

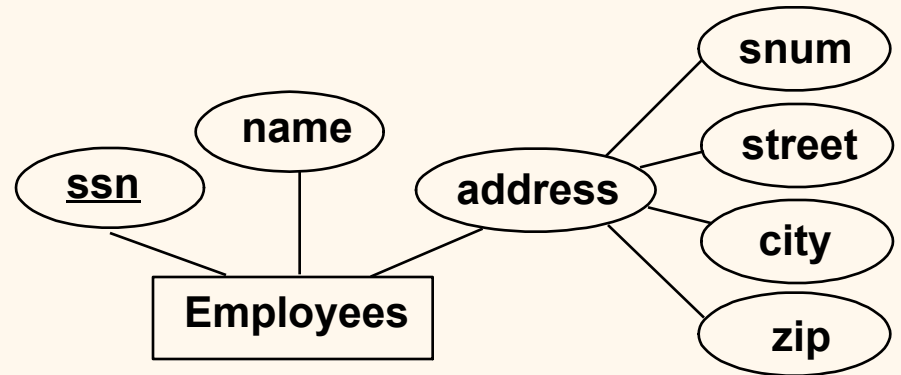
CS122A: Introduction to Data Management

Lecture 8 Introduction to SQL

Instructor: Chen Li

More notes from Quiz 3

Composite attributes



CREATE TABLE Employee

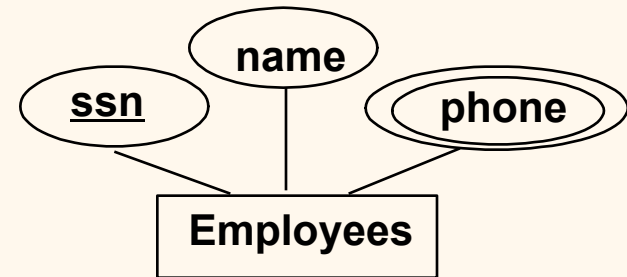
(ssn CHAR(20), name CHAR(20),

sname CHAR(20), street CHAR(20),

city CHAR(20), zip CHAR(10),

PRIMARY KEY (ssn))

Multi-valued attributes



```
CREATE TABLE Employee
```

```
(ssn CHAR(20), name CHAR(20), ...)
```

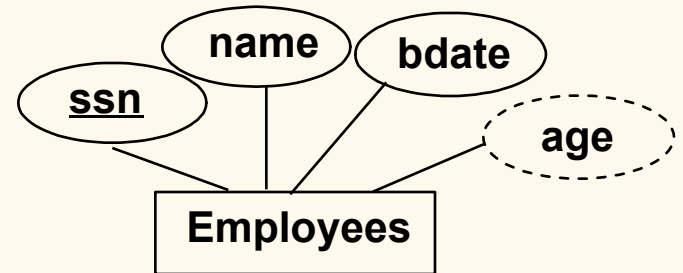
```
CREATE TABLE Phone
```

```
(pid INT, ssn CHAR(20), phone CHAR(15),
```

```
PRIMARY KEY (pid),
```

```
FOREIGN KEY (ssn) REFERENCES Employees)
```

Derived attributes



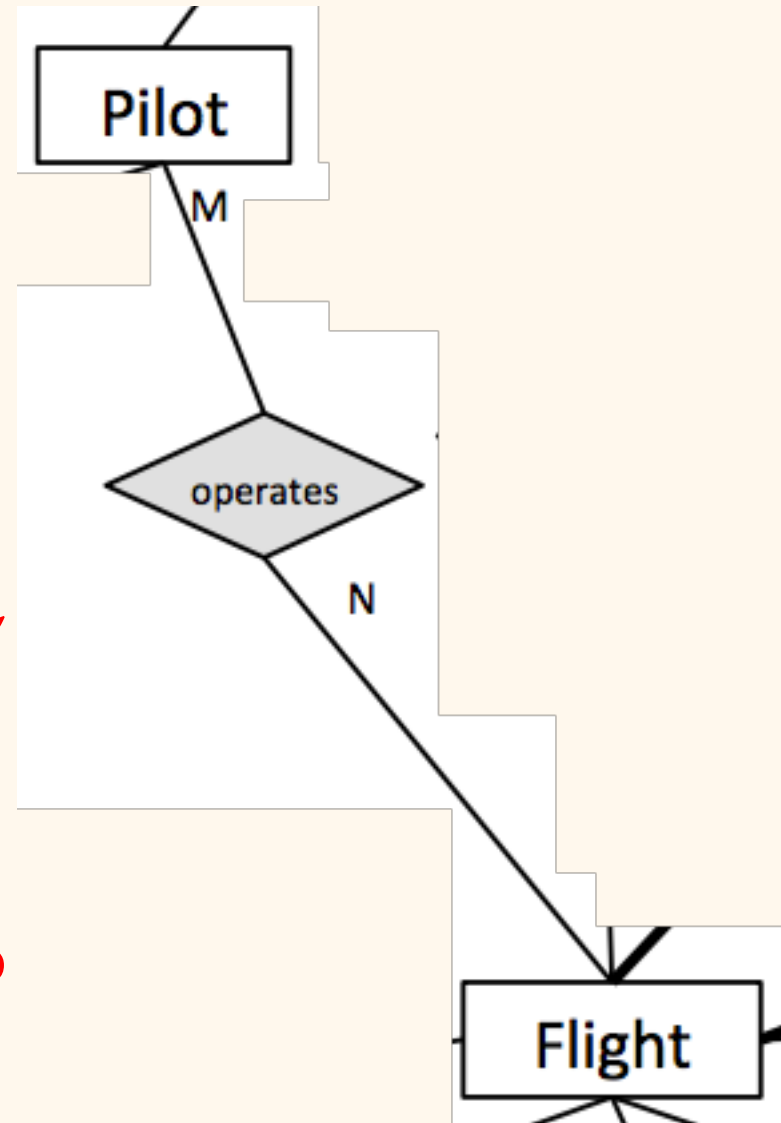
CREATE TABLE Employee

(ssn CHAR(20), name CHAR(20), bdate
DATETIME, **age** INT, PRIMARY KEY (ssn))

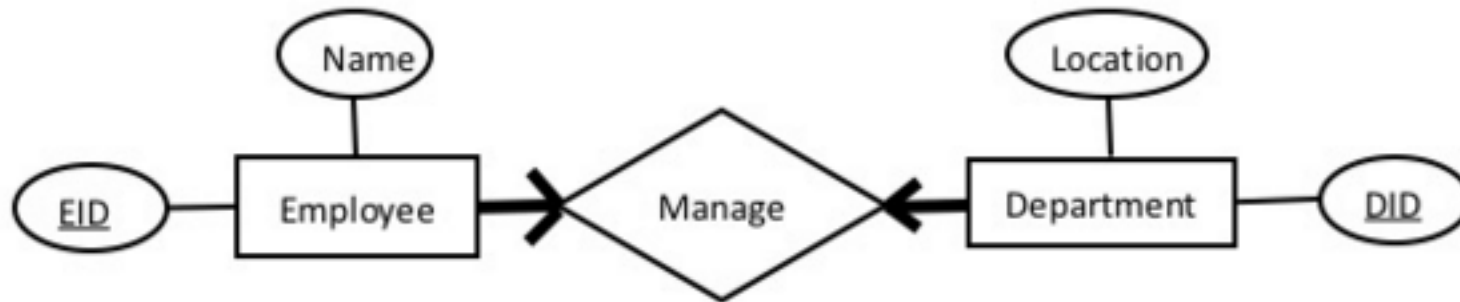
HW1 solution

Q: Why we don't have a total participation constraint from "Flight" to "Attendant"?

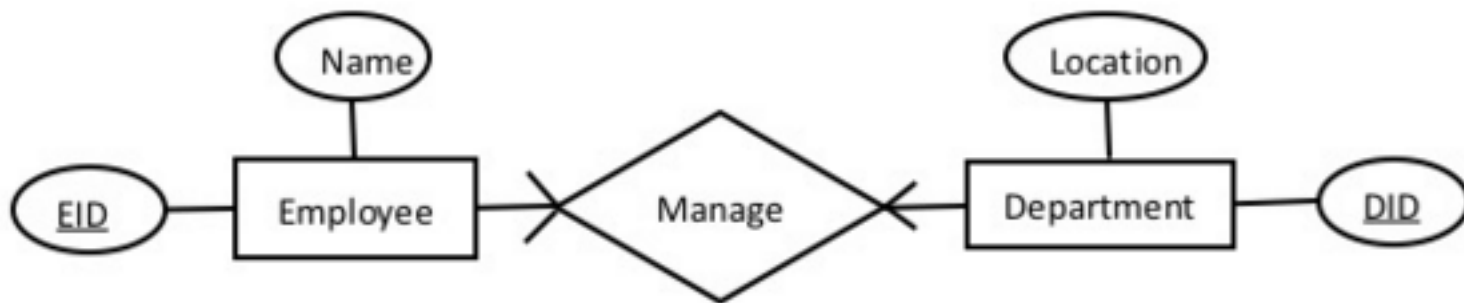
A: "5. Each flight is initialized with departure/arrival airports, projected departure datetime, and projected arrival datetime. Pilots, flight attendants, and an airplane can be assigned later to finalize a flight.



Examples in Quiz 3



- ❖ The solution used two tables;
- ❖ Alternatively we could use one table.



- ❖ Is it a good idea to use one table for this ER?

Next topic: SQL!

SQL-Based DBMSs

- ❖ Commercial RDBMS choices include
 - DB2 (IBM)
 - Oracle
 - SQL Server (Microsoft)
 - Teradata
- ❖ Open source RDBMS options include
 - MySQL
 - PostgreSQL
- ❖ And for so-called “Big Data”, we also have
 - Apache Hive (on Hadoop) + newer wannabees

Example Instances

<i>R1</i>	<u>sid</u>	<u>bid</u>	<u>day</u>
	22	101	10/10/96
	58	103	11/12/96

- ❖ We'll use these instances of our usual Sailors and Reserves relations in our examples.
- ❖ (If the key for the Reserves relation contained only the attributes *sid* and *bid*, how would the semantics differ?)

<i>S1</i>	<u>sid</u>	sname	rating	age
	22	dustin	7	45.0
	31	lubber	8	55.5
	58	rusty	10	35.0

<i>S2</i>	<u>sid</u>	sname	rating	age
	28	yuppy	9	35.0
	31	lubber	8	55.5
	44	guppy	5	35.0
	58	rusty	10	35.0

Example Data in MySQL

Sailors

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	4	25.5
95	Bob	3	63.5
101	Joan	3	NULL
107	Johan...	NULL	35.0

Reserves

sid	bid	date
22	101	1998-10-10
22	102	1998-10-10
22	103	1998-10-08
22	104	1998-10-07
31	102	1998-11-10
31	103	1998-11-06
31	104	1998-11-12
64	101	1998-09-05
64	102	1998-09-08
74	103	1998-09-08
NULL	103	1998-09-09
1	NULL	2001-01-11
1	NULL	2002-02-02

Boats

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Basic SQL Query

SELECT	[DISTINCT] <i>target-list</i>
FROM	<i>relation-list</i>
WHERE	<i>qualification</i>

- ❖ *relation-list* A list of relation names (possibly with a *range-variable* after each name).
- ❖ *target-list* A list of attributes of relations in *relation-list*
- ❖ *qualification* Comparisons ($Attr\ op\ const$ or $Attr1\ op\ Attr2$, where *op* is one of $<$, $<=$, $=$, $>$, $>=$, $<>$) combined using AND, OR and NOT.
- ❖ **DISTINCT** is an optional keyword indicating that the answer should not contain duplicates. Default is that duplicates are not eliminated! (Bags, not sets.)

Conceptual Evaluation Strategy

- ❖ Semantics of an SQL query defined in terms of the following conceptual evaluation strategy:
 - Compute the cross-product of *relation-list*.
 - Discard resulting tuples if they fail *qualifications*.
 - Project out attributes that are not in *target-list*.
 - If **DISTINCT** is specified, eliminate duplicate rows.
- ❖ This strategy is probably the least efficient way to compute a query! An optimizer will find more efficient strategies to compute *the same answers*.

Example of Conceptual Evaluation

```
SELECT S.sname  
FROM Sailors S, Reserves R      ← using S1  
WHERE S.sid=R.sid AND R.bid=103
```

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

A Note on Range Variables

Sailors(sid, sname, rating, age)

Reserves(sid, bid, day)

Boats(bid, bname, color)

- ❖ Really needed only if the same relation appears twice in the FROM clause. The previous query can also be written as:

```
SELECT sname
FROM   Sailors S, Reserves R
WHERE  S.sid=R.sid AND bid=103
```

OR

```
SELECT sname
FROM   Sailors, Reserves
WHERE  Sailors.sid=Reserves.sid
       AND bid=103
```

*It is good style,
however, to use
range variables
always!*

Find sailors who've reserved at least one boat

```
SELECT S.sid  
FROM Sailors S, Reserves R  
WHERE S.sid=R.sid
```

Sailors(sid, sname, rating, age)
Reserves(sid, bid, day)
Boats(bid, bname, color)

- ❖ Would adding DISTINCT to this query make a difference? (With our data? With possible data?)
- ❖ What is the effect of replacing *S.sid* by *S.sname* in the SELECT clause? Would adding DISTINCT to this variant of the query make a difference?

Expressions and Strings

Sailors(sid, sname, rating, age)

```
SELECT S.sname, S.age, S.age+1 AS age1, 7*S.age AS age2
FROM Sailors S
WHERE S.sname LIKE 'B_%B'
```

- ❖ Illustrates use of arithmetic expressions and string pattern matching: *Find triples (of ages of sailors and two fields defined by expressions) for sailors whose names begin and end with B and contain at least three characters.*
- ❖ **AS** provides a way to (re)name fields in result.
- ❖ **LIKE** is used for string matching. ``_`` stands for any one character and ``%`` stands for 0 or more arbitrary characters.

Find sid's of sailors who've reserved a red or a green boat

Sailors(sid, sname, rating, age)

Reserves(sid, bid, day)

Boats(bid, bname, color)

- ❖ If we replace **OR** by **AND** in this first version, what do we get?
- ❖ **UNION**: Can be used to compute the union of any two *union-compatible* sets of tuples (which are themselves the result of SQL queries).
- ❖ Also available: **EXCEPT** (What do we get if we replace **UNION** by **EXCEPT**?)

```
SELECT S.sid
FROM Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid
      AND (B.color='red' OR B.color='green' )
```

```
SELECT S.sid
FROM Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid
      AND B.color='red'
```

UNION

```
SELECT S.sid
FROM Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid
      AND B.color='green'
```

Find sid's of sailors who've reserved a red and a green boat

Sailors(sid, sname, rating, age)

Reserves(sid, bid, day)

Boats(bid, bname, color)

- ❖ **INTERSECT**: Can be used to compute the intersection of any two *union-compatible* sets of tuples.
- ❖ Included in the SQL/92 standard, but some systems don't support it.
- ❖ Contrast symmetry of the UNION and INTERSECT queries with how much the other versions differ.

```
SELECT S.sid
FROM Sailors S, Boats B1, Reserves R1,
     Boats B2, Reserves R2
WHERE S.sid=R1.sid AND R1.bid=B1.bid
     AND S.sid=R2.sid AND R2.bid=B2.bid
     AND (B1.color= 'red' AND B2.color= 'green')
```

```
SELECT S.sid
FROM Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid
     AND B.color= 'red'
```

INTERSECT

```
SELECT S.sid
FROM Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid
     AND B.color= 'green'
```

Key field!