# Introduction to Database Systems: CS312 A Larger Database system

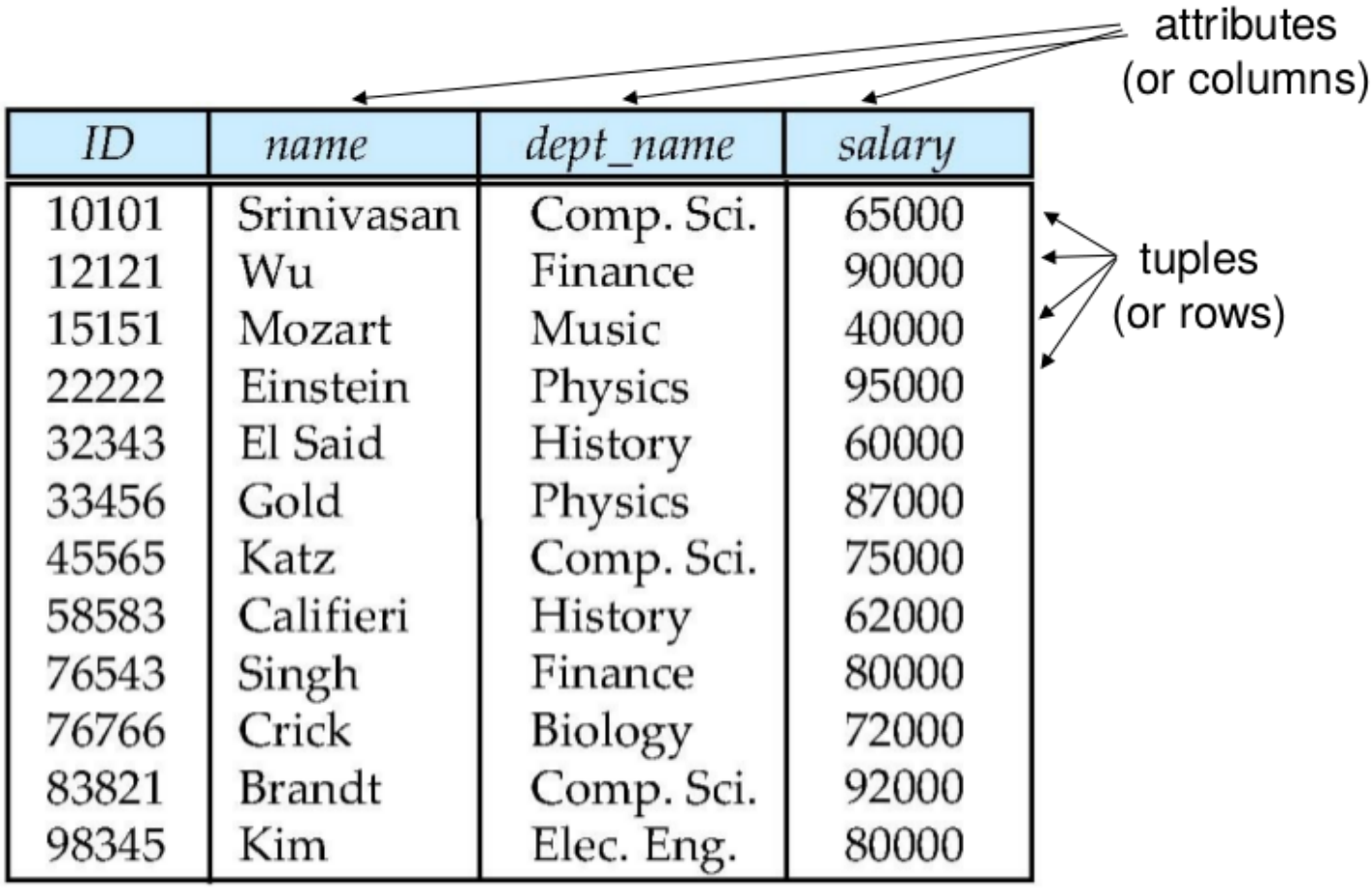
Oliver Bonham-Carter

7 Sept 2020

# 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



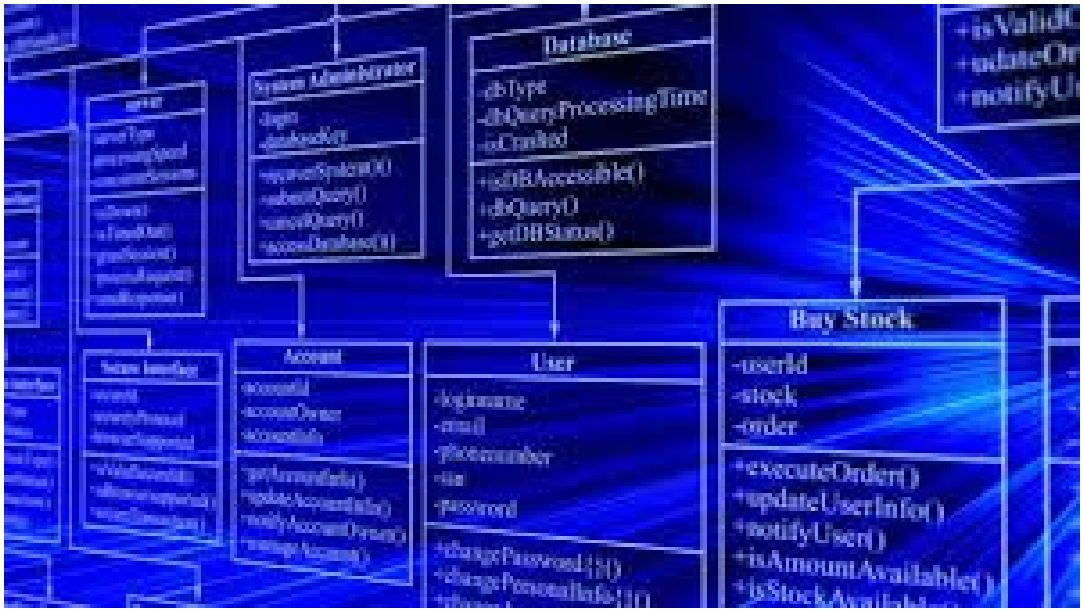
| *ID* | *name* | *dept_name* | *salary* |
|---|---|---|---|
| 10101 | Srinivasan | Comp. Sci. | 65000 |
| 12121 | Wu | Finance | 90000 |
| 15151 | Mozart | Music | 40000 |
| 22222 | Einstein | Physics | 95000 |
| 32343 | El Said | History | 60000 |
| 33456 | Gold | Physics | 87000 |
| 45565 | Katz | 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*)

# 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

# E-R Models ... Entities? Relationships?

## Section 1.6.3, The Entity-Relationship Model



### Entities

- The entity-relationship (E-R) data model uses a collection of basic objects, called **entities**, and relationships exist to connect these objects.
- Entities are defined by attributes (i.e, *column headers* in tables)
- *ID*, *name*, and *salary* are points of information to describe an *instructor* entity

### *Relationships*

- A *relationship* is an association among several *entities*
- For example, a member relationship associates an instructor with her department.
- The set of all *entities* of the same type and the set of all *relationships* of the same type are termed an *entity set* and *relationship set*, respectively.

# E-R Models ... Entities? Relationships?

## Section 1.6.3, The Entity-Relationship Model



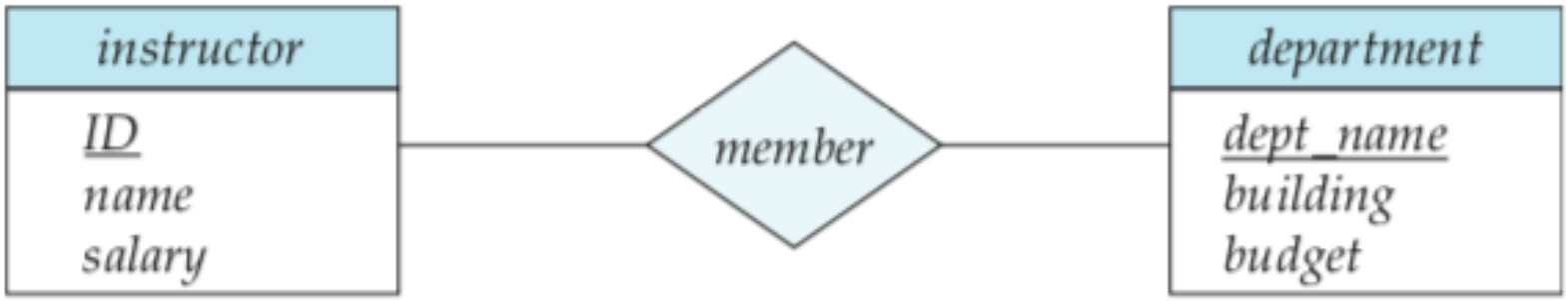
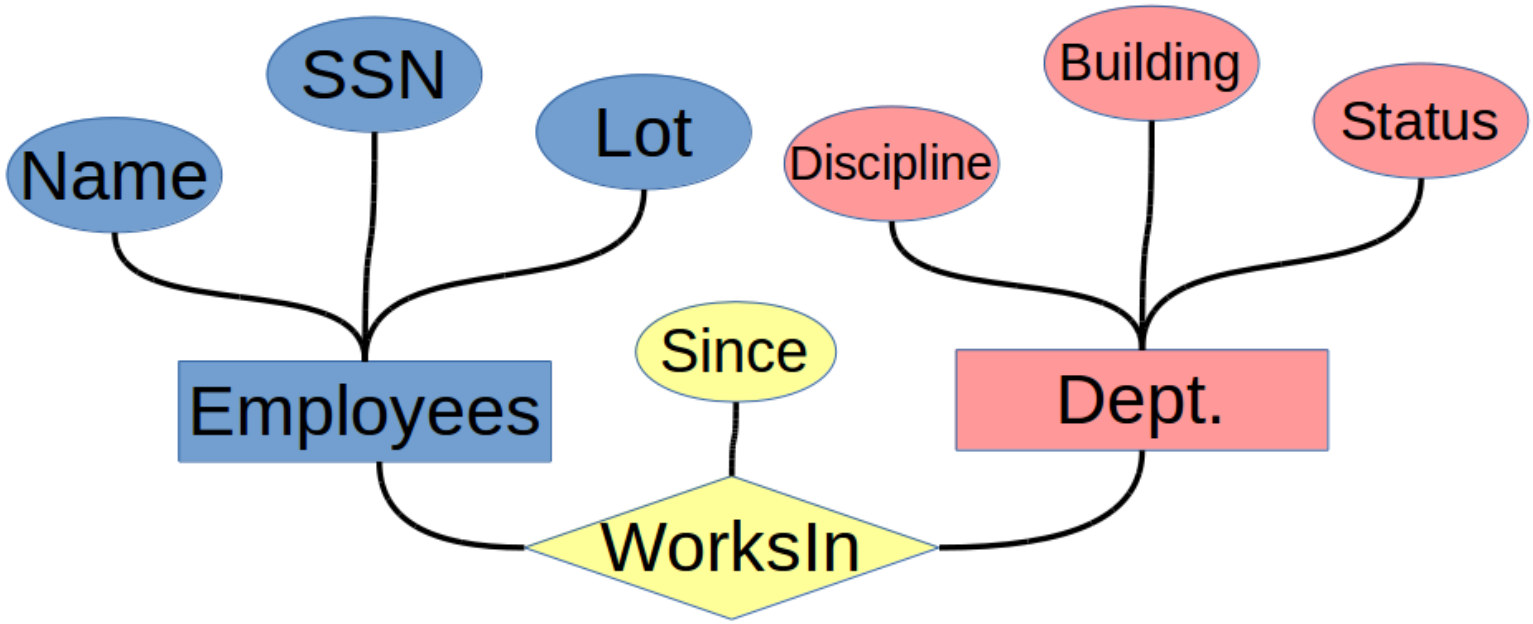



**Figure 1.3** A sample E-R diagram.

- *Entity sets* are represented by a rectangular box with the *entity set* name in the header and the attributes listed below it.
- *Relationship sets* are represented by a diamond connecting a pair of related *entity sets*. The name of the *relationship* is placed inside the diamond.




- Relationship Set: A collection of similar relationships for entities
- Relationship sets can also have *descriptive attributes* (i.e., the "since" attribute of *WorksIn*)

# Displaying schemas

## We create some tables to find schema



- A **schema** resembles a subroutine and describes the table and the data that it contains.

```
CREATE TABLE employees
 ( name VARCHAR(10),
   ssn VARCHAR(10),
   lot VARCHAR(10) );

 CREATE TABLE dept
  ( discipline VARCHAR(10),
    building VARCHAR(10),
    status VARCHAR(10) );

CREATE TABLE workin
  ( since VARCHAR(10) );
```

```
.schema
```

# Null Values

## What needs to be included in a table's values?



### Definition: Null Values

- SQLite NULL is the term used to represent a missing value. A NULL value in a table is a value in a field that appears to be blank.
- A field with a NULL value is a field with no value. It is very important to understand that a NULL value is different than a zero value or a field that contains spaces (i.e, 0 or " ".)

### Null and Non Nulls

- When you CREATE a table, you can specify whether or not NULL or NOT NULL values can be accepted (i.e., inserted into table).
- Once a NOT NULL constraint is attached to a column, then an insertion or an update involving a NULL (i.e., a nothing value) will be rejected (i.e., a constraint violation).
- NOT NULL signifies that the column should always accept an explicit value of the given data type.

# Null Values



## Pseudocode Form

```
CREATE TABLE table_name (
    ...
    column_name type_name NOT NULL,
    ... );
```

## SQL Code

```
CREATE TABLE secretNames(
  name VARCHAR(10),
  codeName VARCHAR(10) NOT NULL,
  role VARCHAR(5)  );
```

# Null Values



## Example

```
INSERT INTO secretNames
   VALUES("SherlockHolmes",221,"Consulting Detective");

INSERT INTO secretNames
   VALUES("JamesWatson",201,"Wing Guy");
```

## Will this command work?

```
INSERT INTO secretNames
   VALUES("BillyBob",NULL,"Wing guy");
```

## Will this command work?

- What happened? Why?

# We've got it!




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

# Let's Make a Persistent Database!



## 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!

# Conditional Queries

## Adding conditional clauses to queries

### 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 birth_date == "08/22/1922";

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**

# 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?