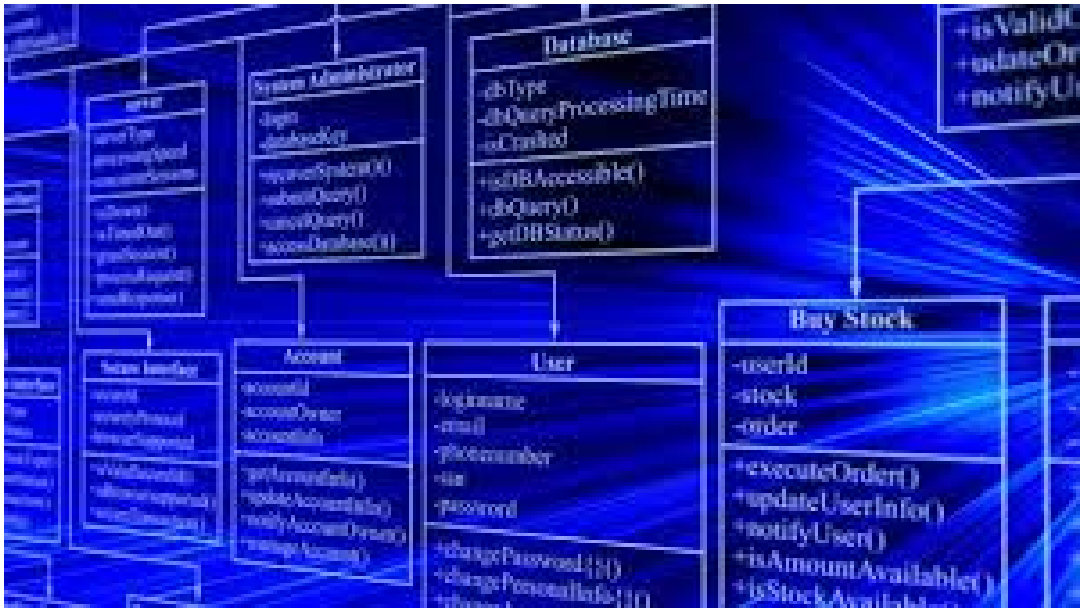
Introduction to Database Systems: CS312 A Larger Database system

Oliver Bonham-Carter

23 January 2019

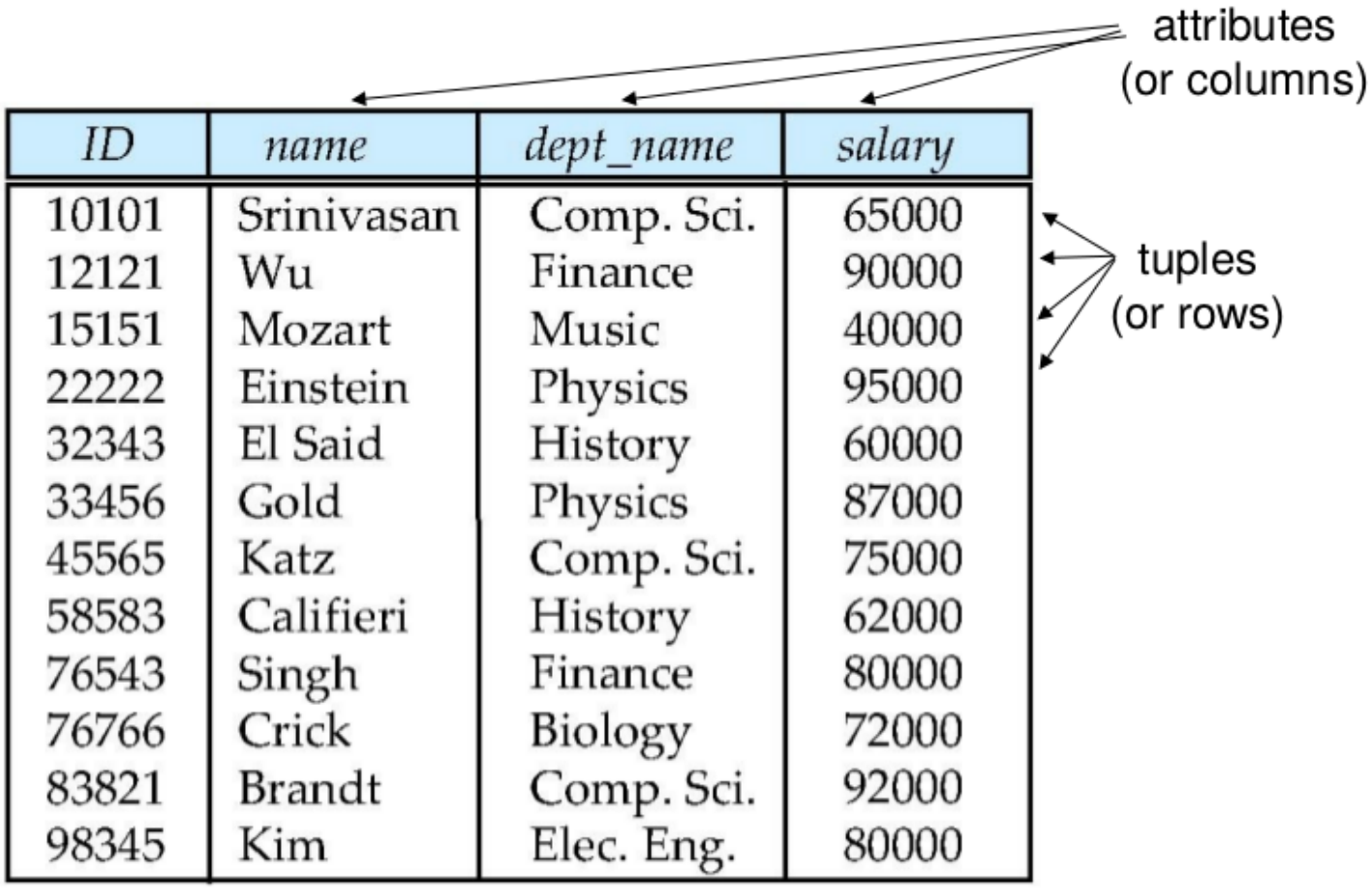
Overview of The Entity-Relationship Model Design

Consider these!



- What is the *data* to store in the database?
- What are the *relationships* between the *entities* of information?
- What is the conceptual *design* of a system to link all this information together: the entity-Relationship (ER) model

ER Model Basics
Tables



ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

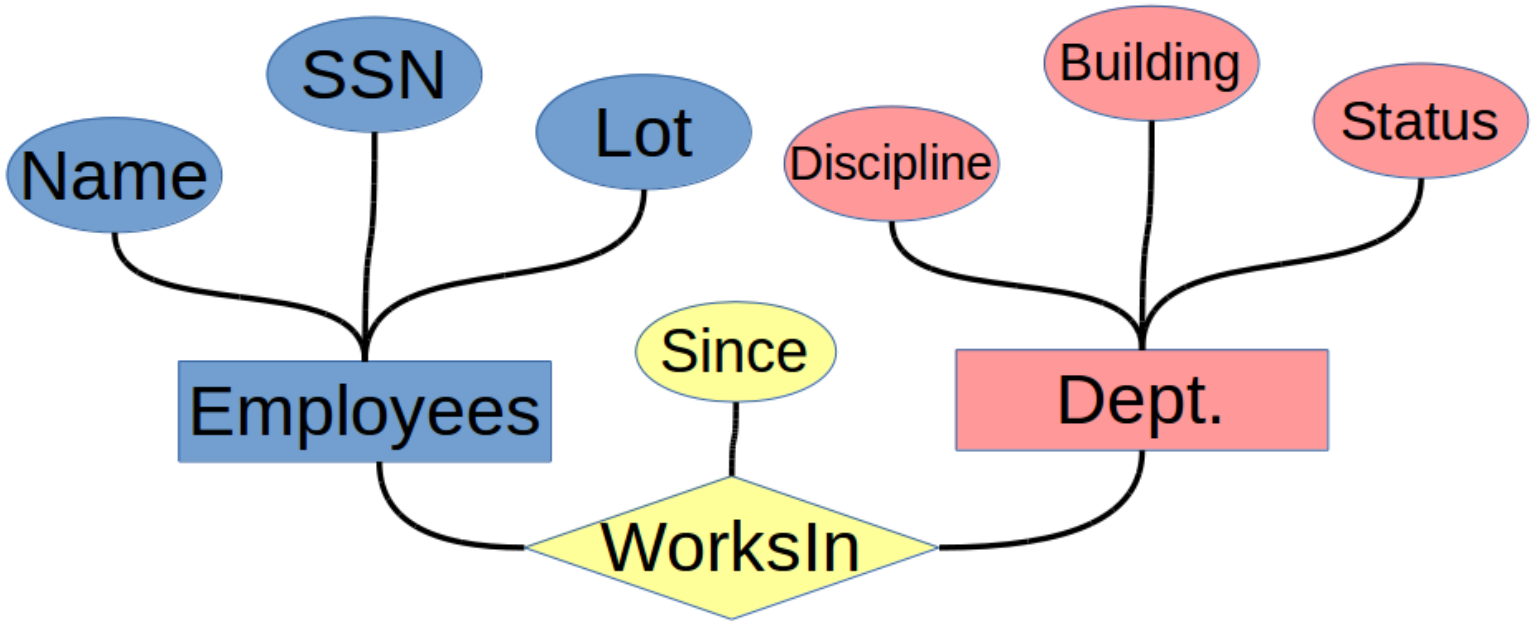
ER Model Basics

Tables

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

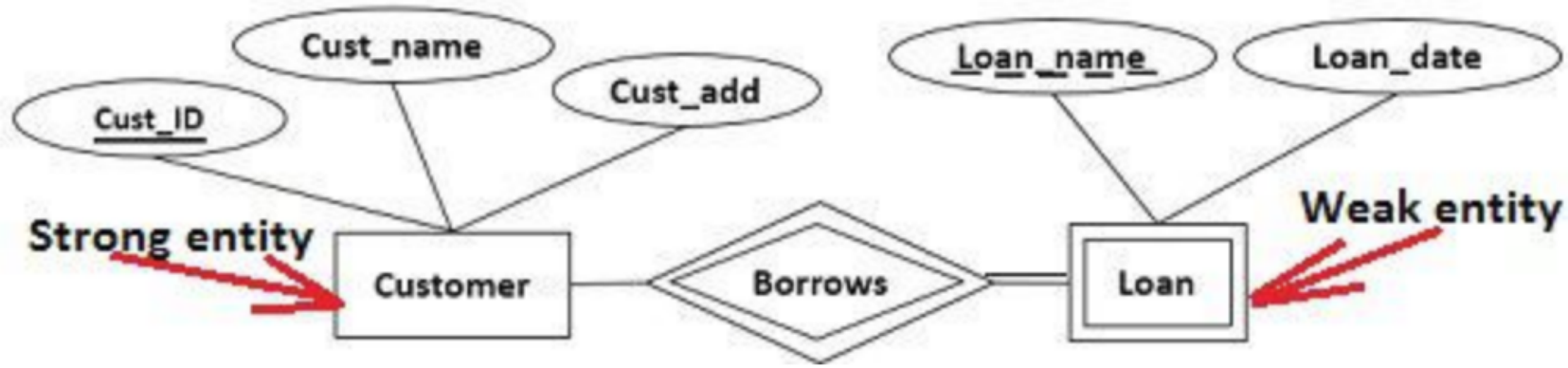
- Four **attributes**: *ID*, *name*, *dept_name* and *salary*
- The **relation instance** refers to a finite set of tuples in the relational database system and represents a relation instance (i.e., a group of observations). Relation instances do not have duplicate tuples.
- The **instance** of the instructor table shown above has 12 tuples, corresponding to 12 instructors (12 *observations*)

ER Model Basics
Schemas and Relationships



- A schema resembles a subroutine and describes the table and the data that it contains.
- Relationship: An association among two of more entities
- Relationship Set: A collection of similar relationships for entities
- Relationship sets can also have *descriptive attributes* (i.e., the "since" attribute of *WorksIn*)

Entity sets



Entity Set

An **entity set** is a set of entities of the same type (e.g., all persons having an account at a bank). Entity sets need not be disjoint. For example, the entity set employee (all employees of a bank) and the entity set customer (all customers of the bank) may have members in common.

Entity sets
Weak set

Loan_name	Loan_date	Amount
Home	20/11/2015	20000
Education	5/10/2015	10000
Home	20/11/2015	20000

Loan Entity Set

Weak entity set: The rows are not unique

In a relational database, a weak entity is an entity that cannot be uniquely identified by its attributes alone due to a lack of a primary key. To distinguish one row from another, a foreign key in conjunction with its attributes, must be employed to create a primary key.

Entity sets
Strong set

Cust_ID	Cust_name	Cust_add
101	John	Xyzzy
103	Ruby	Pqr
109	John	Uvfw

Customer Entity Set

Strong entity set: The rows are unique

Each row in a table is unique thanks to an (acting) primary key (attribute: `Cust_ID`) which guarantees that one entire row cannot be *overwritten* by another.

Entity sets

ID	Tea	Sandwich
JJ	**1**	**Ruban**
OBC	1	PBJ
AM	1	Chicken
GK	1	Chicken
JJ	**1**	**Ruban**
DW	0	PBJ
MC	1	Ruban
JJ	**1**	**Ruban**
SR	1	Ruban
JJ	**1**	**Ruban**
KT	1	Ruban

- **Entity set**: a collection of entities of the same kind
  - (i.e., the preferred sandwiches.)
- One observation (row) **must be different** from another in the table.

ER Model Basics
Keys

- **Primary keys**: Unique identifiers for the row of information sharing a relation (n-tuple).
- **Super keys**: A superkey is a set of attributes within a table whose values can be used to uniquely identify a n-tuple.
- **Candidate keys**: is a minimal set of attributes necessary to identify a n-tuple.
- **SuperKeys**: a set of attributes within a table whose values can be used to uniquely identify a tuple (each row is unique from the other rows)

Keys

You will note the importance of keys once you start storing your data in your own databases!

We've got it!

- Let's build a relational database in SQLite3!!

Let's Try it!

The data

```
1|Ezra|Weston Loomis|Pound|30/10/1885|1/11/1972|USA
2|Arthur|Conan|Doyle|05/22/1859|07/7/1930|UK
3|Ernest|Miller|Hemingway|07/21/1899|07/02/1961|USA
4|John|Edward|Williams|08/22/1922|03/3/1994|USA
```

Attributes

- ID
- first name
- middle name
- last name
- birth date
- death date
- country of origin

Create the file!

The terminal command to open a new database

```
sqlite3 writers.sqlite3
```

```
obonhamcarter$ sqlite3 writers.sqlite3
SQLite version 3.19.3 2017-06-27 16:48:08
Enter ".help" for usage hints.
sqlite>
```

Create the Space

Create Table command

```
CREATE TABLE Writers (
    id INTEGER NOT NULL PRIMARY KEY,
    first_name VARCHAR(15) NOT NULL,
    middle_name VARCHAR(15),
    last_name  VARCHAR(15) NOT NULL,
    birth_date VARCHAR(10) NOT NULL,
    death_date VARCHAR(10),
    country_of_origin VARCHAR(20) NOT NULL
);
```

- Note: $attribute_1$ varchar(n) **NOT NULL**
  - declaration of the size of string (n) contained by entry called, $attribute_1$
  - NOT NULL ensures that this field is not left blank when populating

Add Data for writers Table

Insert Commands

```
INSERT INTO Writers VALUES(1, 'Ezra', 'Weston Loomis', 'Pound', '30/10/1885',
'1/11/1972', 'USA');
INSERT INTO Writers VALUES(2, 'Arthur', 'Conan', 'Doyle', '05/22/1859',
'07/7/1930', 'UK');
INSERT INTO Writers VALUES(3, 'Ernest', 'Miller', 'Hemingway', '07/21/1899',
'07/02/1961', 'USA');
INSERT INTO Writers VALUES(4, 'John', 'Edward', 'Williams', '08/22/1922',
'03/3/1994', 'USA');
```

Tables and Schema

- What is the schema (i.e., the arrangement of data) of your database?
  - Type in **".schema"** and see!
- What are the tables of your database?
  - Type in **".tables"** and see!

Consider this...

Please see the *sandbox* file for code.

- Can you populate your base by adding more data?
- Can you also check that the data was correctly stored in the table?
- Can you run queries to access particular attributes?

Consider this...

What are these queries and how do they work?

Queries to play with using conditional clauses

```
select * from Writers where country_of_origin == "UK";

select * from Writers where country_of_origin == "USA";

select * from Writers where first_name == "Arthur";

select * from Writers where middle_name == "Miller";
```

What else can you query using this code?

Save your DB and exit: **.exit**