Exploratory Data Analysis with R

Basic SQL Queries - Part II

Xuemao Zhang
East Stroudsburg University

November 30, 2022

Outline

- Retrieving data from multiple tables
- Joining tables
- SQL Join Expressions
  - INNER JOIN
  - LEFT JOIN
  - RIGHT JOIN
  - FULL JOIN
  - CROSS JOIN

Retrieving data from multiple tables

- List each required table in the **FROM** clause
- **SELECT** and **WHERE** clauses can refer to attributes of any of these tables
- In general, `conditions` (such as two attribute values being equal) need to 'connect' the tables
- If the same attribute name appears in two relations, we can disambiguate them by using the relation name as well, e.g., surveys.plot_id

Joining tables

- List all species located in plot_id=2:

```
SELECT DISTINCT species.species
FROM species, surveys
WHERE species.species_id = surveys.species_id
AND surveys.plot_id=2;
```

- Find all observations with species=harrisi located in plot_id=2:

```
SELECT *
FROM species, surveys
WHERE species.species_id = surveys.species_id
AND species.species = 'harrisi'
AND surveys.plot_id=2;
```

- Find all observations with genus=Baiomys located in plot_id=2
  - Again, we join the tables by species_id

```
SELECT *
FROM species, surveys
WHERE species.species_id = surveys.species_id
AND species.genus='Baiomys'
AND surveys.plot_id=2;
```

Joining tables

- We can join more than two tables in this way

```
SELECT *
FROM species, surveys, plots
WHERE species.species_id = surveys.species_id
AND surveys.plot_id = plots.plot_id
AND species.species = 'harrisi'
AND surveys.plot_id=2;
```

SQL Join Expressions

- Instead of 'connecting' the tables in the **WHERE** clause, we can use an explicit **JOIN** in the **FROM** clause.
- Types of joins
  - INNER JOIN
  - LEFT JOIN
  - RIGHT JOIN
  - FULL OUTER JOIN
  - CROSS JOIN
- All joins in the last two slides are essentially `INNER JOIN`s

SQL Join Expressions - INNER JOIN

- `INNER JOIN`: The INNER JOIN keyword selects all rows from both tables as long as there is a match between the columns in both tables.
  - INNER JOIN animation
- SQL INNER JOIN Syntax

```
SELECT      column-name(s)
FROM        table1
INNER JOIN  table2
ON          table1.column-name=table2.column-name;
```

- **Note**: `INNER JOIN` can be replaced with `JOIN`

SQL Join Expressions - INNER JOIN

- Find all observations located in `plot_id=2`:

```
SELECT *
FROM species
JOIN surveys
ON species.species_id = surveys.species_id
WHERE plot_id=2;
```

SQL Join Expressions - LEFT JOIN

- `LEFT JOIN`: The LEFT JOIN keyword returns all rows from the left table (table1), with the matching rows in the right table (table2). The result is NULL in the right side when there is no match.
  - LEFT JOIN animation
- SQL LEFT JOIN Syntax

```
SELECT      column-name(s)
FROM        table1
LEFT JOIN   table2
ON          table1.column-name=table2.column-name;
```

- **Note**: In some databases `LEFT JOIN` is called `LEFT OUTER JOIN`.

SQL Join Expressions - LEFT JOIN

- Find all observations with `plot_id=2`

```
SELECT *
FROM surveys
LEFT JOIN species
ON species.species_id = surveys.species_id
WHERE plot_id=2;
```

SQL Join Expressions - LEFT JOIN

- **Note**: The results will be a little different if we interchange the order of surveys and species. It is due to missing values of species_id in surveys

```
SELECT *
FROM species
LEFT JOIN surveys
ON species.species_id = surveys.species_id
WHERE plot_id=2;
```

- Using R

```
species=read.csv("../data/portal_mammals/species.csv", header = T, sep = ",")
surveys=read.csv("../data/portal_mammals/surveys.csv", header = T, sep = ",")
unique(surveys$species_id)
library(dplyr)
data1=left_join(surveys,species, by="species_id")
dim(data1)
data2=left_join(species, surveys, by="species_id")
dim(data2)
```

SQL Join Expressions - RIGHT JOIN

- `RIGHT JOIN`: The RIGHT JOIN keyword returns all rows from the right table (table2), with the matching rows in the left table (table1). The result is NULL in the left side when there is no match.
  - RIGHT JOIN animation
- SQL RIGHT JOIN Syntax

```
SELECT      column-name(s)
FROM        table1
RIGHT JOIN  table2
ON          table1.column-name=table2.column-name;
```

- **Note**: In some databases `RIGHT JOIN` is called `RIGHT OUTER JOIN`.

SQL Join Expressions - RIGHT JOIN

- Find all observations with `plot_id=2`

```
SELECT *
FROM species
RIGHT JOIN surveys
ON species.species_id = surveys.species_id
WHERE plot_id=2;
```

- It is equivalent to

```
SELECT *
FROM surveys
LEFT JOIN species
ON surveys.species_id = species.species_id
WHERE plot_id=2;
```

SQL Join Expressions - FULL JOIN

- `FULL JOIN`: The FULL JOIN keyword returns all rows from the left table (table1) and from the right table (table2). The FULL JOIN keyword combines the result of both LEFT and RIGHT joins.
  - FULL JOIN animation
- SQL FULL JOIN Syntax

```
SELECT      column-name(s)
FROM        table1
FULL JOIN   table2
ON          table1.column-name=table2.column-name;
```

- **Note**: The `FULL JOIN` generally is written as `FULL OUTER JOIN`.

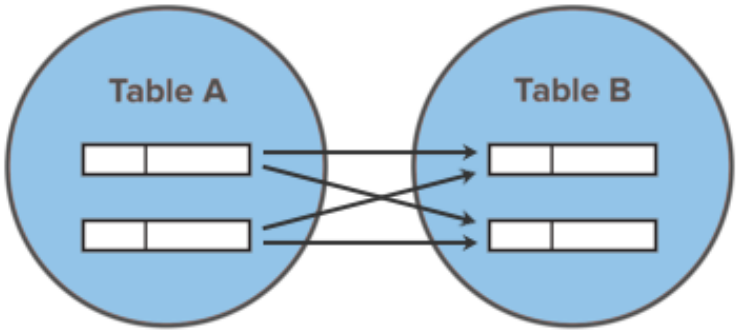
SQL Join Expressions - FULL JOIN

- Find all observations with `plot_id=2` after FULL JOINing the two tables `species` and `surveys`

```
SELECT *
FROM surveys
FULL OUTER JOIN species
ON species.species_id = surveys.species_id
WHERE plot_id=2;
```

SQL Join Expressions - CROSS JOIN

- `CROSS JOIN`: The CROSS JOIN keyword produces a Cartesian product of rows in the two tables.



SQL CROSS JOIN

- SQL CROSS JOIN Syntax

```
SELECT      column-name(s)
FROM        table1
CROSS JOIN  table2;
```

SQL Join Expressions - CROSS JOIN

- Another way to perform cross join

SELECT column-name(s)
FROM table1, table2;

- Cartesian product of the two tables `plots` and `species`

```
SELECT *
FROM species, plots;
```

SQL Join Expressions

- We can join more than two tables. For example,

```
SELECT *
FROM species
LEFT JOIN surveys
ON species.species_id = surveys.species_id
LEFT JOIN plots
ON plots.plot_id = surveys.plot_id
WHERE surveys.plot_id=2;
```

License