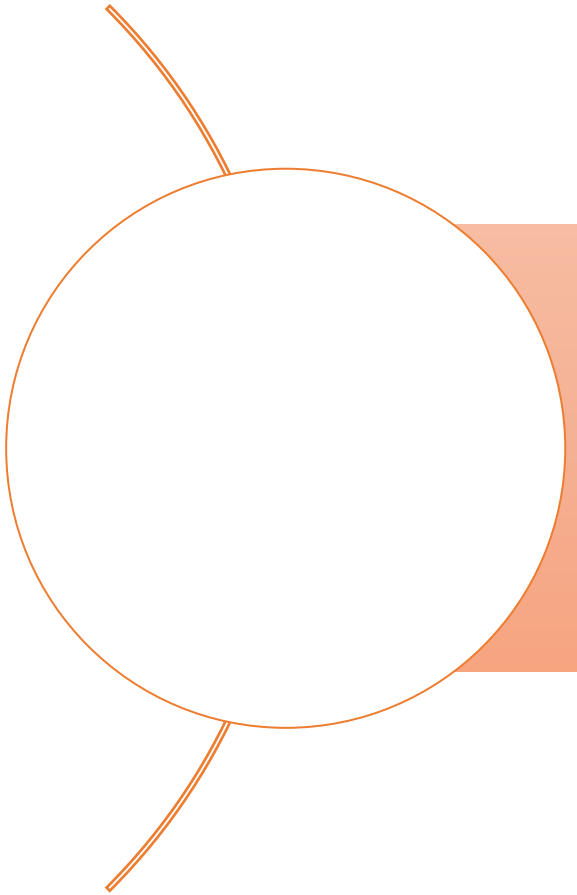
DBMS

Lab-4

Things we will complete today



Query on single relation (continue)

MySQL Comparison Operators

MySQL Comparison Operators

Comparison Operations

Comparison operators are used in the WHERE clause to determine which records to select.

```
1  SELECT
2      column1, column2, columnN
3  FROM
4      table_name
5  [WHERE condition];
```

Comparison Operator	Description
=	Equal
<=>	Equal (Safe to compare NULL values)
<>	Not Equal
!=	Not Equal
>	Greater Than
>=	Greater Than or Equal
<	Less Than
<=	Less Than or Equal
IN ( )	Matches a value in a list
NOT	Negates a condition
BETWEEN	Within a range (inclusive)
IS NULL	NULL value
IS NOT NULL	Non-NULL value
LIKE	Pattern matching with % and _
EXISTS	Condition is met if subquery returns at least one row

The = operator

Example:

contact_id	last_name	website1	website2
1	Johnson	techonthenet.com	<NULL>
2	Anderson	<NULL>	<NULL>
3	Smith	TBD	TDB
4	Jackson	checkyourmath.com	digminecraft.com

```
SELECT *
FROM contacts
WHERE website1 = website2;
```

Because we used the `=` operator, we would get the following results:

contact_id	last_name	website1	website2
3	Smith	TBD	TDB

The <=> operator

Example:

contact_id	last_name	website1	website2
1	Johnson	techonthenet.com	<NULL>
2	Anderson	<NULL>	<NULL>
3	Smith	TBD	TDB
4	Jackson	checkyourmath.com	digminecraft.com

```
SELECT *
FROM contacts
WHERE website1 <=>website2;
```

Because we used the `<=>` operator, we would get the following results:

contact_id	last_name	website1	website2
2	Anderson	<NULL>	<NULL>
3	Smith	TBD	TDB

The Inequality Operator <> and !=

contact_id	last_name	website1	website2
1	Johnson	techonthenet.com	<NULL>
2	Anderson	<NULL>	<NULL>
3	Smith	TBD	TDB
4	Jackson	checkyourmath.com	digminecraft.com

Example:

```
SELECT *
FROM contacts
WHERE last_name <> 'Johnson';
```

```
SELECT *
FROM contacts
WHERE last_name != 'Johnson';
```

contact_id	last_name	website1	website2
2	Anderson	<NULL>	<NULL>
3	Smith	TBD	TDB
4	Jackson	checkyourmath.com	digminecraft.com

08/30/2020 19:04

The >, >=, < and <=

contact_id	last_name	website1	website2
1	Johnson	techonthenet.com	<NULL>
2	Anderson	<NULL>	<NULL>
3	Smith	TBD	TDB
4	Jackson	checkyourmath.com	digminecraft.com

Example:

```
SELECT *
FROM contacts
WHERE contact_id > 50;
```

```
SELECT *
FROM contacts
WHERE contact_id >= 50;
```

```
SELECT *
FROM inventory
WHERE product_id < 300;
```

```
SELECT *
FROM inventory
WHERE product_id <= 300;
```

The in(...) operator

The IN operator allows you to specify multiple values in a WHERE clause.

The IN operator is a shorthand for multiple OR conditions.

```
SELECT column_name(s)
FROM table_name
WHERE column_name = value-1 or column_name=value-2, ...column_name=value-n);
```

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value-1, value-2, ... value-n);
```

The in(…) operator

selects the names of instructors whose names are either “Mozart” or “Einstein”

```
1 SELECT *
2 FROM instructor
3 WHERE `name` IN ('Mozart', 'Einstein');
```

instructor (4×2)

ID	name	dept_name	salary
15151	Mozart	Music	40,000.00
22222	Einstein	Physics	95,000.00

university_db.instructor: 12 rows total

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65,000.00
12121	Wu	Finance	90,000.00
15151	Mozart	Music	40,000.00
22222	Einstein	Physics	95,000.00
32343	El Said	History	60,000.00
33456	Gold	Physics	87,000.00
45565	Katz	Comp. Sci.	75,000.00
58583	Califieri	History	62,000.00
76543	Singh	Finance	80,000.00
76766	Crick	Biology	(NULL)
83821	Brandt	Comp. Sci.	92,000.00
98345	Kim	Elec. Eng.	80,000.00

The in(…) operator

Using ‘in’ selects the names of instructors whose ids are less than 30000

```
1 SELECT *
2 FROM instructor
3 WHERE ID IN (SELECT id FROM instructor WHERE id < 30000);
```

instructor (4×4)

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65,000.00
12121	Wu	Finance	90,000.00
15151	Mozart	Music	40,000.00
22222	Einstein	Physics	95,000.00

university_db.instructor: 12 rows total

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65,000.00
12121	Wu	Finance	90,000.00
15151	Mozart	Music	40,000.00
22222	Einstein	Physics	95,000.00
32343	El Said	History	60,000.00
33456	Gold	Physics	87,000.00
45565	Katz	Comp. Sci.	75,000.00
58583	Califieri	History	62,000.00
76543	Singh	Finance	80,000.00
76766	Crick	Biology	(NULL)
83821	Brandt	Comp. Sci.	92,000.00
98345	Kim	Elec. Eng.	80,000.00

The not in(…) operator

selects the names of instructors whose names are neither “Mozart” nor “Einstein”

```
1 SELECT *
2 FROM instructor
3 WHERE `name` NOT IN ('Mozart', 'Einstein');
```

instructor (4×10)

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65,000.00
12121	Wu	Finance	90,000.00
32343	El Said	History	60,000.00
33456	Gold	Physics	87,000.00
45565	Katz	Comp. Sci.	75,000.00
58583	Califieri	History	62,000.00
76543	Singh	Finance	80,000.00
76766	Crick	Biology	(NULL)
83821	Brandt	Comp. Sci.	92,000.00
98345	Kim	Elec. Eng.	80,000.00

university_db.instructor: 12 rows total

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65,000.00
12121	Wu	Finance	90,000.00
15151	Mozart	Music	40,000.00
22222	Einstein	Physics	95,000.00
32343	El Said	History	60,000.00
33456	Gold	Physics	87,000.00
45565	Katz	Comp. Sci.	75,000.00
58583	Califieri	History	62,000.00
76543	Singh	Finance	80,000.00
76766	Crick	Biology	(NULL)
83821	Brandt	Comp. Sci.	92,000.00
98345	Kim	Elec. Eng.	80,000.00

The **between** operator

between comparison operator is used to simplify **where** clauses that specify that a value be less than or equal to some value and greater than or equal to some other value

university_db.instructor: 12 rows total

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65,000.00
12121	Wu	Finance	90,000.00
15151	Mozart	Music	40,000.00
22222	Einstein	Physics	95,000.00
32343	El Said	History	60,000.00
33456	Gold	Physics	87,000.00
45565	Katz	Comp. Sci.	75,000.00
58583	Califieri	History	62,000.00
76543	Singh	Finance	80,000.00
76766	Crick	Biology	72,000.00
83821	Brandt	Comp. Sci.	92,000.00
98345	Kim	Elec. Eng.	80,000.00

The **between** operator

find the names of instructors with salary amounts between $90,000 and $100,000

```
select name
from instructor
where salary <= 100000 and salary >= 90000;
```

```
1 SELECT `name`, salary
2 FROM instructor
3 WHERE salary BETWEEN 90000 AND 100000;
```

instructor (2×3)

NAME	salary
Wu	90,000.00
Einstein	95,000.00
Brandt	92,000.00

university_db.instructor: 12 rows total

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65,000.00
12121	Wu	Finance	90,000.00
15151	Mozart	Music	40,000.00
22222	Einstein	Physics	95,000.00
32343	El Said	History	60,000.00
33456	Gold	Physics	87,000.00
45565	Katz	Comp. Sci.	75,000.00
58583	Califieri	History	62,000.00
76543	Singh	Finance	80,000.00
76766	Crick	Biology	72,000.00
83821	Brandt	Comp. Sci.	92,000.00
98345	Kim	Elec. Eng.	80,000.00

The **is null** operator

Null values present special problems in relational operations, including arithmetic operations, comparison operations, and set operations.

The result of an arithmetic expression (involving, for example +, −, ∗, or /) is null if any of the input values is null. For example, if a query has an expression

- **and**: The result of *true* **and** *unknown* is *unknown*, *false* **and** *unknown* is *false*, while *unknown* **and** *unknown* is *unknown*.
- **or**: The result of *true* **or** *unknown* is *true*, *false* **or** *unknown* is *unknown*, while *unknown* **or** *unknown* is *unknown*.
- **not**: The result of **not** *unknown* is *unknown*.

university_db.instructor: 12 rows total

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65,000.00
12121	Wu	Finance	90,000.00
15151	Mozart	Music	40,000.00
22222	Einstein	Physics	95,000.00
32343	El Said	History	60,000.00
33456	Gold	Physics	87,000.00
45565	Katz	Comp. Sci.	75,000.00
58583	Califieri	History	62,000.00
76543	Singh	Finance	80,000.00
76766	Crick	Biology	(NULL)
83821	Brandt	Comp. Sci.	92,000.00
98345	Kim	Elec. Eng.	80,000.00

The **is null** operator

SQL uses the special keyword **null** in a predicate to test for a null value

Find the names of instructors whose salary is unknown.

```
1 SELECT * FROM instructor
2 WHERE salary = NULL
```

instructor (4×0)

ID	name	dept_name	salary

```
1 SELECT * FROM instructor
2 WHERE salary IS NULL
3
4
```

instructor (4×1)

ID	name	dept_name	salary
76766	Crick	Biology	(NULL)

university_db.instructor: 12 rows total

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65,000.00
12121	Wu	Finance	90,000.00
15151	Mozart	Music	40,000.00
22222	Einstein	Physics	95,000.00
32343	El Said	History	60,000.00
33456	Gold	Physics	87,000.00
45565	Katz	Comp. Sci.	75,000.00
58583	Califieri	History	62,000.00
76543	Singh	Finance	80,000.00
76766	Crick	Biology	(NULL)
83821	Brandt	Comp. Sci.	92,000.00
98345	Kim	Elec. Eng.	80,000.00

The **is not null** operator

The opposite of **is not null**

Find the names of instructors whose salary we know.

```
1 SELECT * FROM instructor
2 WHERE salary IS NOT NULL
```

instructor (4×11)

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65,000.00
12121	Wu	Finance	90,000.00
15151	Mozart	Music	40,000.00
22222	Einstein	Physics	95,000.00
32343	El Said	History	60,000.00
33456	Gold	Physics	87,000.00
45565	Katz	Comp. Sci.	75,000.00
58583	Califieri	History	62,000.00
76543	Singh	Finance	80,000.00
83821	Brandt	Comp. Sci.	92,000.00
98345	Kim	Elec. Eng.	80,000.00

university_db.instructor: 12 rows total

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65,000.00
12121	Wu	Finance	90,000.00
15151	Mozart	Music	40,000.00
22222	Einstein	Physics	95,000.00
32343	El Said	History	60,000.00
33456	Gold	Physics	87,000.00
45565	Katz	Comp. Sci.	75,000.00
58583	Califieri	History	62,000.00
76543	Singh	Finance	80,000.00
76766	Crick	Biology	(NULL)
83821	Brandt	Comp. Sci.	92,000.00
98345	Kim	Elec. Eng.	80,000.00

The like operator

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column

```
SELECT column1, column2, ...
FROM table_name
WHERE columnN LIKE pattern;
```

There are two wildcards often used in conjunction with the LIKE operator:

- % - The percent sign represents zero, one, or multiple characters
- _ - The underscore represents a single character

The like operator

Find the names of all courses whose title includes the substring '**to**'.

```
1 SELECT * FROM course
2 WHERE title LIKE '%to%'
3
```

course (4×4)

course_id	title	dept_name	credits
BIO-101	Intro. to Biology	Biology	4
CS-101	Intro. to Computer Science	Comp. Sci.	4
EE-181	Intro. to Digital Systems	Elec. Eng.	3
HIS-351	World History	History	3

university_db.course: 13 rows total

course_id	title	dept_name	credits
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

The like operator

- 'Intro%' matches any string beginning with "Intro".
- '%Comp%' matches any string containing "Comp" as a substring, for example, 'Intro. to Computer Science', and 'Computational Biology'.
- '___' matches any string of exactly three characters.
- '___%' matches any string of at least three characters.

university_db.course: 13 rows total

course_id	title	dept_name	credits
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

The like operator

LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	
WHERE CustomerName LIKE '%a'	
WHERE CustomerName LIKE '%or%'	
WHERE CustomerName LIKE '_r%'	
WHERE CustomerName LIKE 'a_%'	
WHERE CustomerName LIKE 'a__%'	
WHERE ContactName LIKE 'a%o'	

The exists operator

The EXISTS operator is used to test for the existence of any record in a subquery.

The EXISTS operator returns true if the subquery returns one or more records

```
SELECT column_name(s)
FROM table_name
WHERE EXISTS
(SELECT column_name FROM table_name WHERE condition);
```

The exists operator

The following SQL statement returns TRUE and lists the suppliers with a product price less than 20

```
SELECT
    SupplierName
FROM
    Suppliers
WHERE
    EXISTS (SELECT ProductName
            FROM Products
            WHERE
                Products.SupplierID = Suppliers.supplierID
                AND
                Price < 20
            );
```

Result:

Number of Records: 1

SupplierName
New Orleans Cajun Delights

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35

SupplierID	SupplierName	ContactName	Address	City	PostalCode	Country
1	Exotic Liquid	Charlotte Cooper	49 Gilbert St.	London	EC1 4SD	UK
2	New Orleans Cajun Delights	Shelley Burke	P.O. Box 78934	New Orleans	70117	USA
3	Grandma Kelly's Homestead	Regina Murphy	707 Oxford Rd.	Ann Arbor	48104	USA
4	Tokyo Traders	Yoshi Nagase	9-8 Sekimai Musashino-shi	Tokyo	100	Japan

08/31/2020 13:06

Demo.

End.