**DASC 5333**

10/15/2024

**Introduction to Relational Algebra and Relational Calculus**

by K. Yue

**1. Introduction (programming paradigm)**

* More theoretical query languages of the relational model:
  1. Relational Algebra (RA): a simple functional language
  2. Relational Calculus (RC): a simple declarative language: what?
  3. Procedural language: how (steps)? Algorithm, performance, optimization
* Provide theoretical foundation for the relational model.
* Not used in practical DBMS.
* Some symbols in RA/RC: [RA\_RC\_Symbols.docx](https://dcm.uhcl.edu/yue/courses/joinDB/Fall2024/notes/rarc/RA_RC_Symbols.docx)

**2. Introduction to Relational algebra**

* Include a set of *basic* and *derived* set-theoretic operations.
* Procedural: specify a sequence of operations.
  + performance optimization is important.
* Operations can be unary or binary.
* The result is also a relation: *closure* property => chained operations.
* RA solutions are algorithmic.

***Example:***

See [toyu\_Ex.pdf](https://dcm.uhcl.edu/yue/courses/joinDB/Fall2024/notes/rarc/toyu_Ex.pdf) and [toyu\_RA\_sol\_even.pdf](https://dcm.uhcl.edu/yue/courses/joinDB/Fall2024/notes/rarc/toyu_RA_sol_even.pdf)

**3. Introduction to Relational Calculus**

* Non-procedural, *declarative*, and high level.
* Two kinds:
  1. Domain Relational Calculus (DRC)
  2. Tuple Relational Calculus (TRC)
* Results specified by the *set builder form*: {s | cond(s)}
* cond(s) is known as a *formula*.
* Constructs:
  1. Variables:
     1. TRC: tuples (bound to tuples): e.g,. s, t, student, class, etc.
     2. DRC: Attributes (bound to domain value): e.g., a, b, c, stuId, fname, etc.
     3. RC's variables are sometime known as 'dummy variables'.
  2. Constants: string, int, etc. E.g., 12, 'csci', 3.7.
  3. Comparison operators: <, <, =, etc.
  4. Boolean operators: and (conjunction, ∧ or just ,), or (disjunction ∨), not (¬), implies (⇒), etc.
  5. Membership functions: belongs to, ∈, not belongs to, ∉, etc.
  6. Quantifiers: there exists (existential, ∃), for all (universal ∀).

***Example:***

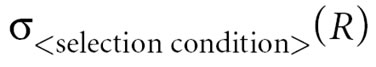
See [toyu\_Ex.pdf](https://dcm.uhcl.edu/yue/courses/joinDB/Fall2024/notes/rarc/toyu_Ex.pdf) and [toyu\_RC\_sol\_odd.pdf](https://dcm.uhcl.edu/yue/courses/joinDB/Fall2024/notes/rarc/toyu_RC_sol_odd.pdf)

**4. Relational Algebra**

**4.1 Basic Operations**

**4.1.1 Select**

1. Unary operation.
2. Select tuples (with the same schema) based on a Boolean condition.
3. Conditions may include attributes in the relational schema.
4. The Boolean expression of the condition can be composite (containing Boolean expressions joining by logical operators).
5. *'Horizontal' subset*.
6. Not to be confused with the Select statement in SQL.



RA: σcond(R) = definition in RC: {t | t ∈ R and cond}, or simply (TRC: tuple Relational Calculus)

σcond(R) = {t | t ∈ R, cond}

**Example:** All information of students majoring in CSCI.

σmajor='CSCI'(Student)

+--------+-------+-------+-------+-------+---------+---------+  
| stuId  | fname | lname | major | minor | credits | advisor |  
+--------+-------+-------+-------+-------+---------+---------+  
| 100000 | Tony  | Hawk  | CSCI  | CINF  |      40 |    1011 |  
| 100001 | Mary  | Hawk  | CSCI  | CINF  |      35 |    1011 |  
| 100002 | David | Hawk  | CSCI  | ITEC  |      66 |    1011 |  
+--------+-------+-------+-------+-------+---------+---------+  
3 rows  
  
In SQL, this is just:

SELECT \*  
FROM Student  
WHERE major = 'CSCI';

**4.1.2 Project**

1. Unary operation
2. Select attributes /columnsfrom tuples.
3. Duplicate results removed (because a relation is a set).
4. 'Vertical' subset.

πc1, .., cm(R) = {s | ∃t ∈ R (t(ci) = s(ci), for 1 <= i <= m)},

or simply

πc1, .., cm(R) = {s | t ∈ R (t(ci) = s(ci), for 1 <= i <= m)}

project

**Example:** Names and majors of students  
  
πLName, FName, Major(Student):

+-----------+---------+-------+  
| FName     | LName   | Major |  
+-----------+---------+-------+  
| Tony      | Hawk    | CSCI  |  
| Mary      | Hawk    | CSCI  |  
| David     | Hawk    | CSCI  |  
| Catherine | Lim     | ITEC  |  
| Larry     | Johnson | ITEC  |  
| Linda     | Johnson | CINF  |  
| Lillian   | Johnson | CINF  |  
| Ben       | Zico    | NULL  |  
| Bill      | Ching   | ARTS  |  
| Linda     | King    | ARTS  |  
+-----------+---------+-------+  
10 rows  
  
**4.1.3.Cartesian Product of sets**

1. Same as the usual definition of the Cartesian Product of two sets.
   1. Remember that a relation is a set.
2. Merge all possible information from two relations.
3. Also called Cross Product or Cross Join.
4. Name ambiguity may be resolved by using full names.
5. The cardinality of a set S is |S|, the number of elements in the set.
6. |RxS|= |R| \* |S|
7. Not very useful in practice as the result can be large and constructing the result can be time consuming.



***Example:***

R(A,B,C) has three tuples. S(A,D) has four tuples.

The result of R \* S always has 12 tuples with the schema (R.A, B, C, S.A, D).

**Example:** in toyu

+--------+-----------+---------+-------+-------+---------+---------+  
| stuId  | fname     | lname   | major | minor | credits | advisor |  
+--------+-----------+---------+-------+-------+---------+---------+  
| 100000 | Tony      | Hawk    | CSCI  | CINF  |      40 |    1011 |  
| 100001 | Mary      | Hawk    | CSCI  | CINF  |      35 |    1011 |  
| 100002 | David     | Hawk    | CSCI  | ITEC  |      66 |    1011 |  
| 100003 | Catherine | Lim     | ITEC  | CINF  |      20 |    1017 |  
| 100004 | Larry     | Johnson | ITEC  | NULL  |      66 |    1017 |  
| 100005 | Linda     | Johnson | CINF  | ENGL  |      13 |    1015 |  
| 100006 | Lillian   | Johnson | CINF  | ITEC  |      18 |    1015 |  
| 100007 | Ben       | Zico    | NULL  | NULL  |      16 |    NULL |  
| 100008 | Bill      | Ching   | ARTS  | ENGL  |      90 |    1018 |  
| 100009 | Linda     | King    | ARTS  | CSCI  |     125 |    1018 |  
+--------+-----------+---------+-------+-------+---------+---------+  
10 rows in set  
  
  
enroll:  
+--------+---------+-------+----------+  
| stuId  | classId | grade | n\_alerts |  
+--------+---------+-------+----------+  
| 100000 |   10000 | A     |        0 |  
| 100001 |   10000 | NULL  |     NULL |  
| 100002 |   10000 | B-    |        3 |  
| 100000 |   10001 | A     |        2 |  
| 100001 |   10001 | A-    |        0 |  
| 100000 |   10002 | B+    |        1 |  
| 100002 |   10002 | B+    |        2 |  
| 100000 |   10003 | C     |        0 |  
| 100002 |   10003 | D     |        4 |  
| 100004 |   10003 | A     |        0 |  
| 100005 |   10003 | NULL  |     NULL |  
| 100000 |   10004 | A-    |        1 |  
| 100004 |   10004 | B+    |     NULL |  
| 100005 |   10004 | A-    |        0 |  
| 100006 |   10004 | C+    |     NULL |  
| 100005 |   10005 | A-    |        0 |  
| 100006 |   10005 | A     |     NULL |  
| 100005 |   10006 | B+    |     NULL |  
| 100007 |   10007 | F     |        4 |  
| 100008 |   10007 | C-    |        0 |  
| 100007 |   10008 | A-    |        0 |  
| 100000 |   11001 | D     |        4 |  
+--------+---------+-------+----------+  
22 rows  
  
  
student \* enroll:  
+--------+-----------+---------+-------+-------+---------+---------+--------+---------+-------+----------+  
| stuId  | fname     | lname   | major | minor | credits | advisor | stuId  | classId | grade | n\_alerts |  
+--------+-----------+---------+-------+-------+---------+---------+--------+---------+-------+----------+  
| 100000 | Tony      | Hawk    | CSCI  | CINF  |      40 |    1011 | 100000 |   10000 | A     |        0 |  
| 100001 | Mary      | Hawk    | CSCI  | CINF  |      35 |    1011 | 100000 |   10000 | A     |        0 |  
| 100002 | David     | Hawk    | CSCI  | ITEC  |      66 |    1011 | 100000 |   10000 | A     |        0 |  
| 100003 | Catherine | Lim     | ITEC  | CINF  |      20 |    1017 | 100000 |   10000 | A     |        0 |  
| 100004 | Larry     | Johnson | ITEC  | NULL  |      66 |    1017 | 100000 |   10000 | A     |        0 |  
| 100005 | Linda     | Johnson | CINF  | ENGL  |      13 |    1015 | 100000 |   10000 | A     |        0 |  
| 100006 | Lillian   | Johnson | CINF  | ITEC  |      18 |    1015 | 100000 |   10000 | A     |        0 |  
| 100007 | Ben       | Zico    | NULL  | NULL  |      16 |    NULL | 100000 |   10000 | A     |        0 |  
| 100008 | Bill      | Ching   | ARTS  | ENGL  |      90 |    1018 | 100000 |   10000 | A     |        0 |  
| 100009 | Linda     | King    | ARTS  | CSCI  |     125 |    1018 | 100000 |   10000 | A     |        0 |  
| 100000 | Tony      | Hawk    | CSCI  | CINF  |      40 |    1011 | 100001 |   10000 | NULL  |     NULL |  
| 100001 | Mary      | Hawk    | CSCI  | CINF  |      35 |    1011 | 100001 |   10000 | NULL  |     NULL |  
| 100002 | David     | Hawk    | CSCI  | ITEC  |      66 |    1011 | 100001 |   10000 | NULL  |     NULL |  
| 100003 | Catherine | Lim     | ITEC  | CINF  |      20 |    1017 | 100001 |   10000 | NULL  |     NULL |  
| 100004 | Larry     | Johnson | ITEC  | NULL  |      66 |    1017 | 100001 |   10000 | NULL  |     NULL |  
| 100005 | Linda     | Johnson | CINF  | ENGL  |      13 |    1015 | 100001 |   10000 | NULL  |     NULL |  
| 100006 | Lillian   | Johnson | CINF  | ITEC  |      18 |    1015 | 100001 |   10000 | NULL  |     NULL |  
| 100007 | Ben       | Zico    | NULL  | NULL  |      16 |    NULL | 100001 |   10000 | NULL  |     NULL |  
| 100008 | Bill      | Ching   | ARTS  | ENGL  |      90 |    1018 | 100001 |   10000 | NULL  |     NULL |  
| 100009 | Linda     | King    | ARTS  | CSCI  |     125 |    1018 | 100001 |   10000 | NULL  |     NULL |  
| 100000 | Tony      | Hawk    | CSCI  | CINF  |      40 |    1011 | 100002 |   10000 | B-    |        3 |  
| 100001 | Mary      | Hawk    | CSCI  | CINF  |      35 |    1011 | 100002 |   10000 | B-    |        3 |  
| 100002 | David     | Hawk    | CSCI  | ITEC  |      66 |    1011 | 100002 |   10000 | B-    |        3 |  
| 100003 | Catherine | Lim     | ITEC  | CINF  |      20 |    1017 | 100002 |   10000 | B-    |        3 |  
| 100004 | Larry     | Johnson | ITEC  | NULL  |      66 |    1017 | 100002 |   10000 | B-    |        3 |  
| 100005 | Linda     | Johnson | CINF  | ENGL  |      13 |    1015 | 100002 |   10000 | B-    |        3 |  
| 100006 | Lillian   | Johnson | CINF  | ITEC  |      18 |    1015 | 100002 |   10000 | B-    |        3 |  
| 100007 | Ben       | Zico    | NULL  | NULL  |      16 |    NULL | 100002 |   10000 | B-    |        3 |  
| 100008 | Bill      | Ching   | ARTS  | ENGL  |      90 |    1018 | 100002 |   10000 | B-    |        3 |  
| 100009 | Linda     | King    | ARTS  | CSCI  |     125 |    1018 | 100002 |   10000 | B-    |        3 |  
| 100000 | Tony      | Hawk    | CSCI  | CINF  |      40 |    1011 | 100000 |   10001 | A     |        2 |  
| 100001 | Mary      | Hawk    | CSCI  | CINF  |      35 |    1011 | 100000 |   10001 | A     |        2 |  
| 100002 | David     | Hawk    | CSCI  | ITEC  |      66 |    1011 | 100000 |   10001 | A     |        2 |  
| 100003 | Catherine | Lim     | ITEC  | CINF  |      20 |    1017 | 100000 |   10001 | A     |        2 |  
| 100004 | Larry     | Johnson | ITEC  | NULL  |      66 |    1017 | 100000 |   10001 | A     |        2 |  
| 100005 | Linda     | Johnson | CINF  | ENGL  |      13 |    1015 | 100000 |   10001 | A     |        2 |  
| 100006 | Lillian   | Johnson | CINF  | ITEC  |      18 |    1015 | 100000 |   10001 | A     |        2 |  
| 100007 | Ben       | Zico    | NULL  | NULL  |      16 |    NULL | 100000 |   10001 | A     |        2 |  
| 100008 | Bill      | Ching   | ARTS  | ENGL  |      90 |    1018 | 100000 |   10001 | A     |        2 |  
| 100009 | Linda     | King    | ARTS  | CSCI  |     125 |    1018 | 100000 |   10001 | A     |        2 |  
| 100000 | Tony      | Hawk    | CSCI  | CINF  |      40 |    1011 | 100001 |   10001 | A-    |        0 |  
| 100001 | Mary      | Hawk    | CSCI  | CINF  |      35 |    1011 | 100001 |   10001 | A-    |        0 |  
| 100002 | David     | Hawk    | CSCI  | ITEC  |      66 |    1011 | 100001 |   10001 | A-    |        0 |  
| 100003 | Catherine | Lim     | ITEC  | CINF  |      20 |    1017 | 100001 |   10001 | A-    |        0 |  
| 100004 | Larry     | Johnson | ITEC  | NULL  |      66 |    1017 | 100001 |   10001 | A-    |        0 |  
| 100005 | Linda     | Johnson | CINF  | ENGL  |      13 |    1015 | 100001 |   10001 | A-    |        0 |  
| 100006 | Lillian   | Johnson | CINF  | ITEC  |      18 |    1015 | 100001 |   10001 | A-    |        0 |  
| 100007 | Ben       | Zico    | NULL  | NULL  |      16 |    NULL | 100001 |   10001 | A-    |        0 |  
| 100008 | Bill      | Ching   | ARTS  | ENGL  |      90 |    1018 | 100001 |   10001 | A-    |        0 |  
| 100009 | Linda     | King    | ARTS  | CSCI  |     125 |    1018 | 100001 |   10001 | A-    |        0 |  
| 100000 | Tony      | Hawk    | CSCI  | CINF  |      40 |    1011 | 100000 |   10002 | B+    |        1 |  
| 100001 | Mary      | Hawk    | CSCI  | CINF  |      35 |    1011 | 100000 |   10002 | B+    |        1 |  
| 100002 | David     | Hawk    | CSCI  | ITEC  |      66 |    1011 | 100000 |   10002 | B+    |        1 |  
| 100003 | Catherine | Lim     | ITEC  | CINF  |      20 |    1017 | 100000 |   10002 | B+    |        1 |  
| 100004 | Larry     | Johnson | ITEC  | NULL  |      66 |    1017 | 100000 |   10002 | B+    |        1 |  
| 100005 | Linda     | Johnson | CINF  | ENGL  |      13 |    1015 | 100000 |   10002 | B+    |        1 |  
| 100006 | Lillian   | Johnson | CINF  | ITEC  |      18 |    1015 | 100000 |   10002 | B+    |        1 |  
| 100007 | Ben       | Zico    | NULL  | NULL  |      16 |    NULL | 100000 |   10002 | B+    |        1 |  
| 100008 | Bill      | Ching   | ARTS  | ENGL  |      90 |    1018 | 100000 |   10002 | B+    |        1 |  
| 100009 | Linda     | King    | ARTS  | CSCI  |     125 |    1018 | 100000 |   10002 | B+    |        1 |  
| 100000 | Tony      | Hawk    | CSCI  | CINF  |      40 |    1011 | 100002 |   10002 | B+    |        2 |  
| 100001 | Mary      | Hawk    | CSCI  | CINF  |      35 |    1011 | 100002 |   10002 | B+    |        2 |  
| 100002 | David     | Hawk    | CSCI  | ITEC  |      66 |    1011 | 100002 |   10002 | B+    |        2 |  
| 100003 | Catherine | Lim     | ITEC  | CINF  |      20 |    1017 | 100002 |   10002 | B+    |        2 |  
| 100004 | Larry     | Johnson | ITEC  | NULL  |      66 |    1017 | 100002 |   10002 | B+    |        2 |  
| 100005 | Linda     | Johnson | CINF  | ENGL  |      13 |    1015 | 100002 |   10002 | B+    |        2 |  
| 100006 | Lillian   | Johnson | CINF  | ITEC  |      18 |    1015 | 100002 |   10002 | B+    |        2 |  
| 100007 | Ben       | Zico    | NULL  | NULL  |      16 |    NULL | 100002 |   10002 | B+    |        2 |  
| 100008 | Bill      | Ching   | ARTS  | ENGL  |      90 |    1018 | 100002 |   10002 | B+    |        2 |  
| 100009 | Linda     | King    | ARTS  | CSCI  |     125 |    1018 | 100002 |   10002 | B+    |        2 |  
| 100000 | Tony      | Hawk    | CSCI  | CINF  |      40 |    1011 | 100000 |   10003 | C     |        0 |  
| 100001 | Mary      | Hawk    | CSCI  | CINF  |      35 |    1011 | 100000 |   10003 | C     |        0 |  
| 100002 | David     | Hawk    | CSCI  | ITEC  |      66 |    1011 | 100000 |   10003 | C     |        0 |  
| 100003 | Catherine | Lim     | ITEC  | CINF  |      20 |    1017 | 100000 |   10003 | C     |        0 |  
| 100004 | Larry     | Johnson | ITEC  | NULL  |      66 |    1017 | 100000 |   10003 | C     |        0 |  
| 100005 | Linda     | Johnson | CINF  | ENGL  |      13 |    1015 | 100000 |   10003 | C     |        0 |  
| 100006 | Lillian   | Johnson | CINF  | ITEC  |      18 |    1015 | 100000 |   10003 | C     |        0 |  
| 100007 | Ben       | Zico    | NULL  | NULL  |      16 |    NULL | 100000 |   10003 | C     |        0 |  
| 100008 | Bill      | Ching   | ARTS  | ENGL  |      90 |    1018 | 100000 |   10003 | C     |        0 |  
| 100009 | Linda     | King    | ARTS  | CSCI  |     125 |    1018 | 100000 |   10003 | C     |        0 |  
| 100000 | Tony      | Hawk    | CSCI  | CINF  |      40 |    1011 | 100002 |   10003 | D     |        4 |  
| 100001 | Mary      | Hawk    | CSCI  | CINF  |      35 |    1011 | 100002 |   10003 | D     |        4 |  
| 100002 | David     | Hawk    | CSCI  | ITEC  |      66 |    1011 | 100002 |   10003 | D     |        4 |  
| 100003 | Catherine | Lim     | ITEC  | CINF  |      20 |    1017 | 100002 |   10003 | D     |        4 |  
| 100004 | Larry     | Johnson | ITEC  | NULL  |      66 |    1017 | 100002 |   10003 | D     |        4 |  
| 100005 | Linda     | Johnson | CINF  | ENGL  |      13 |    1015 | 100002 |   10003 | D     |        4 |  
| 100006 | Lillian   | Johnson | CINF  | ITEC  |      18 |    1015 | 100002 |   10003 | D     |        4 |  
| 100007 | Ben       | Zico    | NULL  | NULL  |      16 |    NULL | 100002 |   10003 | D     |        4 |  
| 100008 | Bill      | Ching   | ARTS  | ENGL  |      90 |    1018 | 100002 |   10003 | D     |        4 |  
| 100009 | Linda     | King    | ARTS  | CSCI  |     125 |    1018 | 100002 |   10003 | D     |        4 |  
| 100000 | Tony      | Hawk    | CSCI  | CINF  |      40 |    1011 | 100004 |   10003 | A     |        0 |  
| 100001 | Mary      | Hawk    | CSCI  | CINF  |      35 |    1011 | 100004 |   10003 | A     |        0 |  
| 100002 | David     | Hawk    | CSCI  | ITEC  |      66 |    1011 | 100004 |   10003 | A     |        0 |  
| 100003 | Catherine | Lim     | ITEC  | CINF  |      20 |    1017 | 100004 |   10003 | A     |        0 |  
| 100004 | Larry     | Johnson | ITEC  | NULL  |      66 |    1017 | 100004 |   10003 | A     |        0 |  
| 100005 | Linda     | Johnson | CINF  | ENGL  |      13 |    1015 | 100004 |   10003 | A     |        0 |  
| 100006 | Lillian   | Johnson | CINF  | ITEC  |      18 |    1015 | 100004 |   10003 | A     |        0 |  
| 100007 | Ben       | Zico    | NULL  | NULL  |      16 |    NULL | 100004 |   10003 | A     |        0 |  
| 100008 | Bill      | Ching   | ARTS  | ENGL  |      90 |    1018 | 100004 |   10003 | A     |        0 |  
| 100009 | Linda     | King    | ARTS  | CSCI  |     125 |    1018 | 100004 |   10003 | A     |        0 |  
| 100000 | Tony      | Hawk    | CSCI  | CINF  |      40 |    1011 | 100005 |   10003 | NULL  |     NULL |  
| 100001 | Mary      | Hawk    | CSCI  | CINF  |      35 |    1011 | 100005 |   10003 | NULL  |     NULL |  
| 100002 | David     | Hawk    | CSCI  | ITEC  |      66 |    1011 | 100005 |   10003 | NULL  |     NULL |  
| 100003 | Catherine | Lim     | ITEC  | CINF  |      20 |    1017 | 100005 |   10003 | NULL  |     NULL |  
| 100004 | Larry     | Johnson | ITEC  | NULL  |      66 |    1017 | 100005 |   10003 | NULL  |     NULL |  
| 100005 | Linda     | Johnson | CINF  | ENGL  |      13 |    1015 | 100005 |   10003 | NULL  |     NULL |  
| 100006 | Lillian   | Johnson | CINF  | ITEC  |      18 |    1015 | 100005 |   10003 | NULL  |     NULL |  
| 100007 | Ben       | Zico    | NULL  | NULL  |      16 |    NULL | 100005 |   10003 | NULL  |     NULL |  
| 100008 | Bill      | Ching   | ARTS  | ENGL  |      90 |    1018 | 100005 |   10003 | NULL  |     NULL |  
| 100009 | Linda     | King    | ARTS  | CSCI  |     125 |    1018 | 100005 |   10003 | NULL  |     NULL |  
| 100000 | Tony      | Hawk    | CSCI  | CINF  |      40 |    1011 | 100000 |   10004 | A-    |        1 |  
| 100001 | Mary      | Hawk    | CSCI  | CINF  |      35 |    1011 | 100000 |   10004 | A-    |        1 |  
| 100002 | David     | Hawk    | CSCI  | ITEC  |      66 |    1011 | 100000 |   10004 | A-    |        1 |  
| 100003 | Catherine | Lim     | ITEC  | CINF  |      20 |    1017 | 100000 |   10004 | A-    |        1 |  
| 100004 | Larry     | Johnson | ITEC  | NULL  |      66 |    1017 | 100000 |   10004 | A-    |        1 |  
| 100005 | Linda     | Johnson | CINF  | ENGL  |      13 |    1015 | 100000 |   10004 | A-    |        1 |  
| 100006 | Lillian   | Johnson | CINF  | ITEC  |      18 |    1015 | 100000 |   10004 | A-    |        1 |  
| 100007 | Ben       | Zico    | NULL  | NULL  |      16 |    NULL | 100000 |   10004 | A-    |        1 |  
| 100008 | Bill      | Ching   | ARTS  | ENGL  |      90 |    1018 | 100000 |   10004 | A-    |        1 |  
| 100009 | Linda     | King    | ARTS  | CSCI  |     125 |    1018 | 100000 |   10004 | A-    |        1 |  
| 100000 | Tony      | Hawk    | CSCI  | CINF  |      40 |    1011 | 100004 |   10004 | B+    |     NULL |  
| 100001 | Mary      | Hawk    | CSCI  | CINF  |      35 |    1011 | 100004 |   10004 | B+    |     NULL |  
| 100002 | David     | Hawk    | CSCI  | ITEC  |      66 |    1011 | 100004 |   10004 | B+    |     NULL |  
| 100003 | Catherine | Lim     | ITEC  | CINF  |      20 |    1017 | 100004 |   10004 | B+    |     NULL |  
| 100004 | Larry     | Johnson | ITEC  | NULL  |      66 |    1017 | 100004 |   10004 | B+    |     NULL |  
| 100005 | Linda     | Johnson | CINF  | ENGL  |      13 |    1015 | 100004 |   10004 | B+    |     NULL |  
| 100006 | Lillian   | Johnson | CINF  | ITEC  |      18 |    1015 | 100004 |   10004 | B+    |     NULL |  
| 100007 | Ben       | Zico    | NULL  | NULL  |      16 |    NULL | 100004 |   10004 | B+    |     NULL |  
| 100008 | Bill      | Ching   | ARTS  | ENGL  |      90 |    1018 | 100004 |   10004 | B+    |     NULL |  
| 100009 | Linda     | King    | ARTS  | CSCI  |     125 |    1018 | 100004 |   10004 | B+    |     NULL |  
| 100000 | Tony      | Hawk    | CSCI  | CINF  |      40 |    1011 | 100005 |   10004 | A-    |        0 |  
| 100001 | Mary      | Hawk    | CSCI  | CINF  |      35 |    1011 | 100005 |   10004 | A-    |        0 |  
| 100002 | David     | Hawk    | CSCI  | ITEC  |      66 |    1011 | 100005 |   10004 | A-    |        0 |  
| 100003 | Catherine | Lim     | ITEC  | CINF  |      20 |    1017 | 100005 |   10004 | A-    |        0 |  
| 100004 | Larry     | Johnson | ITEC  | NULL  |      66 |    1017 | 100005 |   10004 | A-    |        0 |  
| 100005 | Linda     | Johnson | CINF  | ENGL  |      13 |    1015 | 100005 |   10004 | A-    |        0 |  
| 100006 | Lillian   | Johnson | CINF  | ITEC  |      18 |    1015 | 100005 |   10004 | A-    |        0 |  
| 100007 | Ben       | Zico    | NULL  | NULL  |      16 |    NULL | 100005 |   10004 | A-    |        0 |  
| 100008 | Bill      | Ching   | ARTS  | ENGL  |      90 |    1018 | 100005 |   10004 | A-    |        0 |  
| 100009 | Linda     | King    | ARTS  | CSCI  |     125 |    1018 | 100005 |   10004 | A-    |        0 |  
| 100000 | Tony      | Hawk    | CSCI  | CINF  |      40 |    1011 | 100006 |   10004 | C+    |     NULL |  
| 100001 | Mary      | Hawk    | CSCI  | CINF  |      35 |    1011 | 100006 |   10004 | C+    |     NULL |  
| 100002 | David     | Hawk    | CSCI  | ITEC  |      66 |    1011 | 100006 |   10004 | C+    |     NULL |  
| 100003 | Catherine | Lim     | ITEC  | CINF  |      20 |    1017 | 100006 |   10004 | C+    |     NULL |  
| 100004 | Larry     | Johnson | ITEC  | NULL  |      66 |    1017 | 100006 |   10004 | C+    |     NULL |  
| 100005 | Linda     | Johnson | CINF  | ENGL  |      13 |    1015 | 100006 |   10004 | C+    |     NULL |  
| 100006 | Lillian   | Johnson | CINF  | ITEC  |      18 |    1015 | 100006 |   10004 | C+    |     NULL |  
| 100007 | Ben       | Zico    | NULL  | NULL  |      16 |    NULL | 100006 |   10004 | C+    |     NULL |  
| 100008 | Bill      | Ching   | ARTS  | ENGL  |      90 |    1018 | 100006 |   10004 | C+    |     NULL |  
| 100009 | Linda     | King    | ARTS  | CSCI  |     125 |    1018 | 100006 |   10004 | C+    |     NULL |  
| 100000 | Tony      | Hawk    | CSCI  | CINF  |      40 |    1011 | 100005 |   10005 | A-    |        0 |  
| 100001 | Mary      | Hawk    | CSCI  | CINF  |      35 |    1011 | 100005 |   10005 | A-    |        0 |  
| 100002 | David     | Hawk    | CSCI  | ITEC  |      66 |    1011 | 100005 |   10005 | A-    |        0 |  
| 100003 | Catherine | Lim     | ITEC  | CINF  |      20 |    1017 | 100005 |   10005 | A-    |        0 |  
| 100004 | Larry     | Johnson | ITEC  | NULL  |      66 |    1017 | 100005 |   10005 | A-    |        0 |  
| 100005 | Linda     | Johnson | CINF  | ENGL  |      13 |    1015 | 100005 |   10005 | A-    |        0 |  
| 100006 | Lillian   | Johnson | CINF  | ITEC  |      18 |    1015 | 100005 |   10005 | A-    |        0 |  
| 100007 | Ben       | Zico    | NULL  | NULL  |      16 |    NULL | 100005 |   10005 | A-    |        0 |  
| 100008 | Bill      | Ching   | ARTS  | ENGL  |      90 |    1018 | 100005 |   10005 | A-    |        0 |  
| 100009 | Linda     | King    | ARTS  | CSCI  |     125 |    1018 | 100005 |   10005 | A-    |        0 |  
| 100000 | Tony      | Hawk    | CSCI  | CINF  |      40 |    1011 | 100006 |   10005 | A     |     NULL |  
| 100001 | Mary      | Hawk    | CSCI  | CINF  |      35 |    1011 | 100006 |   10005 | A     |     NULL |  
| 100002 | David     | Hawk    | CSCI  | ITEC  |      66 |    1011 | 100006 |   10005 | A     |     NULL |  
| 100003 | Catherine | Lim     | ITEC  | CINF  |      20 |    1017 | 100006 |   10005 | A     |     NULL |  
| 100004 | Larry     | Johnson | ITEC  | NULL  |      66 |    1017 | 100006 |   10005 | A     |     NULL |  
| 100005 | Linda     | Johnson | CINF  | ENGL  |      13 |    1015 | 100006 |   10005 | A     |     NULL |  
| 100006 | Lillian   | Johnson | CINF  | ITEC  |      18 |    1015 | 100006 |   10005 | A     |     NULL |  
| 100007 | Ben       | Zico    | NULL  | NULL  |      16 |    NULL | 100006 |   10005 | A     |     NULL |  
| 100008 | Bill      | Ching   | ARTS  | ENGL  |      90 |    1018 | 100006 |   10005 | A     |     NULL |  
| 100009 | Linda     | King    | ARTS  | CSCI  |     125 |    1018 | 100006 |   10005 | A     |     NULL |  
| 100000 | Tony      | Hawk    | CSCI  | CINF  |      40 |    1011 | 100005 |   10006 | B+    |     NULL |  
| 100001 | Mary      | Hawk    | CSCI  | CINF  |      35 |    1011 | 100005 |   10006 | B+    |     NULL |  
| 100002 | David     | Hawk    | CSCI  | ITEC  |      66 |    1011 | 100005 |   10006 | B+    |     NULL |  
| 100003 | Catherine | Lim     | ITEC  | CINF  |      20 |    1017 | 100005 |   10006 | B+    |     NULL |  
| 100004 | Larry     | Johnson | ITEC  | NULL  |      66 |    1017 | 100005 |   10006 | B+    |     NULL |  
| 100005 | Linda     | Johnson | CINF  | ENGL  |      13 |    1015 | 100005 |   10006 | B+    |     NULL |  
| 100006 | Lillian   | Johnson | CINF  | ITEC  |      18 |    1015 | 100005 |   10006 | B+    |     NULL |  
| 100007 | Ben       | Zico    | NULL  | NULL  |      16 |    NULL | 100005 |   10006 | B+    |     NULL |  
| 100008 | Bill      | Ching   | ARTS  | ENGL  |      90 |    1018 | 100005 |   10006 | B+    |     NULL |  
| 100009 | Linda     | King    | ARTS  | CSCI  |     125 |    1018 | 100005 |   10006 | B+    |     NULL |  
| 100000 | Tony      | Hawk    | CSCI  | CINF  |      40 |    1011 | 100007 |   10007 | F     |        4 |  
| 100001 | Mary      | Hawk    | CSCI  | CINF  |      35 |    1011 | 100007 |   10007 | F     |        4 |  
| 100002 | David     | Hawk    | CSCI  | ITEC  |      66 |    1011 | 100007 |   10007 | F     |        4 |  
| 100003 | Catherine | Lim     | ITEC  | CINF  |      20 |    1017 | 100007 |   10007 | F     |        4 |  
| 100004 | Larry     | Johnson | ITEC  | NULL  |      66 |    1017 | 100007 |   10007 | F     |        4 |  
| 100005 | Linda     | Johnson | CINF  | ENGL  |      13 |    1015 | 100007 |   10007 | F     |        4 |  
| 100006 | Lillian   | Johnson | CINF  | ITEC  |      18 |    1015 | 100007 |   10007 | F     |        4 |  
| 100007 | Ben       | Zico    | NULL  | NULL  |      16 |    NULL | 100007 |   10007 | F     |        4 |  
| 100008 | Bill      | Ching   | ARTS  | ENGL  |      90 |    1018 | 100007 |   10007 | F     |        4 |  
| 100009 | Linda     | King    | ARTS  | CSCI  |     125 |    1018 | 100007 |   10007 | F     |        4 |  
| 100000 | Tony      | Hawk    | CSCI  | CINF  |      40 |    1011 | 100008 |   10007 | C-    |        0 |  
| 100001 | Mary      | Hawk    | CSCI  | CINF  |      35 |    1011 | 100008 |   10007 | C-    |        0 |  
| 100002 | David     | Hawk    | CSCI  | ITEC  |      66 |    1011 | 100008 |   10007 | C-    |        0 |  
| 100003 | Catherine | Lim     | ITEC  | CINF  |      20 |    1017 | 100008 |   10007 | C-    |        0 |  
| 100004 | Larry     | Johnson | ITEC  | NULL  |      66 |    1017 | 100008 |   10007 | C-    |        0 |  
| 100005 | Linda     | Johnson | CINF  | ENGL  |      13 |    1015 | 100008 |   10007 | C-    |        0 |  
| 100006 | Lillian   | Johnson | CINF  | ITEC  |      18 |    1015 | 100008 |   10007 | C-    |        0 |  
| 100007 | Ben       | Zico    | NULL  | NULL  |      16 |    NULL | 100008 |   10007 | C-    |        0 |  
| 100008 | Bill      | Ching   | ARTS  | ENGL  |      90 |    1018 | 100008 |   10007 | C-    |        0 |  
| 100009 | Linda     | King    | ARTS  | CSCI  |     125 |    1018 | 100008 |   10007 | C-    |        0 |  
| 100000 | Tony      | Hawk    | CSCI  | CINF  |      40 |    1011 | 100007 |   10008 | A-    |        0 |  
| 100001 | Mary      | Hawk    | CSCI  | CINF  |      35 |    1011 | 100007 |   10008 | A-    |        0 |  
| 100002 | David     | Hawk    | CSCI  | ITEC  |      66 |    1011 | 100007 |   10008 | A-    |        0 |  
| 100003 | Catherine | Lim     | ITEC  | CINF  |      20 |    1017 | 100007 |   10008 | A-    |        0 |  
| 100004 | Larry     | Johnson | ITEC  | NULL  |      66 |    1017 | 100007 |   10008 | A-    |        0 |  
| 100005 | Linda     | Johnson | CINF  | ENGL  |      13 |    1015 | 100007 |   10008 | A-    |        0 |  
| 100006 | Lillian   | Johnson | CINF  | ITEC  |      18 |    1015 | 100007 |   10008 | A-    |        0 |  
| 100007 | Ben       | Zico    | NULL  | NULL  |      16 |    NULL | 100007 |   10008 | A-    |        0 |  
| 100008 | Bill      | Ching   | ARTS  | ENGL  |      90 |    1018 | 100007 |   10008 | A-    |        0 |  
| 100009 | Linda     | King    | ARTS  | CSCI  |     125 |    1018 | 100007 |   10008 | A-    |        0 |  
| 100000 | Tony      | Hawk    | CSCI  | CINF  |      40 |    1011 | 100000 |   11001 | D     |        4 |  
| 100001 | Mary      | Hawk    | CSCI  | CINF  |      35 |    1011 | 100000 |   11001 | D     |        4 |  
| 100002 | David     | Hawk    | CSCI  | ITEC  |      66 |    1011 | 100000 |   11001 | D     |        4 |  
| 100003 | Catherine | Lim     | ITEC  | CINF  |      20 |    1017 | 100000 |   11001 | D     |        4 |  
| 100004 | Larry     | Johnson | ITEC  | NULL  |      66 |    1017 | 100000 |   11001 | D     |        4 |  
| 100005 | Linda     | Johnson | CINF  | ENGL  |      13 |    1015 | 100000 |   11001 | D     |        4 |  
| 100006 | Lillian   | Johnson | CINF  | ITEC  |      18 |    1015 | 100000 |   11001 | D     |        4 |  
| 100007 | Ben       | Zico    | NULL  | NULL  |      16 |    NULL | 100000 |   11001 | D     |        4 |  
| 100008 | Bill      | Ching   | ARTS  | ENGL  |      90 |    1018 | 100000 |   11001 | D     |        4 |  
| 100009 | Linda     | King    | ARTS  | CSCI  |     125 |    1018 | 100000 |   11001 | D     |        4 |  
+--------+-----------+---------+-------+-------+---------+---------+--------+---------+-------+----------+  
220 row  
  
R \* S in SQL:

SELECT R.\*, S.\*  
FROM R, S; -- note that there is no join condition.

**4.1.4 Union**

1. The set union operator.
2. Condition for R U S: R and S must be *union compatible*. Their relation schema must have compatible schema with the same structures. Each corresponding attribute must have the same types (domains).

R U S = {t | t ∈ R V t ∈ S}

**Example:**

Suppose StaffID and FacultyID are union compatible.

 πStaffID(Staff) U πFacultyID(Faculty)

**Example:** All information of students majoring in CSCI or ARTS.

σ(major='CSCI') (Student) U σ(major='ARTS') (Student)

or

σ(major='CSCI') V (major='ARTS') (Student)

+--------+-------+-------+-------+-------+---------+---------+  
| stuId  | fname | lname | major | minor | credits | advisor |  
+--------+-------+-------+-------+-------+---------+---------+  
| 100008 | Bill  | Ching | ARTS  | ENGL  |      90 |    1018 |  
| 100009 | Linda | King  | ARTS  | CSCI  |     125 |    1018 |  
| 100000 | Tony  | Hawk  | CSCI  | CINF  |      40 |    1011 |  
| 100001 | Mary  | Hawk  | CSCI  | CINF  |      35 |    1011 |  
| 100002 | David | Hawk  | CSCI  | ITEC  |      66 |    1011 |  
+--------+-------+-------+-------+-------+---------+---------+  
5 rows

**4.1.5 Difference (Minus)**

1. The set difference operator.
2. R - S: R and S must be *union compatible*.

R - S = {t | t ∈ R and not (t ∈ S)}

or

R - S = {t | t ∈ R, t ∉ S}

**Example:** Information of all students majoring in CSCI but not those taken credits less than 40.

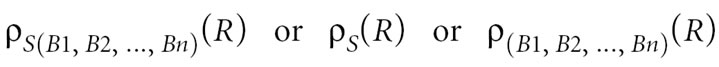
σmajor='CSCI'(Student) - σcredit <40 (Student)

+--------+-------+-------+-------+-------+---------+---------+  
| stuId  | fname | lname | major | minor | credits | advisor |  
+--------+-------+-------+-------+-------+---------+---------+  
| 100000 | Tony  | Hawk  | CSCI  | CINF  |      40 |    1011 |  
| 100002 | David | Hawk  | CSCI  | ITEC  |      66 |    1011 |  
+--------+-------+-------+-------+-------+---------+---------+  
2 rows  
  
Note that this is the same as:

σmajor='CSCI' and credit >=40(Student)

**4.1.6 Rename**

1. Rename the names of selected attributes in a relation.
2. Maybe used to rename attributes before a set operation.
3. Notation in Elmarsi (a popular db textbook):



* A better notation includes the original name and the new name.

Example:

ρ(FacultyId, department <- FacId, deptCode) (Faculty)

SELECT facId AS facultyId, fname, lname, deptCode AS department, `rank`

FROM Faculty;  
  
+-----------+----------+----------+------------+---------------------+  
| facultyId | fname    | lname    | department | rank                |  
+-----------+----------+----------+------------+---------------------+  
|      1011 | Paul     | Smith    | CSCI       | Professor           |  
|      1012 | Mary     | Tran     | CSCI       | Associate Professor |  
|      1013 | David    | Love     | CSCI       | NULL                |  
|      1014 | Sharon   | Mannes   | CSCI       | Assistant Professor |  
|      1015 | Daniel   | Kim      | CINF       | Professor           |  
|      1016 | Andrew   | Byre     | CINF       | Associate Professor |  
|      1017 | Deborah  | Gump     | ITEC       | Professor           |  
|      1018 | Art      | Allister | ARTS       | Assistant Professor |  
|      1019 | Benjamin | Yu       | ITEC       | Lecturer            |  
|      1020 | Katrina  | Bajaj    | ENGL       | Lecturer            |  
|      1021 | Jorginlo | Neymar   | ACCT       | Assistant Professor |  
+-----------+----------+----------+------------+---------------------+  
11 rows

* The basic set of operations is *complete*. Other relational algebra operations can be derived from them.

12. Show the student names and their major names for all students who have received a grade A in a class offered by a faculty from the CSCI department.

(12) π fName, lName, deptName(student |x| ρmajor <- deptCode(department) |x| σgrade=’A’(enroll) |x| class |x| σdeptCode=’CSCI’(faculty))

Common attributes in RA: same names.

Columns with the same meaning: major, deptCode

**4.2. Derived Operations**

**4.2.1 Theta-join**

1. Allow the application of condition on Cartesian product.
2. There are still redundant data on common attributes.
3. Allow the query engine to throw away tuples not in the result immediately.
4. Conceptually, a Cartesian Product followed by a selection Θ.
5. Not usually used.

R1 ⋈ΘR2 = σΘ(R1 \* R2)

**Example:** All related information of students with 70 or more credits and a grade A or better in some courses.

Student ⋈(credits >= 70 and grade = 'A') Enroll

+--------+-------+-------+-------+-------+---------+---------+--------+---------+-------+----------+  
| stuId  | fname | lname | major | minor | credits | advisor | stuId  | classId | grade | n\_alerts |  
+--------+-------+-------+-------+-------+---------+---------+--------+---------+-------+----------+  
| 100008 | Bill  | Ching | ARTS  | ENGL  |      90 |    1018 | 100000 |   10000 | A     |        0 |  
| 100008 | Bill  | Ching | ARTS  | ENGL  |      90 |    1018 | 100000 |   10001 | A     |        2 |  
| 100008 | Bill  | Ching | ARTS  | ENGL  |      90 |    1018 | 100004 |   10003 | A     |        0 |  
| 100008 | Bill  | Ching | ARTS  | ENGL  |      90 |    1018 | 100006 |   10005 | A     |     NULL |  
| 100009 | Linda | King  | ARTS  | CSCI  |     125 |    1018 | 100000 |   10000 | A     |        0 |  
| 100009 | Linda | King  | ARTS  | CSCI  |     125 |    1018 | 100000 |   10001 | A     |        2 |  
| 100009 | Linda | King  | ARTS  | CSCI  |     125 |    1018 | 100004 |   10003 | A     |        0 |  
| 100009 | Linda | King  | ARTS  | CSCI  |     125 |    1018 | 100006 |   10005 | A     |     NULL |  
+--------+-------+-------+-------+-------+---------+---------+--------+---------+-------+----------+  
8 rows in set (0.01 sec)  
  
  
 **4.2.2 Equi-join**

1. Theta-join where the condition involves only equality comparisons.
2. There are still redundant data on *common attributes*.
3. Common attributes are attributes that have the same *names*. The attributes may not have the same meaning.
4. Not usually used.

**Example:**

Student |x| (Student.StuId = Enrol.StuId) Enroll

+--------+---------+---------+-------+-------+---------+---------+--------+---------+-------+----------+  
| stuId  | fname   | lname   | major | minor | credits | advisor | stuId  | classId | grade | n\_alerts |  
+--------+---------+---------+-------+-------+---------+---------+--------+---------+-------+----------+  
| 100000 | Tony    | Hawk    | CSCI  | CINF  |      40 |    1011 | 100000 |   10000 | A     |        0 |  
| 100000 | Tony    | Hawk    | CSCI  | CINF  |      40 |    1011 | 100000 |   10001 | A     |        2 |  
| 100000 | Tony    | Hawk    | CSCI  | CINF  |      40 |    1011 | 100000 |   10002 | B+    |        1 |  
| 100000 | Tony    | Hawk    | CSCI  | CINF  |      40 |    1011 | 100000 |   10003 | C     |        0 |  
| 100000 | Tony    | Hawk    | CSCI  | CINF  |      40 |    1011 | 100000 |   10004 | A-    |        1 |  
| 100000 | Tony    | Hawk    | CSCI  | CINF  |      40 |    1011 | 100000 |   11001 | D     |        4 |  
| 100001 | Mary    | Hawk    | CSCI  | CINF  |      35 |    1011 | 100001 |   10000 | NULL  |     NULL |  
| 100001 | Mary    | Hawk    | CSCI  | CINF  |      35 |    1011 | 100001 |   10001 | A-    |        0 |  
| 100002 | David   | Hawk    | CSCI  | ITEC  |      66 |    1011 | 100002 |   10000 | B-    |        3 |  
| 100002 | David   | Hawk    | CSCI  | ITEC  |      66 |    1011 | 100002 |   10002 | B+    |        2 |  
| 100002 | David   | Hawk    | CSCI  | ITEC  |      66 |    1011 | 100002 |   10003 | D     |        4 |  
| 100004 | Larry   | Johnson | ITEC  | NULL  |      66 |    1017 | 100004 |   10003 | A     |        0 |  
| 100004 | Larry   | Johnson | ITEC  | NULL  |      66 |    1017 | 100004 |   10004 | B+    |     NULL |  
| 100005 | Linda   | Johnson | CINF  | ENGL  |      13 |    1015 | 100005 |   10003 | NULL  |     NULL |  
| 100005 | Linda   | Johnson | CINF  | ENGL  |      13 |    1015 | 100005 |   10004 | A-    |        0 |  
| 100005 | Linda   | Johnson | CINF  | ENGL  |      13 |    1015 | 100005 |   10005 | A-    |        0 |  
| 100005 | Linda   | Johnson | CINF  | ENGL  |      13 |    1015 | 100005 |   10006 | B+    |     NULL |  
| 100006 | Lillian | Johnson | CINF  | ITEC  |      18 |    1015 | 100006 |   10004 | C+    |     NULL |  
| 100006 | Lillian | Johnson | CINF  | ITEC  |      18 |    1015 | 100006 |   10005 | A     |     NULL |  
| 100007 | Ben     | Zico    | NULL  | NULL  |      16 |    NULL | 100007 |   10007 | F     |        4 |  
| 100007 | Ben     | Zico    | NULL  | NULL  |      16 |    NULL | 100007 |   10008 | A-    |        0 |  
| 100008 | Bill    | Ching   | ARTS  | ENGL  |      90 |    1018 | 100008 |   10007 | C-    |        0 |  
+--------+---------+---------+-------+-------+---------+---------+--------+---------+-------+----------+  
22 rows

It is important to note the difference between names and meanings. Consider

student(stuId, ... advisorFacId, ..., createTime) and

faculty(facId, ..., createTime)

1. The attributes createTime in student and faculty have the same name, but different meaning.
   1. Student(createTime) is the time the student row is inserted into the student table.
   2. Faculty(createTime) is the time the faculty row is inserted into the faculty table
2. student(advisorFacId) and faculty(facId) have different names but the same meaning. In fact, student(advisorFacId) is a foreign key that references faculty(facId).

**4.2.3 Natural Join**

1. Remove redundant common attributes from equi-joins
   1. Equi-join on all common attributes.
   2. Projection to remove redundant common attributes.
2. Used very frequently to combine two tables.
3. If two relations do not share any common attributes, their natural join is the same as their Cartesian Product.

8. Show the names of students who have enrolled in 10000.

(8) π fName, lName(student |x| σclassId = 10000(enroll))

Performance optimization: minimize [1] # of operations; [2] sizes of the intermediate results.

π fName, lName(student |x| σclassId = 10000(enroll)) -- [1] # of operations: 3; σclassId = 10000(enroll)) : 3 rows. sizes of the intermediate results:small

π fName, lName(σclassId = 10000(student |x| enroll)) -- [2] # of operations: 3; sizes of the intermediate results: large, student |x| enroll: 242 tuples in our case.

π fName, lName(σclassId = 10000(student |x| π stuId (enroll))) -- [3] error

π fName, lName(student |x| σclassId = 10000(π stuId (enroll))) -- [4] error

π fName, lName(student |x| π stuId (σclassId = 10000(enroll))) -- [5] # of operations: 4; sizes of the intermediate results: smallest

Let C1, C2, ... Cm be the common attributes of R and S.

R |x| S = πA1, A2, .. Al(σR.C1=S.C1,.., R.Cm=S.Cm(R\*S))

where A1, A2, ... Al is the list of attributes in R\*S except S.C1, S.C2,.. S.Cm.

**Example:**

The schema of R(A,B) |x| S(A,C) is ABC. The schema of R(A,B) \* S(A,C) is {R.A, B, S.A, C}.

**Example:**

Student |x| Enroll:

+--------+---------+---------+-------+-------+---------+---------+---------+-------+----------+  
| stuId  | fname   | lname   | major | minor | credits | advisor | classId | grade | n\_alerts |  
+--------+---------+---------+-------+-------+---------+---------+---------+-------+----------+  
| 100000 | Tony    | Hawk    | CSCI  | CINF  |      40 |    1011 |   10000 | A     |        0 |  
| 100000 | Tony    | Hawk    | CSCI  | CINF  |      40 |    1011 |   10001 | A     |        2 |  
| 100000 | Tony    | Hawk    | CSCI  | CINF  |      40 |    1011 |   10002 | B+    |        1 |  
| 100000 | Tony    | Hawk    | CSCI  | CINF  |      40 |    1011 |   10003 | C     |        0 |  
| 100000 | Tony    | Hawk    | CSCI  | CINF  |      40 |    1011 |   10004 | A-    |        1 |  
| 100000 | Tony    | Hawk    | CSCI  | CINF  |      40 |    1011 |   11001 | D     |        4 |  
| 100001 | Mary    | Hawk    | CSCI  | CINF  |      35 |    1011 |   10000 | NULL  |     NULL |  
| 100001 | Mary    | Hawk    | CSCI  | CINF  |      35 |    1011 |   10001 | A-    |        0 |  
| 100002 | David   | Hawk    | CSCI  | ITEC  |      66 |    1011 |   10000 | B-    |        3 |  
| 100002 | David   | Hawk    | CSCI  | ITEC  |      66 |    1011 |   10002 | B+    |        2 |  
| 100002 | David   | Hawk    | CSCI  | ITEC  |      66 |    1011 |   10003 | D     |        4 |  
| 100004 | Larry   | Johnson | ITEC  | NULL  |      66 |    1017 |   10003 | A     |        0 |  
| 100004 | Larry   | Johnson | ITEC  | NULL  |      66 |    1017 |   10004 | B+    |     NULL |  
| 100005 | Linda   | Johnson | CINF  | ENGL  |      13 |    1015 |   10003 | NULL  |     NULL |  
| 100005 | Linda   | Johnson | CINF  | ENGL  |      13 |    1015 |   10004 | A-    |        0 |  
| 100005 | Linda   | Johnson | CINF  | ENGL  |      13 |    1015 |   10005 | A-    |        0 |  
| 100005 | Linda   | Johnson | CINF  | ENGL  |      13 |    1015 |   10006 | B+    |     NULL |  
| 100006 | Lillian | Johnson | CINF  | ITEC  |      18 |    1015 |   10004 | C+    |     NULL |  
| 100006 | Lillian | Johnson | CINF  | ITEC  |      18 |    1015 |   10005 | A     |     NULL |  
| 100007 | Ben     | Zico    | NULL  | NULL  |      16 |    NULL |   10007 | F     |        4 |  
| 100007 | Ben     | Zico    | NULL  | NULL  |      16 |    NULL |   10008 | A-    |        0 |  
| 100008 | Bill    | Ching   | ARTS  | ENGL  |      90 |    1018 |   10007 | C-    |        0 |  
+--------+---------+---------+-------+-------+---------+---------+---------+-------+----------+  
22 rows  
  
**Exercise:**

Let the cardinality of R(A,B) be 5 and the cardinality of S(A,C) be 6. What is the range of the cardinality of R(A,B) |x| S(A,C)?

**4.2.4 Other Joins** (Additional Materials)

1. Some other joins are left join, right join, outer join, inner join and semi-join.
2. They can be defined through relational algebra expressions based on the basic operations.
3. Look them up when needs arise. For example: <https://en.wikipedia.org/wiki/Relational_algebra>

**4.2.5 Division** (Additional Materials)

1. R / S or R ÷ S.
2. Condition: the domain of S is a proper subset of R.
3. Let the schemes of R, S and T be dom(R), dom(S) and dom(T) = dom(R) - dom(S) respectively.
4. R / S = {t | t ∈ dom(T), ꓯs ∈ S (ꓱr ∈ R (r = st))}.
5. In term of basic RA operations, R / S = πR-S(R) - πR-S((πR-S(R) \* S) - R)

**Example:**

Find the student id of all students who enrolled in all courses offered by the faculty '1014':

Stuid and classNumber information (who is enrolled in which class):

π(stuId, classId) (Enroll): rows added to Class.

+--------+---------+  
| stuId  | classId |  
+--------+---------+  
| 100000 |   10000 |  
| 100000 |   10001 |  
| 100000 |   10002 |  
| 100000 |   10003 |  
| 100000 |   10004 |  
| 100000 |   11001 |  
| 100001 |   10000 |  
| 100001 |   10001 |  
| 100002 |   10000 |  
| 100002 |   10002 |  
| 100002 |   10003 |  
| 100004 |   10003 |  
| 100004 |   10004 |  
| 100005 |   10003 |  
| 100005 |   10004 |  
| 100005 |   10005 |  
| 100005 |   10006 |  
| 100006 |   10004 |  
| 100006 |   10005 |  
| 100007 |   10007 |  
| 100007 |   10008 |  
| 100008 |   10007 |  
+--------+---------+  
22 rows

Classes offered by faculty '1014':

π(classId) (σ(facId='1014) (Class)):

+---------+  
| classId |  
+---------+  
|   10003 |  
|   10004 |  
+---------+  
2 rows

Solution:

π(stuId, classId) (Enroll) / π(stuId, classId) (Enroll):

+--------+  
| stuId  |  
+--------+  
| 100000 |  
| 100004 |  
| 100005 |  
+--------+  
3 rows

**4.3 Query Optimization**

* Since RA is operational and thus algorithmic, there are multiple solutions with varying performance.
* Some heuristics for constructing effective RA solutions:
  1. Minimize the number of RA operations.
  2. Minimize the sizes of the intermediate results.
* In SQL:
  1. A SQL query execution plan breaks down a query into basic execution steps (based on RA).
  2. A SQL optimizer selects one from a list of execution plans.
  3. In MySQL, use the EXPLAIN statement to obtain information about the execution plan: <https://dev.mysql.com/doc/refman/8.1/en/explain-output.html>.

**4.4 Epilog**

Some shortcomings of Relational Algebra:

1. Cannot navigate tuples.
2. Cannot deal with recursion.
   1. e.g., for the relation Employee(SSN, Supervisor\_SSN, ...), find all supervisors (direct or indirect).
   2. May extend to logical databases, e.g. Datalog.
3. No group functions.
   1. e.g., Show the available total quantities of all parts.
4. Operations are too simple, resulting in long sequences.

**5. Relational Calculus  
  
5.1 Review**

* Non-procedural, *declarative*, and high level.
* Two kinds:
  1. Domain Relational Calculus (DRC): MS Access
  2. Tuple Relational Calculus (TRC): SQL;

1. Show all student names.

TRC:

{ (s.fName, s.lName) | s ϵ student }: variable s: a tuple/row

SELECT s.fName, s.lNme

FROM student AS s; -- s ϵ student

DRC:

{(fName, lName) | (stuIf, lName, fName, major, minor, ach, grade) ϵ student}: 7variables (attributes)

{(fName, lName) | (\_, lName, fName, \_, \_, \_, \_) ϵ student}: \_: anonymous variables (placeholders)

* Results specified by the *set builder form*: {s | cond(s)}
* cond(s) is known as a *formula*.
* Constructs:
  1. Variables:
     1. TRC: tuples (bound to tuples): e.g,. s, t, student, class, etc.
     2. DRC: Attributes (bound to domain value): e.g., a, b, c, stuId, fname, etc.
     3. RC's variable is sometime known as 'dummy variable'.
  2. Constants: string, int, etc., E.g., 12, 'csci', 3.7.
  3. Comparison operators: <, <, =, etc.
  4. Boolean operators: and (conjunction, ∧ or just ,), or (disjunction ∨), not (¬), implies (⇒), etc.
  5. Membership functions: belongs to, ∈, not belongs to, ∉, etc.
  6. Quantifiers: there exists (existential, ∃), for all (universal ∀).

**5.2 More RC**

* An *atom* can be thought of as a simple Boolean expression:
  + e ∈ R, or
  + x op y where x and y are attributes or constants, and op is a comparison operation.
* A *formula* is either an atom or formula connected by Boolean operator or qualifiers.
* A formula that is not an atom can be thought of a compound Boolean expression.
* A variable is *bound* if it appears in qualifier expressions. Otherwise, it is a *free* variable.
* Free variables can only appear in the LHS of |.
* All RA expressions can be expressed in RC.
* RA and RC have the same expressive power.
* Any query language that can express all RA is known to be relational complete.
* Relational Calculus expressions need to be *safe*: results should be a*finite* set of tuples.
* Care should be taken especially for the negation operation. E.g. {s |¬ (s ∈ Student) } is unsafe.
* For a given implementation of relational calculus:
  + There may be restrictions in supported constructs.
  + There may be certain *canonical* (good, supported) requirements: e.g. *conjunction* (joined by the and operator) of *disjunction* (joined by the or operator).

**Example:**

{i | i ∈ I ∧ i % 2 =0}  
{i | i ∈ I, i % 2 =0} -- set builder form.

{t | ∃r ∈R, r.firstname = t.firstname, r.lastname = t.lastname}

* t is a *free* variable.
* It will have two attributes: t.firstname and t.lastname.

Alternatively, we can use the set builder form in the LHS before |:

{(r.firstname, r.lastname) | r ∈ R}

R(A,B,C,D) / S(C,D)

{(a,b) | (∀(c,d) ∈ S) (a,b,c,d) ∈ R)}

***Exercises:***

How do you use RC to implement RA operations?

**5.3 TRC**

* The variables in TRC are tuples.
* SQL is based on TRC.

**5.4. DRC**

* The variables in DRC are attributes (domain values).
* Query By Example (QBE) is based on DRC.

**Exercise:**

Work on some of the query questions listed in the [toyu Query Exercise](https://dcm.uhcl.edu/yue/courses/joinDB/Fall2024/notes/rarc/toyu_Ex.pdf) in DRC and TRC.

**More SQL**

by K. Yue

**1. More SQL Features**

**1.1 Prepared Statements**

* MySQL supports server-side prepared statements.
* A prepared statement is used in the following sequence.
  1. The statement is prepared.

SET @sql = "SELECT \* FROM toyu.student"; -- @sql: session variable

PREPARE stmt FROM @sql;

* 1. The prepared statements are executed one to many times, including using different values on placeholders.

EXECUTE stmt;

EXECUTE stmt;

* 1. The statement is deallocated.

DEALLOCATE PREPARE stmt;

* Prepared Statements have the following benefits:
  1. They are compiled once and can be executed many times.
  2. With the use of placeholders, they protect against SQL injections.
  3. They can be used to execute dynamic SQL statements.
     1. However, dynamic SQL statements can be serious security holes and should be used carefully.
* In MySQL, prepared statements can be used in
  1. Applications: such as SQL clients (e.g. MySQL prompt) or Python programs.
  2. SQL scripts: such as invoked by the source command.
* There are vendor-dependent restrictions in prepared statements.
* For examples, in MySQL,
  1. Constants or user variables should be used as the parameter values to the placeholders of the prepared statements.
  2. user variables should be used in the INTO clause of the SQL query in the prepared SELECT statement.

***Example:***

Try the following code in MySQL prompt.

-- Prepared statements.  
SET @sql = "SELECT \* FROM toyu.student";  
PREPARE stmt FROM @sql;  
EXECUTE stmt;  
DEALLOCATE PREPARE stmt;  
  
-- with placeholders.  
SET @sql = "SELECT \* FROM toyu.student WHERE major = ? AND ach >= ?";  
PREPARE stmt FROM @sql;  
SET @major = 'CSCI';  
SET @ach = 38;  
EXECUTE stmt USING @major, @ach;  
  
EXECUTE stmt USING 'CSCI', 38;  
  
SET @major = 'CINF';  
SET @ach = 15;  
EXECUTE stmt USING @major, @ach;  
  
SET @major = 'ITEC';  
SET @ach = 25;  
EXECUTE stmt USING @major, @ach;  
  
DEALLOCATE PREPARE stmt;

**2. Views**

* Views are *virtual* tables *derived* from other tables.
* In MySQL, "views are stored queries that when invoked produce a *result set*. A view acts as a virtual table." See: <https://dev.mysql.com/doc/refman/8.2/en/views.html>
* Some advantages of using views:
  1. Better data abstraction: hiding unnecessary information.
  2. Logical data independence
  3. Better consistency
  4. More security control
  5. Possibly more efficient
* Some disadvantages:
  1. More work
  2. Complicated especially when views are updated.
  3. Performance: view processing may not be optimized, e.g.,
  4. A view can be treated as a subquery by the db engine.
  5. Cannot create index for view in MySQL.
* See the [three layered DB architecture](https://dcm.uhcl.edu/yue/courses/joinDB/Fall2024/notes/intro/DBBasics.html).
* MySQL Create View Manual: search for "mysql view manual"

CREATE  
    [OR REPLACE]  
    [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]  
    [DEFINER = { user | CURRENT\_USER }]  
    [SQL SECURITY { DEFINER | INVOKER }]  
    VIEW view\_name [(column\_list)]  
    AS select\_statement  
    [WITH [CASCADED | LOCAL] CHECK OPTION]

* There are some limitations in views. For example:
  + In earlier versions of MySQL, subqueries are not allowed in the SELECT clause in the CREATE VIEW statement.
* You may need to define intermediate views accordingly.

***Example:***

Execute the following code and ensure that you understand the result.

CREATE OR REPLACE VIEW school\_summary(  
   schoolCode, schoolName, n\_departments) AS  
SELECT s.schoolCode, s.schoolName,  
COUNT(d.deptCode) AS n\_departments  
FROM school AS s LEFT JOIN department AS d ON (s.schoolCode = d.schoolCode)  
GROUP BY s.schoolCode, s.schoolName;  
  
SHOW CREATE VIEW school\_summary;  
-- Note something like "ALGORITHM=UNDEFINED DEFINER=`yue`@`localhost` SQL SECURITY DEFINER"  
-- (default values) may be added.

DESC school\_summary;  
  
SELECT \*  
FROM school\_summary  
WHERE n\_departments > 0;

DROP VIEW school\_summary;

Notes:

* "CREATE OR REPLACE" can replace existing view definition.
* There are no data types explicitly specified for the columns in the example.
* The column list is optional.
* MySQL supports specification of algorithm for processing views:
  + MERGE: view definition merged into the containing query (inlining)
  + TEMPTABLE: view results stored in temporary table to be used in the containing query.
* MySQL supports updatable views. Use them carefully. E.g., the view 'user' in 'mysql' is updatable.
  + In general, "for a view to be updatable, there must be a one-to-one relationship between the rows in the view and the rows in the underlying table."

**3. Stored Subroutines**

* Subroutines stored by the DBMS can be called.
* Subroutines can be
  1. functions: return a value, or
  2. procedures:
     1. do not return a value, and
     2. work by side effects.
* Known as SQL/PSM (SQL/Persistent Stored Modules) in MySQL.
  1. Can execute SQL statements.
  2. Include general programming constructs.
* Some advantages of stored subroutines:
  1. Enforcing application constraints and requirements.
  2. Providing consistency and security control.
  3. Possible performance optimization by both the developers and the DBMS.
  4. Sharing among DB applications, especially when they use different languages.
* Some disadvantages of stored procedures:
  1. Use up DB server's resources.
  2. Potentially inefficient because of limitations in language constructs.
  3. Potentially harder to develop because of relative lack of libraries and difficulty in debugging.
  4. Can be vendor specific.
* In general, stored subroutines should be used more often.
* Some general points:
  1. A stored subroutine is associated with a particular database.
  2. Stored functions cannot be recursive.

**3.1 Stored Procedures**

***Example:***

Execute the following code and ensure that you understand the result.

-- A very simple stored procedure.  
-- Redefine the delimiter to end the procedure.  
DELIMITER //  
  
CREATE OR REPLACE PROCEDURE deptInfo(IN dCode VARCHAR(4), OUT numFaculty INT)  
BEGIN  
   --  Display some information.  
   SELECT d.deptName, d.SchoolCode, t1.n\_majors, t2.n\_minors  
   FROM department AS d INNER JOIN  
      (SELECT COUNT(stuId) AS n\_majors  
      FROM student  
      WHERE major = dCode) AS t1 INNER JOIN  
      (SELECT COUNT(stuId) AS n\_minors  
      FROM student  
      WHERE minor = dCode) AS t2  
   WHERE d.deptCode = dCode;  
  
   -- MySQL does not direct sending output to console.  
   -- It is necessary to use a SQL statement.  
   SELECT 'Debuggin comment can be put here.';  
   SELECT CONCAT('Faculty in the department: ', dCode) AS faculty;  
  
   SELECT \*  
   FROM faculty AS f  
   WHERE f.deptCode = dCode;  
  
   SELECT COUNT(f.facId) INTO numFaculty  
   FROM faculty AS f  
   WHERE f.deptCode = dCode;  
END //  
  
DELIMITER ;  
  
SHOW CREATE PROCEDURE deptInfo;  
  
SET @numFaculty = 0;  
SET @dCode = 'CSCI';  
CALL deptInfo(@dCode, @numFaculty);  
SELECT @dCode, @numFaculty;  
  
SET @dCode = 'ITEC';  
CALL deptInfo(@dCode, @numFaculty);  
SELECT @dCode, @numFaculty;  
  
DROP PROCEDURE deptInfo;

Note:

1. Use of the DELIMITER command to redefine '//' as the delimiter indicating the end of the stored procedure. Otherwise, the default ';' is the delimiter indicating the end of the stored procedure.
2. Two parameters for this procedure: one using IN and one using OUT as the parameter passing mechanism.
3. A procedure does not return any value and accomplish its goal through *side effects*.
4. Side effects include:
   1. Return SELECT results in the procedure body.
   2. Copy numFaculty out upon completion.

***Example: using toyu***

DELIMITER //  
CREATE OR REPLACE PROCEDURE AddNewCourse(  
    IN course\_id INT,  
    IN rubric CHAR(4),  
    IN course\_number CHAR(4),  
    IN course\_title VARCHAR(80),  
    IN credits TINYINT  
)  
BEGIN  
    INSERT INTO Course (courseId, rubric, number, title, credits)  
    VALUES (course\_id, rubric, course\_number, course\_title, credits);  
END //  
DELIMITER ;

CALL AddNewCourse(3009, 'CSCI', '4436', 'Systems Administration', 3);

**3.2 Stored Functions**

***Example:***

Execute the following code and ensure that you understand the result.  
  
  
-- A simple function  
DELIMITER //  
  
CREATE OR REPLACE FUNCTION n\_major(dCode varchar(4)) RETURNS INT  
READS SQL DATA  
BEGIN  
   DECLARE count INT DEFAULT 0;  
  
   SELECT COUNT(\*) INTO count  
   FROM student  
   WHERE major = dCode;  
  
   RETURN count;  
END //  
  
DELIMITER ;  
  
SHOW CREATE FUNCTION n\_major;  
  
SELECT n\_major('CSCI');  
SELECT n\_major('ITEC');  
  
Note:

* A function returns a value and does not accomplish its goal through side effects.
* Note the argument passed into the function.
* Formal arguments of a function use the IN parameter passing mode.
* Note the variable declaration and assignment.
* The scope of the variable is within the function.
* 'READS SQL DATA' describes the characteristic of the subroutine: it reads but not writes SQL data.

***Example:***

The following example functions are included in the script of creating toyu. Function calls are added.

-- get the full name of a student.  
DELIMITER //  
CREATE OR REPLACE FUNCTION GetStudentFullName(  
    student\_id INT  
)  
RETURNS VARCHAR(61)  
DETERMINISTIC  
BEGIN  
    DECLARE full\_name VARCHAR(61);  
    SELECT CONCAT(fname, ' ', lname) INTO full\_name  
    FROM Student  
    WHERE stuId = student\_id;  
    RETURN full\_name;  
END //  
DELIMITER ;  
  
SELECT GetStudentFullName(100000);  
SELECT GetStudentFullName(100001);  
  
-- get the full name of a department code  
DELIMITER //  
CREATE OR REPLACE FUNCTION GetDepartmentName(  
    dept\_code CHAR(4)  
)  
RETURNS VARCHAR(30)  
DETERMINISTIC  
BEGIN  
    DECLARE dept\_name VARCHAR(30);  
    SELECT d.deptName INTO dept\_name  
    FROM Department d  
    WHERE d.deptCode = dept\_code;  
    RETURN dept\_name;  
END //  
DELIMITER ;  
  
SELECT GetDepartmentName('CSCI');  
SELECT GetDepartmentName('ITEC');  
  
-- Compute and return the GPA of a student  
DELIMITER //  
CREATE OR REPLACE FUNCTION GetStudentGPA(  
    student\_id INT  
)  
RETURNS DECIMAL(3,2)  
DETERMINISTIC  
BEGIN  
    DECLARE gpa DECIMAL(3,2);  
    SELECT ROUND(SUM(g.gradePoint \* co.credits) / SUM(co.credits), 2) INTO gpa  
    FROM Enroll e  
    JOIN Class c ON e.classId = c.classId  
    JOIN Course co ON c.courseId = co.courseId  
    JOIN Grade g ON e.grade = g.grade  
    WHERE e.stuId = student\_id;  
    RETURN gpa;  
END //  
DELIMITER ;  
  
SELECT GetStudentGPA(100000);  
SELECT GetStudentGPA(100001);  
SELECT GetStudentGPA(100002);  
  
-- Return the top n students in a major department with GPA.  
DELIMITER //  
CREATE OR REPLACE FUNCTION GetTopStudentsInDepartment(  
    dept\_code CHAR(4),  
    top\_count INT  
)  
RETURNS VARCHAR(1000)  
DETERMINISTIC  
BEGIN  
    DECLARE student\_list VARCHAR(1000);  
    SET student\_list = '';  
     
    WITH temp AS(  
        SELECT s.stuId, CONCAT(s.fname, ' ', s.lname) AS student,  
            ROUND(SUM(g.gradePoint \* co.credits) / SUM(co.credits), 2) AS gpa  
        FROM Student s  
        JOIN Department d ON s.major = d.deptCode  
        JOIN Enroll e ON s.stuId = e.stuId  
        JOIN Class c ON e.classId = c.classId  
        JOIN Course co ON c.courseId = co.courseId  
        JOIN Grade g ON e.grade = g.grade  
        WHERE d.deptCode = dept\_code  
        GROUP BY s.stuId, student  
        ORDER BY gpa DESC  
        LIMIT top\_count)  
    SELECT GROUP\_CONCAT(CONCAT(temp.student, '(', temp.stuId, '):', temp.gpa) SEPARATOR ', ')  
        INTO student\_list  
    FROM temp  
    LIMIT 1;  
  
    RETURN student\_list;  
END //  
DELIMITER ;  
  
SELECT GetTopStudentsInDepartment('CSCI', 2);  
SELECT GetTopStudentsInDepartment('CSCI', 3);

**3.3 Cursors**

* In stored subroutines in MySQL, various high level language constructs are supported, such as variable declarations, conditional statements, control statements, etc.
* In particular, cursors are supported for allowing iteration through the result sets.

***Example:***

Execute the following code and ensure that you understand the result. The example is artifically constructed as there are better ways.  
  
-- using cursor.  
DELIMITER //  
  
CREATE FUNCTION major\_students(dept VARCHAR(4))  
RETURNS VARCHAR(1000)  
READS SQL DATA  
BEGIN  
DECLARE result VARCHAR(1000) DEFAULT '';  
DECLARE name VARCHAR(41) DEFAULT '';  
DECLARE done INT DEFAULT FALSE;  
  
DECLARE cursor\_1 CURSOR FOR  
SELECT DISTINCT CONCAT(fName, ' ', lName) AS name  
FROM student  
WHERE major = dept;  
  
DECLARE continue handler FOR NOT FOUND SET done = TRUE;  
  
OPEN cursor\_1;  
SET result = '';   
  
compute\_loop: LOOP  
-- This is needed as there are other SQL statements that may set done to true.  
SET done = false;  
FETCH cursor\_1 INTO name;  
IF done THEN  
LEAVE compute\_loop;  
END IF;  
  
IF (result <> '') THEN  
SET result = CONCAT(result, ', ');  
END IF;  
  
SET result = CONCAT(result, name);  
END LOOP;  
  
CLOSE cursor\_1;  
  
RETURN result;  
END //  
  
DELIMITER ;  
  
SELECT major\_students('CSCI');  
SELECT major\_students('CINF');  
  
DROP FUNCTION major\_students;

* Note that the example is only used to demonstrate cursors in a stored function. A much better solution is to simply use GROUP\_CONCAT, such as:

SELECT GROUP\_CONCAT(CONCAT(fName, ' ', lName) SEPARATOR ', ') AS majors  
FROM student  
WHERE major = 'CSCI';

**4. Triggers**

* Triggers allow *event-driven* programming.
* A trigger is activated when certain events occur. Unlike stored routines, triggers are not explicitly called.
* Four questions to ask for an event model:
  1. What are the events?
  2. What information can be obtained from the events?
  3. What actions can be performed to handle events?
  4. How do the events propagate?

**4.1 The event model for triggers**

* Events:
  1. Inserting a row
  2. Updating a row
  3. Deleting a row
* Actions can be executed:
  1. Before the event
  2. After the event
* Information can be obtained through two keywords:
  1. old:
     + the old value of the row before the event.
     + for update and delete events
  2. new:
     + the new value of the row after the event.
     + for update and insert events
* Triggers do not propagate.

Advantages of triggers

1. The event driven model suits certain kinds of tasks better.
2. Can ease the implementation of consistency check.
3. Can ease the implementation of business logic and integrity check.

Disadvantages:

1. Relatively invisible, and possibly overlooked by developers.
2. Relatively difficult to debug.
3. Potential performance issues.
4. Potential complicated interactions.

***Example:***

An example of using trigger is for auditing, e.g. <https://vladmihalcea.com/mysql-audit-logging-triggers/>

Execute the following code for auditing and ensure that you understand the result.

-- trigger  
  
CREATE TABLE courseUpdate(  
   cuId        INT NOT NULL AUTO\_INCREMENT,  
   `type`      CHAR(1),  
    courseId    INT NOT NULL,  
    rubric      CHAR(4) NOT NULL,  
    number      CHAR(4) NOT NULL,  
    title       VARCHAR(80) NOT NULL,  
    credits     TINYINT NULL,  
    ts          TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,  
    CONSTRAINT CourseUpdate\_courseId\_pk PRIMARY KEY (cuId),  
    CONSTRAINT CourseUpdate\_deptCode\_fk FOREIGN KEY (rubric)  
        REFERENCES Department(deptCode));  
       
       
-- trigger example.  
DELIMITER $$  
  
CREATE TRIGGER update\_Course AFTER UPDATE ON Course FOR EACH ROW  
BEGIN  
   -- code should be more sophisticated.  
   INSERT INTO courseUpdate(`type`, courseId, rubric, number, title, credits)  
   VALUES('U', new.courseId, new.rubric, new.number, new.title, new.credits);  
END $$  
  
  
CREATE TRIGGER insert\_Course AFTER INSERT ON Course FOR EACH ROW  
BEGIN  
   -- code should be more sophisticated.  
   INSERT INTO courseUpdate(`type`, courseId, rubric, number, title, credits)  
   VALUES('I', new.courseId, new.rubric, new.number, new.title, new.credits);  
END $$  
  
DELIMITER ;  
  
SELECT \* FROM Course;  
  
UPDATE Course  
SET number = 2315  
WHERE courseId = 2000;  
  
INSERT INTO Course(courseId, rubric, number, title, credits) VALUES  
    (4000,'CSCI',3532,'Advanced Data Structures',3);  
    
UPDATE Course  
SET number = 3341  
WHERE courseId = 4000;  
  
SELECT \* FROM course;  
SELECT \* FROM courseUpdate;  
  
-- Clean up.  
DROP TRIGGER update\_Course;  
DROP TRIGGER insert\_Course;  
  
DROP TABLE courseUpdate;  
  
DELETE FROM Course  
WHERE CourseId = 4000;  
  
UPDATE Course  
SET number = 3333  
WHERE courseId = 2000;  
  
SELECT \* FROM Course;

**5. System Catalog**

* Named collections of *meta-data* in SQL DBMS.
* Important for DB administrators and developers to know.
* Usually stored in relations.
* Common terms with similar meaning: data dictionary, catalog, etc.
* Three major databases that are included with MySQL: mysql, information\_schema and performance\_schema.

***Example:***

mysql> show databases;  
+--------------------+  
| Database           |  
+--------------------+  
| information\_schema |  
| mysql              |  
| performance\_schema |  
| swim              |  
| toytu              |  
| world              |  
| yue\_exp            |  
+--------------------+  
7 rows in set (0.00 sec)  
>

**5.1 mysql database**

* MySQL database stores information about the system. It is the system schema.
* See: <https://dev.mysql.com/doc/refman/8.1/en/system-schema.html>.
* However, many of these system tables cannot be accessed directly. They should be access through other means (such as information\_schema, see below)

***Example:***

SELECT DISTINCT u.user, u.password  
FROM mysql.user AS u  
WHERE u.host = 'localhost';

**5.2 information\_schema database**

* Store information about all databases in MySQL.
* Contents are system*views* using the MEMORY storage engine and they cannot be updated.
* Triggers cannot be defined on tables in information\_schema.

***Example:***

mysql> SELECT table\_name, table\_type, row\_format, table\_rows, avg\_row\_length  
    -> FROM information\_schema.tables  
    -> WHERE table\_schema = 'information\_schema'  
    -> ORDER BY table\_name DESC;  
+---------------------------------------+-------------+------------+------------+----------------+  
| table\_name                            | table\_type  | row\_format | table\_rows | avg\_row\_length |  
+---------------------------------------+-------------+------------+------------+----------------+  
| VIEWS                                 | SYSTEM VIEW | Dynamic    |       NULL |              0 |  
| USER\_PRIVILEGES                       | SYSTEM VIEW | Fixed      |       NULL |           1986 |  
| TRIGGERS                              | SYSTEM VIEW | Dynamic    |       NULL |              0 |  
| TABLE\_PRIVILEGES                      | SYSTEM VIEW | Fixed      |       NULL |           2372 |  
| TABLE\_CONSTRAINTS                     | SYSTEM VIEW | Fixed      |       NULL |           2504 |  
| TABLESPACES                           | SYSTEM VIEW | Fixed      |       NULL |           6951 |  
| TABLES                                | SYSTEM VIEW | Fixed      |       NULL |           9450 |  
| STATISTICS                            | SYSTEM VIEW | Fixed      |       NULL |           5753 |  
| SESSION\_VARIABLES                     | SYSTEM VIEW | Fixed      |       NULL |           3268 |  
| SESSION\_STATUS                        | SYSTEM VIEW | Fixed      |       NULL |           3268 |  
| SCHEMA\_PRIVILEGES                     | SYSTEM VIEW | Fixed      |       NULL |           2179 |  
| SCHEMATA                              | SYSTEM VIEW | Fixed      |       NULL |           3464 |  
| ROUTINES                              | SYSTEM VIEW | Dynamic    |       NULL |              0 |  
| REFERENTIAL\_CONSTRAINTS               | SYSTEM VIEW | Fixed      |       NULL |           4814 |  
| PROFILING                             | SYSTEM VIEW | Fixed      |       NULL |            308 |  
| PROCESSLIST                           | SYSTEM VIEW | Dynamic    |       NULL |              0 |  
| PLUGINS                               | SYSTEM VIEW | Dynamic    |       NULL |              0 |  
| PARTITIONS                            | SYSTEM VIEW | Dynamic    |       NULL |              0 |  
| PARAMETERS                            | SYSTEM VIEW | Dynamic    |       NULL |              0 |  
| KEY\_COLUMN\_USAGE                      | SYSTEM VIEW | Fixed      |       NULL |           4637 |  
| INNODB\_TRX                            | SYSTEM VIEW | Fixed      |       NULL |           4534 |  
| INNODB\_LOCK\_WAITS                     | SYSTEM VIEW | Fixed      |       NULL |            599 |  
| INNODB\_LOCKS                          | SYSTEM VIEW | Fixed      |       NULL |          31244 |  
| INNODB\_CMP\_RESET                      | SYSTEM VIEW | Fixed      |       NULL |             25 |  
| INNODB\_CMPMEM\_RESET                   | SYSTEM VIEW | Fixed      |       NULL |             29 |  
| INNODB\_CMPMEM                         | SYSTEM VIEW | Fixed      |       NULL |             29 |  
| INNODB\_CMP                            | SYSTEM VIEW | Fixed      |       NULL |             25 |  
| GLOBAL\_VARIABLES                      | SYSTEM VIEW | Fixed      |       NULL |           3268 |  
| GLOBAL\_STATUS                         | SYSTEM VIEW | Fixed      |       NULL |           3268 |  
| FILES                                 | SYSTEM VIEW | Fixed      |       NULL |           2677 |  
| EVENTS                                | SYSTEM VIEW | Dynamic    |       NULL |              0 |  
| ENGINES                               | SYSTEM VIEW | Fixed      |       NULL |            490 |  
| COLUMN\_PRIVILEGES                     | SYSTEM VIEW | Fixed      |       NULL |           2565 |  
| COLUMNS                               | SYSTEM VIEW | Dynamic    |       NULL |              0 |  
| COLLATION\_CHARACTER\_SET\_APPLICABILITY | SYSTEM VIEW | Fixed      |       NULL |            195 |  
| COLLATIONS                            | SYSTEM VIEW | Fixed      |       NULL |            231 |  
| CHARACTER\_SETS                        | SYSTEM VIEW | Fixed      |       NULL |            384 |  
+---------------------------------------+-------------+------------+------------+----------------+  
37 rows in set (0.08 sec)

* It is more flexible than the SHOW command.

***Example:***

Execute the following code and ensure that you understand the results.

-- System Catalog  
-- Getting selected columns from information\_schema  
SELECT table\_name, table\_type, row\_format, table\_rows, avg\_row\_length  
FROM information\_schema.tables  
WHERE table\_schema = 'information\_schema'  
ORDER BY table\_name DESC;  
  
-- databases and tables  
SELECT t.TABLE\_SCHEMA AS `schema`, COUNT(t.TABLE\_NAME) AS num\_tables  
FROM information\_schema.tables AS t  
GROUP BY `schema`  
ORDER BY num\_tables DESC;  
  
SELECT t.TABLE\_SCHEMA AS `schema`, t.ENGINE, COUNT(t.TABLE\_NAME) AS num\_tables  
FROM information\_schema.tables t  
GROUP BY `schema`, t.ENGINE  
ORDER BY `schema`, num\_tables DESC;

***Example:***

DROP SCHEMA IF EXISTS dbtool;  
CREATE SCHEMA dbtool;  
USE dbtool;  
  
-- Return the number of columns of a table in a schema (database) in the output parameter column\_count  
DELIMITER //  
CREATE OR REPLACE PROCEDURE count\_columns(  
    IN schema\_name VARCHAR(64),  
    IN table\_name VARCHAR(64),  
    OUT column\_count INT  
)  
BEGIN  
    SET @\_\_cc\_query = CONCAT('SELECT COUNT(\*) INTO @\_\_cc\_column\_count FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_SCHEMA = ? AND TABLE\_NAME = ?');  
    PREPARE stmt FROM @\_\_cc\_query;  
    SET @\_\_cc\_schema\_name = schema\_name;  
    SET @\_\_cc\_table\_name = table\_name;  
    
    EXECUTE stmt USING @\_\_cc\_schema\_name, @\_\_cc\_table\_name;  
    SET column\_count = @\_\_cc\_column\_count;  
    DEALLOCATE PREPARE stmt;  
END //  
DELIMITER ;  
  
CALL count\_columns('toyu', 'student', @column\_count);  
SELECT @column\_count;  
CALL count\_columns('swim', 'swimmer', @column\_count);  
SELECT @column\_count;

**5.3 performance\_schema database**

* contain performance related server data.