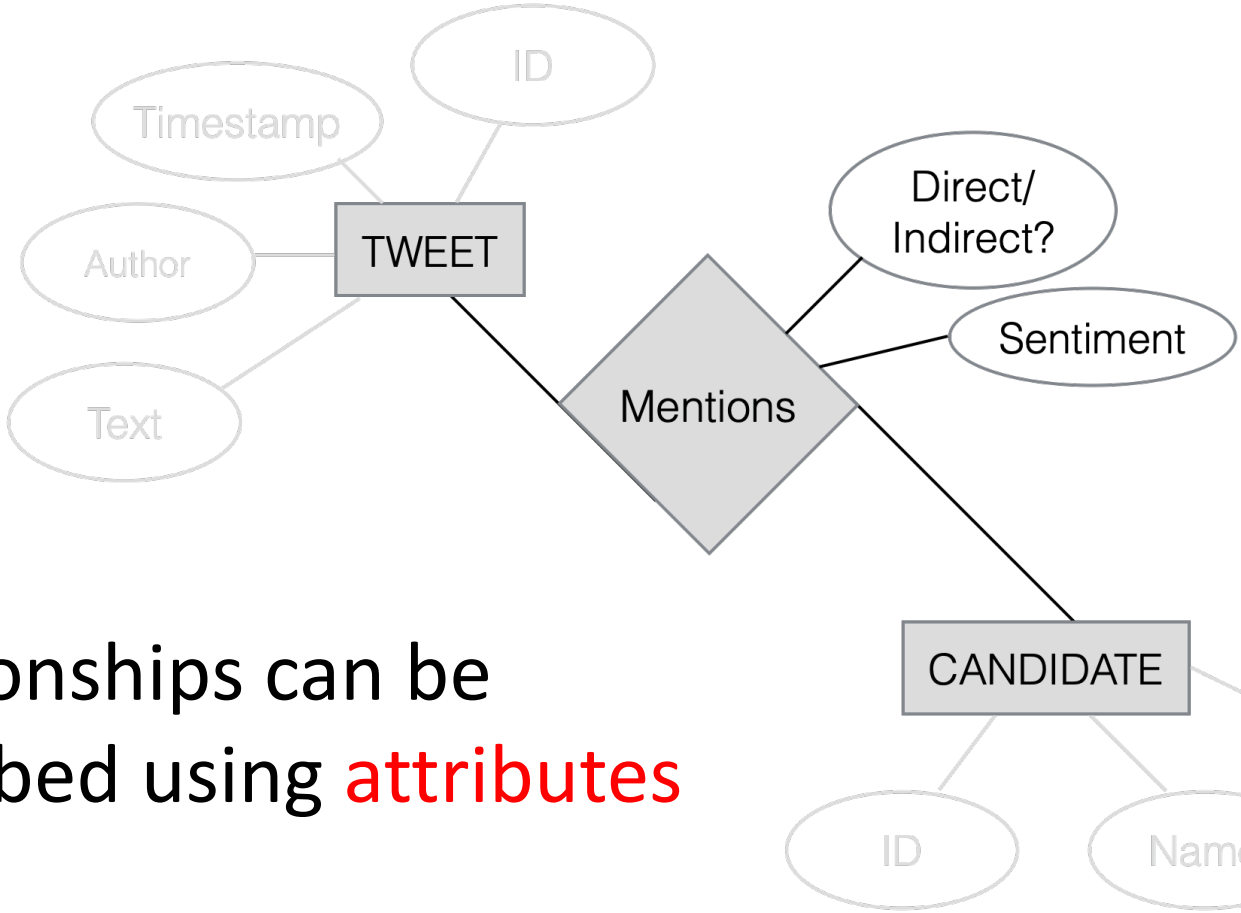


Relationships



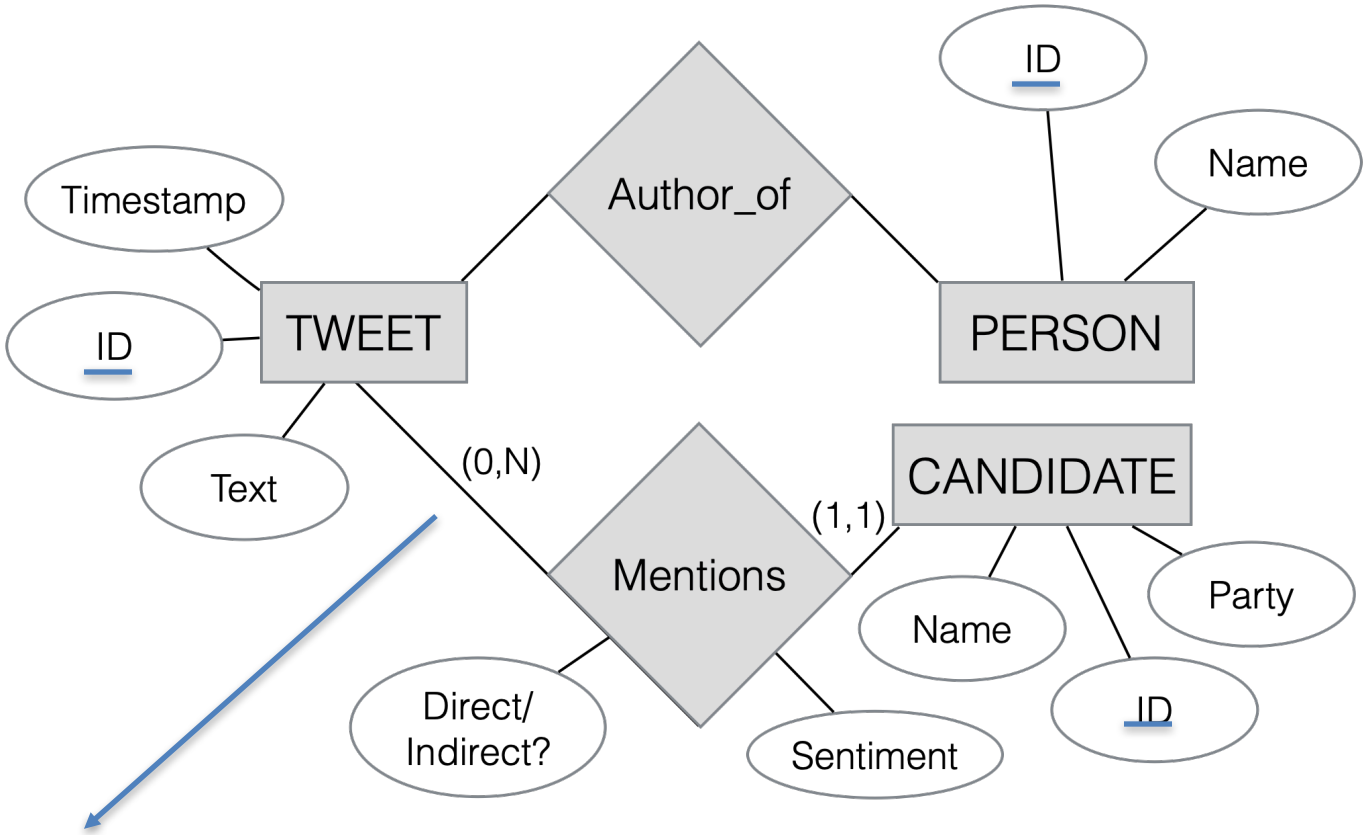
- Can think of relationships between entities as mergers of corresponding attributes

Attributes of relationships



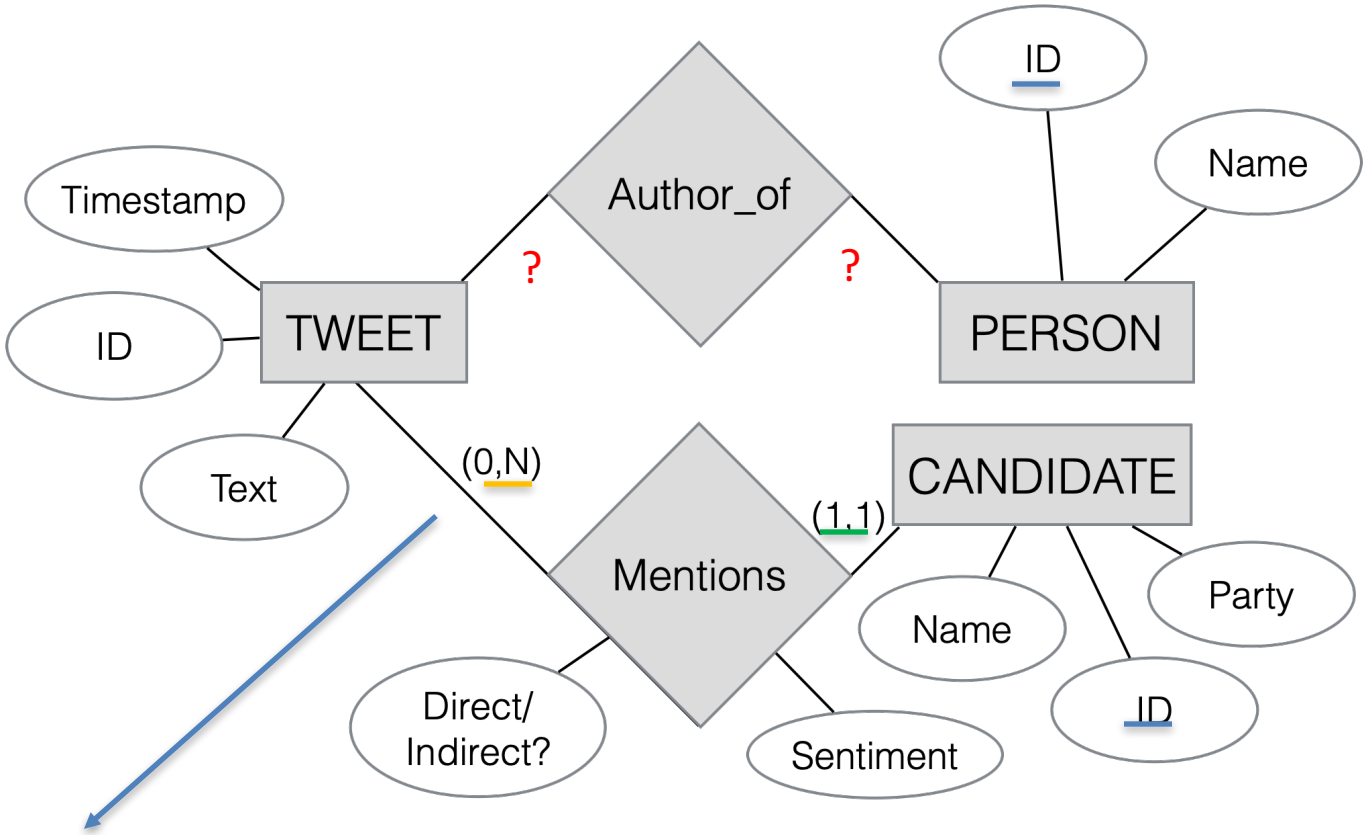
- Relationships can be described using attributes

Attributes of relationships



(min, max) number of entities involved in a single relationship

Attributes of relationships



- each mention corresponds to exactly one candidate
- each tweet can mention no or many candidates

Design Decision

- The ER model provides formalism and tools for conceptual database representation and design
- It is not a strict algorithmic process
- Ample room for design decisions
- Multiple possible choices may be equally as correct!
- Design your model, evaluate and repeat

Relational model

T1

A	B	C	D	E

T2

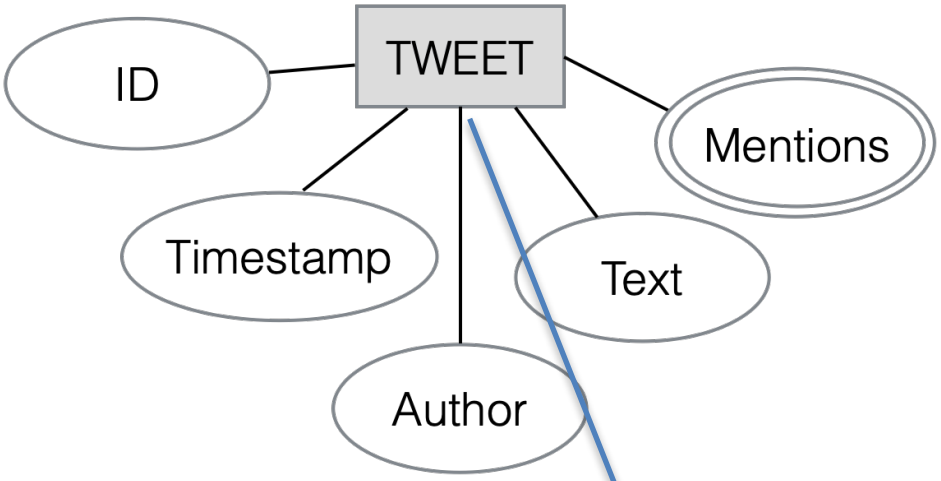
A	B	C	D	E

T3

F	P	G	L	E

- Think of DBMS as a set of tables
- Each table is called a relation

Relational model

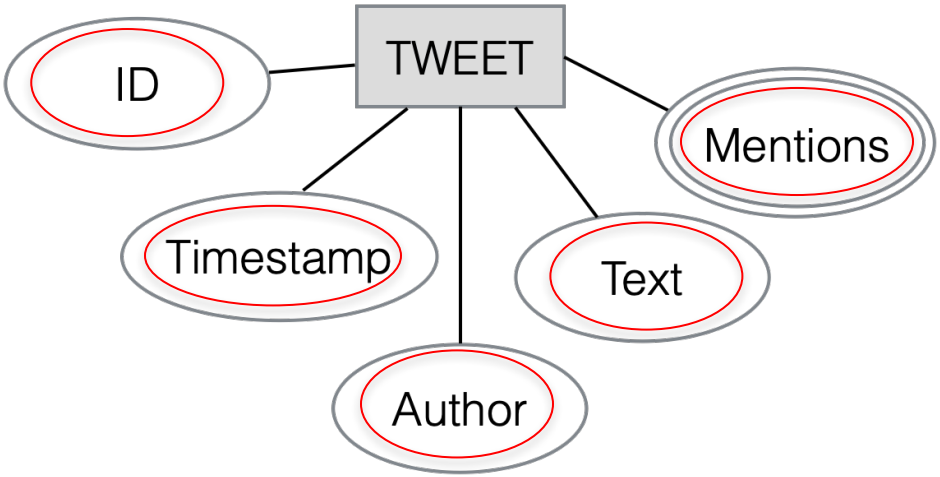


Name of the relation

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



Attributes of the 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

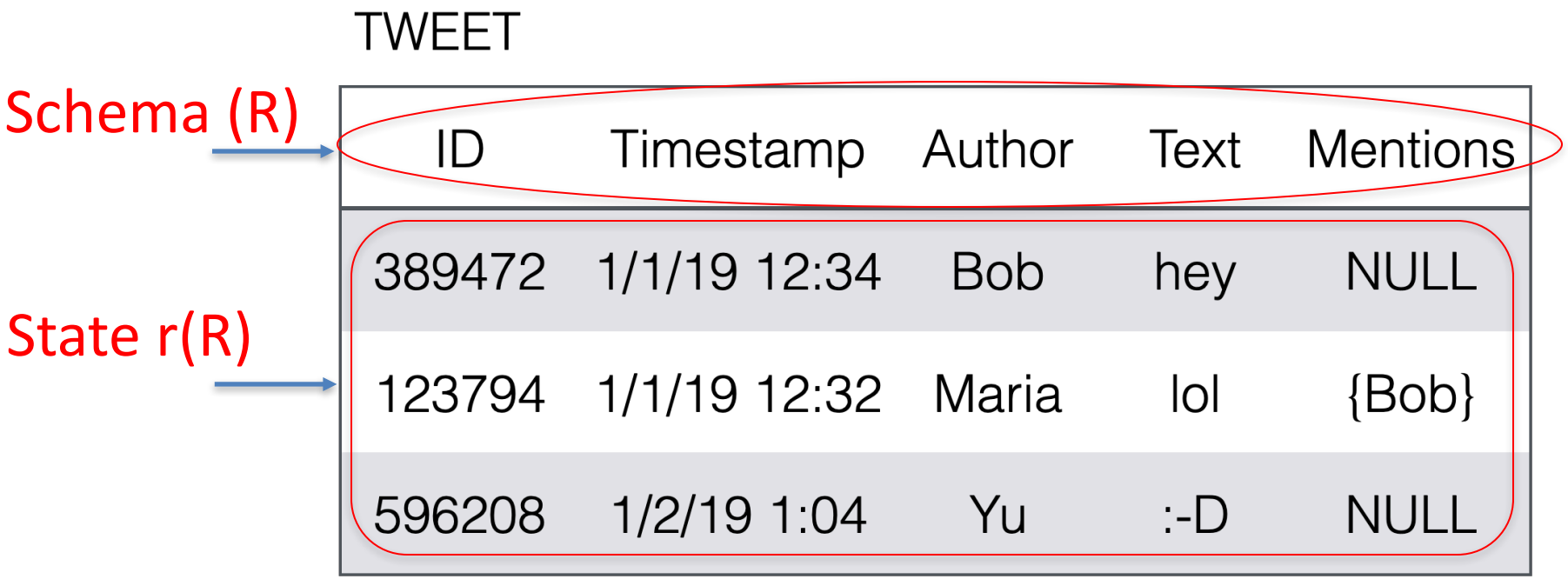
Relational model

TWEET

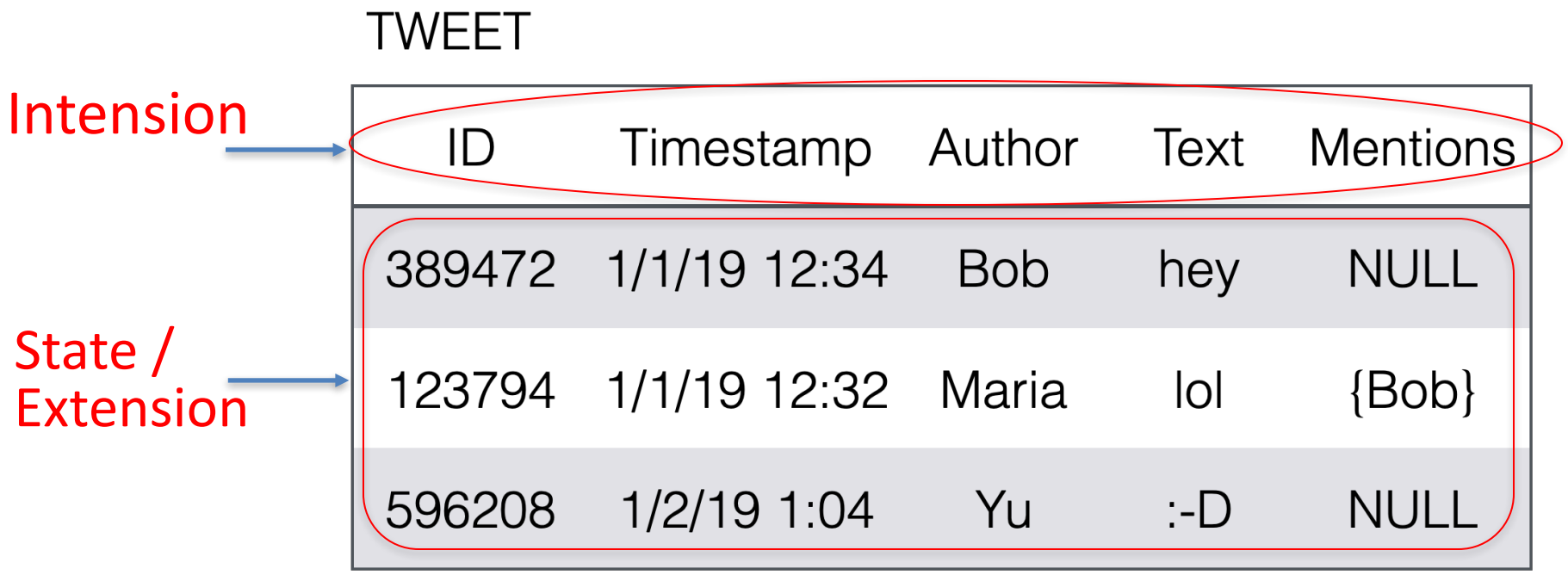
ID	Timestamp	Author	Text	Mentions
389472	1/1/19 12:34	Bob	hey	NULL
123794	1/1/19 12:32	Maria	lol	{Bob}
596208	1/2/19 1:04	Yu	:-D	NULL

- Each attribute has is own domain
  - I.e., a set of possible values it can assume
  - E.g, Dates in the GG/MM/YYYY format, text, integer numbers with 6 digits, dollar amount...
- Domain is part of the specification of an attribute

Relational Schema and State



Relational Schema and State

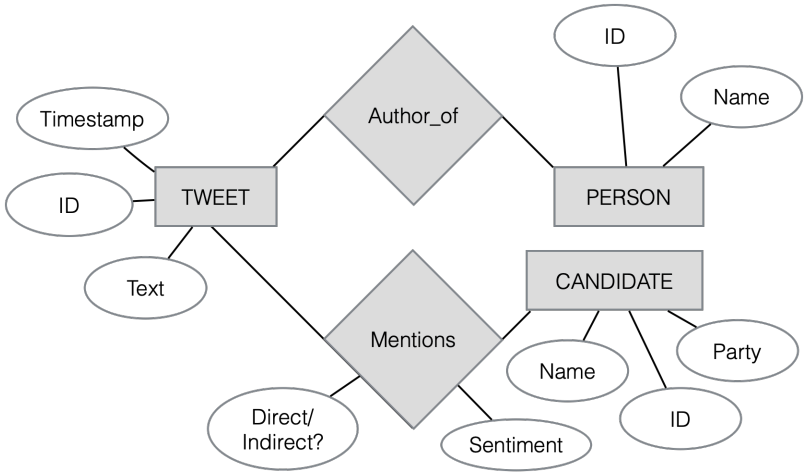


SQL

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

From model to implementation

So how do we go from here...



... to here?

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

Creating tables

TWEET: <ID, Time, Text>

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

- For each relation (table) we specify name and attributes each with their own type
- We create the scheme or intension of the relation

Data Definition: Data Types

- Numeric: INT, FLOAT, REAL, DOUBLE
- Character Strings: CHAR(n), VARCHAR(n), CLOB(size)
  - CHAR is fixed with, VARCHAR is not
  - 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
- Opportune choice of data type leads to improved performance and better memory utilization

https://www.w3schools.com/sql/sql_datatypes.asp

Deciding types

TWEET: <ID, Time, Text>

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

- Should text be CHAR(n), VARCHAR(n),CLOB(SIZE)?
  - CHAR(n) fixed allocation: can be faster as can use static memory address and does not require size check
  - VARCHAR(n) requires 2-more bits and can be slightly slower (dynamic address, size check), but, on average allows to save on memory
  - When uncertain on the size of a field better to use adjustable type – I.e., VARCHAR(140)!

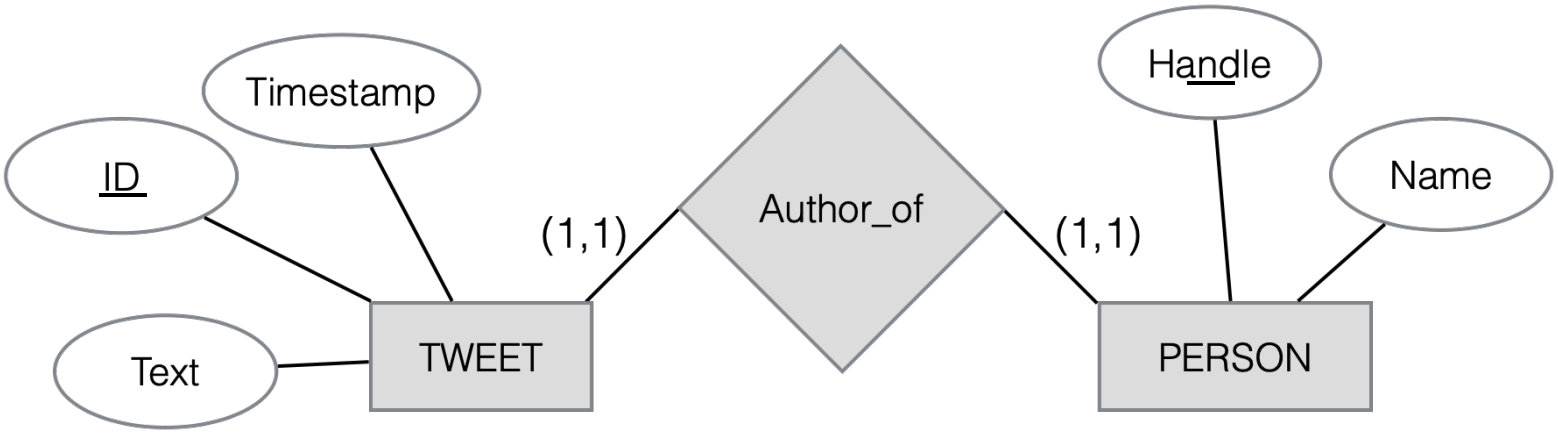
Deciding types

PERSON: <Handle, Name, ProfilePic, ProfilePage>

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

- What types should be used for ProfilePic and ProfilePage?

Relations form relationships

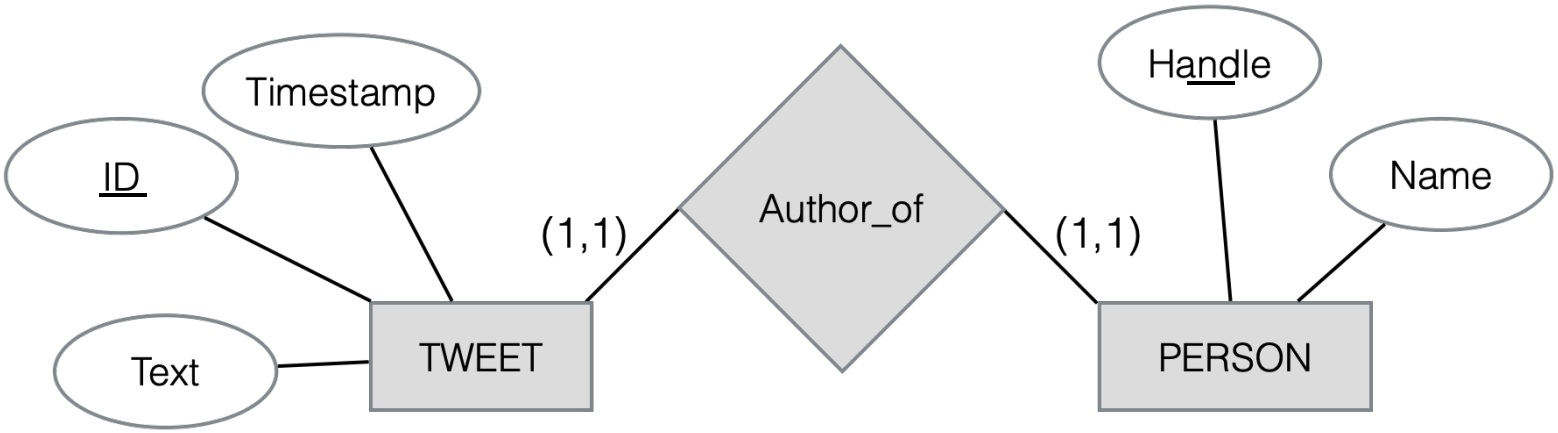


- So far we constructed tables from entities, can we do the same for relationships?

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

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

Relations form relationships




```
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)

Insert record in a table

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

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

- Once a relation scheme has been initialized, we can populate it adding tuples

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

Incomplete insertions

- Is possible to insert tuples while specifying only the values of some attributes

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

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

- The values of the missing attributes are initialized as "NULL"

Default attributes values

- Is possible to specify a default value for each attribute, which is then used instead of “NULL”

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

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

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

Default attributes values

- We can define some attributes so that all records cannot have a corresponding NULL value

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

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

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

Not admissible as ID would be identified as null

Primary Keys

TWEET: <ID, Time, Text>

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

- Some attributes can be marked as PRIMARY
- Forces records to have the corresponding attribute have NON NULL and unique value
- Primary keys are designed as identifiers

Primary Keys

```
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)
);
```

- It is also possible to first create the attributes and then designing one as PRIMARY
- The two definitions are equivalent

Primary Keys

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

- Tuples of attributes can be designed as PRIMARY KEY
- Values for all the Key attributes should not be NULL
- The combination of their values must be unique

Quiz 1

(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”);
```

Quiz 2

(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 (should be INT)

Quiz 3

(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");
```

Primary Key value cannot be NULL

Quiz 4

(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");
```

Unconventional but technically fine

Quiz 5

(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)
);
```

- Relations constructed from relationships using the key primary attributes of the former as connections

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 allow connections between tables by allowing referencing of their attributes
- They formalize pointers between values of different relations
- They do not imply a copy of said values

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
- Generally, stick to the rule of making Foreign Keys reference a Priority Key
  - If you can't do this, try reorganizing your DB to make it possible (if you are in a position to do this)

Updating tuples

```
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

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

Deleting tuples

COSTUMER

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany
2	Ana Trujillo Emparedados y helados	Juan	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Juan	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

```
DELETE FROM COSTUMER WHERE Country = Mexico
```

- A single update operation can be used to delete multiple records!
  - All those that satisfy the criteria specified by WHERE

Deletions without WHERE specification

COSTUMER

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany
2	Ana Trujillo Emparedados y helados	Juan	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Juan	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

DELETE FROM COSTUMER

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country

- CAREFUL: omitting WHERE specification will lead to ALL records in the relation to be deleted!

Updating tuples

CONTACTS

ID	Name	Phone number
389472	Anna	1578987326
123879	Bob	6745678123
780947	Charles	9874537677
479908	Devin	4014444489

```
UPDATE CONTACTS SET Phone Number = "451678453" WHERE Name = "Bob"
```

Target Relation

Attributes to update with respective values

Selection criteria

CONTACTS

ID	Name	Phone number
389472	Anna	1578987326
123879	Bob	451678453
780947	Charles	9874537677
479908	Devin	4014444489

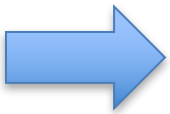
Updating Multiple tuples

```
UPDATE CONTACTS
SET
Phone Number = "451678453"
ID = "0000"
WHERE  Name = "Bob"
```

- Multiple attributes can be updated with a single instruction
- All records selected by the WHERE criteria will be affected

CONTACTS

ID	Name	Phone number
389472	Anna	1578987326
123879	Bob	6745678123
780947	Charles	9874537677
479908	Devin	4014444489
800084	Bob	7364982909



CONTACTS

ID	Name	Phone number
389472	Anna	1578987326
0000	Bob	451678453
780947	Charles	9874537677
479908	Devin	4014444489
0000	Bob	451678453

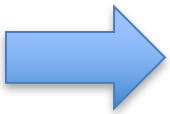
Updates without WHERE specification

```
UPDATE CONTACTS
SET
Phone Number = “451678453”
ID = “0000”
```

- CAREFUL: omitting WHERE specification will lead to ALL records in the relation being modified!

CONTACTS

ID	Name	Phone number
389472	Anna	1578987326
123879	Bob	6745678123
780947	Charles	9874537677
479908	Devin	4014444489
800084	Bob	7364982909



CONTACTS

ID	Name	Phone number
389472	Anna	1578987326
0000	Bob	451678453
780947	Charles	9874537677
479908	Devin	4014444489
0000	Bob	451678453

Data 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

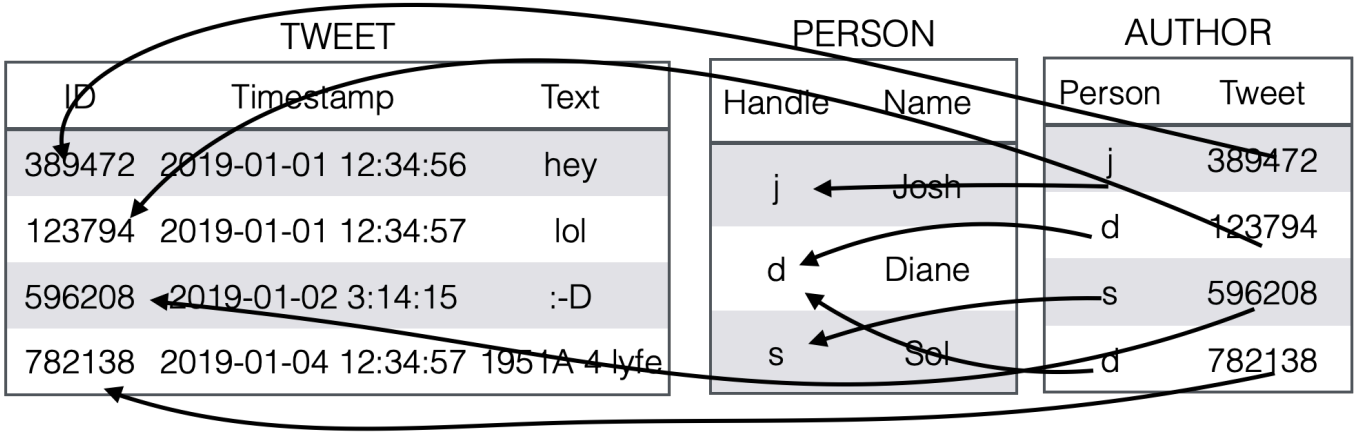
Data integrity

```
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),
);
```

Foreign key establish references and pointers



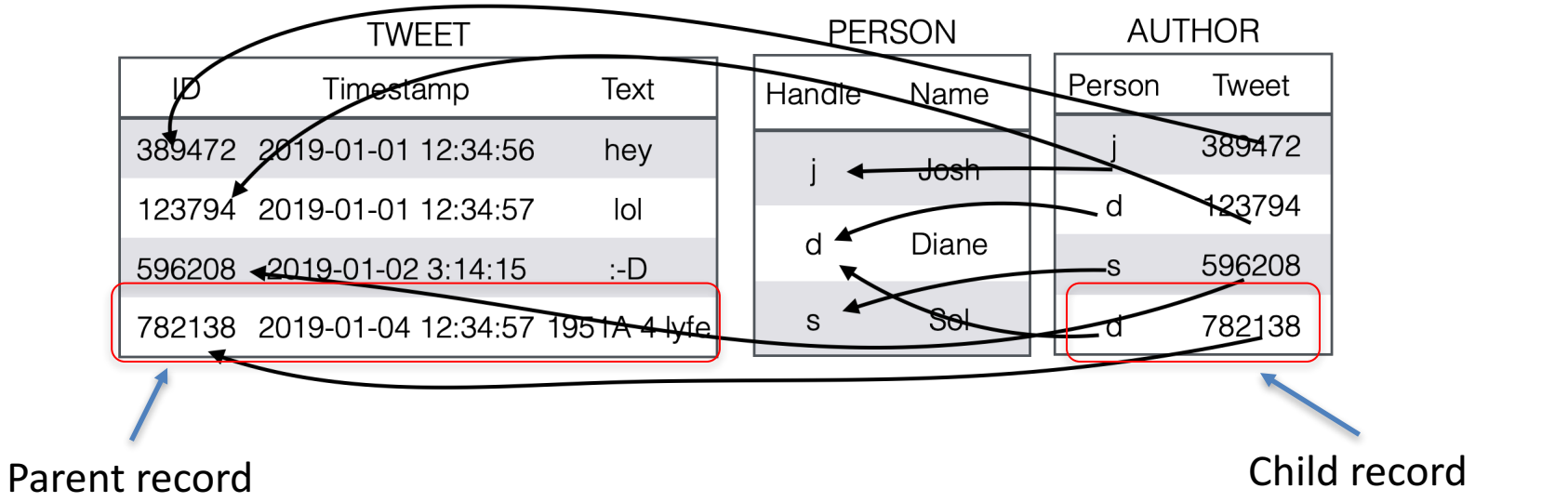
Data integrity

```
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),
);
```

Foreign key establish references and pointers



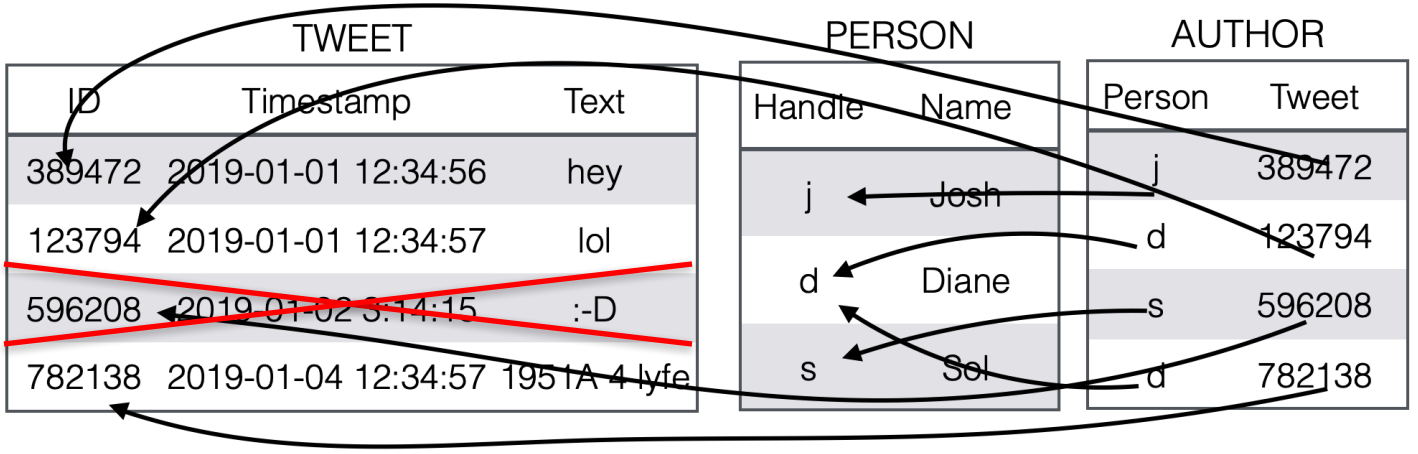
Data integrity

```
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),
);
```

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



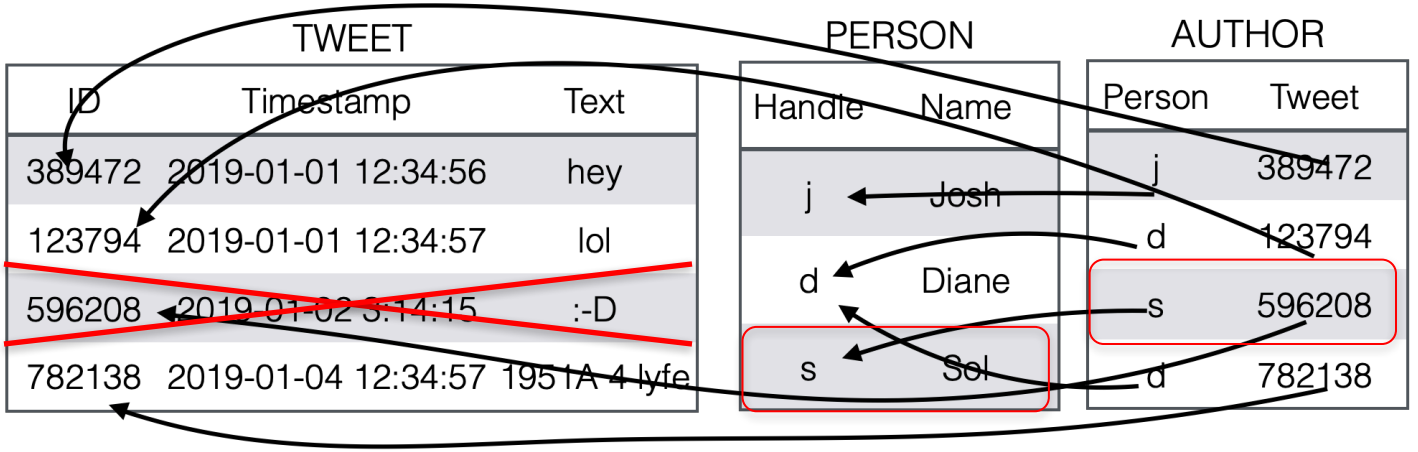
Data integrity

```
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),
);
```

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



What happens to the records referencing the deleted one?

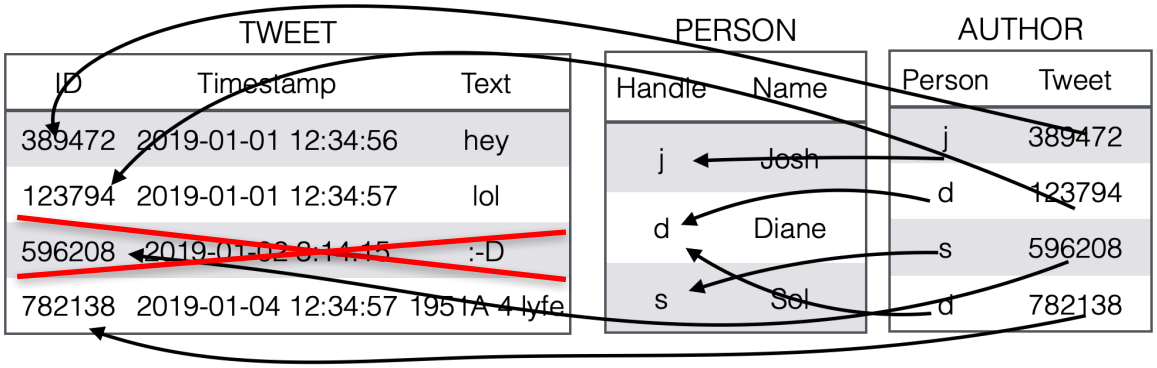
Data integrity: Default behavior

```
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"
```



By default, the deletion of a record who is a parent of another record in a child relation table will not be allowed

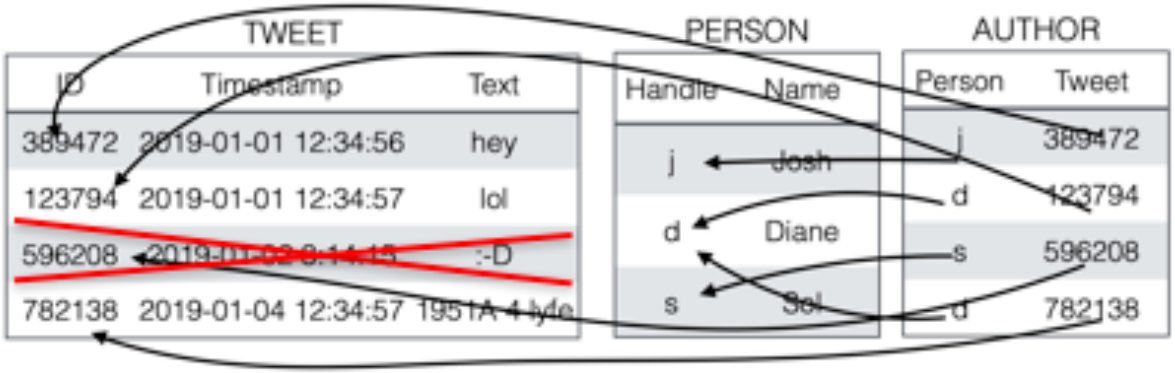
Data integrity: ON DELETE RESTRICT

```
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"
```



- By default, the deletion of a record who is a parent of another record will not be allowed
- Equivalent to specifying the ON DELETE RESTRICT option

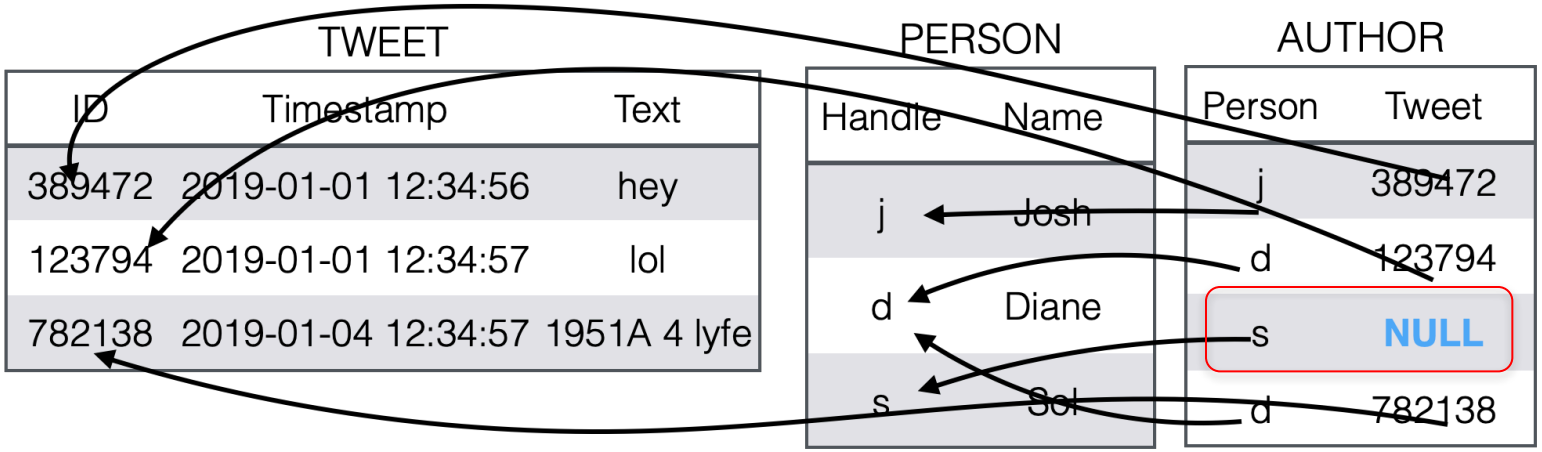
Data integrity: ON DELETE SET NULL

```
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"
```



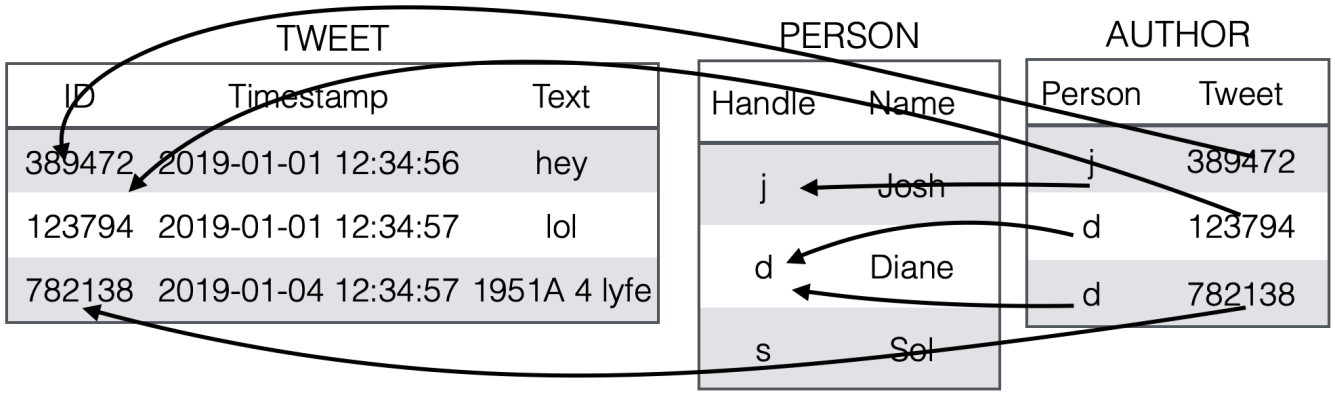
Data integrity: ON DELETE CASCADE

```
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"
```



Propagates deletion to all records which would otherwise be set to "NULL"

Data integrity: ON UPDATE CASCADE

```
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) ON UPDATE CASCADE,
);
```

```
UPDATE PERSON SET Handle = "x" WHERE Name = "Josh"
```

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

Propagates updates of parent records to all child records

Data integrity: ON UPDATE CASCADE

```
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) ON UPDATE CASCADE,
);
```

```
UPDATE PERSON SET Handle = "x" WHERE Name = "Josh"
```

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
x	~~Josh~~
d	Diane
s	~~Sol~~

AUTHOR

Person	Tweet
x	389472
d	123794
s	596208
d	782138

Propagates updates of parent records to all child records

Data integrity

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

Is it possible to specify actions both on deletions and update

Quiz 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"
```

Quiz time

	PERSON		TWEET			RETWEET	
	Handle	Name	ID	Text	Author	Person	Tweet
(a)	j	Josh	3	:-D	d		
	d	Diane					

	PERSON		TWEET			RETWEET	
	Handle	Name	ID	Text	Author	Person	Tweet
(b)	j	Josh	3	:-D	d	NULL	NULL
	d	Diane				NULL	NULL
						NULL	NULL

	PERSON		TWEET			RETWEET	
	Handle	Name	ID	Text	Author	Person	Tweet
(c)	j	Josh	3	:-D	d	j	NULL
	d	Diane				j	NULL
						d	NULL