

Tutorial

Levin Noronha

Sep 13, 2021

Slides by: Zheng Zheng
Modified by: Levin Noronha

Introduction

- TAs
 - Morteza Alipour Langouri
 - Email: alipoum@mcmaster.ca
 - Office hours: Fri 2:00pm – 3pm on MS Teams
 - Levin Noronha
 - Email: noronl@mcmaster.ca
 - Office hours: Thurs 2:30pm – 3:30pm on MS Teams
 - Lucia Cristiano
 - Email: cristial@mcmaster.ca
 - Office hours: Tue 3:00pm – 4:00pm on MS Teams
- Tutorials time
 - (T01) Tues 1:30pm - 2:20pm ETB 238
 - (T02) Mon 12:30pm - 1:20pm ETB 235
 - (T03) Wed 11:30am - 12:20pm ETB 235

Tutorial Agenda

- Relational Model and Keys
- Connect to DB2 servers and troubleshooting

Relational Model

- A model for representing data as relations (i.e., tables)
- **Relational database** is a collection of relations (or tables)

Relation

- **Relation schema** specifies the table name, name of all attributes, and the domain of each attribute
 - Students(*sid*: integer, *name*: string, *login*: string, *age*: integer, *gpa*: real)
- **Relation instance** (or simply **relation**) is a table

<i>sid</i>	<i>name</i>	<i>login</i>	<i>age</i>	<i>gpa</i>
50000	Dave	dave@cs	19	3.2
53666	Jones	jones@cs	18	3.3
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.7
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Definitions

- A **table** (or **relation**) consists of rows and columns.
- **Columns**, also known as **fields** and **attributes**, represent basic data components.
- **Rows**, also known as **tuples** or **records**, are a set of related attributes.

<i>sid</i>	<i>name</i>	<i>login</i>	<i>age</i>	<i>gpa</i>
50000	Dave	dave@cs	19	3.2
53666	Jones	jones@cs	18	3.3
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.7
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Keys

- Superkey: A set of attributes K is a superkey for a relation R if R cannot contain two distinct tuples t_1 and t_2 such that $t_1[K] = t_2[K]$
 - If K is a superkey, then so is any superset of K

Example

<i>sid</i>	<i>name</i>	<i>login</i>	<i>age</i>	<i>gpa</i>
50000	Dave	dave@cs	19	3.2
53666	Jones	jones@cs	18	3.3
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.7
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

- Superkeys:
 - {name, age}
 - {login}
 - {name, login}
 - {sid}
 - {sid, name, login, age, gpa}
 - etc...

Keys

- Superkey: A set of attributes K is a superkey for a relation R if R cannot contain two distinct tuples t_1 and t_2 such that $t_1[K] = t_2[K]$
 - If K is a superkey, then so is any superset of K
- Candidate Key: K is a candidate key for R if K is a minimal superkey

Example

<i>sid</i>	<i>name</i>	<i>login</i>	<i>age</i>	<i>gpa</i>
50000	Dave	dave@cs	19	3.2
53666	Jones	jones@cs	18	3.3
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.7
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

- Superkeys:

- {name, age}
- {login}
- {name, login}
- {sid}
- {sid, name, login, age, gpa}
- etc...

- Candidate keys:

- {login}
- {sid}
- {name, age}*
• {age, gpa}*

* Enforcing the highlighted keys might prevent the addition of new student tuples to the table. For e.g., {age, gpa} would enforce that two students can have the same age or gpa, but not both!

Keys

- Superkey: A set of attributes K is a superkey for a relation R if R cannot contain two distinct tuples t_1 and t_2 such that $t_1[K] = t_2[K]$
 - If K is a superkey, then so is any superset of K
- Candidate Key: K is a candidate key for R if K is a minimal superkey
- Primary Key: One candidate key that is chosen by the database designer as the principal means of identifying tuples within a relation
 - Only one candidate key can be primary key

Example

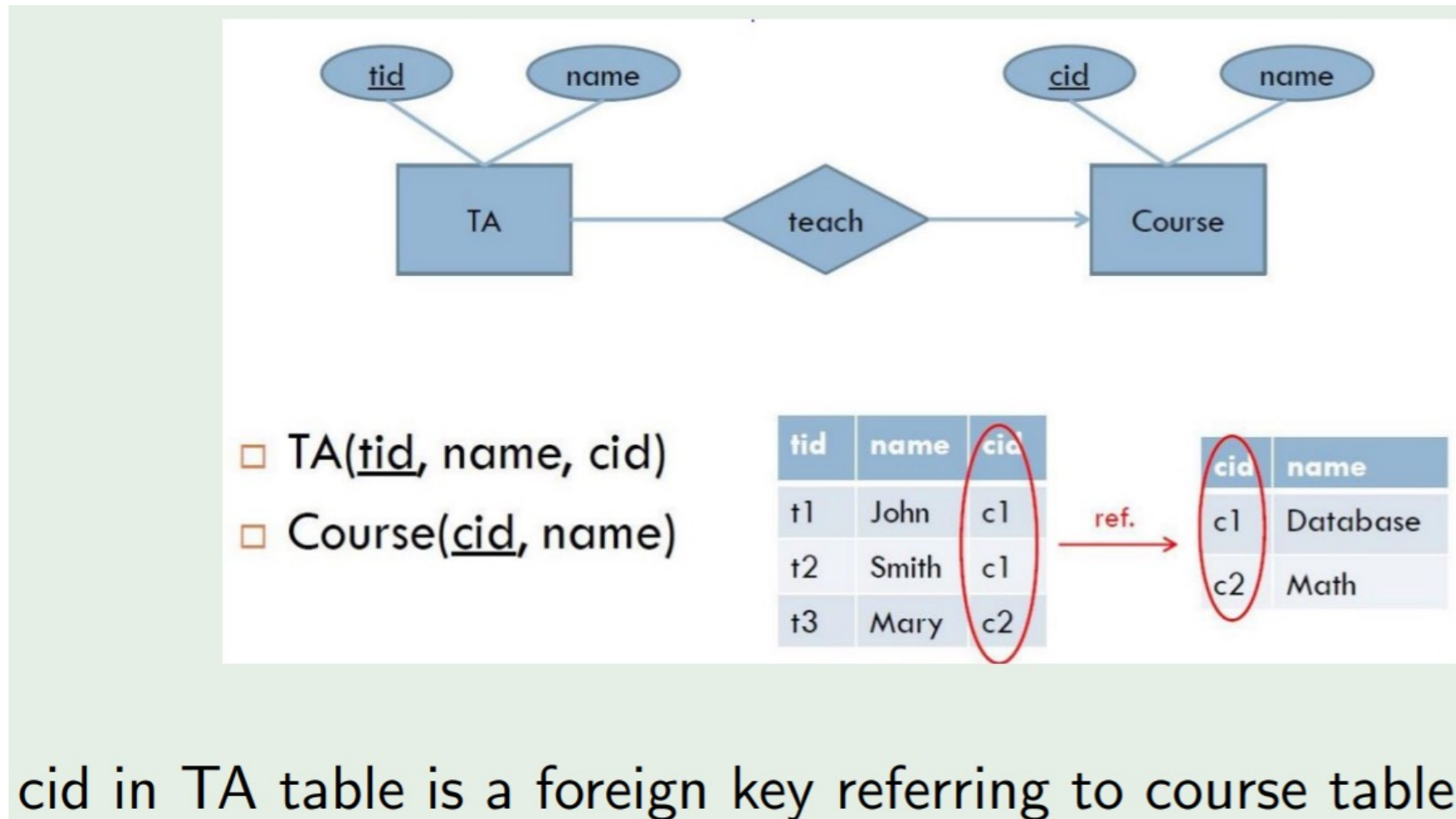
<i>sid</i>	<i>name</i>	<i>login</i>	<i>age</i>	<i>gpa</i>
50000	Dave	dave@cs	19	3.2
53666	Jones	jones@cs	18	3.3
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.7
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

- Superkeys:
 - {name, age}
 - {login}
 - {name, login}
 - {sid}
 - {sid, name, login, age, gpa}
 - etc...
- Candidate keys:
 - {login}
 - {sid}
 - {name, age}
 - {age, gpa}
- Primary keys:
 - {sid}

Keys

- **Super key:** A set of attributes K is a super key for a relation R if it can uniquely identify a tuple, i.e., R cannot contain two distinct tuples t_1 and t_2 such that $t_1[K] = t_2[K]$
 - If K is a super key, then so is any superset of K
- **Candidate Key:** K is a candidate key for R if K is a minimal super key, i.e., no attributes can be removed from K without losing the unique identification property.
- **Primary Key:** One candidate key that is chosen by the database designer as the principal means of identifying tuples within a relation
 - Only one candidate key can be the primary key
- **Foreign Key:** A foreign key requires that the values on a set X of attributes of a relation R_1 must appear as values of the primary key of another relation R_2

Example



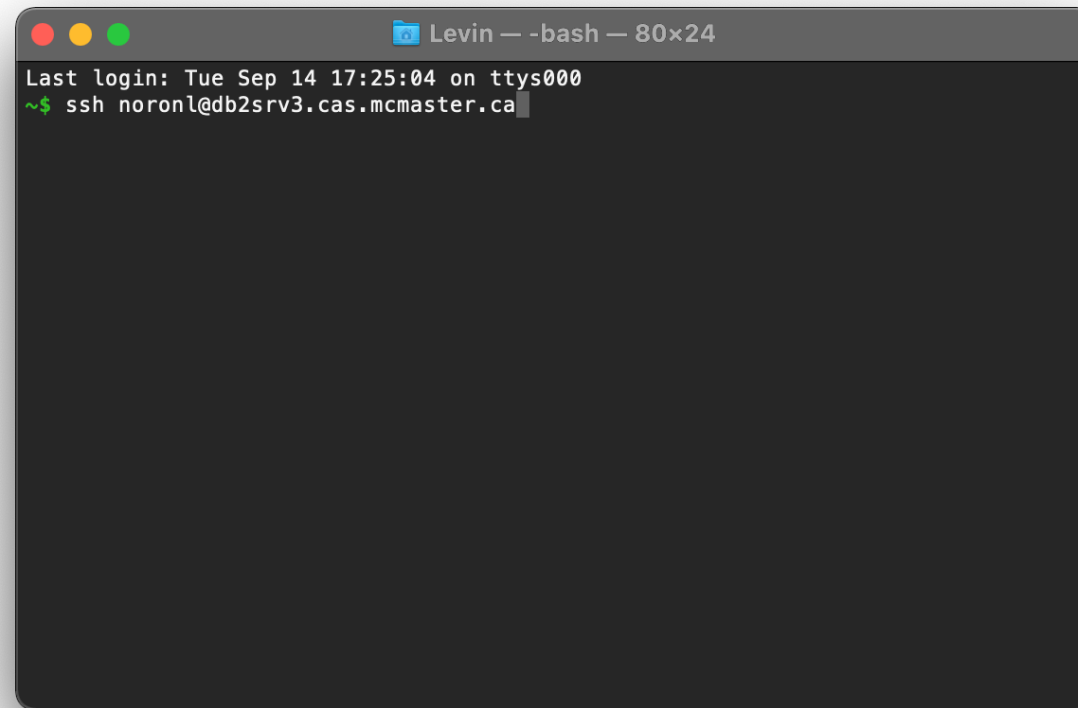
Environment setup

- Department DB2 server
 - db2srv3.cas.mcmaster.ca
- To access db2srv3, you must either:
 - a) Use the McMaster VPN, or
 - b) SSH to mills.mcmaster.ca first, then SSH to a DB2 server

How to connect to db2srv3.cas.mcmaster.ca

- Windows: use PuTTY, Xshell or another SSH client.
- macOS/Linux: In a bash terminal, type:
 - `ssh <macid>@db2srv3.cas.mcmaster.ca`

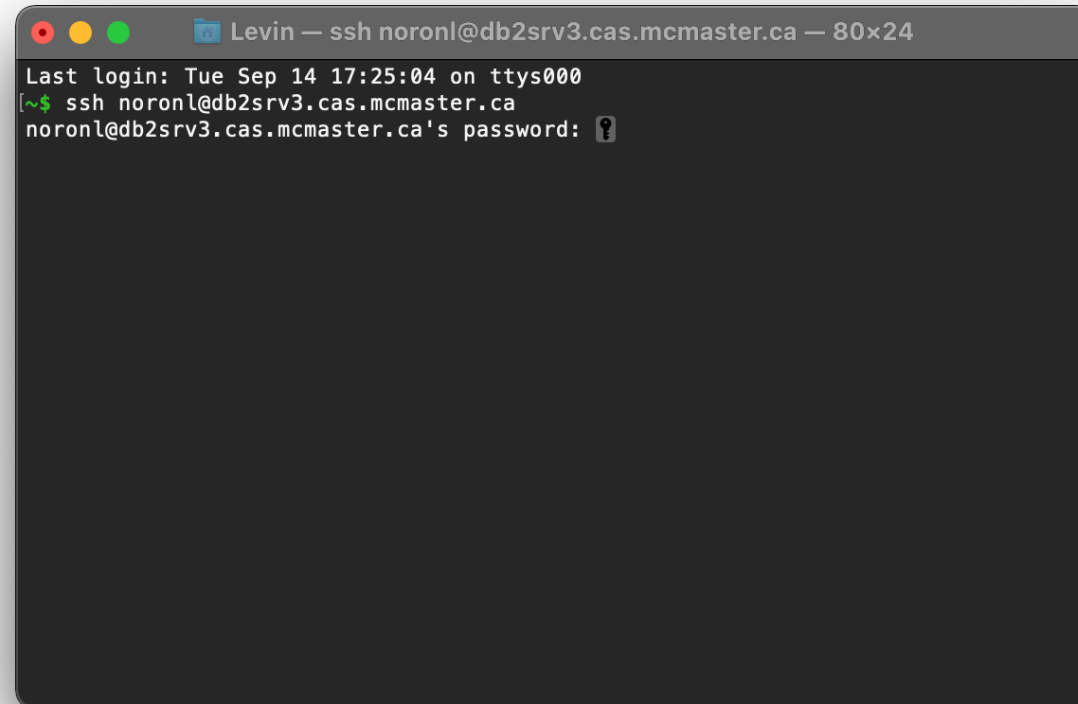
Bash example (macOS, Linux)



A screenshot of a terminal window on a macOS system. The window title bar shows 'Levin — -bash — 80x24'. The terminal content displays the last login information: 'Last login: Tue Sep 14 17:25:04 on ttys000'. Below this, the prompt '~\$' is followed by the command 'ssh noronl@db2srv3.cas.mcmaster.ca' which has been entered but not yet executed, as indicated by the cursor at the end of the line.

```
Levin — -bash — 80x24
Last login: Tue Sep 14 17:25:04 on ttys000
~$ ssh noronl@db2srv3.cas.mcmaster.ca
```

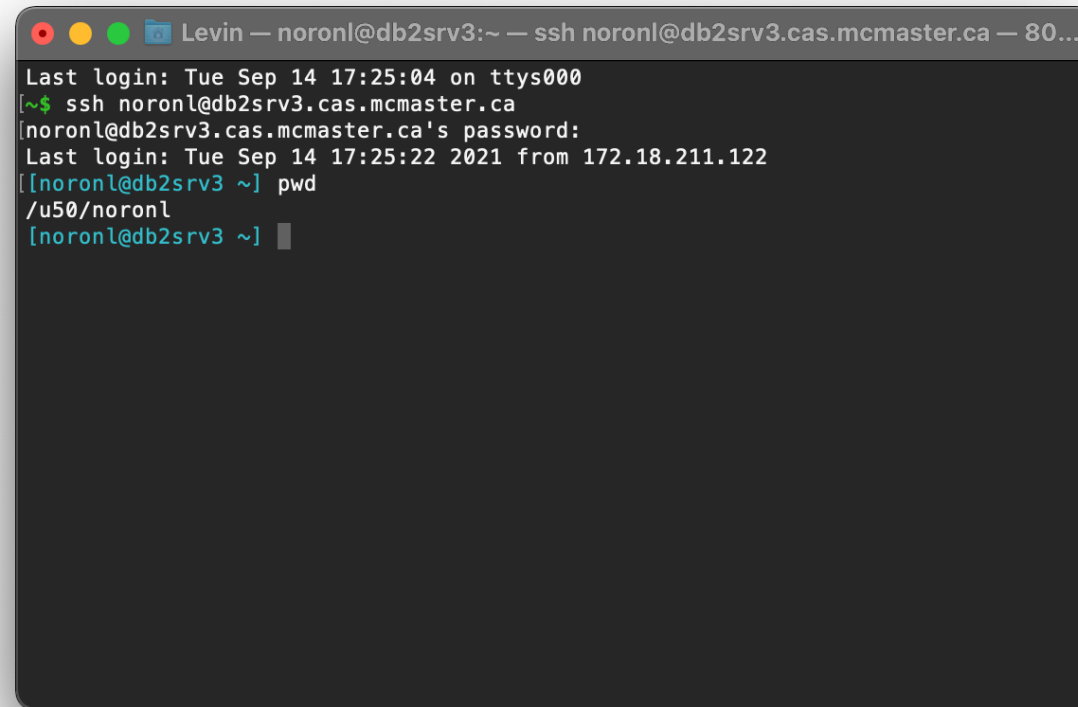
Bash example (macOS, Linux)

A terminal window with a dark background and light text. The title bar at the top reads "Levin — ssh noronl@db2srv3.cas.mcmaster.ca — 80x24". The terminal content shows a login sequence: "Last login: Tue Sep 14 17:25:04 on ttys000", followed by a prompt "[~]\$ ssh noronl@db2srv3.cas.mcmaster.ca" and the response "noronl@db2srv3.cas.mcmaster.ca's password: " with a cursor icon.

```
Levin — ssh noronl@db2srv3.cas.mcmaster.ca — 80x24
Last login: Tue Sep 14 17:25:04 on ttys000
[~]$ ssh noronl@db2srv3.cas.mcmaster.ca
noronl@db2srv3.cas.mcmaster.ca's password: 
```

In most cases, your CAS account password should be the same as your macOS password.

Bash example (macOS, Linux)

A terminal window with a dark background and light text. The title bar shows 'Levin — noronl@db2srv3:~ — ssh noronl@db2srv3.cas.mcmaster.ca — 80...'. The terminal content shows a successful SSH login with a last login timestamp and IP address. The user then runs the 'pwd' command, which outputs the home directory path. The prompt is '[noronl@db2srv3 ~]' followed by a cursor.

```
Levin — noronl@db2srv3:~ — ssh noronl@db2srv3.cas.mcmaster.ca — 80...
Last login: Tue Sep 14 17:25:04 on ttys000
[~$ ssh noronl@db2srv3.cas.mcmaster.ca
[noronl@db2srv3.cas.mcmaster.ca's password:
Last login: Tue Sep 14 17:25:22 2021 from 172.18.211.122
[noronl@db2srv3 ~] pwd
/u50/noronl
[noronl@db2srv3 ~]
```

IMPORTANT: Verify that you have a home directory (/u50/<macID>) on the DB2 server using command “pwd”

Troubleshooting

- If you have any login problems with your CAS account
 - 1) Follow the instructions on <https://www.cas.mcmaster.ca/account/>
 - 2) If 1) doesn't resolve the problem, please contact the TAs
- All login problems should be fixed before Sept. 22
 - Otherwise, you will not be able to complete the assignments

Change your CAS password

- Your new password must be at least 8 characters long and contain at least one letter and one digit.
- You will receive an e-mail confirming the password change.
- If you don't know your CAS password, you can use your MacID to authenticate and reset CAS account here: <https://www.cas.mcmaster.ca/reset>

Username:

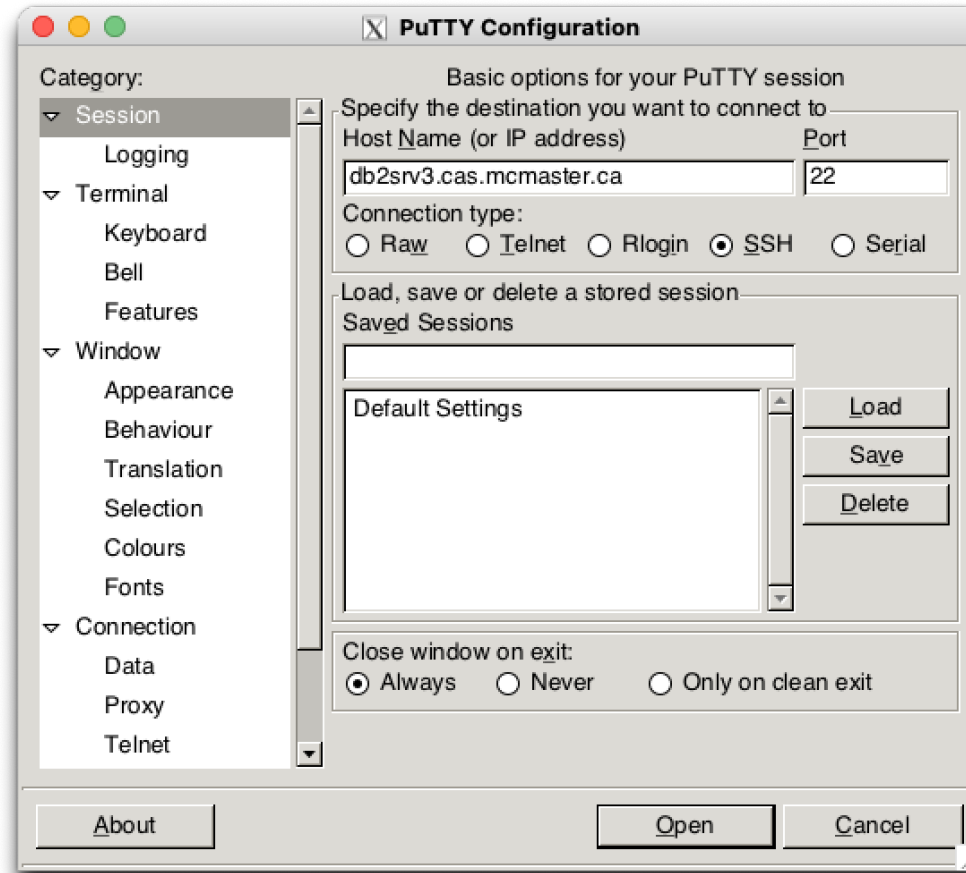
Old password:

New password:

New password (confirm):

Change Password

PuTTY example (Windows)



PuTTY example (Windows)



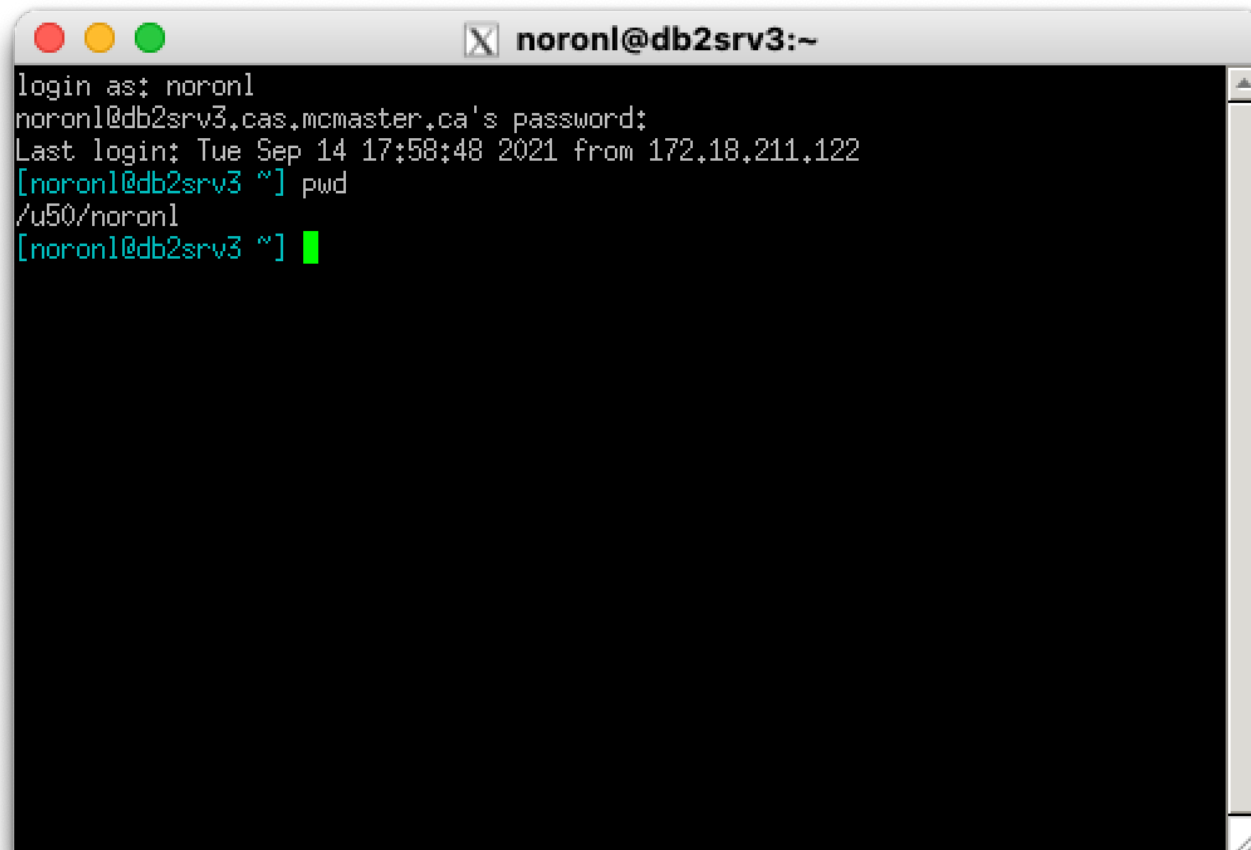
In most cases, your CAS account password should be the same as your macOS password.

PuTTY example (Windows)



In most cases, your CAS account password should be the same as your macOS password.

PuTTY example (Windows)



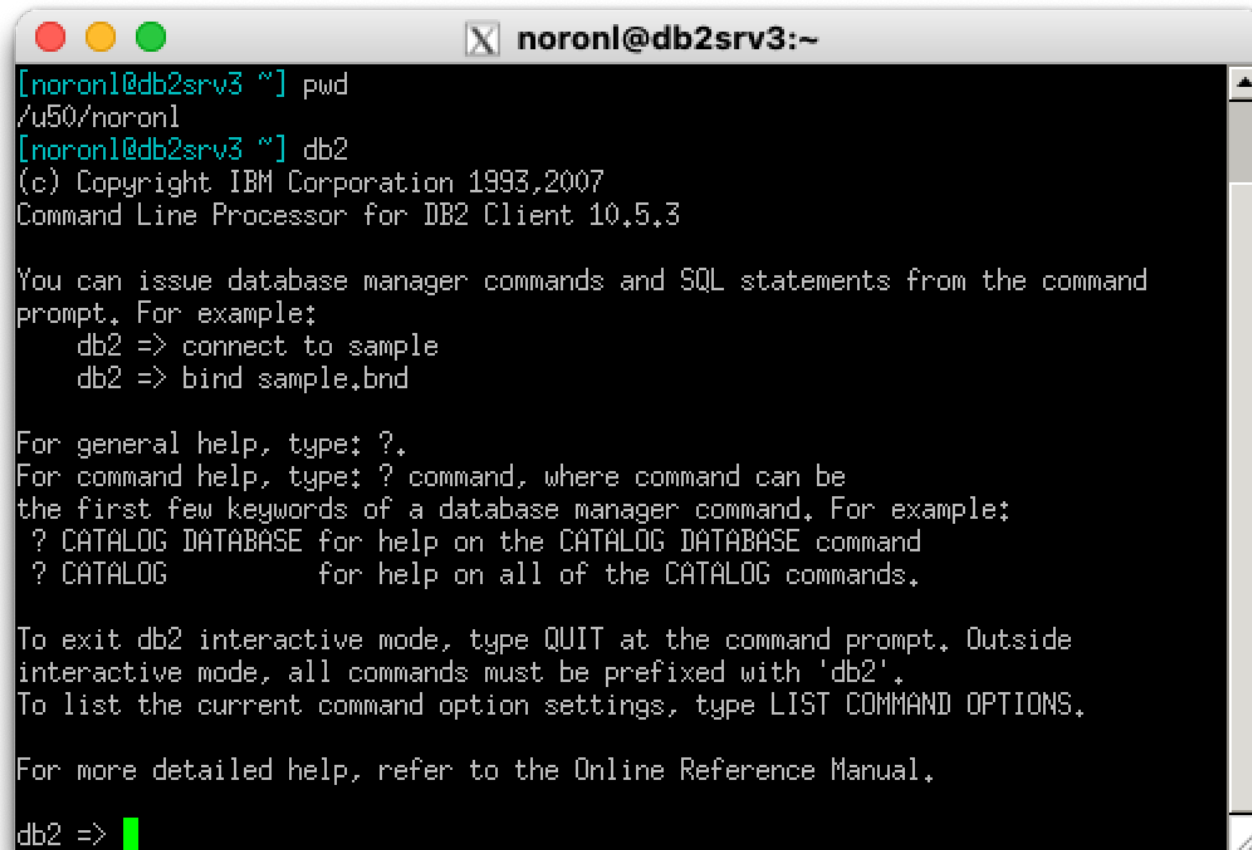
```
login as: noronl
noronl@db2srv3.cas.mcmaster.ca's password:
Last login: Tue Sep 14 17:58:48 2021 from 172.18.211.122
[noronl@db2srv3 ~] pwd
/u50/noronl
[noronl@db2srv3 ~] █
```

IMPORTANT: Verify that you have a home directory (/u50/<macID>) on the DB2 server using command “pwd”

Basic Commands

- Start application by typing “db2”

Example

A terminal window titled 'noronl@db2srv3:~' with standard macOS window controls (red, yellow, green buttons). The terminal shows the execution of 'pwd' and 'db2' commands. The 'db2' command outputs copyright information and a list of example commands. It also provides instructions on how to use '?' for help and 'QUIT' to exit. The prompt 'db2 =>' is followed by a green cursor.

```
[noronl@db2srv3 ~] pwd
/u50/noronl
[noronl@db2srv3 ~] db2
(c) Copyright IBM Corporation 1993,2007
Command Line Processor for DB2 Client 10.5.3

You can issue database manager commands and SQL statements from the command
prompt. For example:
    db2 => connect to sample
    db2 => bind sample.bnd

For general help, type: ?.
For command help, type: ? command, where command can be
the first few keywords of a database manager command. For example:
    ? CATALOG DATABASE for help on the CATALOG DATABASE command
    ? CATALOG           for help on all of the CATALOG commands.

To exit db2 interactive mode, type QUIT at the command prompt. Outside
interactive mode, all commands must be prefixed with 'db2'.
To list the current command option settings, type LIST COMMAND OPTIONS.

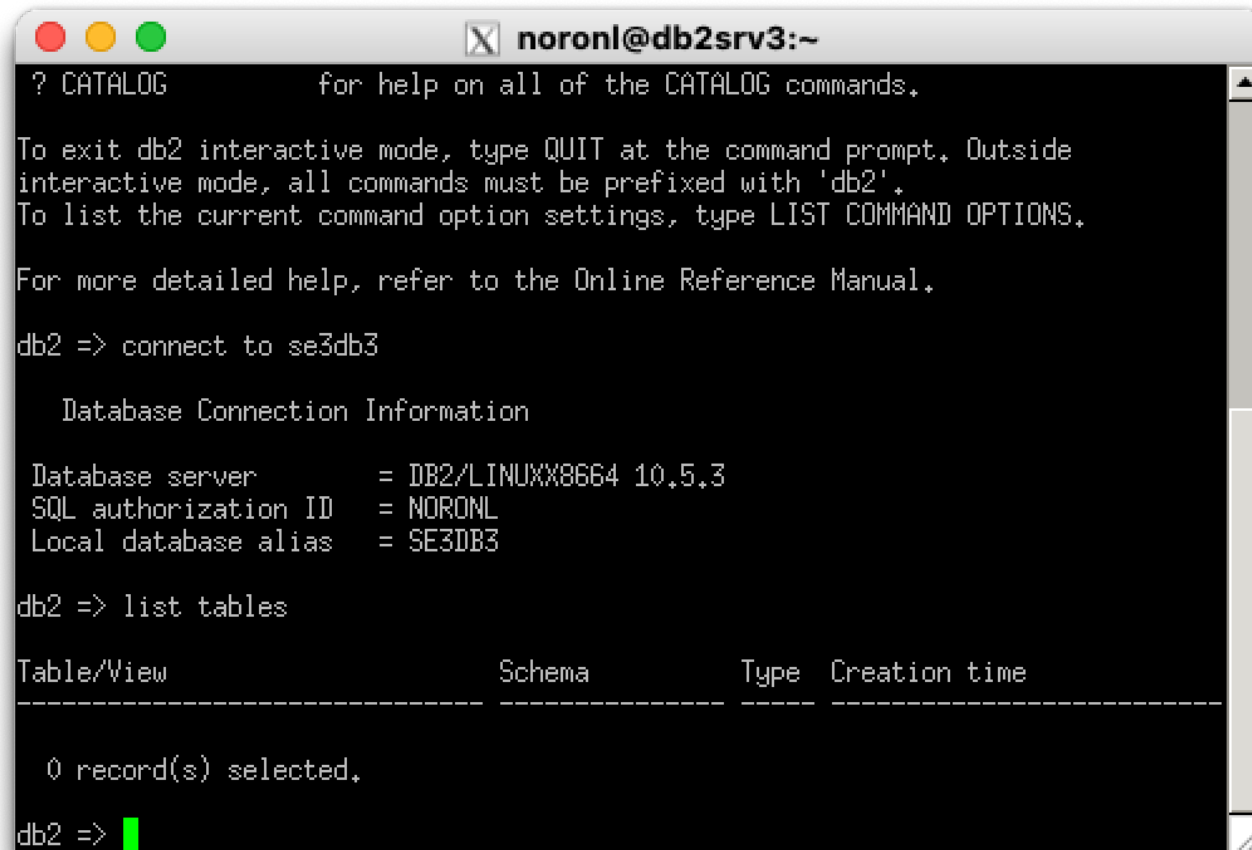
For more detailed help, refer to the Online Reference Manual.

db2 => █
```

Basic Commands

- Write a comment: - - this is a comment
- Make a connection to database: connect to se3db3
- List tables: list tables

Example



```
noronl@db2srv3:~  
? CATALOG          for help on all of the CATALOG commands.  
  
To exit db2 interactive mode, type QUIT at the command prompt. Outside  
interactive mode, all commands must be prefixed with 'db2'.  
To list the current command option settings, type LIST COMMAND OPTIONS.  
  
For more detailed help, refer to the Online Reference Manual.  
  
db2 => connect to se3db3  
  
      Database Connection Information  
  
Database server      = DB2/LINUX8664 10.5.3  
SQL authorization ID = NORONL  
Local database alias = SE3DB3  
  
db2 => list tables  
  
Table/View          Schema      Type      Creation time  
-----  
  
0 record(s) selected.  
  
db2 => █
```

Upload script

- Windows: use WinSCP client if you use Putty; use Xftp client if you use Xshell.
- macOS, Linux: use scp command
 - For e.g., `scp <files_src_path> <macid>@mills.mcmaster.ca:<dest_path >`
 - `scp /Users/script.ddl <macid>@db2srv3.mcmaster.ca:/u50/path/`
- Run your script in server
 - `db2 -tnf script.ddl`

SE3DB3 TUTORIAL

Lucia Cristiano
Sept 20-22, 2021

Introduction

- Expectations:
 - I will repeat the questions that students in the room ask before I answer
 - I will pause occasionally to look at the chat, please unmute yourself or ask questions at these points

Outline

- Review of Keys
- Referential Integrity
- ER Terms
- ER Diagram Notations
- Relationship Types
- Relationship Degree
- Participation Constraints
- ER Design Example
- Assignment 1 Tips
- Contact

Review of Keys

Employee Relation

Employee ID	Name	DOB	Department No
123	John Smith	02/25/1978	5
456	Alice Doe	04/06/1984	3
789	John Smith	09/17/1990	3

List candidates key(s), super key(s) and designate a primary key:

Referential Integrity

- Referential integrity is used to guarantee that attributes in one relation refer to existing tuples in another relation referenced in a relationship

Employee ID	Name	DOB	Department No
123	John Smith	02/25/1978	5
456	Alice Doe	04/06/1984	3
789	John Smith	09/17/1990	3

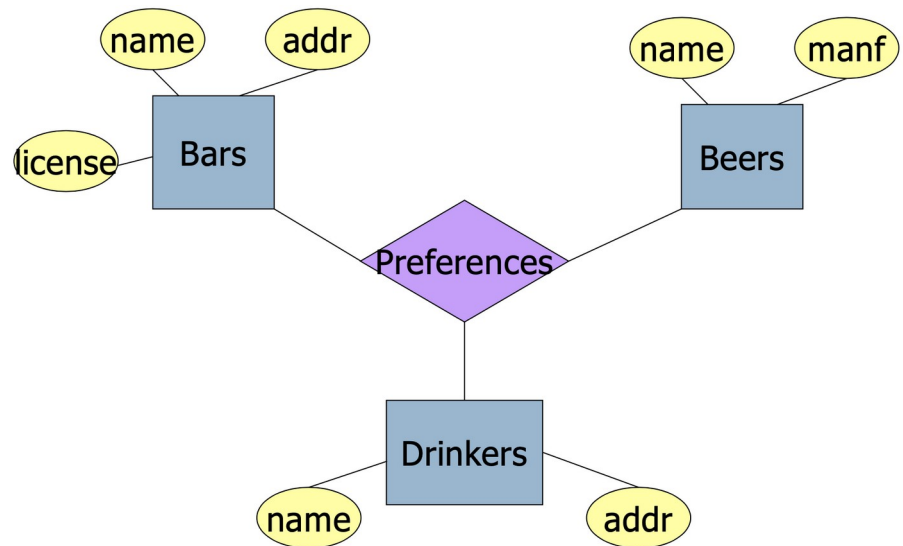
Is there a foreign key or referential integrity violation between these relations?

Department No	Name	Manager	# employees
5	Finance	Jane Mae	10
2	Sales	Bob Brown	15
4	Development	Sally Roe	30

ER Terms

- Entity: Is a “thing” or object
- Attribute: Is a property of an entity set. Usually, a simple value
 - Has a domain: a list of values under the that attribute
- Relationship: Association between entity sets

List entities, attributes and relationships:

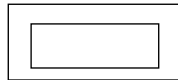


ER Diagram Notations

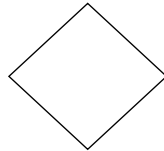
- Entity



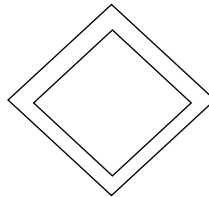
- Weak Entity



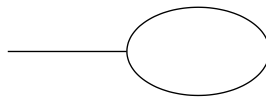
- Relationship



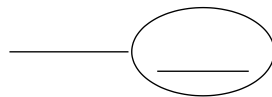
- Identifying Relationship



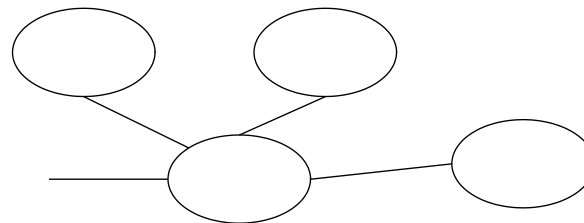
- Attribute



- Key Attribute

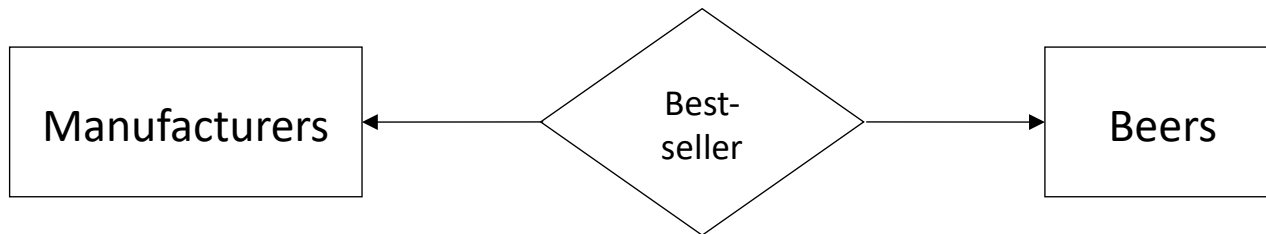


- Composite Attribute



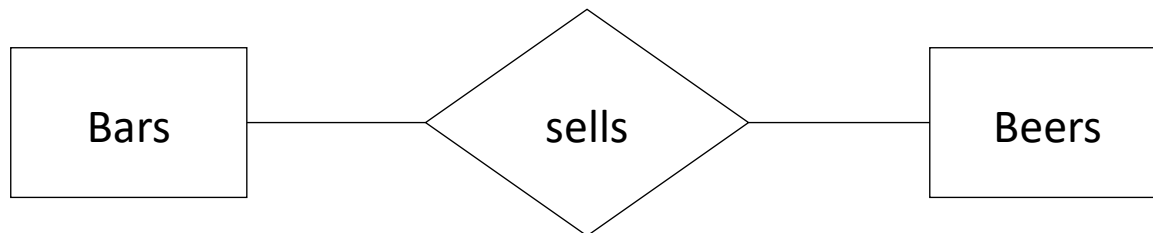
One-to-One Relationship

- Each entity of either entity set is related to at most one entity of the other set.
- E.g., an manufacturer has exactly one best-seller beer.



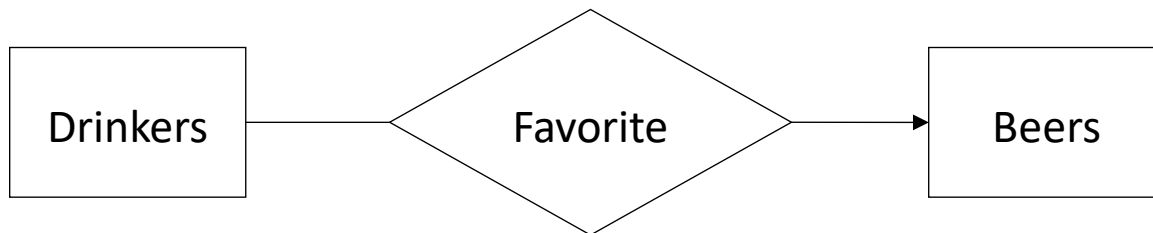
Many-to-Many Relationship

- An entity of either set can be connected to many entities of the other set.
- E.g., a bar sells many beers, and a beer is sold by many bars.



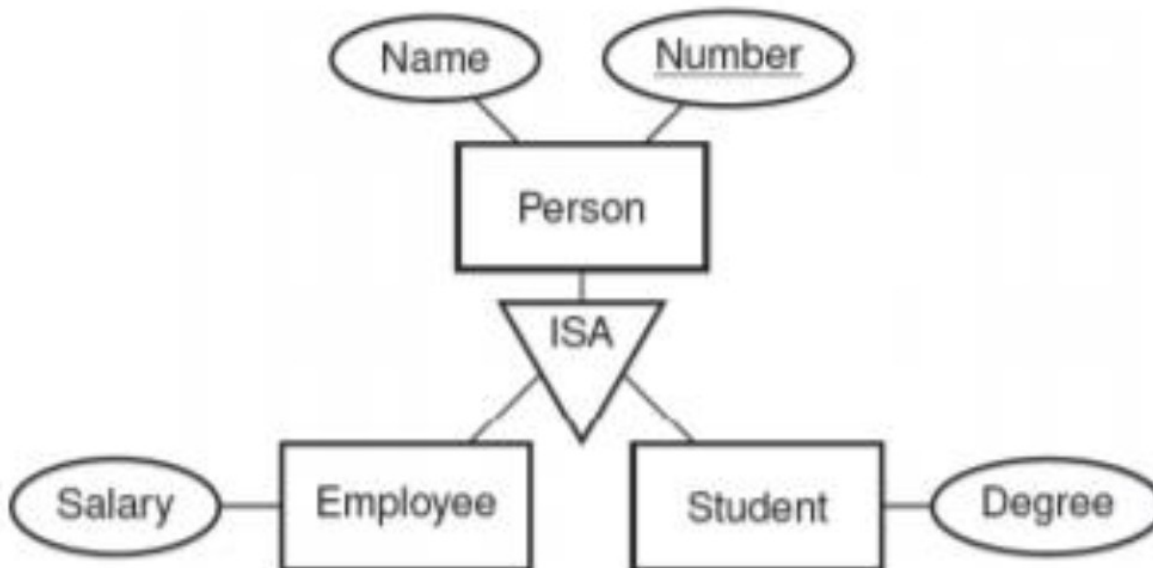
Many-to-One Relationship

- Each entity of the first set is connected to at most one entity of the second set. But an entity of the second set can be connected to zero, one, or many entities of the first set.
- E.g., a drinker has only one most favorite beer, and a beer can be the most favorite of many drinkers.



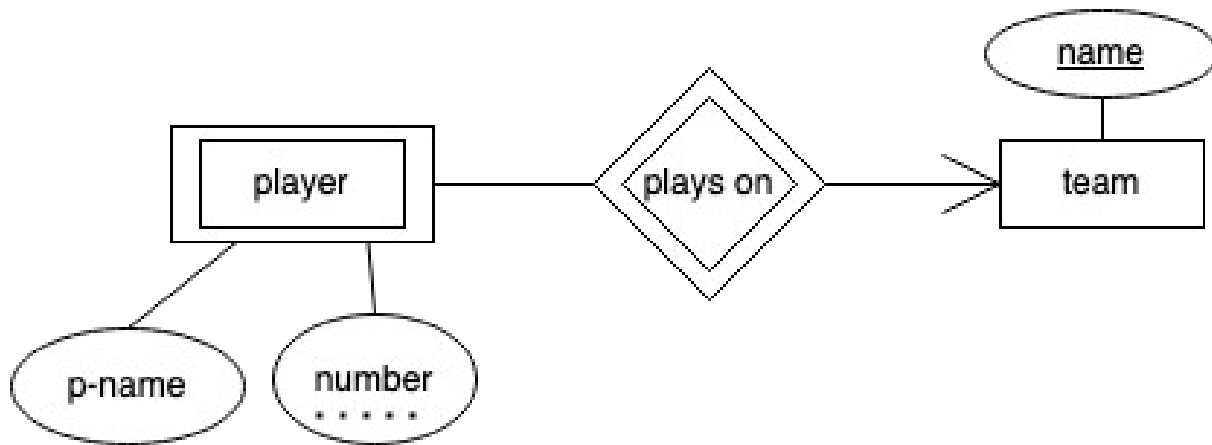
Subclasses/ISA Relationship

- Subclass are a special type of entity set that inherit properties from a parent entity
- The subclass must have all attributes of the parent as well as having additional properties



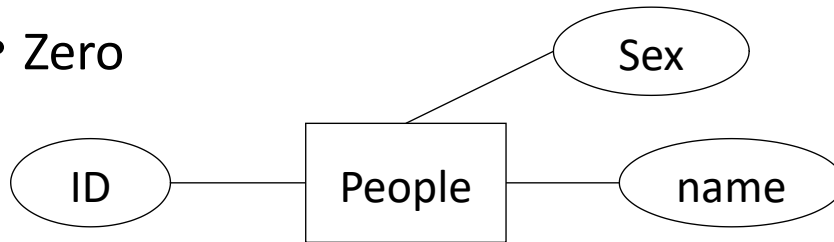
Weak Entity Sets

- Weak entities do not have enough information to have its own primary key
- The weak entity relies on the supporting entity for its identification
 - The key of the weak entity is a composite key made of the primary key of the supporting entity and the partial key of the weak entity

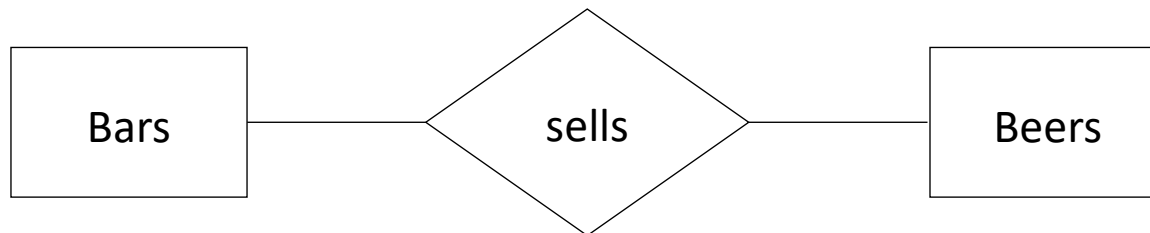


Relationship Degree(1)

- Zero

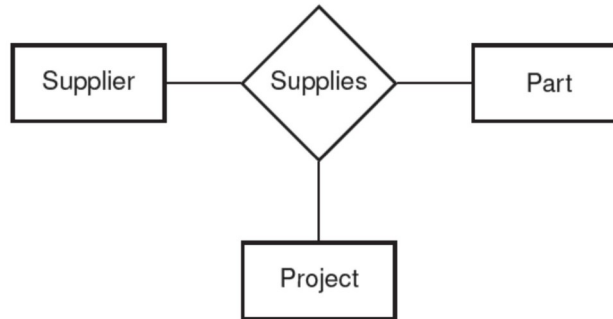


- Binary

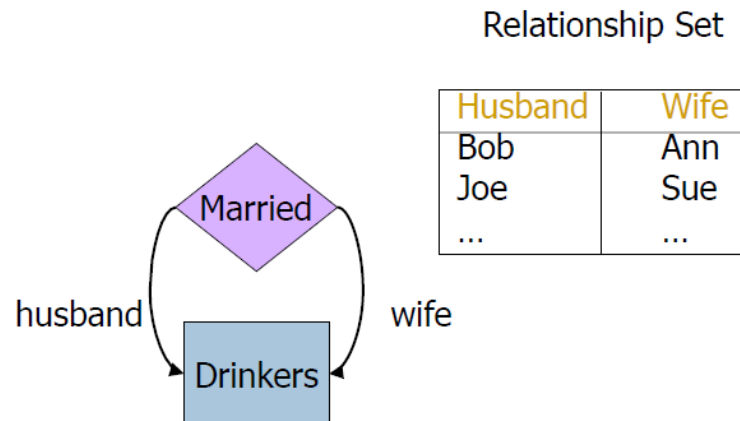


Relationship Degree(2)

- Ternary

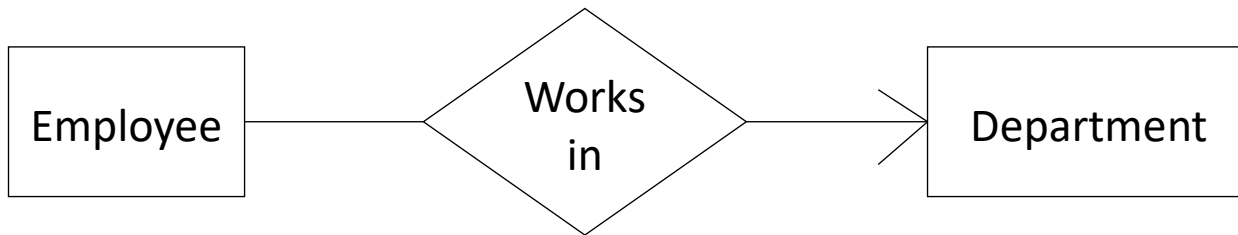


- Roles(Cyclical or recursive relationship)

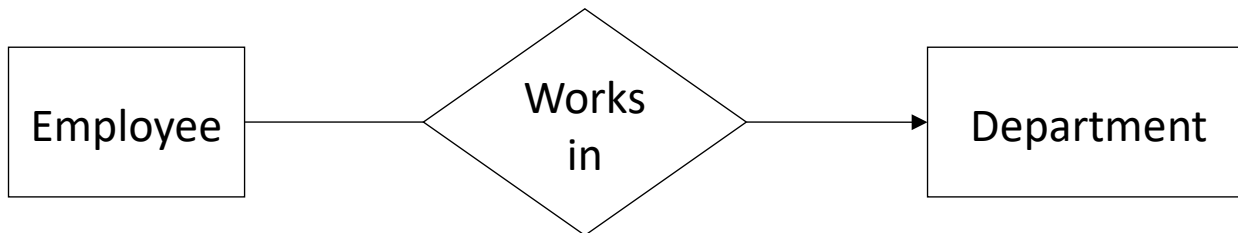


Participation

- Totally participation: At least one or more entities are in the relationship. Mandatory



- Partially participation: zero or more entities are in the relationship. Optional

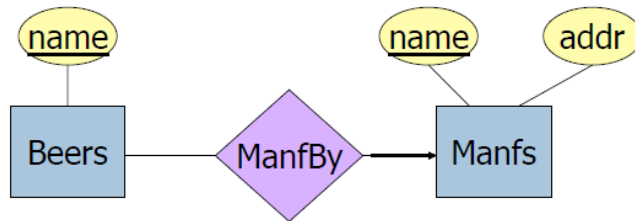


ER Schema Mapping(4)

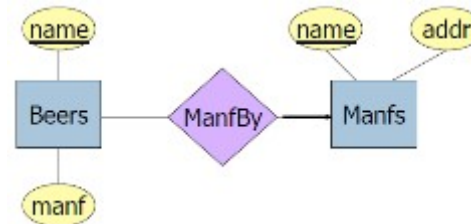
- Design Techniques

- Avoid redundancy

- Good



- Bad



- Limit the use of weak entity sets

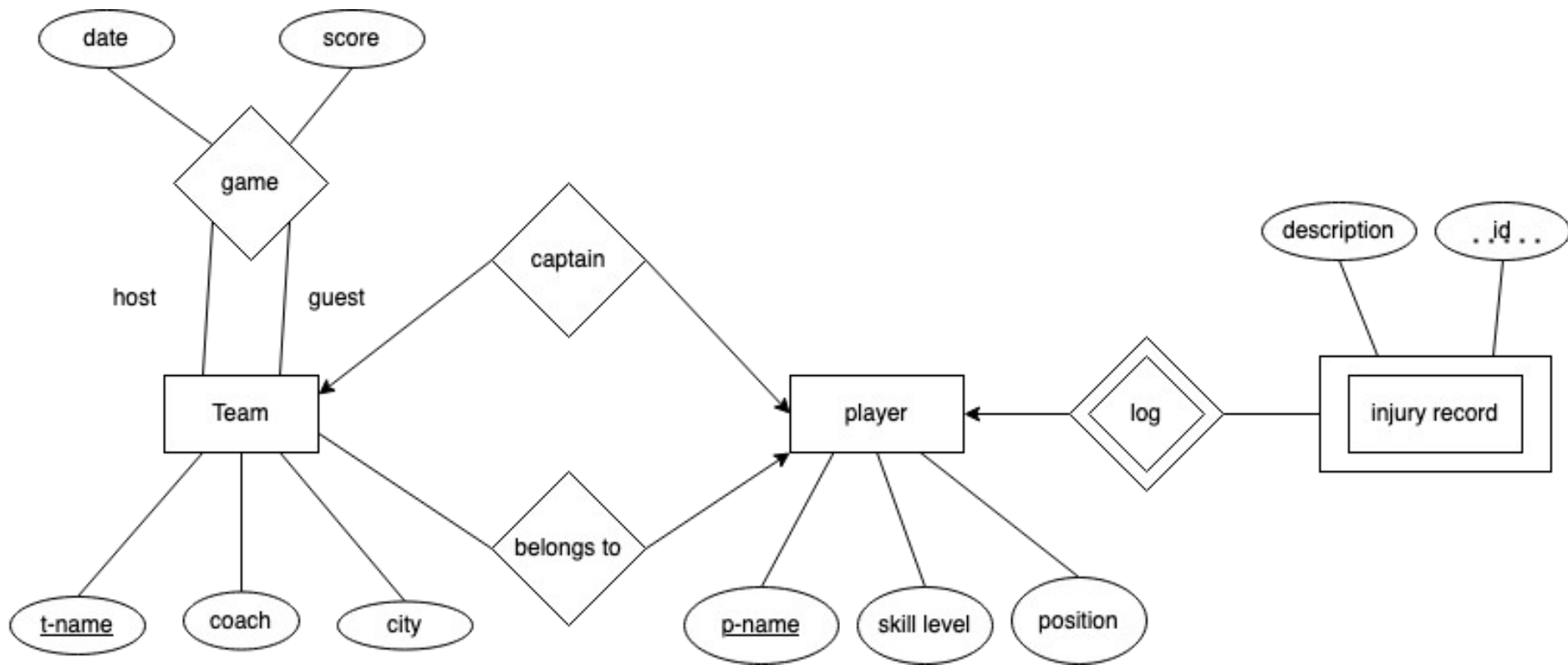
- Do not use an entity set when an attribute will do

An Example

Suppose you are given the following requirements for a simple database for the National Hockey League (NHL):

- the NHL has many teams,
- each team has a name, a city, a coach, a captain, and a set of players,
- each player belongs to only one team,
- each player has a name, a position (such as left wing or goalie), a skill level, and a set of injury records,
- a team captain is also a player,
- a game is played between two teams (referred to as `host_team` and `guest_team`) and has a date (such as May 11th, 1999) and a score (such as 4 to 2).
- Note: name is the primary key for team and player

Solution



Assignment 1 Tips

Suggested drawing software: (some are paid with free trials)

- [Lucid Chart](#)
- [diagrams.net](#)
- [Visual Paradigm](#)
- [ERD Plus](#)
- Pay attention to the description, look for words that indicate relationships, constraints, special cases etc.

Contact

If you have any questions or feedback, please email me or attend my office hours:

Lucia Cristiano

- Email: cristial@mcmaster.ca
- Office hours: Tue 3:00pm – 4:00pm on MS Teams

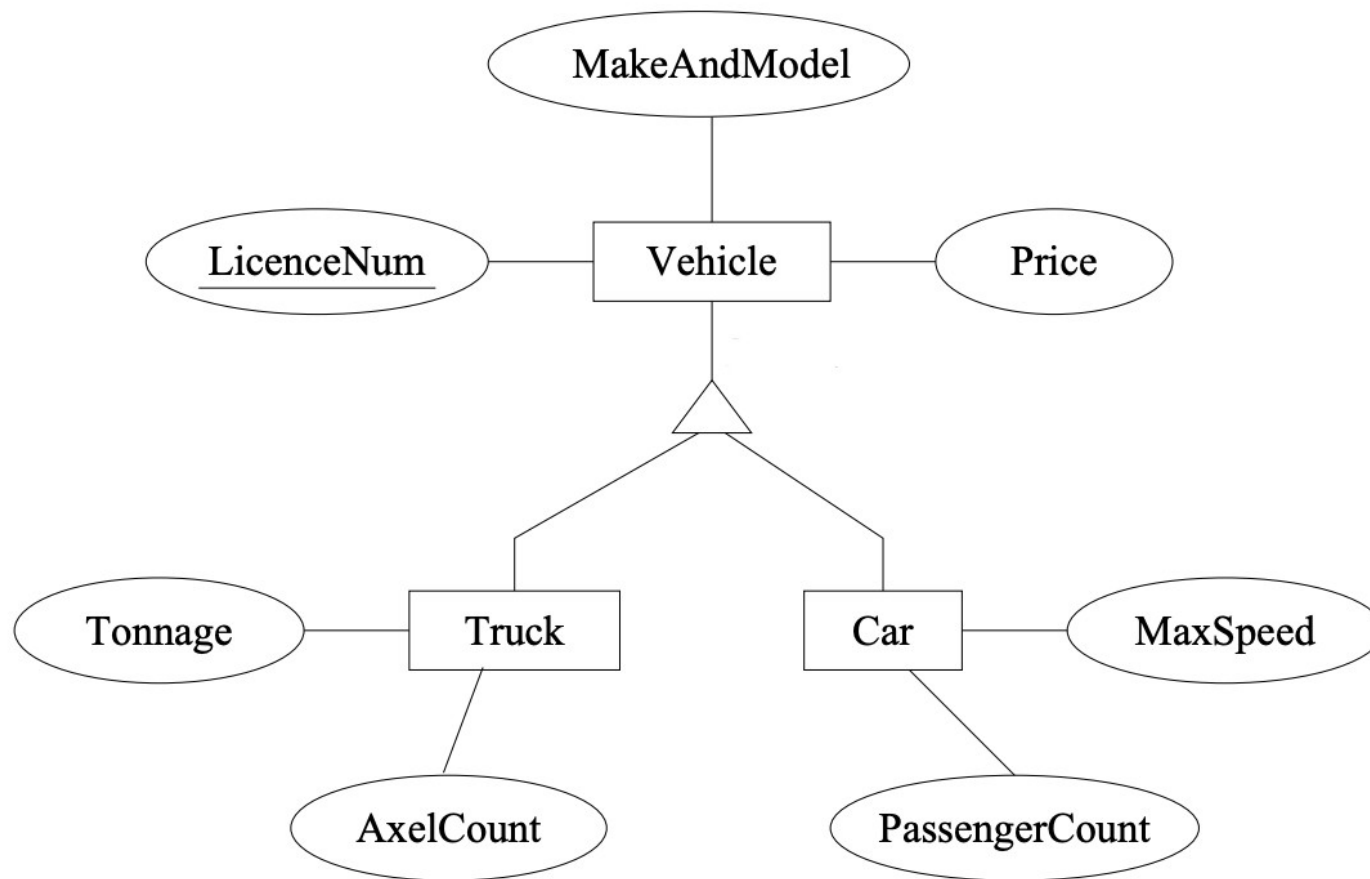
DataBase Tutorial

Sep. 27, 2021
Morteza Alipour

Outline

- ER Schema Mapping
- Aggregation Function
- Group by/Having/Joins/Views
- Questions

IS-A



Vehicle

<u>LicenceNum</u>	MakeAndModel	Price
-------------------	--------------	-------

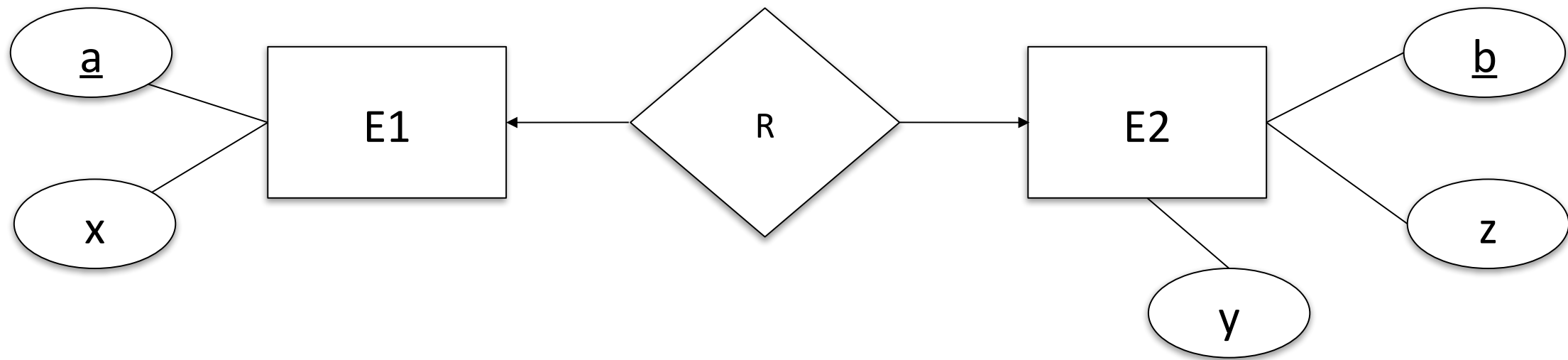
Truck

<u>LicenceNum</u>	Tonnage	AxelCount
-------------------	---------	-----------

Car

<u>LicenceNum</u>	MaxSpeed	PassengerCount
-------------------	----------	----------------

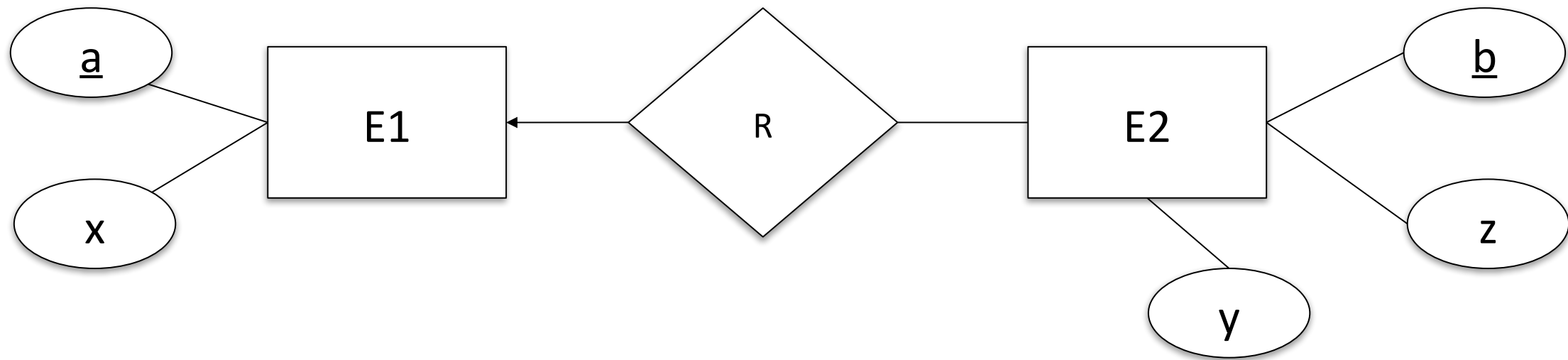
One-to-One Relationship



$E1: (\underline{a}, x, b), E2: (\underline{b}, y, z)$

Or $E1: (\underline{a}, x), E2: (\underline{b}, y, z, a)$

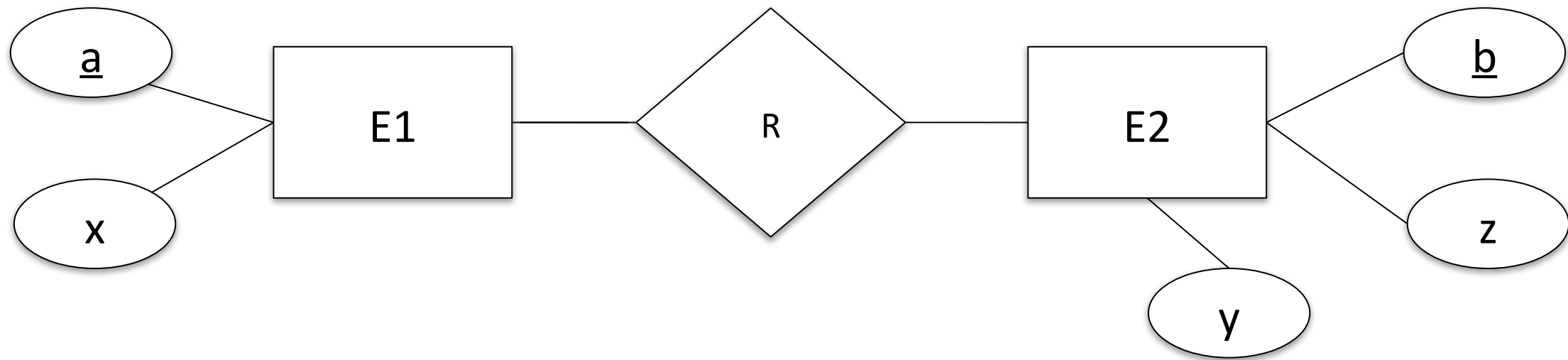
Many-to-One Relationship



E1: (a, x), E2: (b, y, z, **a**)

Note: the primary key of E1 is the foreign key of E2

Many-to-Many Relationship



E1: (a, x), E2: (b, y, z), R: (a, b)

Aggregation Functions

- Aggregate function is a function where the values of multiple rows are grouped together as input on certain criteria to form a single output value.
- Max, Min, Sum, Count, Avg
- Aggregate functions often need an added GROUP BY statement.

Example

- If you only want to return the SUM:

```
SELECT SUM(aggregate_expression)
FROM tables
WHERE conditions;
```

- If you want to return the several attributes and SUM:

```
SELECT expression1, expression2, ... expression_n,
       SUM(aggregate_expression)
FROM tables
WHERE conditions
GROUP BY expression1, expression2, ... expression_n;
```

- Find out salary of all employees whose salary is above \$25,000 / year. (Only return SUM)

```
SELECT SUM(salary) AS "Total Salary"  
FROM employees  
WHERE salary > 25000;
```

- Return the name of the department and the total sales (in the associated department).

```
SELECT department, SUM(sales) AS "Total sales"  
FROM order_details  
GROUP BY department;
```

Group by

- The SQL GROUP BY clause can be used in a SELECT statement to collect data across multiple records and group the results by one or more columns.
- It is often used with Aggregation Functions

```
SELECT expression1, expression2, ... expression_n,  
       aggregate_function (aggregate_expression)  
FROM tables  
WHERE conditions  
GROUP BY expression1, expression2, ... expression_n;
```

Example: group by and Aggregation

Functions

- Uses the COUNT function to return the department and the number of employees (in the department) that make over \$25,000 / year.

```
SELECT department, COUNT(*) AS "Number of employees"  
FROM employees  
WHERE salary > 25000  
GROUP BY department;
```

- uses the MIN function to return the name of each department and the minimum salary in the department.

```
SELECT department, MIN(salary) AS "Lowest salary"  
FROM employees  
GROUP BY department;
```

Having

- The SQL HAVING Clause is used in combination with the GROUP BY Clause to restrict the groups of returned rows to only those whose the condition is TRUE.

```
SELECT expression1, expression2, ... expression_n,  
       aggregate_function (aggregate_expression)  
FROM tables  
WHERE conditions  
GROUP BY expression1, expression2, ... expression_n  
HAVING having_condition;
```


Example

- use the SQL SUM function to return the name of the department and the total sales (in the associated department). The SQL HAVING clause will filter the results so that only departments with sales greater than \$1000 will be returned.

```
SELECT department, SUM(sales) AS "Total sales"  
FROM order_details  
GROUP BY department  
HAVING SUM(sales) > 1000;
```

Statement Order

```
SELECT column1, column2  
FROM table1, table2  
WHERE [ conditions ]  
GROUP BY column1, column2  
HAVING [ conditions ]  
ORDER BY column1, column2;
```

Inner Join

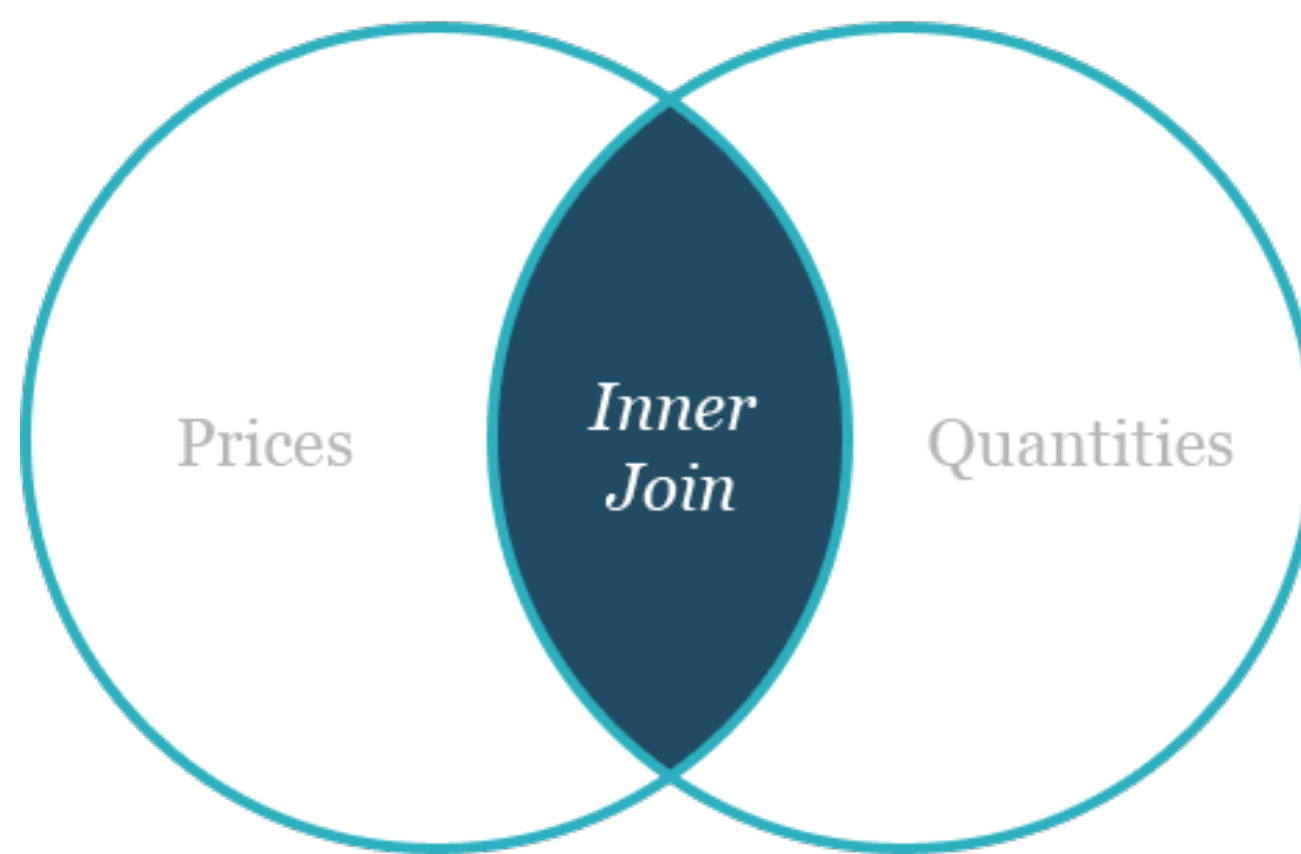


TABLE 1: PRICES

PRODUCT	PRICE
Potatoes	\$3
Avocados	\$4
Kiwis	\$2
Onions	\$1
Melons	\$5
Oranges	\$5
Tomatoes	\$6

TABLE 2: QUANTITIES

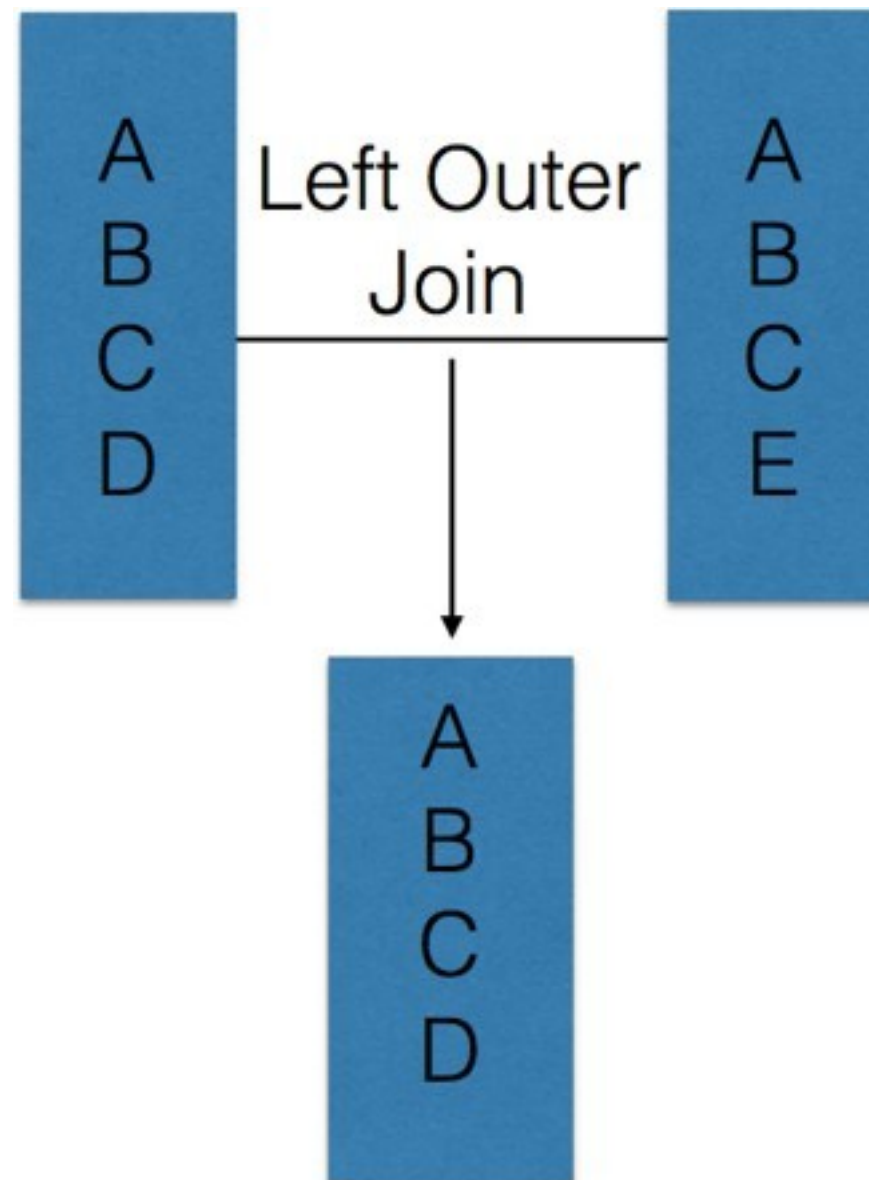
PRODUCT	QUANTITY
Potatoes	45
Avocados	63
Kiwis	19
Onions	20
Melons	66
Broccoli	27
Squash	92

```
SELECT Prices.*, Quantities.Quantity  
FROM Prices INNER JOIN Quantities  
ON Prices.Product = Quantities.Product;
```

QUERY RESULT FOR INNER JOIN

PRODUCT	PRICE	QUANTITY
Potatoes	\$3	45
Avocados	\$4	63
Kiwis	\$2	19
Onions	\$1	20
Melons	\$5	66

Left Outer Joins



Left Outer Join

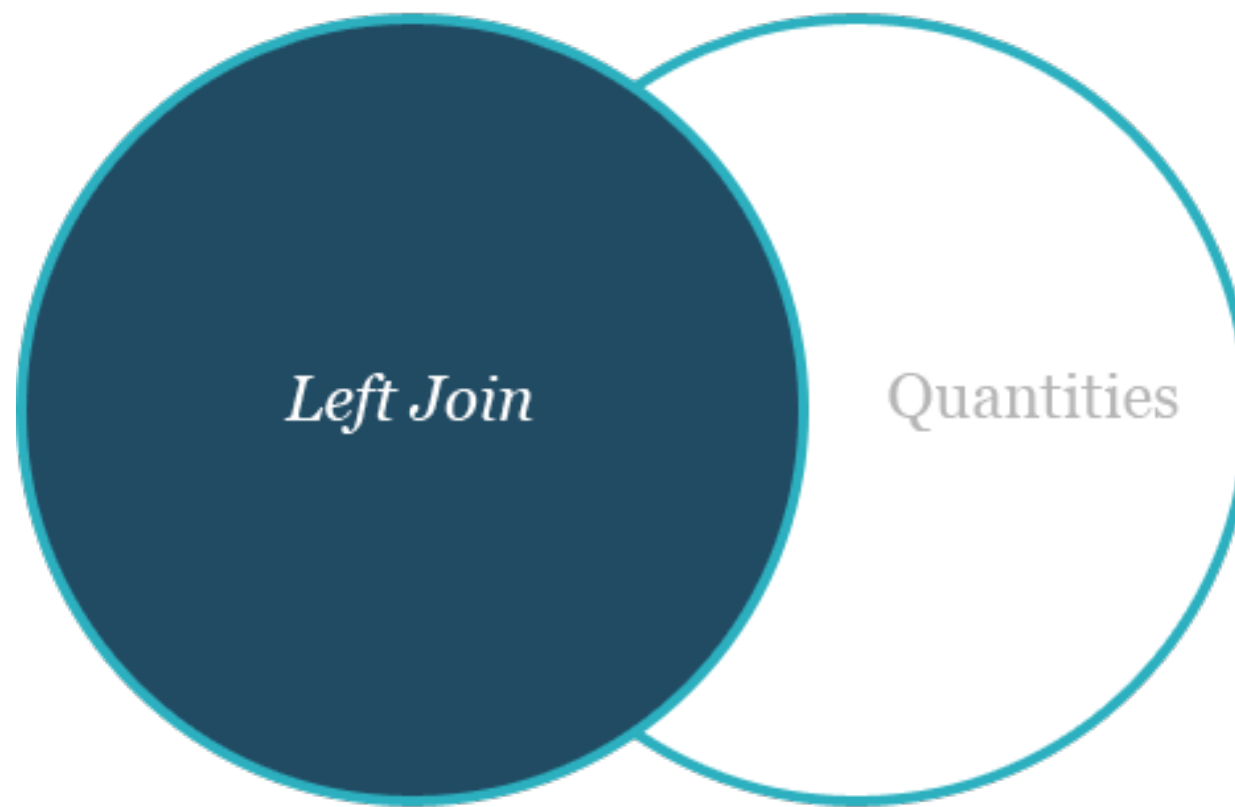


TABLE 1: PRICES

PRODUCT	PRICE
Potatoes	\$3
Avocados	\$4
Kiwis	\$2
Onions	\$1
Melons	\$5
Oranges	\$5
Tomatoes	\$6

TABLE 2: QUANTITIES

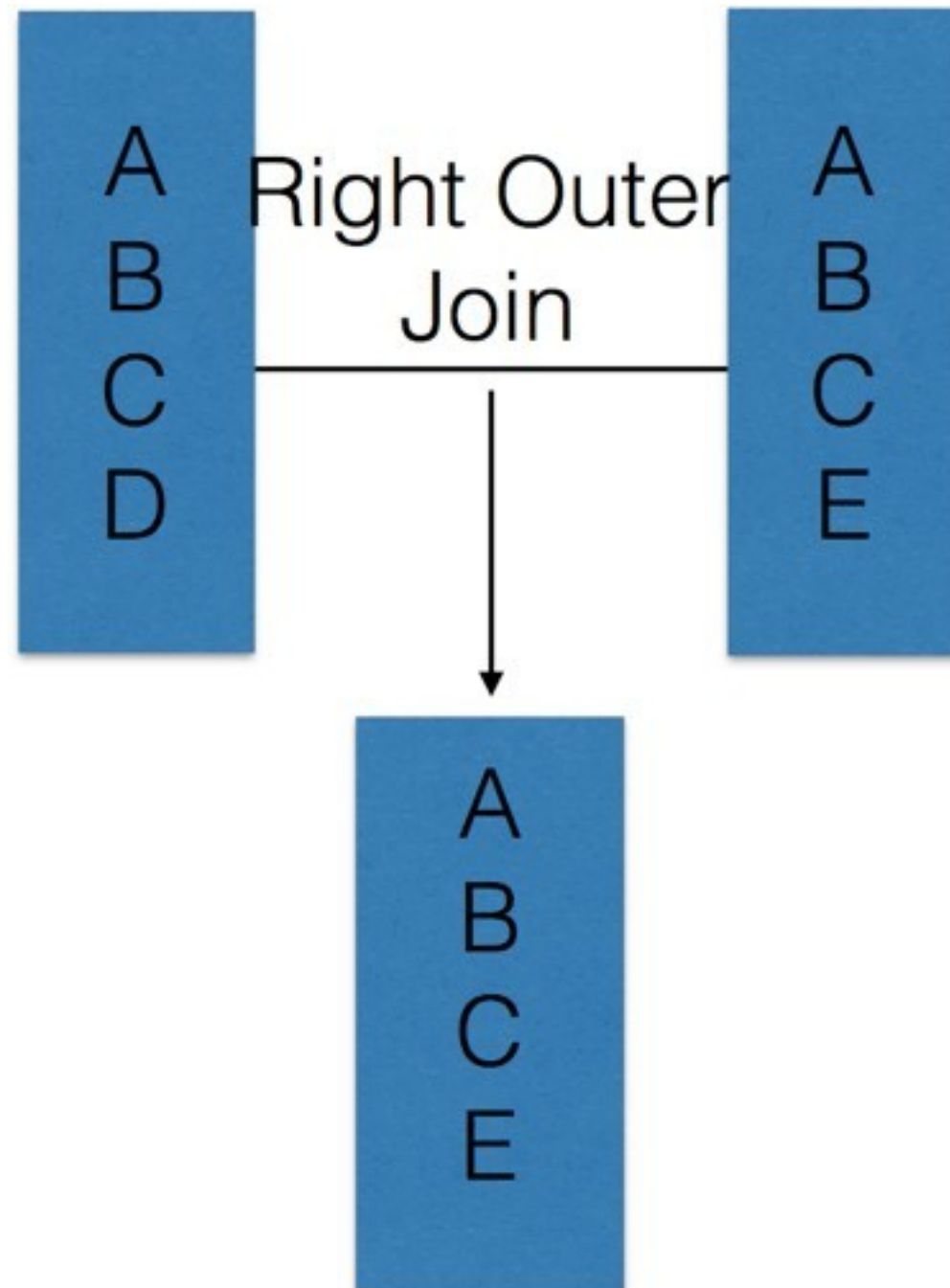
PRODUCT	QUANTITY
Potatoes	45
Avocados	63
Kiwis	19
Onions	20
Melons	66
Broccoli	27
Squash	92

```
SELECT Prices.*, Quantities.Quantity  
FROM Prices LEFT OUTER JOIN Quantities  
ON Prices.Product = Quantities.Product;
```

QUERY RESULT FOR LEFT OUTER JOIN

PRODUCT	PRICE	QUANTITY
Potatoes	\$3	45
Avocados	\$4	63
Kiwis	\$2	19
Onions	\$1	20
Melons	\$5	66
Oranges	\$5	NULL
Tomatoes	\$6	NULL

Right Outer Joins



Right Outer Join

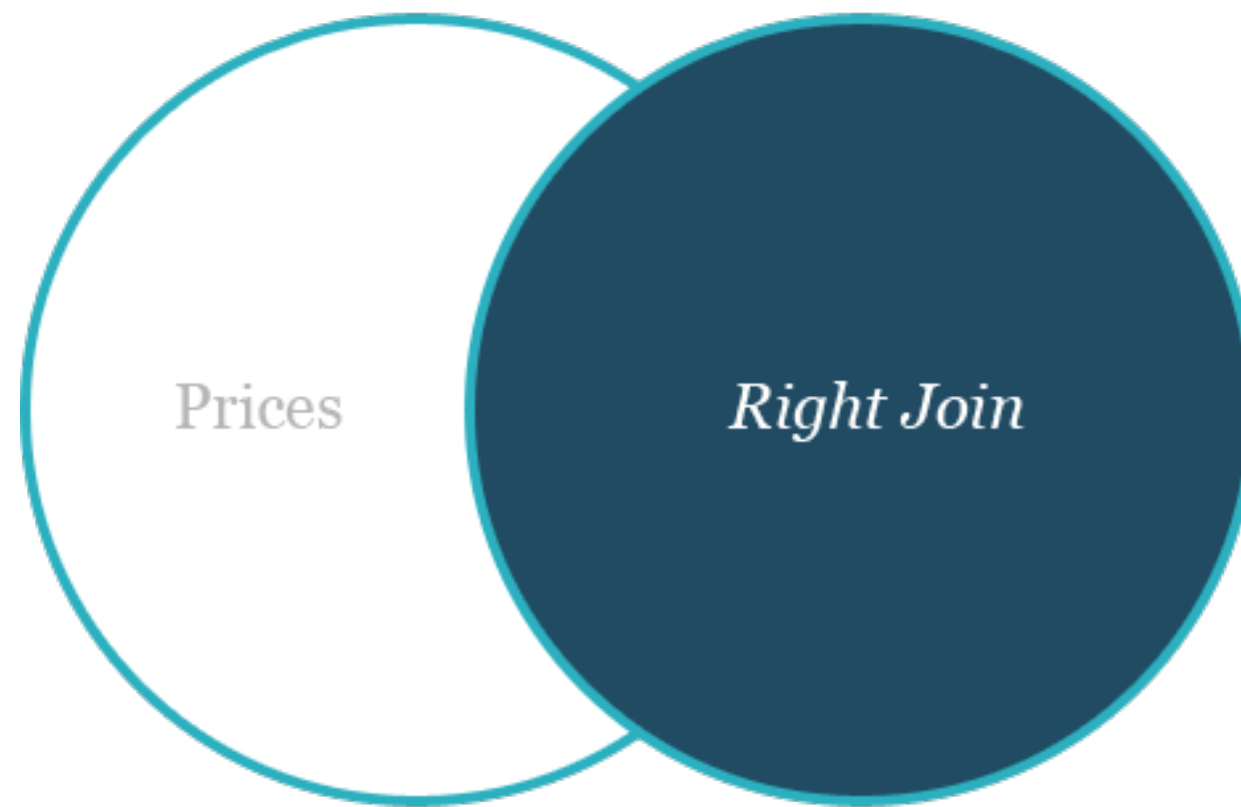
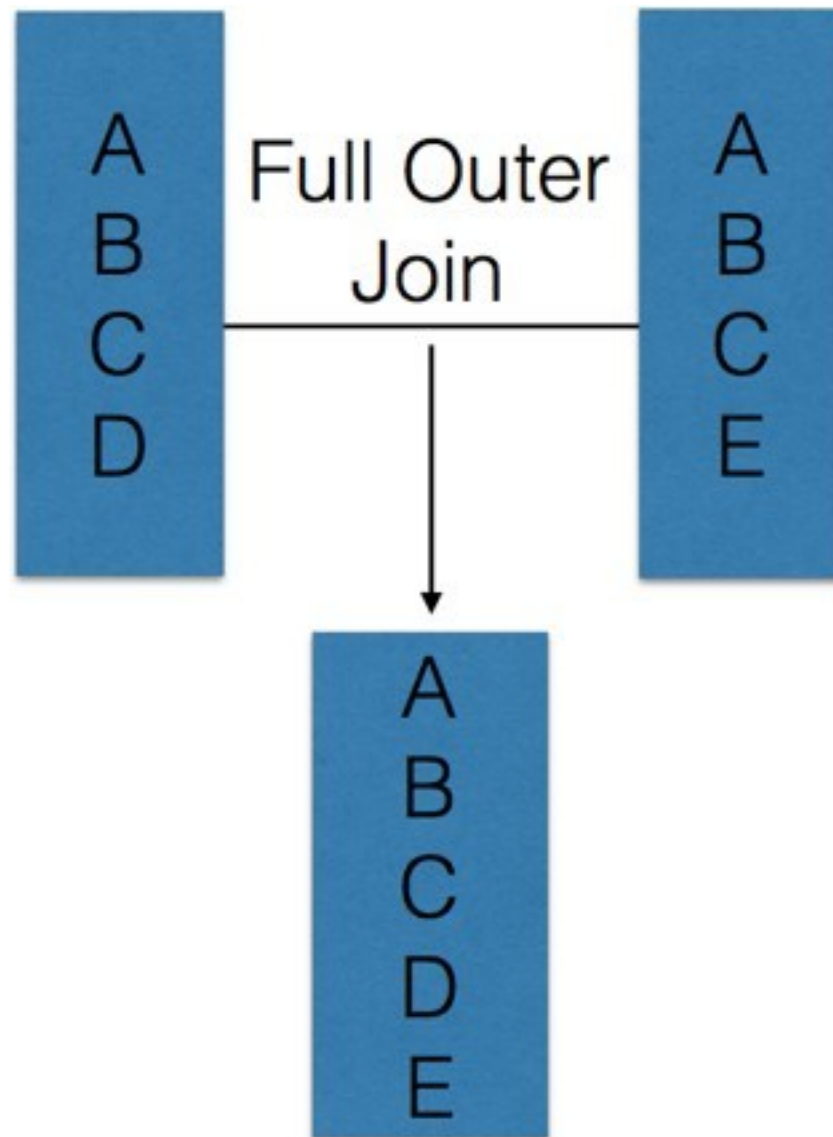


TABLE 1: PRICES		TABLE 2: QUANTITIES	
PRODUCT	PRICE	PRODUCT	QUANTITY
Potatoes	\$3	Potatoes	45
Avocados	\$4	Avocados	63
Kiwis	\$2	Kiwis	19
Onions	\$1	Onions	20
Melons	\$5	Melons	66
Oranges	\$5	Broccoli	27
Tomatoes	\$6	Squash	92

```
SELECT Prices.*, Quantities.Quantity
FROM Prices RIGHT OUTER JOIN Quantities
ON Prices.Product = Quantities.Product;
```

QUERY RESULT FOR RIGHT OUTER JOIN			
PRICE	PRODUCT	QUANTITY	
\$3	Potatoes	45	
\$4	Avocados	63	
\$2	Kiwis	19	
\$1	Onions	20	
\$5	Melons	66	
NULL	Broccoli	27	
NULL	Squash	92	

Full Outer Joins



Full Outer Join

Full Join

TABLE 1: PRICES

PRODUCT	PRICE
Potatoes	\$3
Avocados	\$4
Kiwis	\$2
Onions	\$1
Melons	\$5
Oranges	\$5
Tomatoes	\$6

TABLE 2: QUANTITIES

PRODUCT	QUANTITY
Potatoes	45
Avocados	63
Kiwis	19
Onions	20
Melons	66
Broccoli	27
Squash	92

```
SELECT Prices.*, Quantities.Quantity  
FROM Prices FULL OUTER JOIN Quantities  
ON Prices.Product = Quantities.Product;
```

QUERY RESULT FOR FULL OUTER JOIN

PRICES.PRODUCT	PRICE	QUANTITIES.PRODUCT	QUANTITY
Potatoes	\$3	Potatoes	45
Avocados	\$4	Avocados	63
Kiwis	\$2	Kiwis	19
Onions	\$1	Onions	20
Melons	\$5	Melons	66
Oranges	\$5	NULL	NULL
Tomatoes	\$6	NULL	NULL
NULL	NULL	Broccoli	27
NULL	NULL	Squash	92

SFWRENG 3DB3 Tutorial

Week 4

Oct 4, 2021

Levin Noronha

Select-From-Where

SELECT desired attributes

FROM one or more tables

WHERE condition about tuples of the tables

- Example schema
 - Beers(name, manf)
 - Bars(name, addr, license)
 - Drinkers(name, addr, phone)
 - Likes(drinker, beer)
 - Sells(bar, beer, price)
 - Frequents(drinker, bar)



Complex Conditions in WHERE Clause

- Boolean operators AND, OR, NOT
- Comparisons =, <>, <, >, <=, >=
- Example: Using `Sells(bar, beer, price)`, find the price Joe's Bar charges for Bud

```
SELECT price
FROM Sells
WHERE bar = 'Joe's Bar' AND beer = 'Bud' ;
```

- SQL includes a **between** comparison operator
- Example: Find the names of all instructors with salary between \$90,000 and \$100,000 (that is, `salary >= 90000 AND salary <= 100000`)

```
select name
from instructor
where salary between 90000 and 100000
```

LIKE operator

- Search for a pattern in a column of string values
 - WHERE column_name LIKE pattern
- Pattern can include wildcards like:
 - “%”: wildcard that represents 0 or more characters
 - “_”: wildcard that represents 1 character
 - and some other wildcards commonly found in regex
- For e.g., using Drinkers(name, address, phone), find drinkers whose names begins with “J” and ends with “n”
 - SELECT name
FROM Drinkers
WHERE name LIKE “J%n”

NULL values

- Columns use NULL in place of missing or inapplicable values
- Cannot use comparison operators to test for NULL
- Find drinkers who do not have an address
 - SELECT name
FROM Drinkers
WHERE address IS NULL
- Find drinkers who have an address
 - SELECT name
FROM Drinkers
WHERE address IS NOT NULL

Multi-relational Queries

- Combine data from multiple relations
- Distinguish attributes of the same name using “<relation>.<attribute>”

CROSS JOIN

- VERY EXPENSIVE. NOT COMMONLY USED.
- Produces cartesian product of two relations
- Number of rows in result = number of rows in first table * number of rows in second table
- For e.g.,

SELECT * FROM beers, drinkers

SELECT * FROM beers CROSS JOIN drinkers

name	manf
Bud	Budweiser
Bud Light	Coors

name	addr	phone
Barry	1 King St	123
John	2 Queen St	345



name	manf	name	addr	phone
Bud	Budweiser	Barry	1 King St	123
Bud Light	Coors	John	2 Queen St	345
Bud	Budweiser	Barry	1 King St	123
Bud Light	Coors	John	2 Queen St	345

EQUI JOIN

- JOIN tables Likes and Frequent to find each drinker's favourite drink and bar
- Add a WHERE clause

```
SELECT l.drinker, l.beer, f.bar
FROM likes l, frequent f
WHERE l.drinker = f.drinker
```

```
SELECT l.drinker, l.beer, f.bar
FROM likes l JOIN frequent f
ON l.drinker = f.drinker
```

- Note: JOIN is interchangeable with INNER JOIN

Likes		Frequent	
drinker	beer	drinker	bar
Barry	Bud	Barry	Joe's
John	Bud Light	John	Callaghan's
		Mary	Joe's



l.drinker	l.beer	f.drinker	f.bar
Barry	Bud	Barry	Joe's
John	Bud Light	John	Callaghan's



l.drinker	l.beer	f.bar
Barry	Bud	Joe's
John	Bud Light	Callaghan's

Subqueries

- Using `Sells(bar, beer, price)`, find the bars that serve Miller for the same price Joe charges for Bud

```
SELECT bar
```

```
FROM Sells
```

```
WHERE beer = 'Miller' AND price
```

- Find the price Joe charges for Bud.
- Find the bars that serve Miller at that price.

`Sells(bar, beer, price)`

```
= (SELECT price  
   FROM Sells  
   WHERE bar = 'Joe's Bar'  
   AND beer = 'Bud');
```

The price at
which Joe
sells Bud



ANY and ALL operators

- Perform comparison between a single value and a list of values
- Can be used with WHERE, and HAVING
- Syntax:
 - ```
SELECT ...
FROM table
WHERE column_name COMPARE_OP ANY/ALL
 (SELECT column_name
 FROM ...
 WHERE ...)
```
- ANY returns TRUE if any of subquery values satisfy the condition
- ALL returns TRUE if all of subquery values satisfy the condition



# ANY operator

- Using Sells(bar, beer, price) and Likes(drinker, beer), find drinkers that like beers sold at price greater than 10

```
SELECT DISTINCT drinker
FROM Likes
WHERE beer = ANY
 (SELECT DISTINCT beer
 FROM Sells
 WHERE price > 10)
```

# ALL operator

- Using Sells(bar, beer, price) and Beers(name, manf), find the beer manufacturer(s) whose beer(s) sell at the highest price

```
SELECT DISTINCT manf
```

```
FROM Sells s INNER JOIN Beers b ON s.beer = b.name
```

```
WHERE price >= ALL
```

```
 (SELECT price
```

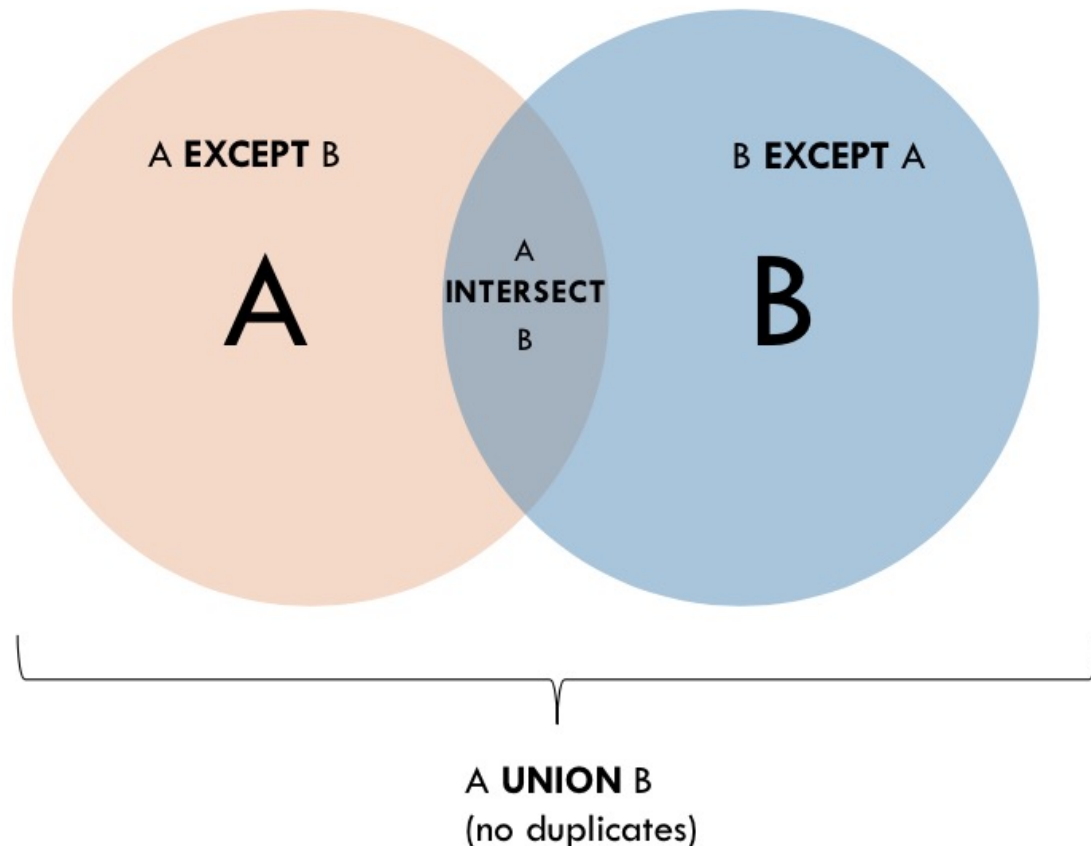
```
 FROM Sells)
```

# IN operator

- Allows multiple values to be specified in WHERE clause
- WHERE col IN (value1, value2, ...)
- WHERE col IN (SELECT col FROM ...)
- Using Drinkers(name, addr, phone) and  
Frequents(drinker, bar), find drinkers who do not  
frequent bars
  - SELECT name  
FROM Drinkers  
WHERE name NOT IN (Select drinker FROM Frequents)

# Union, Intersection, Difference

- ( $\langle \text{subquery} \rangle$ ) UNION/INTERSECT/EXCEPT ( $\langle \text{subquery} \rangle$ )



Note: requires both relations to have the same number of columns with compatible types

| name      | manf      |
|-----------|-----------|
| Bud       | Budweiser |
| Bud Light | Budweiser |
| Coors     | Coors     |
| Heineken  | Heineken  |

INTERSECT

| name      | manf      |
|-----------|-----------|
| Bud       | Budweiser |
| Bud Light | Budweiser |

=

| name      | manf      |
|-----------|-----------|
| Bud       | Budweiser |
| Bud Light | Budweiser |

| name      | manf      |
|-----------|-----------|
| Bud       | Budweiser |
| Bud Light | Budweiser |
| Coors     | Coors     |
| Heineken  | Heineken  |

EXCEPT

| name      | manf      |
|-----------|-----------|
| Bud       | Budweiser |
| Bud Light | Budweiser |

=

| name     | manf     |
|----------|----------|
| Coors    | Coors    |
| Heineken | Heineken |

| name      | manf      |
|-----------|-----------|
| Bud       | Budweiser |
| Bud Light | Budweiser |
| Coors     | Coors     |
| Heineken  | Heineken  |

UNION

| name  | manf      |
|-------|-----------|
| Bud   | Budweiser |
| Boxer | Boxer     |

=

| name      | manf      |
|-----------|-----------|
| Bud       | Budweiser |
| Bud Light | Budweiser |
| Coors     | Coors     |
| Heineken  | Heineken  |
| Boxer     | Boxer     |

Note: to preserve duplicates, use UNION ALL instead.

# EXCEPT example

Note: EXCEPT requires both relations to have the same number of columns with compatible types

- Find beer manufacturers that only sell one beer

(SELECT \* FROM Beers)

EXCEPT

(SELECT b1.name, b2.manf FROM Beers b1, Beers b2

WHERE b1.manf = b2.manf AND b1.name <> b2.name)

Beers

| name      | manf      |
|-----------|-----------|
| Bud       | Budweiser |
| Bud Light | Budweiser |
| Coors     | Coors     |
| Heineken  | Heineken  |

EXCEPT

| name      | manf      |
|-----------|-----------|
| Bud       | Budweiser |
| Bud Light | Budweiser |

=

| name     | manf     |
|----------|----------|
| Coors    | Coors    |
| Heineken | Heineken |



# SE3DB3 TUTORIAL

Lucia Cristiano  
Oct 18-20, 2021

# Assignment 1

A1 marks are up on Avenue. I was the TA who marked it please contact me if you have any questions

Lucia Cristiano

- Email: [cristial@mcmaster.ca](mailto:cristial@mcmaster.ca)
- Office hours: Tue 3:00pm – 4:00pm on MS Teams

# Outline

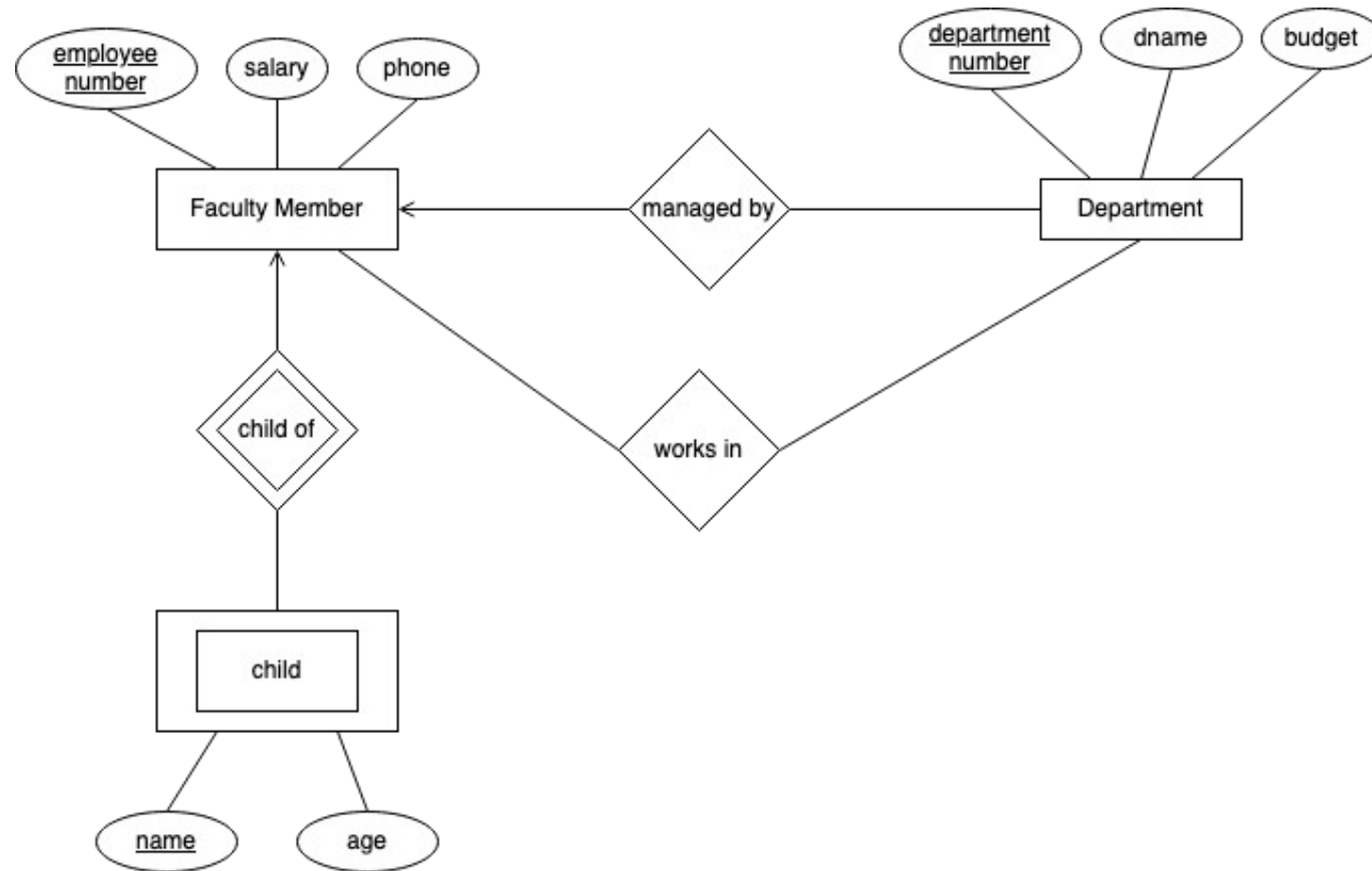
- ERD question
- DDL SQL Statements
- Create a SQL query
- What does this query return?
- Contact

# ERD Question

University database that stores info about faculty member with employee number, address and phone, departments with department number, department name, and budget, as well as children of employees with name and age. Faculty work in departments. Each department is managed by a faculty member. A child is identified by name when parent is known. Assume that only 1 parent works for the university.

# ERD Question

**ANSWER**



# SQL DDL Questions

Employee(eid, ename, age, salary)

Works(eid, did, pcttime)

Department(did, dname, budget, managerid)

Write a create table statement so that every department has a manager

# SQL DDL Questions

Write a create table statement so that every department has a manager

## **ANSWER**

```
CREATE TABLE Department(
 did INT NOT NULL PRIMARY KEY,
 dname VARCHAR(20),
 budget REAL,
 managerId INT NOT NULL REFERENCES Employee);
```

# SQL DDL Questions

Employee(eid, ename, age, salary)

Works(eid, did, pcttime)

Department(did, dname, budget, managerid)

Write a statement to delete employee with eid = 5.



# SQL DDL Questions

Write a statement to delete employee with eid = 5.

## **ANSWER**

```
DELETE FROM Employee
WHERE eid = 5;
```

# SQL DDL Questions

Employee(eid, ename, age, salary)

Works(eid, did, pcttime)

Department(did, dname, budget, managerid)

Write a statement to add department finance with did = 10, budget = 30000 and managerid = 102

# SQL DDL Questions

Write a statement to add department finance with did = 10, budget = 30000 and managerid = 102

## **ANSWER**

```
INSERT INTO Department(did, dname, budget,managerid)
VALUES(10, 'Fianance', 30000, 102);
```

# SQL DDL Questions

Consider the following CREATE TABLE definition:

```
CREATE TABLE Midterm
(A INT NOT NULL,
 B INT NOT NULL,
 C INT NOT NULL,
 PRIMARY KEY (A),
 FOREIGN KEY (B) REFERENCES Midterm(A) ON DELETE CASCADE ON UPDATE CASCADE,
 FOREIGN KEY (C) REFERENCES Midterm(A) ON DELETE CASCADE ON UPDATE RESTRICT)
```

Consider the following instance table Midterm:

| A | B | C |
|---|---|---|
| 4 | 3 | 3 |
| 3 | 4 | 3 |

a) **What is the result** of the following statement:

```
UPDATE Midterm
SET B = B+1
WHERE B in (SELECT A FROM Midterm)
```

# SQL DDL Questions

Consider the following CREATE TABLE definition:

```
CREATE TABLE Midterm
(A INT NOT NULL,
 B INT NOT NULL,
 C INT NOT NULL,
 PRIMARY KEY (A),
 FOREIGN KEY (B) REFERENCES Midterm(A) ON DELETE CASCADE ON UPDATE CASCADE,
 FOREIGN KEY (C) REFERENCES Midterm(A) ON DELETE CASCADE ON UPDATE RESTRICT)
```

Consider the following instance table Midterm:

| A | B | C |
|---|---|---|
| 4 | 3 | 3 |
| 3 | 4 | 3 |

a) **What is the result** of the following statement:

```
UPDATE Midterm
SET B = B+1
WHERE B in (SELECT A FROM Midterm)
```

## ANSWER

This results in an error as the foreign key constraint is violated.

# Create a SQL Query

- Write a query that finds the numbers, names, and ages of employees who earn more than 40k
  - Schema: Employee(Number, Name, Age, Salary)

# Create a SQL Query

- Write a query that finds the numbers, names, and ages of employees who earn more than 40k
  - Schema: Employee(Number, Name, Age, Salary)

## **ANSWER**

```
SELECT Number, Name, Ages
FROM Employee
WHERE Salary > 40,000
```

# Create a SQL Query

- Write a query that finds beer and average price for each beer.
  - Schema: Sells(bar, beer, price)



# Create a SQL Query

- Write a query that finds beer and average price for each beer.
  - Schema: Sells(bar, beer, price)

## **ANSWER**

```
SELECT beer, AVG(price)
FROM Sells
GROUP BY beer
```

# Create a SQL Query

3. Find the names of all classes that either meet in room R128 or have five or more students enrolled.

Student(snum: integer, sname: string, major: string, level: string, age: integer)

Class(name: string, meets\_at: time, room: string, fid: integer)

Enrolled(snum: integer, cname: string)

Faculty(fid: integer, fname: string, deptid: integer)

# Create a SQL Query

3. Find the names of all classes that either meet in room R128 or have five or more students enrolled.

```
SELECT C.name
FROM Class C
WHERE C.room = 'R128'
 OR C.name IN (SELECT E.cname
 FROM Enrolled E
 GROUP BY E.cname
 HAVING COUNT (*) >= 5)
```

# What does this query return?

```
SELECT DISTINCT R.A
FROM R
WHERE R.A NOT IN (
 SELECT DISTINCT S.B AS A
 FROM S
 WHERE S.B = S.C);
```

R

| A | B |
|---|---|
| 1 | 2 |
| 3 | 4 |
| 1 | 3 |

S

| B | C |
|---|---|
| 1 | 3 |
| 2 | 4 |

# What does this query return?

```
SELECT DISTINCT R.A
FROM R
WHERE R.A NOT IN (
 SELECT DISTINCT S.B AS A
 FROM S
 WHERE S.B = S.C);
```

**ANSWER**

| A |
|---|
| 1 |
| 3 |

R

| A | B |
|---|---|
| 1 | 2 |
| 3 | 4 |
| 1 | 3 |

S

| B | C |
|---|---|
| 1 | 3 |
| 2 | 4 |

# What does this query return?

```
SELECT DISTINCT R.A, S.C, avg(R.B), as av
FROM R, S
WHERE R.B < 4
GROUP BY R.A S.C
HAVING max(R.B) >= 2;
```

R

| A | B |
|---|---|
| 1 | 2 |
| 3 | 4 |
| 1 | 3 |

S

| B | C |
|---|---|
| 1 | 3 |
| 2 | 4 |

# What does this query return?

R

| A | B |
|---|---|
| 1 | 2 |
| 3 | 4 |
| 1 | 3 |

S

| B | C |
|---|---|
| 1 | 3 |
| 2 | 4 |

Then remove the values  
of  $R.B < 4$   
(WHERE clause)

First Join R and S on B

| A | R.B | S.B | C |
|---|-----|-----|---|
| 1 | 2   | 1   | 3 |
| 1 | 2   | 2   | 4 |
| 3 | 4   | 1   | 3 |
| 3 | 4   | 2   | 4 |
| 1 | 3   | 1   | 3 |
| 1 | 3   | 2   | 4 |

| A | R.B | S.B | C |
|---|-----|-----|---|
| 1 | 2   | 1   | 3 |
| 1 | 2   | 2   | 4 |
| 1 | 3   | 1   | 3 |
| 1 | 3   | 2   | 4 |

Group by R.A and R.C

| A | R.B | S.B | C |
|---|-----|-----|---|
| 1 | 2   | 1   | 3 |
| 1 | 2   | 1   | 3 |
| 1 | 3   | 2   | 4 |
| 1 | 3   | 2   | 4 |

Final table

| A | C | av  |
|---|---|-----|
| 1 | 3 | 2.5 |
| 1 | 4 | 2.5 |

# Contact

If you have any questions or feedback, please email me or attend my office hours:

Lucia Cristiano

- Email: [cristial@mcmaster.ca](mailto:cristial@mcmaster.ca)
- Office hours: Tue 3:00pm – 4:00pm on MS Teams