Introduction to SQL

Thomas Bittner
bittner3@buffalo.edu

session 2

recap

the SELECT command

The `SELECT` statement is used to pull information from a table.

```
SELECT what_to_select
FROM which_table
WHERE conditions_to_satisfy;
```

what_to_select indicates what you want to see. This can be a list of columns, or * to indicate "all columns."

which_table indicates the table from which you want to retrieve data.

The `WHERE` clause is optional. If it is present, *conditions_to_satisfy* specifies one or more conditions that rows must satisfy to qualify for retrieval.

exercise

- list complete table pet
- list the column 'owner'
- list the columns 'name' and 'owner'
- list rows where 'name' is 'Bowser'
- list rows where name is Bowser or Claws
- list the columns name and owner in rows where name is Bowser or Claws
- which animals were born during or after 1998
- list all female dogs
- list all male cats and female dogs

exercise

- list complete table: `select * from pet;`
- list the column 'owner': `select owner from pet;`
- list the columns 'name' and 'owner': `select name, owner from pet;`
- list rows where 'name' is 'Bowser': `select * from pet WHERE name = 'Bowser';`
- list rows where name is Bowser or Claws: `select * from pet WHERE name = 'Bowser' or name = 'Claws';`
- list the columns name and owner in rows where name is Bowser or Claws: `select name, owner from pet WHERE name = 'Bowser' or name = 'Claws';`

exercises

list all female dogs

```
sql> SELECT * FROM pet WHERE species = 'dog' AND sex = 'f';

+-------+--------+---------+------+------------+-------+
| name  | owner  | species | sex  | birth      | death |
+-------+--------+---------+------+------------+-------+
| Buffy | Harold | dog     | f    | 1989-05-13 | NULL  |
+-------+--------+---------+------+------------+-------+
1 row in set (0.00 sec)
```

exercises

list all male cats and female dogs

```
sql> SELECT * FROM pet WHERE (species = 'cat' AND sex = 'm')
     OR (species = 'dog' AND sex = 'f');
+-------+--------+---------+------+------------+-------+
| name  | owner  | species | sex  | birth      | death |
+-------+--------+---------+------+------------+-------+
| Claws | Gwen   | cat     | m    | 1994-03-17 | NULL  |
| Buffy | Harold | dog     | f    | 1989-05-13 | NULL  |
+-------+--------+---------+------+------------+-------+
2 rows in set (0.00 sec)
```

Sorting rows

Sorting Rows

List animal names and birthdays, sorted by date

```
sql> SELECT name, birth FROM pet ORDER BY
birth;
```

Sorting Rows

```
sql> SELECT name, birth FROM pet ORDER BY
birth;
+----------+------------+
| name     | birth      |
+----------+------------+
| Buffy    | 1989-05-13 |
| Bowser   | 1989-08-31 |
| Fluffy   | 1993-02-04 |
| Claws    | 1994-03-17 |
| Puffball | 1999-03-30 |
+----------+------------+
5 rows in set (0.08 sec)
```

Sorting Rows

List in descending order:

```
sql> SELECT name, birth FROM pet ORDER BY birth
DESC;
```

Sorting Rows

```
sql> SELECT name, birth FROM pet ORDER BY birth
DESC;
+----------+------------+
| name     | birth      |
+----------+------------+
| Puffball | 1999-03-30 |
| Claws    | 1994-03-17 |
| Fluffy   | 1993-02-04 |
| Bowser   | 1989-08-31 |
| Buffy    | 1989-05-13 |
+----------+------------+
```

Sorting Rows

sort by type of animal in ascending order, then by birth date within animal type in descending order (youngest animals first)

Sorting Rows

sort by type of animal in ascending order, then by birth date within animal type in descending order (youngest animals first)

```
sql> SELECT name, species, birth FROM pet ORDER BY
species, birth DESC;
```

Sorting Rows

sort by type of animal in ascending order, then by birth date within animal type in descending order (youngest animals first)

```
sql> SELECT name, species, birth FROM pet
    ->  ORDER BY species, birth DESC;
+----------+---------+------------+
| name     | species | birth      |
+----------+---------+------------+
| Claws    | cat     | 1994-03-17 |
| Fluffy   | cat     | 1993-02-04 |
| Bowser   | dog     | 1989-08-31 |
| Buffy    | dog     | 1989-05-13 |
| Puffball | hamster | 1999-03-30 |
+----------+---------+------------+
```

Working with `NULL` Values

The `NULL` value can be surprising until you get used to it.

Conceptually, `NULL` means “a missing unknown value” and it is treated somewhat differently from other values.

To test for `NULL`, you cannot use the arithmetic comparison operators such as `=`, `<`, or `<>`.

```
sql> SELECT COUNT(*) FROM
ALL_TABLES WHERE
    1 = NULL or
1 <> NULL or
1 < NULL or
1 > NULL or
NULL = NULL;
```

???????

Working with `NULL` Values

The `NULL` value can be surprising until you get used to it.

Conceptually, `NULL` means “a missing unknown value” and it is treated somewhat differently from other values.

To test for `NULL`, you cannot use the arithmetic comparison operators such as `=`, `<`, or `<>`.

```
SQL> SELECT COUNT(*) FROM ALL_TABLES WHERE  1 = NULL or 1 <>
NULL or 1 < NULL or 1 > NULL or NULL = NULL;

  COUNT(*)
----------
         0
```

Working with NULL Values

Use the IS NULL and IS NOT NULL operators!!!

```
SQL> select name from pet where death is null;

NAME
--------------------
Puffball
Fluffy
Claws
Buffy
```

Working with NULL Values

Use the IS NULL and IS NOT NULL operators!!!

```
SQL> select name from pet where death is not null;

NAME
--------------------
Bowser
```

Working with more than one table

loading the event table

- execute
  SQL> @/ubfs/myfiles/b/i/bittner3/ubunix/GIS-design/pet_event.sql;
- you can find the file pet_event.sql on ublearns.

list the event table.

```
sql> select * from event;
```

name	date	type	remark
Fluffy	1995-05-15	litter	4 kittens, 3 female, 1 male
Buffy	1993-06-23	litter	5 puppies, 2 female, 3 male
Buffy	1994-06-19	litter	3 puppies, 3 female
Chirpy	1999-03-21	vet	needed beak straightened
Slim	1997-08-03	vet	broken rib
Bowser	1991-10-12	kennel	
Fang	1991-10-12	kennel	
Fang	1998-08-28	birthday	Gave him a new chew toy
Claws	1998-03-17	birthday	Gave him a new flea collar
Whistler	1998-12-09	birthday	First birthday

Queries using two tables

```
sql> SELECT pet.name, owner, event.name,
type FROM pet INNER JOIN event ON pet.name =
event.name;
```

databasename.attributename

Queries using two tables

```
sql> SELECT pet.name, owner, event.name,
type FROM pet INNER JOIN event ON pet.name =
event.name;
```

```
+--------+--------+--------+----------+
| name   | owner  | name   | type     |
+--------+--------+--------+----------+
| Fluffy | Harold | Fluffy | litter   |
| Buffy  | Harold | Buffy  | litter   |
| Buffy  | Harold | Buffy  | litter   |
| Bowser | Diane  | Bowser | kennel   |
| Claws  | Gwen   | Claws  | birthday |
+--------+--------+--------+----------+
```

Queries using two tables

```
sql> SELECT pet.name, owner, event.name, type
FROM
pet INNER JOIN event ON pet.name = event.name;
```

```
+--------+--------+--------+----------+
| name   | owner  | name   | type     |
+--------+--------+--------+----------+
| Fluffy | Harold | Fluffy | litter   |
| Buffy  | Harold | Buffy  | litter   |
| Buffy  | Harold | Buffy  | litter   |
| Bowser | Diane  | Bowser | kennel   |
| Claws  | Gwen   | Claws  | birthday |
+--------+--------+--------+----------+
```

Queries using two tables

```
mysql> SELECT pet.name, owner, event.name, type
FROM
pet INNER JOIN event ON pet.name = event.name;
```

```
+--------+--------+--------+----------+
| name   | owner  | name   | type     |
+--------+--------+--------+----------+
| Fluffy | Harold | Fluffy | litter   |
| Buffy  | Harold | Buffy  | litter   |
| Buffy  | Harold | Buffy  | litter   |
| Bowser | Diane  | Bowser | kennel   |
| Claws  | Gwen   | Claws  | birthday |
+--------+--------+--------+----------+
```

inner join

- The query uses an **INNER JOIN** to combine the tables.
- An **INNER JOIN** allows for rows from either table to appear in the result if and only if both tables meet the conditions specified in the ON clause.
- In this example, the **ON** clause specifies that the **name** column in the **pet** table must match the **name** column in the **event** table.
- If a name appears in one table but not the other, the row will not appear in the result because the condition in the **ON** clause fails.

Queries using two tables

- Suppose that you want to find out the ages at which each pet had its litters.
- The litter date of the mother is in the **event** table, but to calculate her age on that date you need her birth date, which is stored in the **pet** table.
- This means the query requires both tables

Queries using two tables

databasename.attributename

```
SELECT
  pet.name,
  (to_char(edate,'YYYY')-to_char(birth,'YYYY')) AS age,
  remark
FROM pet INNER JOIN event ON pet.name = event.name
WHERE event.type = 'litter';
```

Queries using two tables

```
SELECT
  pet.name,
  (to_char(edate,'YYYY')-to_char(birth,'YYYY')) AS age,
  remark
FROM pet INNER JOIN event ON pet.name = event.name
WHERE event.type = 'litter';
```

Queries using two tables

```
SELECT
  pet.name,
  (to_char(edate,'YYYY')-to_char(birth,'YYYY')) AS age,
  remark
FROM pet INNER JOIN event ON pet.name = event.name
WHERE event.type = 'litter';
```

```
SELECT
  pet.name,
  (to_char(edate,'YYYY')-to_char(birth,'YYYY'))
AS age,
  remark
FROM pet INNER JOIN event ON pet.name =
event.name
WHERE event.type = 'litter';
```

```
+--------+------+-----------------------------+
| name   | age  | remark                      |
+--------+------+-----------------------------+
| Fluffy |    2 | 4 kittens, 3 female, 1 male |
| Buffy  |    4 | 5 puppies, 2 female, 3 male |
| Buffy  |    5 | 3 puppies, 3 female         |
+--------+------+-----------------------------+
```

The country database

Create and fill the following database tables:

COUNTRY	Name	Cont	Pop (millions)	GDP (billions)	Life-Exp	Shape
	Canada	NAM	30.1	658.0	77.08	Polygonid-1
	Mexico	NAM	107.5	694.3	69.36	Polygonid-2
	Brazil	SAM	183.3	1004.0	65.60	Polygonid-3
	Cuba	NAM	11.7	16.9	75.95	Polygonid-4
	USA	NAM	270.0	8003.0	75.75	Polygonid-5
	Argentina	SAM	36.3	348.2	70.75	Polygonid-6

(a) Country

CITY	Name	Country	Pop (millions)	Capital	Shape
	Havana	Cuba	2.1	Y	Pointid-1
	Washington, D.C.	USA	3.2	Y	Pointid-2
	Monterrey	Mexico	2.0	N	Pointid-3
	Toronto	Canada	3.4	N	Pointid-4
	Brasilia	Brazil	1.5	Y	Pointid-5
	Rosario	Argentina	1.1	N	Pointid-6
	Ottawa	Canada	0.8	Y	Pointid-7
	Mexico City	Mexico	14.1	Y	Pointid-8
	Buenos Aires	Argentina	10.75	Y	Pointid-9

(b) City

RIVER	Name	Origin	Length (kilometers)	Shape
	Rio Parana	Brazil	2600	LineStringid-1
	St. Lawrence	USA	1200	LineStringid-2
	Rio Grande	USA	3000	LineStringid-3
	Mississippi	USA	6000	LineStringid-4

(c) River