The Power of Relational Algebra in Database Management System

Ebole Alpha F
Department of Computer Science, Lagos State Polytechnic, Lagos state Nigeria.

Illuno Christiana
Department of Mathematics and Statistics, Lagos State Polytechnic, Lagos State Nigeria

***Abstract*: The fundamental set of operations for the relational model is known as the relational algebra. These operations permit a user to specify basic retrieval requirements. The result of retrieval is a new relation, which may have been formed from one or more relations. The algebra operations thus produce new relations, which can be further manipulated using operations of the same algebra. A sequence of relational algebra operations forms a relational algebra expression, whose result will also be a relation that represents the result of a database query (or retrieval request). A language based on operators and a domain of values operators map values taken from the domain into other domain values hence, an expression involving operators and arguments produces a value in the field when the domain is a set of all relations. The operators are project, union, set difference, Cartesian product, select, rename, set intersection, natural join, and assignment, to generate and get the relational algebra. We refer to the expression as a query and the value produced as the query result.**

Keywords: Language, expression, operators and argument.

I. Introduction

Relational Algebra is an off- shoot of first-order logic. In 1970 E.F. Codd while working for IBM proposed how Relational Algebra can be used as a basis for database query language. Since then Relational Algebra has find extensive use in the development of query languages, the most popular of them being Structural Query Language (SQL). The beauty of Relational Algebra is that each of its operators takes as input one or more relational tables and outputs a relational table as result, which can again act as an input to another operator.

This unique characteristic makes it simple to construct complex queries using the relational operators. SQL being based on relational algebra follows a similar approach when building query. Relational algebra is a notation for specifying queries about the contents of relations. The relational algebra is a procedural query language that consists of a set of operations that take one or two relations as input and produce a new relation as their result.

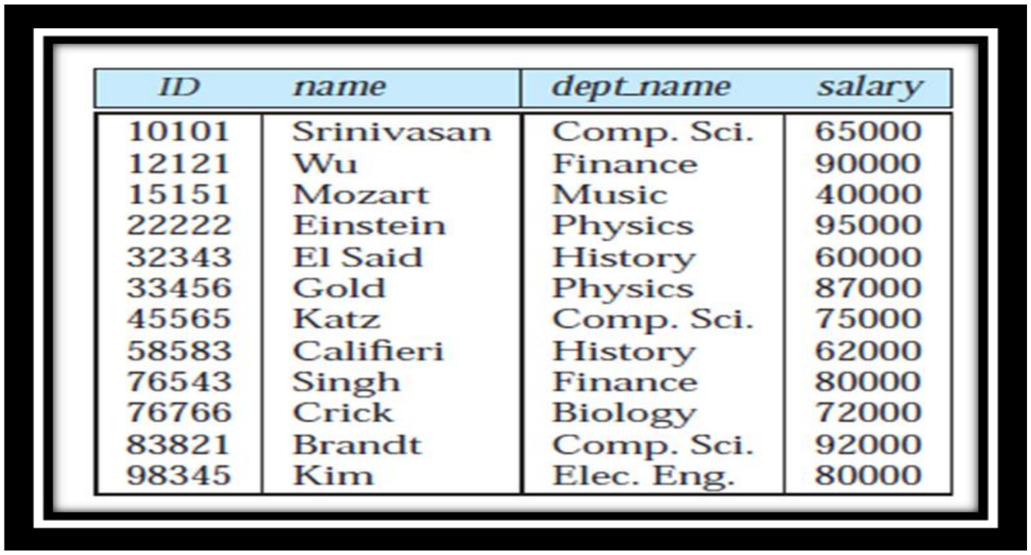
The fundamental operations in the relational algebra are select, project, union, set difference, Cartesian product, and rename. In addition to the fundamental operations, there are several other operations—namely, set intersection, natural join, and assignment. The select, project, and rename operations are called unary operations, because they operate on one relation. The other three operations operate on pairs of relations and are, therefore, called binary operations.

II. Background Studies

A relational system is a system in which, the data is perceived by the user as tables and the operators available to the user are operators that derive "new" tables from "old" ones. When Codd (1970) introduced his relational model of data, he formulated it mathematical analysis in terms of domains, tuples, and relations. It original interest was on data independence, not query languages. Two years later, Codd (1972) gave a detailed definition of Relational Algebra (RA) in a collection of eight unary and binary operations (restriction, projection, product, join, division, union, intersection, and difference) that can be performed on relations. In a survey of database educators, Robbert and Ricardo (2003) reported that only 70% included RA in their courses. Elmasri and Navathe (2006), Silberschatz, Korth, and Sudarshan (2006), and Ullman and Widom (2008) all define RA concepts in mathematical terms. There are several problems with the mathematical representation of RA queries. Most database students are uncomfortable with mathematics in general, and with the math notation used in RA queries in particular. RA math expressions in textbooks employ symbols from other languages, subscripts, and both prefix and infix operators in unfamiliar and potentially confusing ways. Several operations are often combined into a single expression to make the query language appear more "algebraic". This practice disguises the order of separate query operations, and hides the procedural nature of RA.

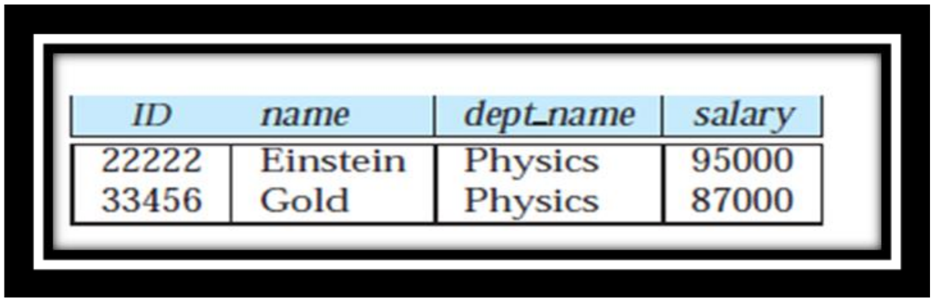
III. The Select Operation

The select operation selects tuples that satisfy a given predicate. We use the lowercase Greek letter sigma α to denote selection. The predicate appears as a subscript to the input. The argument relation is in parentheses after the output. Thus, to select those tuples of the instructor relation where the instructor is in the "Physics" department, we write: If the instructor relation is as shown in Figure 1 then the relation that results from the preceding query is as shown in Figure 2.



ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

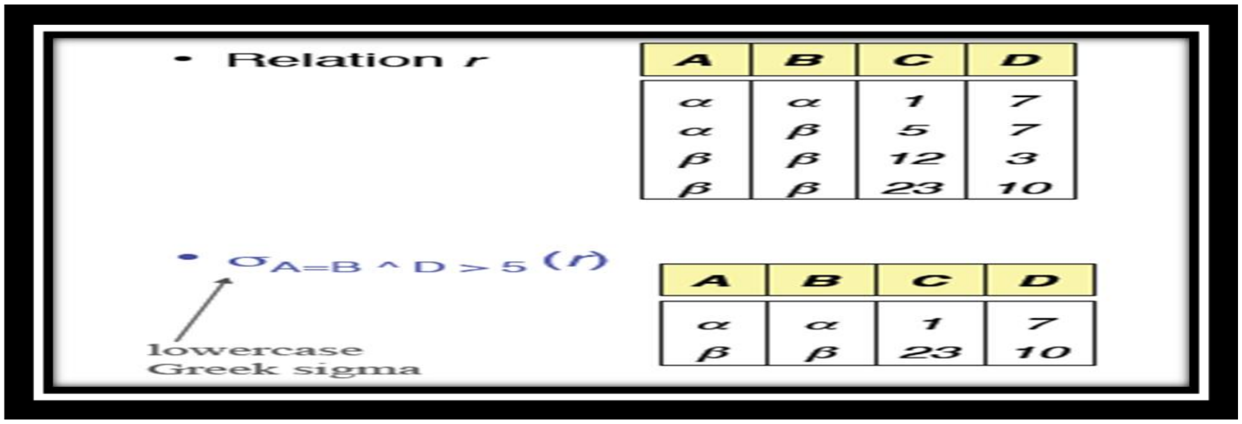
Figure 1



ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
33456	Gold	Physics	87000

Figure 2

The select operation allows us to retrieve some rows of a relation (by "some" I mean anywhere from none of them to all of them)



- Relation *r*

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

- $\sigma_{A=B \wedge D > 5}(r)$

lowercase Greek sigma

A	B	C	D
α	α	1	7
β	β	23	10

Figure 3

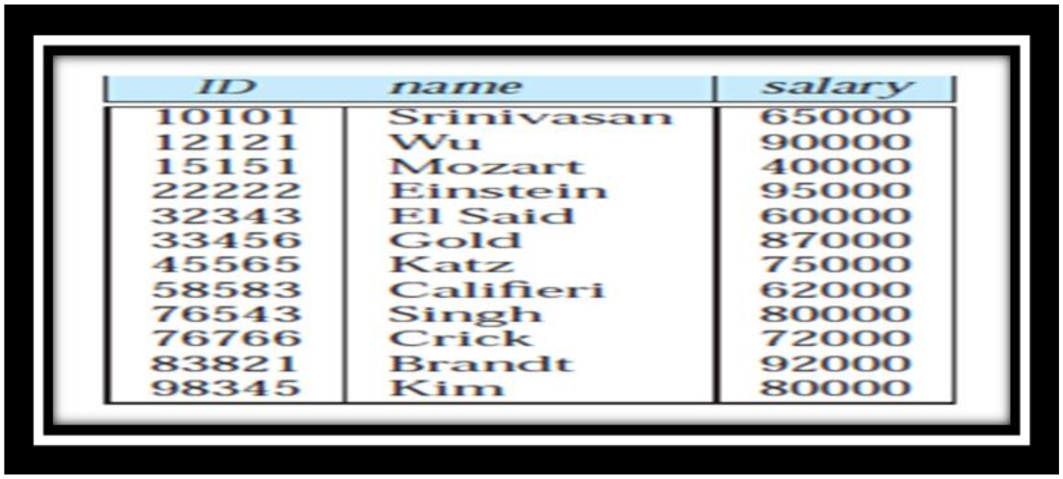
IV. The Project Operation

A unary operation is used to select particular columns of a table. Written as $\Pi_{a_1, a_2, \ldots a_n}$ (R), where a_1;------ an corresponds to a set of attribute names, the Project operator produces tuples in R which are restricted to the columns a_1------------:an .

Suppose we want to list all instructors' ID, name, and salary, but do not care about the dept name. The project operation allows us to produce this relation. The project operation is a unary operation that returns its argument relation, with certain attributes left out. Since a relation is a set, any duplicate rows are eliminated. Projection is denoted by the uppercase Greek letter pi (□). We list those attributes that we wish to appear in the result as a subscript to. The argument relation follows in parentheses. We write the query to produce such a list as:

$$\Pi_{ID,\ name,\ salary}(instructor)$$

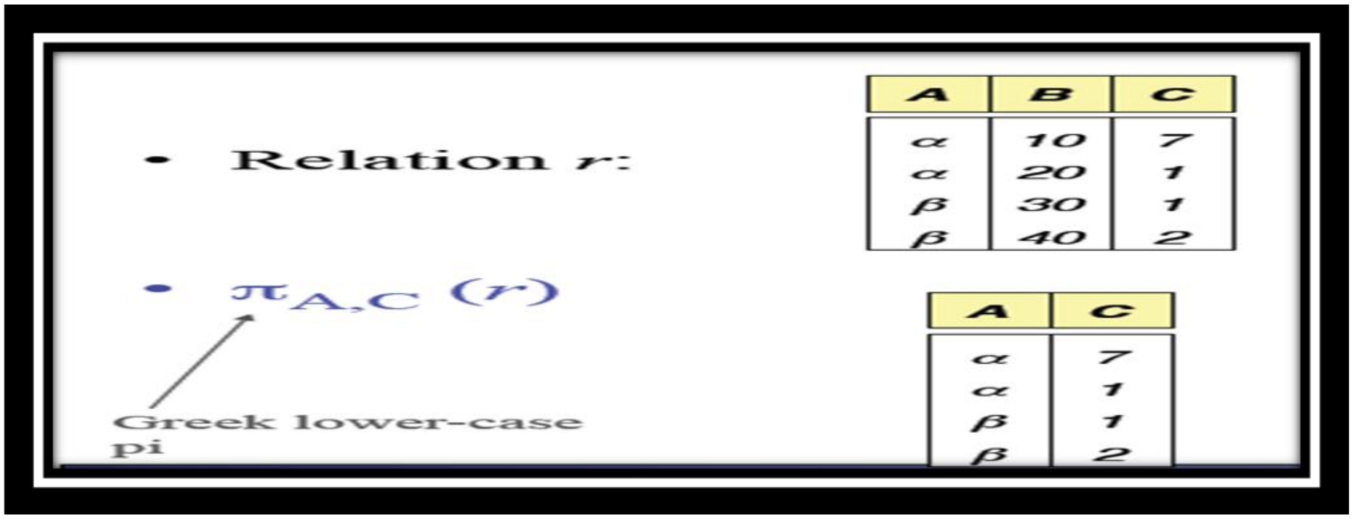
The project operation allows us to retrieve some columns of a relation (by "some" It means anywhere from none of them to all of them) Here columns A and C have been retrieved from figure 1 to give figure 4.



ID	name	salary
10101	Srinivasan	65000
12121	Wu	90000
15151	Mozart	40000
22222	Einstein	95000
32343	El Said	60000
33456	Gold	87000
45565	Katz	75000
58583	Califieri	62000
76543	Singh	80000
76766	Crick	72000
83821	Brandt	92000
98345	Kim	80000

Figure 4

Another example to further explain the Project operator is show below.



- Relation r:

A	B	C
α	10	7
α	20	1
β	30	1
β	40	2

- $\pi_{A,C}(r)$

A	C
α	7
α	1
β	1
β	2

Greek lower-case pi

Figure 5

V. The Union Operation

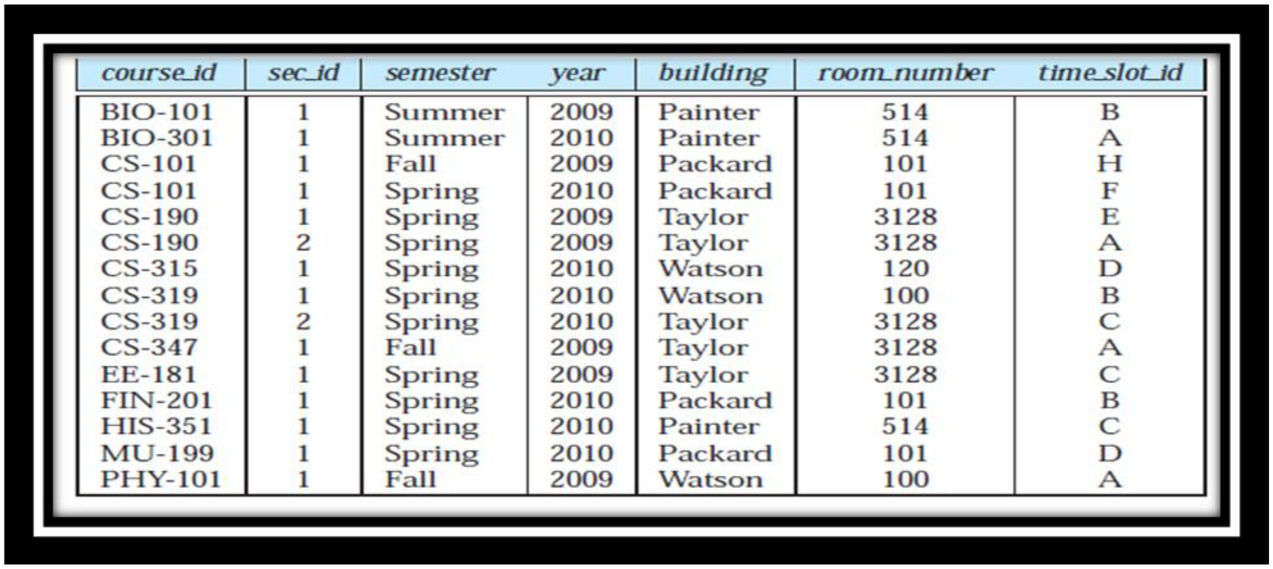
Consider a query to find the set of all courses taught in the Fall 2009 semester, the Spring 2010 semester, or both. The information is contained in the section relation to find the set of all courses taught in the fall 2009

$$\Pi_{course\_id}\ (\sigma_{semester = \text{“Fall”} \land year=2009}\ (section))$$

To find the set of all courses taught in the Spring 2010 semester, we write

$$\Pi_{course\_id}\ (\sigma_{semester = \text{“Spring”} \land year=2010}\ (section))$$

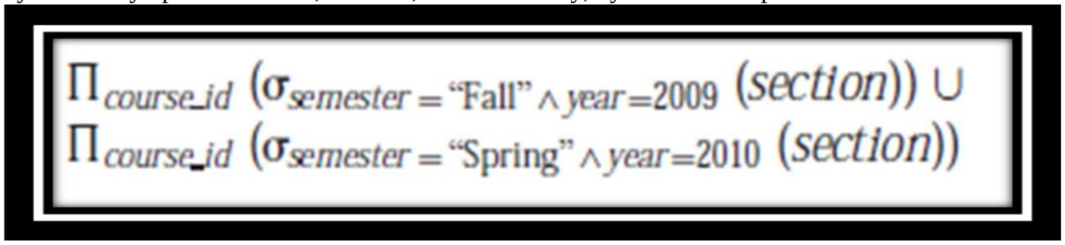
To answer the query, we need the union of these two sets; that is, we need all section IDs that appear in either or both of the two relations. We find these data



course_id	sec_id	semester	year	building	room_number	time_slot_id
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A

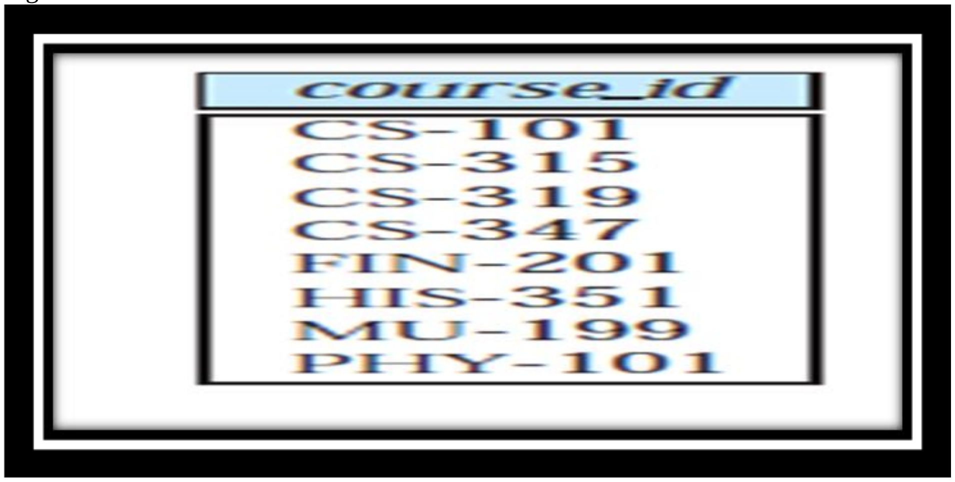
Figure 6

by the binary operation union, denoted, as in set theory, by ∪. So the expression needed is:



$$\Pi_{course\_id}\ (\sigma_{semester = \text{“Fall”} \land year=2009}\ (section)) \cup$$
$$\Pi_{course\_id}\ (\sigma_{semester = \text{“Spring”} \land year=2010}\ (section))$$

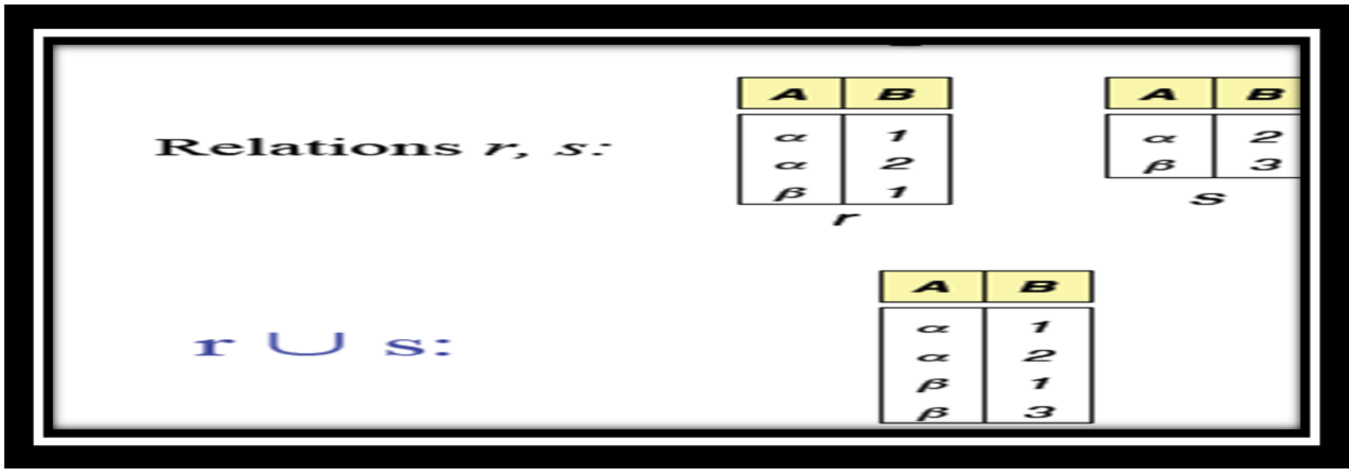
Figure 7



course_id
CS-101
CS-315
CS-319
CS-347
FIN-201
HIS-351
MU-199
PHY-101

Figure 8

The union operation also concatenates two relations, and removes duplicate rows (since relations are sets). Here there are two rows with A = α and B = 2. So one was discarded



Relations r, s:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

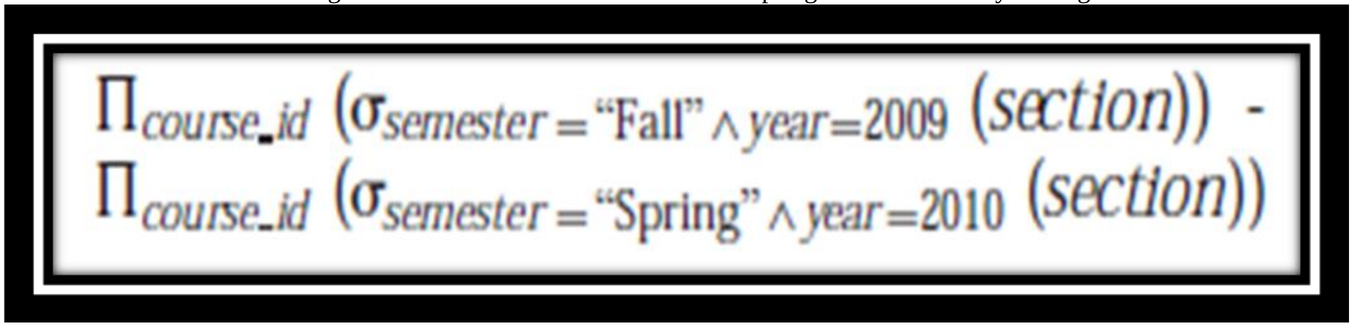
$r \cup s$:

A	B
α	1
α	2
β	1
β	3

Figure 9

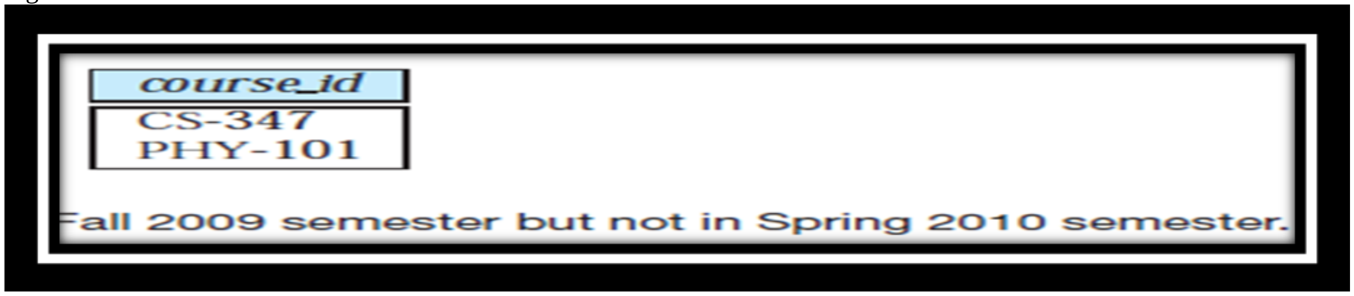
VI. The Set-Difference Operation

The set-difference operation, denoted by −, allows us to find tuples that are in one relation but are not in another. The expression r − s produces a relation containing those tuples in r but not in s.
We can find all the courses taught in the Fall 2009 semester but not in Spring 2010 semester by writing:



$$\Pi_{course\_id}\left(\sigma_{semester=\text{“Fall”} \wedge year=2009}(section)\right) - \Pi_{course\_id}\left(\sigma_{semester=\text{“Spring”} \wedge year=2010}(section)\right)$$

Figure 10

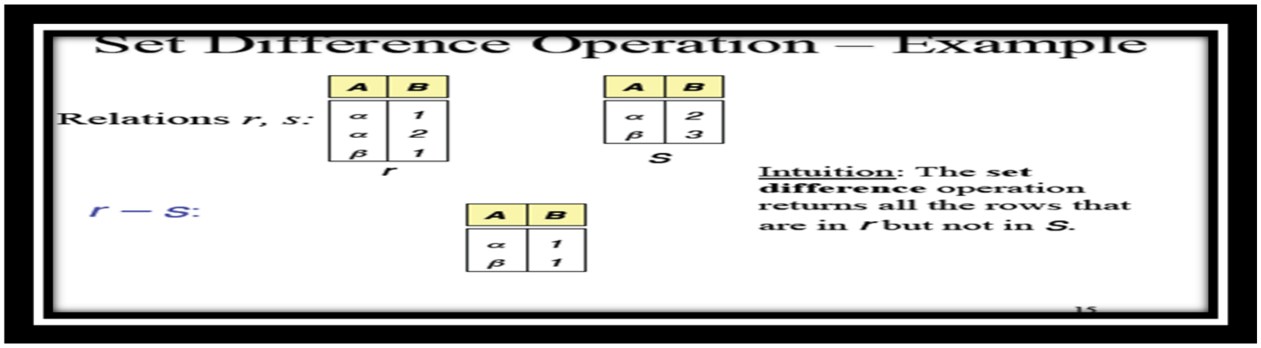


course_id
CS-347
PHY-101

Fall 2009 semester but not in Spring 2010 semester.

Figure 11

Another example to illustrate this operator is shown below.



Set Difference Operation – Example

Relations r, s:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

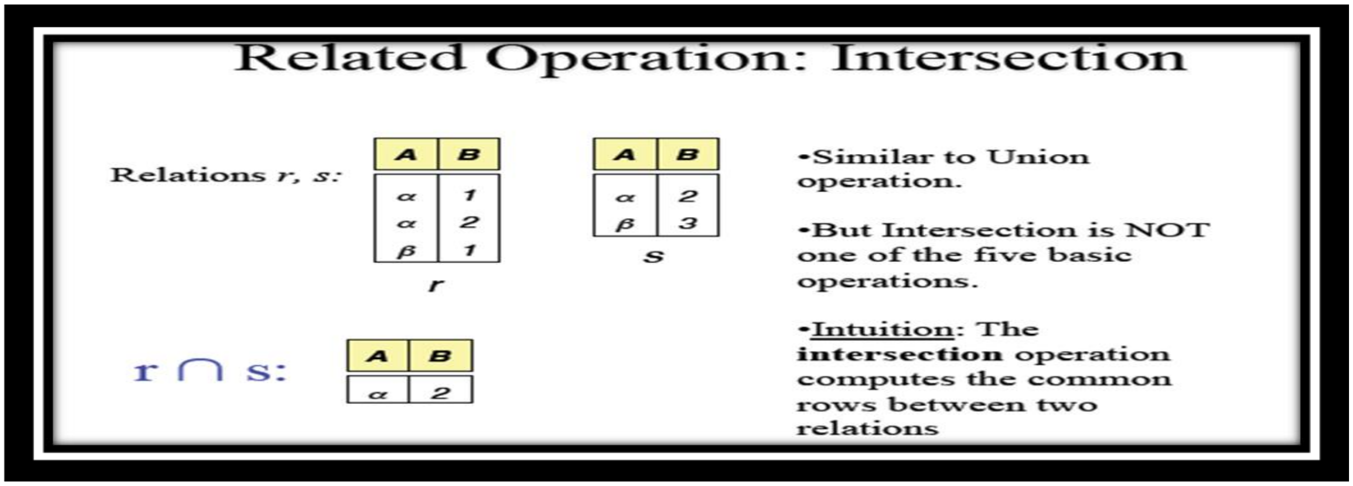
$r - s$:

A	B
α	1
β	1

Intuition: The set **difference** operation returns all the rows that are in r but not in S.

15

Figure 12



Related Operation: Intersection

Relations *r*, *s*:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

r ∩ s:

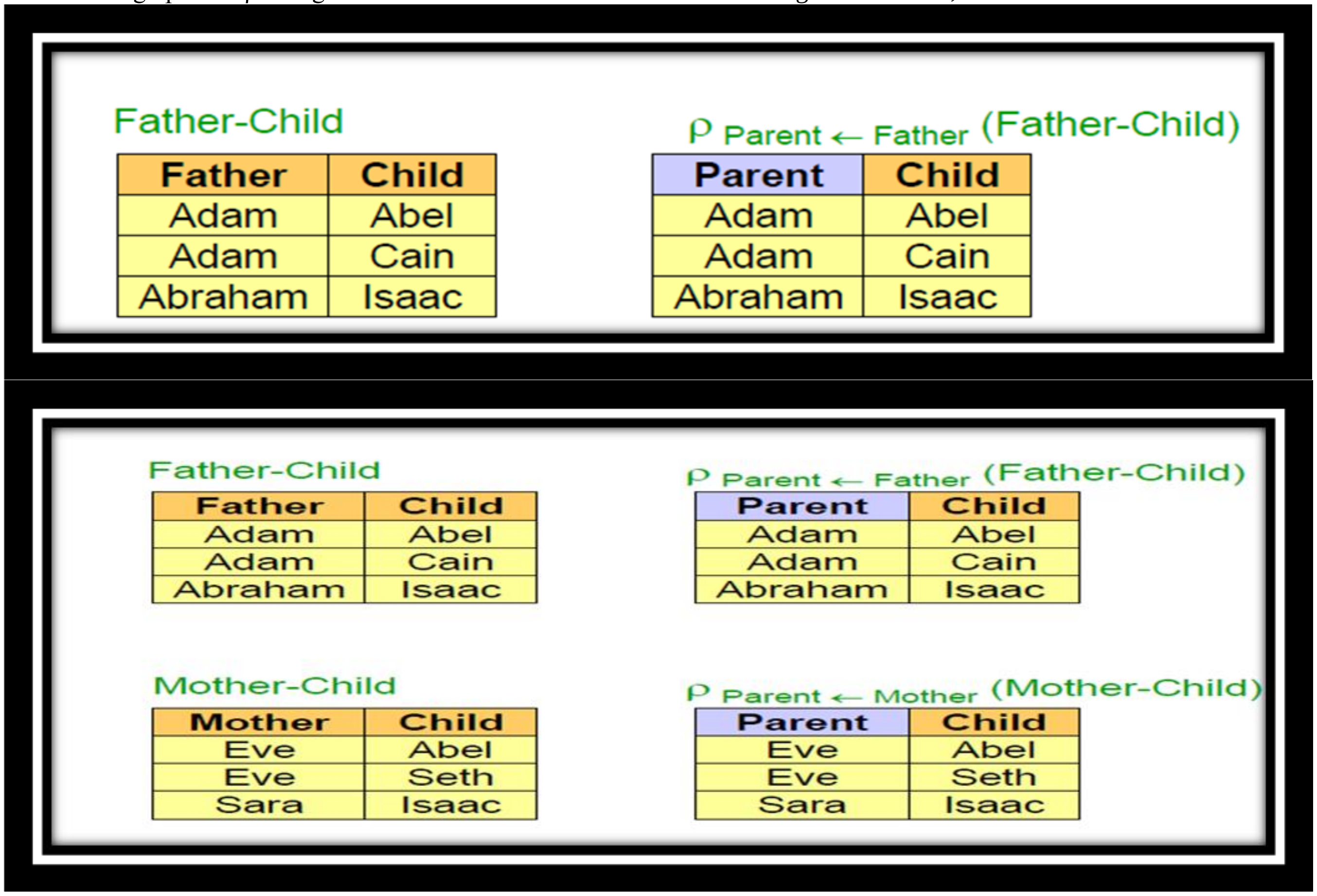
A	B
α	2

- Similar to Union operation.
- But Intersection is NOT one of the five basic operations.
- <u>Intuition</u>: The **intersection** operation computes the common rows between two relations

Figure 13

VII. Renaming

The renaming operator ρ changes the name of one or more attributes. It changes the schema, but not the instance of a relation



Father-Child

Father	Child
Adam	Abel
Adam	Cain
Abraham	Isaac

$\rho_{\text{Parent} \leftarrow \text{Father}}$ (Father-Child)

Parent	Child
Adam	Abel
Adam	Cain
Abraham	Isaac

Father-Child

Father	Child
Adam	Abel
Adam	Cain
Abraham	Isaac

$\rho_{\text{Parent} \leftarrow \text{Father}}$ (Father-Child)

Parent	Child
Adam	Abel
Adam	Cain
Abraham	Isaac

Mother-Child

Mother	Child
Eve	Abel
Eve	Seth
Sara	Isaac

$\rho_{\text{Parent} \leftarrow \text{Mother}}$ (Mother-Child)

Parent	Child
Eve	Abel
Eve	Seth
Sara	Isaac

Figure 14

The Cartesian product operation

Output all pairs of rows from the two input relations (regardless of whether or not they have the same values on common attributes) The Cartesian-product operation, denoted by a cross (×), allows us to combine information from any two relations. We write the Cartesian product of relations

U and V as U × V. = M

As shown in the table below respectivily

For example:

U

A	B	C
1	2	3
6	7	8
9	7	8

Figure 15

V

B	C	D
2	3	4
2	3	5
7	8	10

Figure 16

M

A	U.B	U.C	V.B	V.C	D
1	2	3	2	3	4
1	2	3	2	3	5
1	2	3	7	8	10
6	7	8	2	3	4
6	7	8	2	3	5
6	7	8	7	8	10
9	7	8	2	3	4
9	7	8	2	3	5
9	7	8	7	8	10

Figure 17

VIII. The natural join

This is a binary operation that allows us to combine certain selections and a Cartesian product into one operation. It is denoted by the join symbol. The natural-join operation forms a Cartesian product of its two arguments, performs a selection forcing equality on those attributes that appear in both relation schemas, and finally removes duplicate attributes.

U and V must have a corresponding element before it can be possible.

For example:

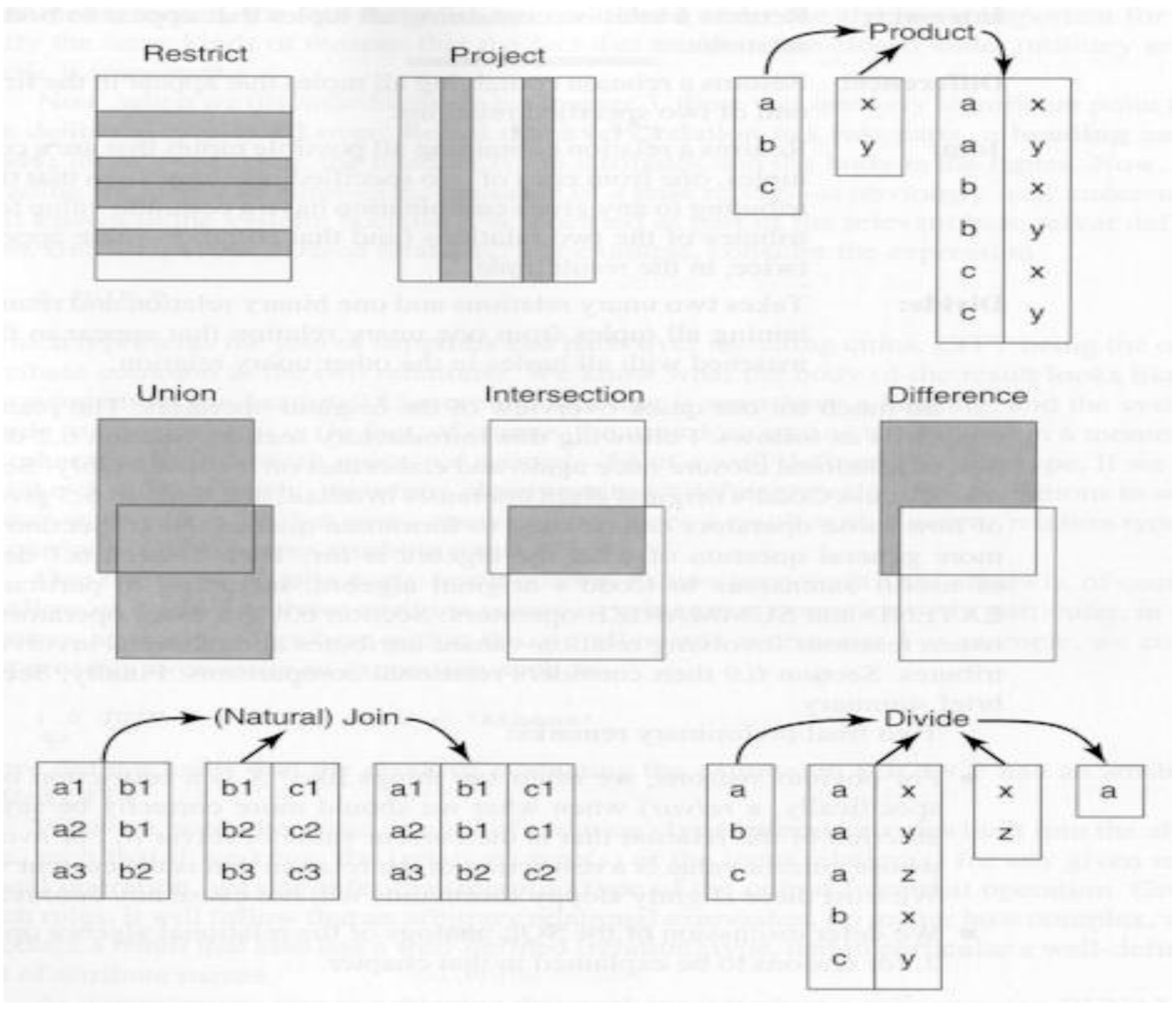
$U \infty_{\ A < D}\ V$

A	U.B	U.C	V.B	V.C	D
1	2	3	2	3	4
1	2	3	2	3	5
6	7	8	7	8	10
9	7	8	7	8	10
1	2	3	7	8	10

Figure 18

IX. SUMMERY

SQL is based on a mathematical body of knowledge, Relational algebra (RA), which serves as an intermediate language for the DBMS. Essentially, when a declarative SQL statement is received, and parsed, it will be translated into an expression in RA. Such an expression is then analyzed and optimized by a query optimizer to become an equivalent but more efficient algorithm, or, a query execution plan. Such a plan is then converted to a piece of executable code. The mathematical nature of the relational algebra makes such analysis and optimization and proof of equivalence possible. Thus, RA is a key to understand the internal working of SQL statements, and is also helpful to design SQL statements. A relational algebraic expression consists of a combination of some simple operators. There are two groups of operators, some basic ones such as Select, Project, Union, Difference, Intersection, and Cartesian product; together with several derived ones: Join, and Division. We use a combination of these operators to come up with a data access program.



X. RECOMMENDATION

The huge success of relational databases has inspired the development of Structured Query Language (SQL) - a query language designed for interaction with relational databases. Over the past two decades, SQL has gradually evolved from its first commercial use into a computer product. SQL has been accepted as the industry standard for database programming language. It is used in systems of various sizes - from mainframes to personal computers and even handheld devices.

XI. CONTRIBUTION TO KNOWLEDGE

SQL is built on the concept of Relational Algebra. Relational Algebra can be seen as the mathematics which underpins the SQL operations and acts as a formal description on the behaviour of relational databases. It bears resemblance to normal algebra (as in x3 + y2) but uses relations as values instead of numbers. The inner, lower-level operations of relational databases are, or are similar to, Relational Algebra operations. Gaining expertise in Relational Algebra is the foundation needed for the student to effectively craft queries in any commercially available languages.

REFERENCES

Zhao, L., Sakr, S., & Liu, A. (2012). On the Spectrum of Web Scale Data Management. Cloud Computing: Methodology, Systems and Applications [M], 488-506.

Abadi, D. J., Madden, S. R., & Hachem, N. (2008, June). Column-stores vs. row-stores: How different are they really?. In Proceedings of the 2008 ACM SIGMOD international conference on Management of data (pp. 967-980). ACM.

Rahm, T. B. E. (2013). XML Data Management. Datenbanksysteme in Büro, Technik und Wissenschaft: 9. GI-Fachtagung Oldenburg, 7.-9. März 2001, 264.

Curé, O., Hecht, R., Le Duc, C., & Lamolle, M. (2011). Data integration over nosql stores using access path based mappings. In Database and Expert Systems Applications (pp. 481-495). Springer Berlin/Heidelberg.

Padhy, R. P., Patra, M. R., & Satapathy, S. C. (2011). RDBMS to NoSQL: reviewing some next-generation non-relational database's. International Journal of Advanced Engineering Science and Technologies, 11(1), 15-30.

Valduriez, P. (2011). Principles of distributed data management in 2020?. InDatabase and Expert Systems Applications (pp. 1-11). Springer Berlin/Heidelberg.

Borkar, V., & Carey, M. J. (2012, December). Big data technologies circa 2012. In Proceedings of the 18th International Conference on Management of Data (pp. 12-14). Computer Society of India.

Li, S. Y., Chang, C. M., Tsai, Y. Y., Chen, S., Tsai, J., & Tsai, W. L. (2013). A Distribution Management System for Relational Databases in Cloud Environments. Journal of Electronic Science and Technology, 2, 010.

Yuan, L. Y., Wu, L., You, J. H., & Chi, Y. (2014, November). Rubato DB: A highly scalable staged grid database system for OLTP and big data applications. In Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (pp. 1-10). ACM.