Relational databases are typically defined using relational algebra, which is a procedural query language. Relational databases have a structure to identify the relationships between the information that is stored there, those relationships are often created using the relational algebra technique. Relational algebra involves using operations to create relationships between one or more attributes to later be used as an attribute itself. To join multiple relationships using one or more operation, you must find the correct sequence. There are various operations to achieve the results desired. Below are some examples of common operations.

**Select**, σ, which is used to *fetch* rows, or *tuples,* from tables, the *relation*, under a given condition. **Project,** Π, is another operation that is used to project only a particular set of columns, or *attributes* from a *relation*, that are asked for and will also remove duplicate data from the *attributes*. The **Union**, ∪, operation *fetches* data from two *relations*, for this operation to work best, the relations must both have the same number of *attributes*.

For example, in the sample database for the auto dealer. If we wanted to find the cost of a car on the lot that was made in Italy, first I would need to check inventory, then I would need to check the vehicle ID number. From there I would need to check the make ID number to reference to the Make table to check for the country of origin. Now if I also wanted to check for a car made in Italy and also comes in a shade of red, I would have to also check the Color Table from the inventory table to check for a color name that contains the word 'Red'

Order of execution often does not matter for most algebraic operations. For example, it wouldn't matter much if we first checked for the color red and then for a car made in Italy or vice versa, we would achieve the same results either way. There is a way to optimize these queries however to improve speed and efficiency. To improve efficiency, you want to minimize the transitional results and find the best order of execution of the operations. Identify first the relations you will need to achieve the correct query result and then identify the operations that would be needed to achieve the shortest result in between all the tables in the database.

If we want to select an item in inventory, from a certain country and a certain color, you must first identify the tables needed; inventory, vehicle, and make. Since we only want cars from the country = 'Italy', we can Select Italy from the Make table and send it to a new table called MakeItaly. An example of a relational algebra expression is below:

$\text{MakeItaly} = \sigma_{\text{country=Italy}}(\text{Make})$

If we want to select a vehicle that is made in Italy and is the color red, you would create an expression similar to below:

$\Pi_{\text{Vehicle}}(\sigma_{\text{country='Italy'} \wedge \text{color='red'}}(\text{Vehicle} \bowtie \text{Make}))$

As we rely more on cloud data solutions and information systems to maintain business function, we are seeing an increase in the amount of information that gets stored in databases and data warehouses across the world. The ability to efficiently and successfully retrieve all that data is

essential for it to be successful. There must be the ability identify and create the proper relationships between these data entities and attributes in order to create efficient queries and to filter that mass amount of information. To achieve this, we must first understand how all the data relates to each other. Large data warehouses and cloud storage are becoming increasingly popular using perabytes of data a day and use relational algebra in every relational database instance. Relational algebra is an invaluable and crucial function of any query made to popular relational databases using SQL. If you find yourself involved in the database engine software, cloud systems and/or relational databases, this relational algebra lesson will come in handy again.

## Additional Lectures on Relational Algebra

- **Relational Algebra Playlist** (https://www.youtube.com/playlist?list=PL5pwvOxs9AGWZPBXmHU76G2HLPtb_Wdz9) - 6 Videos (35:26 total)

  From **Stanford Dbclass** (https://www.youtube.com/channel/UC5ZAemhQUQuNqW3c9Jkw8ug). (2014). *Relational Algebra Playlist*. YouTube. Retrieved from **https://www.youtube.com/playlist?list=PL5pwvOxs9AGWZPBXmHU76G2HLPtb_Wdz9** (https://www.youtube.com/playlist?list=PL5pwvOxs9AGWZPBXmHU76G2HLPtb_Wdz9)

## About RelaX

The relational algebra calculator **RelaX** (https://dbis-uibk.github.io/relax/landing) is a neat tool created by the University of Innsbruck to experiment and validate your relational algebra expressions.

Any expression which works in RelaX will be considered valid in this course.

RelaX is very helpful in decomposing expressions but it is also strict about data types and operator usage. For example, any attribute of string data type needs to be in single quotes only and any operands and parentheses should be separated with spaces. You can use either symbols or words to specify operators.

Please note that while RelaX can convert the relational algebra equations directly into SQL, you should always check your work for accuracy.

**Special Note: string expressions can only be encased by single quotation marks. Double quotation marks will result in an error in ReLaX. ReLaX also expects a space on either side of operators, e.g. '='.**

## Activity