CCINFOM

Module 2 Lesson 03: Different ways to use data from different sources

Estimated Time to consume material – 6.0 Hour

Requirements for this lesson

1. Installed MYSQL and MYSQL Workbench
2. Downloaded MYSQL Script File (dbsimple.sql)

Lesson 3: Different ways to use data from different sources

There are many ways we use data from different sources or tables. These are (a) Cartesian Product, (b) Inner Joins or Joins (c) Outer Joins or Left Joins, (d) Union, (e) Intersect, and (f) Set Difference. For this lesson, we will only be discussing four (4) out of the six ways – A, B, C and D. We will go back to learn about Intersect and Set Difference when we are in SQL Level 4.

To understand these different ways to use data from different sources or tables, we will be using the database described below for the entire lesson. The script file to create this database is in the Google Folder on Learning Materials shared to the entire class. In this lesson, the ways to use data from different sources will be illustrated and the SQL SELECT statement to implement it, will be discussed in succeeding lessons.

Fruits

fruitcode	description	qty
APPLE	Red Fruit	20
BANANA	Yellow Fruit	10
MANGO	Yelow Fruit	10
TOMATO	Red Fruit	20

Students

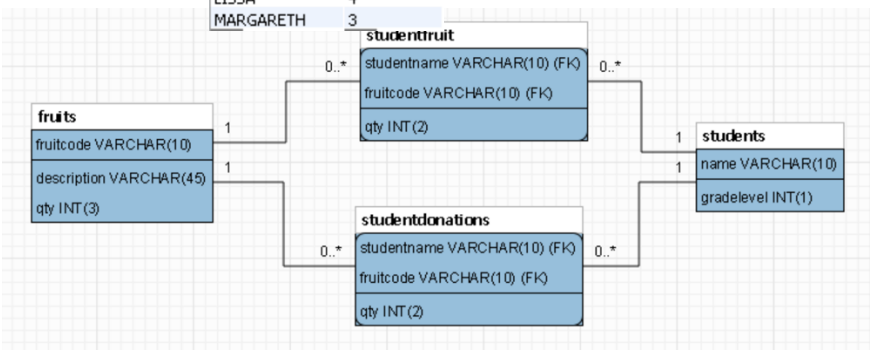
name	gradelevel
ALEXANDRA	2
ANGELA	6
CHRISTIAN	4
GERONE	2
JEHOVAH	5
LEON	5
LISSA	4
MARGARETH	3

StudentDonation

studentname	fruitcode	qty
ANGELA	MANGO	4
GERONE	BANANA	10
LEON	APPLE	4
LISSA	APPLE	4

StudentFruits

studentname	fruitcode	qty
ANGELA	MANGO	3
JEHOVAH	APPLE	2
LEON	APPLE	3



CARTESIAN PRODUCT (CP or ,)

To use data from two data sources/tables, the very basic way is to do cartesian product. To better understand what cartesian product is, let's start with an example.

If we say "**FRUITS, STUDENTDONATIONS**", then we are using the cartesian product of data coming from the two tables like this

Fruits

fruitcode	description	qty
APPLE	Red Fruit	20
BANANA	Yellow Fruit	10
MANGO	Yelow Fruit	10
TOMATO	Red Fruit	20

,

StudentDonation

studentname	fruitcode	qty
ANGELA	MANGO	4
GERONE	BANANA	10
LEON	APPLE	4
LISSA	APPLE	4

=

fruitcode	description	qty	studentname	fruitcode	qty
APPLE	Red Fruit	20	ANGELA	MANGO	4
BANANA	Yellow Fruit	10	ANGELA	MANGO	4
MANGO	Yelow Fruit	10	ANGELA	MANGO	4
TOMATO	Red Fruit	20	ANGELA	MANGO	4
APPLE	Red Fruit	20	GERONE	BANANA	10
BANANA	Yellow Fruit	10	GERONE	BANANA	10
MANGO	Yelow Fruit	10	GERONE	BANANA	10
TOMATO	Red Fruit	20	GERONE	BANANA	10
APPLE	Red Fruit	20	LEON	APPLE	4
BANANA	Yellow Fruit	10	LEON	APPLE	4
MANGO	Yelow Fruit	10	LEON	APPLE	4
TOMATO	Red Fruit	20	LEON	APPLE	4
APPLE	Red Fruit	20	LISSA	APPLE	4
BANANA	Yellow Fruit	10	LISSA	APPLE	4
MANGO	Yelow Fruit	10	LISSA	APPLE	4
TOMATO	Red Fruit	20	LISSA	APPLE	4

SELECT *
FROM fruits, studentdonations;

What did you notice when we use the two tables in cartesian product?

Every record of FRUITS is combined with every record of STUDENTDONATIONS. As such, if there are 4 records of FRUITS, and 4 records of STUDENTDONATIONS, the cartesian product will generate a single table containing 16 records.

fruitcode	description	qty	studentname	fruitcode	qty
APPLE	Red Fruit	20	ANGELA	MANGO	4

But does each resulting record have sense? Let's take one of the records in the result as shown on the right. It doesn't make sense to have a STUDENTDONATIONS record for a mango and combine it to a record of an APPLE. This is why cartesian product is NOT USUALLY USED to fulfill information requirements. Most of the resulting records when two data sources are combined using cartesian product may not have any useful sense. Just count the number of results that do not make sense compared to the number of results that make sense? In the result, there are 12 results that do not make sense, and only 4 results that make sense.

The other reason why cartesian product is being avoided is the amount of memory being consumed by the result of the cartesian product. Just imagine if there are 100,000 records of Table A, and 200,000 records of Table B. A cartesian product would require a memory that can store 100,000 x 200,000 = 20 billion records.

INNER JOIN (JOIN)
Inner join is combining records from one table with records from another table that pass a set of conditions that will produce results with useful sense. In our example in cartesian product, some of the records do not make sense since the fruitcode of records of STUDENTDONATIONS is combined with a different fruitcode of records of FRUITS. What will make the combined records make sense is, if the fruitcode in STUDENTDONATIONS match the fruitcode in FRUITS. Join actually does this, as shown below. When combining, only combinations that pass the condition stated in the join will be part of the result.

Fruits

fruitcode	description	qty
APPLE	Red Fruit	20
BANANA	Yellow Fruit	10
MANGO	Yelow Fruit	10
TOMATO	Red Fruit	20

join

On fruits.fruitcode=studentdonations.fruitcode

StudentDonation

studentname	fruitcode	qty
ANGELA	MANGO	4
GERONE	BANANA	10
LEON	APPLE	4
LISSA	APPLE	4

=

fruitcode	description	qty	studentname	fruitcode	qty
MANGO	Yelow Fruit	10	ANGELA	MANGO	4
BANANA	Yellow Fruit	10	GERONE	BANANA	10
APPLE	Red Fruit	20	LEON	APPLE	4
APPLE	Red Fruit	20	LISSA	APPLE	4

```
SELECT  *
FROM    fruits    JOIN    studentdonations   ON    fruits.fruitcode = studentdonations.fruitcode;
```

FRUITS and STUDENTDONATIONS are "*joinable*" since based on the data model, they are related to each other, and there is a field representing the relationship – the fruitcode. And if you have observed, fruitcode is the field used in the join condition. JOINS is the usual way of combining records from two tables. Most of the information requirements use JOIN when there is a need to use data coming from several tables in the database. Some says, that JOIN is simply like a CARTESIAN PRODUCT with condition. For now, we will accept this thought, but In the future, we will discover that this isn't really the same.

OUTER JOIN (LEFT JOIN)
Outer join is combining **all records** from one table with records from another table on a set of conditions that will produce results with useful sense. The key difference between INNER JOIN and OUTER JOIN is that all records from one table is part of the result even if it doesn't have any combining record from another table. Let's illustrate this using an example below.

Fruits

fruitcode	description	qty
APPLE	Red Fruit	20
BANANA	Yellow Fruit	10
MANGO	Yelow Fruit	10
TOMATO	Red Fruit	20

left join

On fruits.fruitcode=studentdonations.fruitcode

StudentDonation

studentname	fruitcode	qty
ANGELA	MANGO	4
GERONE	BANANA	10
LEON	APPLE	4
LISSA	APPLE	4

=

fruitcode	description	qty	studentname	fruitcode	qty
MANGO	Yelow Fruit	10	ANGELA	MANGO	4
BANANA	Yellow Fruit	10	GERONE	BANANA	10
APPLE	Red Fruit	20	LEON	APPLE	4
APPLE	Red Fruit	20	LISSA	APPLE	4
TOMATO	Red Fruit	20	NULL	NULL	NULL

```
SELECT  *
FROM    fruits    LEFT JOIN    studentdonations   ON    fruits.fruitcode = studentdonations.fruitcode;
```

Notice that in the result, The TOMATO record in FRUITS is part of the result, even if there are no records in STUDENTDONATIONS having TOMATO. This is what OUTER JOIN is doing. In this example, all the records of FRUITS will be part of the result even if they do not have a combining record from STUDENTDONATIONS. There are many information requirements that required the use of OUTER JOIN, particularly that of looking for information about something and if or not they have related records.

An example could be the list of students and the number of courses they have enrolled in. Since there maybe students that have not enrolled in any courses yet, they should still be part of the result regardless if they have enrolled or not. But if the requirement needs the list of enrolled students and the number of courses they have enrolled in, then the requirement gives us a clue that this will require an inner join instead, since we only need records of students that has a relationship with courses (enrolled relationship).

Let us explore OUTER JOIN further, what if the OUTER JOIN is like what is shown below, is the result the same as the left join above?

StudentDonation

studentname	fruitcode	qty
ANGELA	MANGO	4
GERONE	BANANA	10
LEON	APPLE	4
LISSA	APPLE	4

left join

On fruits.fruitcode=studentdonations.fruitcode

Fruits

fruitcode	description	qty
APPLE	Red Fruit	20
BANANA	Yellow Fruit	10
MANGO	Yelow Fruit	10
TOMATO	Red Fruit	20

```
SELECT   *
FROM     studentdonations   LEFT JOIN        fruits     ON        fruits.fruitcode = studentdonations.fruitcode;
```

The answer is NO. The result of this outer join is shown below.
All the records of STUDENTDONATIONS is part of the result regardless if they have a combining record from the FRUITS table.

studentname	fruitcode	qty	fruitcode	description	qty
ANGELA	MANGO	4	MANGO	Yelow Fruit	10
GERONE	BANANA	10	BANANA	Yellow Fruit	10
LEON	APPLE	4	APPLE	Red Fruit	20
LISSA	APPLE	4	APPLE	Red Fruit	20

UNION
If you have observed, cartesian product, inner and outer joins are horizontal combination of records from two or more data source/table. Horizontal combinations results to a table with the columns of one data source added with columns of the other data source. In our sample, FRUITS have 3 columns, STUDENTDONATIONS have 3 columns, and the result of combinations (cartesian product, inner or outer join) will always have 6 columns.

Union is another way of combining data from two or more data sources/tables. But this time it is performing a vertical combination and requiring the two data sources/table having the same number of column, same names and data types. To better understand union, we have to illustrate UNION through an example.

StudentDonation

studentname	fruitcode	qty
ANGELA	MANGO	4
GERONE	BANANA	10
LEON	APPLE	4
LISSA	APPLE	4

union

StudentFruit

studentname	fruitcode	qty
ANGELA	MANGO	3
JEHOVAH	APPLE	2
LEON	APPLE	3

=

studentname	fruitcode	qty
ANGELA	MANGO	4
GERONE	BANANA	10
LEON	APPLE	4
LISSA	APPLE	4
ANGELA	MANGO	3
JEHOVAH	APPLE	2
LEON	APPLE	3

```
SELECT   *
FROM     studentdonations
UNION
SELECT   *
FROM     studentfruit;
```

If you will look at the result, the number of columns is still the same as the number of columns of the two data sources/table. The result is also the vertical combination of the two data sources/tables – four (4) rows from STUDENTDONATIONS and three (3) rows from STUDENTFRUIT. No conditions to use in combining.

EXERCISE:
Given the database in this lesson, write the output of combining records from two tables. Put your answers in an Excel File, using separate worksheet per item in the exercise item and save it using <section>-<lastname>-<firstname>-M02L03.xls. Just be ready with the file, in case your teacher will collect it for formative assessment. Prepare your questions and clarifications as these are important indicators that you went through this exercise.

1.	students	,	studentfruit	
2.	students	JOIN	studentfruit	ON students.name = studentfruit.studentname
3.	students	LEFT JOIN	studentfruit	ON students.name = studentfruit.studentname
4.	studentfruit	UNION	studentdonations	
5.	students	UNION	studentfruit	