**Relational Calculus** (Last updated 2020-02-14)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Relational Calculus**  These notes are based on E. F. Codd's language Alpha which provided the mathematical foundation for **SQL**. It is based on predicate calculus which you covered in CS2233 Discrete Math.  We will only consider the RANGE and GET commands. (We will discuss insert, delete, and update when we discuss SQL.) | Consider the following relations:   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Student   |  |  | | --- | --- | | **ST#** | **Classif** | | 100 | FR | | 200 | SO | | 300 | JR | | 400 | SR | | 500 | FR |   Course   |  |  | | --- | --- | | C# | Title | | CS 1713 | Intro to CS | | MAT 1224 | Calculus I | | ENG 1043 | English I | | HIS 1033 | History of the World Part I | | Enrollment   |  |  |  |  | | --- | --- | --- | --- | | **ST#** | **C#** | **Midterm** | **Final** | | 100 | CS 1713 | 95 | 80 | | 100 | MAT 1224 | 100 | 100 | | 100 | ENG 1043 | 85 | 85 | | 100 | HIS 1033 | 80 | 85 | | 200 | CS 1713 | 91 | 90 | | 200 | MAT 1224 | 50 | 80 | | 300 | CS 1713 | 65 | 80 | | 500 | CS 1713 | 100 | 80 | | 500 | HIS 1033 | 95 | 90 | | |
| **Relational Calculus Statements**  **RANGE** *Relation TupleVariable*  restricts the *TupleVariable* to the specified *Relation*.  **GET** *Workspace (TargetColumnList) : QualifyingExpression.*  gets a result which will contain the specified columns of rows from 1 or more relations satisfying the qualifying expression and places the result in *Workspace.*  ***Workspace*** *-* names the resulting workspace of the newly formed temporary relation  ***TargetColumnList*** *-* a list of columns qualified by tuple variables which will be the columns in the result.  ***QualifyingExpression*** *-* an optional expression which if true will cause the corresponding row to be placed in the *Workspace* | **Example 1:** **Projection** - Get the C# for courses in which students are enrolled.  RANGE Enrollment e  GET W (e.C#)   |  | | --- | | **C#** | | CS 1713 | | MAT 1224 | | ENG 1043 | | HIS 1033 |   **Example 2:** **Projection** - Get the ST# and Midterm grades of each student.  RANGE Enrollment e //(W = any variable name)  GET W (e.ST#, e.Midterm)   |  |  | | --- | --- | | **ST#** | **Midterm** | | 100 | 95 | | 100 | 100 | | 100 | 85 | | 100 | 80 | | 200 | 91 | | 200 | 50 | | 300 | 65 | | 500 | 100 | | 500 | 95 | |
| **Selection**  We specify a condition after the colon. | **Example 3:** **Selection -** get enrollment information for students who has a midterm grade > 90.  RANGE Enrollment e  GET W (e.ST#, e.C#, e.Midterm, e.Final): e.Midterm > 90   |  |  |  |  | | --- | --- | --- | --- | | **ST#** | **C#** | **Midterm** | **Final** | | 100 | CS 1713 | 95 | 80 | | 100 | MAT 1224 | 100 | 100 | | 200 | CS 1713 | 91 | 90 | | 500 | CS 1713 | 100 | 80 | | 500 | HIS 1033 | 95 | 90 |   **Example 4:** **Selection -** get enrollment information of students who made an A on both exams.  Enrollment[Midterm >= 90 ^ Final >= 90]  RANGE Enrollment e  GET W (e.ST#, e.C#, e.Midterm, e.Final):   e.Midterm > 90 ^ e.Final >= 90 |
| **Join** | **Example 5:** **Join -** join the Student and Enrollment relations which have the same ST#  RANGE Enrollment e  RANGE Student s  GET W (s.ST#, s.Classif, e.C#, e.Midterm, e.Final):   s.ST# = e.ST#   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **ST#** | **Classif** | **C#** | **Midterm** | **Final** | | 100 | FR | CS 1713 | 95 | 80 | | 100 | FR | MAT 1224 | 100 | 100 | | 100 | FR | ENG 1043 | 85 | 85 | | 100 | FR | HIS 1033 | 80 | 85 | | 200 | SO | CS 1713 | 91 | 90 | | 200 | SO | MAT 1224 | 50 | 80 | | 300 | JR | CS 1713 | 65 | 80 | | 500 | FR | CS 1713 | 100 | 80 | | 500 | FR | HIS 1033 | 95 | 90 | |

|  |  |
| --- | --- |
| **Qualified Retrievals Using Quantifiers**  **Existential quantifier** - there exists at least one.  Symbol: **∃t**  The predicate is true if there is **at least one** tuple where the condition is **true**; otherwise, it is false.  **Universal quantifier** - for all cases, it must be true.  Symbol: **∀t**  The predicate **must** **be true for all the tuples** in the relation. The predicate is false if there is at least one tuple where the condition is false; otherwise, it is true. | **Example 6: Existential explanation**  If t is a tuple variable representing tuples (i.e., rows) in the Student table, the predicate ∃t (t.CLASSIF = 'FR') is true if there is at least one student who has a classif of FR.  **Example 7: Universal explanation**  If t is a tuple variable representing tuples in the Student table, the predicate  ∀t (t.CLASSIF = 'FR') is true if **all** students are FR classifs.  That could be true if all the students are FR classifs or what other case? If Student doesn't have any students, the predicate would be true. |
| **Existential Quantifier Needed**  **Table Referenced in  *QualifyingExpression* is not referenced in  *TargetColumnList***  To execute the Get, it will automatically loop through the table(s) listed in the *TargetColumnList*. We had to introduce an existential quantifier for Enrollment since Enrollment wasn’t part of the *TargetColumnList*. | **Example 8: Use of Existential Quantifier**  Get the C# and Title of courses in which student 200 is enrolled.  RANGE Enrollment e  RANGE Course c  GET W8 (c.C#, c.Title): ∃e (e.C#=c.C# ^ e.ST# = 200)  Consider a course. Include its C# and TITLE in the result if there exists an enrollment for that course having ST# = 200. |
| In example 9, the **Get** will **automatically loop** through **each Student**. If the  ***QualifyingExpression*** is true for a student, its classif is included in the result. | **Example 9: Use of Existential Quantifier**  Get the **classif** of students enrolled in HIS 1033.  RANGE Student s  RANGE Enrollment e  Get W9 (s.Classif):∃e (s.ST#=e.ST# ^ e.C# = ‘HIS1033’)  Consider a student. Include its classif in the result if there exists an enrollment for that student having C# = "HIS1033". |

|  |  |
| --- | --- |
| **ALL**  Universal quantifier is used to answer queries where **all** of something must be present.  We must first identify what requires all occurrences. In example 10, we must have all courses. | **Example 10: Use of Universal Quantifier**  Get the ST# and Classif of students who are enrolled in all of the courses.  RANGE Course c  RANGE Enrollment e  RANGE Student s  Get W (s.ST#, s.Classif): ∀c(∃e (e.C#=c.C# ^ e.ST# = s.ST#))  Consider a student. Include its S# and classif in the result if for all courses, there exists an enrollment for the course and that student.  The query could have been written like this:  Get W (s.ST#, s.Classif): ∀c∃e (e.C#=c.C# ^ e.ST# = s.ST#)  This would be incorrect:  Get W (s.ST#, s.Classif): ∃e∀c(e.C#=c.C# ^ e.ST# = s.ST#)  This last incorrect query means something like:  Consider a student. Include its S# and classif in the result if there exists an enrollment where the C# matches all of the courses for that student. One C# can't match ALL of the C# in Course unless Course has zero or one rows. |
| **Only**  Examine the query on the right. What are we universally quantifying? Enrollment.   |  |  |  |  | | --- | --- | --- | --- | | **ST#** | **C#** | **Midterm** | **Final** | | 100 | CS 1713 | 95 | 80 | | 100 | MAT 1224 | 100 | 100 | | 100 | ENG 1043 | 85 | 85 | | 100 | HIS 1033 | 80 | 85 | | 200 | CS 1713 | 91 | 90 | | 200 | MAT 1224 | 50 | 80 | | 300 | CS 1713 | 65 | 80 | | 500 | CS 1713 | 100 | 80 | | 500 | HIS 1033 | 95 | 90 |  |  |  | | --- | --- | | **ST#** | **Classif** | | 100 | FR | | 200 | SO | | 300 | JR | | 400 | SR | | 500 | FR | | **Example 11:** Find the ST# and Classif of students who **are only** enrolled in CS 1713.  RANGE Enrollment e  RANGE Student s for all Enrollments if st# matches and c  Get W (s.ST#, s.Classif): ∀e(e.ST#=s.ST# ^ e.C#="CS1713")  Why doesn't that query work? It says that all rows in Enrollment must be our ST# we are considering.  Consider ST# 100: the query must be true for ALL enrollments   * 1st row: matches ST# 100 and C# CS1713 * 2nd row: reject (as expected) since C# doesn't match   Consider ST# 200: the query must be true for ALL enrollments   * 1st row: matches C# CS1713, but ST# 100 <> 200, reject …. hmmmm   Consider ST# 300: the query must be true for ALL enrollments   * 1st row: matches C# CS1713, but ST# 100 <> 300, reject * But ST# 300 should have been in our answer!!!!!   …  We actually want to match all of a subset of Enrollment -- only the rows for the ST# we are considering.  RANGE Enrollment e  RANGE Student s  Get W (s.ST#, s.Classif): ∀e(e.ST# ≠ s.ST# v e.C#="CS1713")  Now when the ST# doesn't match, we don't reject the row; instead, the left part of the OR matched so the condition is TRUE for that one row. We continue with subsequent rows. If our ST# matches, the left part is FALSE, so we must consider the right part and see if the C# is "CS1713".  Consider ST# 100: the query must be true for ALL enrollments   * 1st row: FALSE for left part, TRUE for right part so the OR is TRUE * 2nd row: reject (as expected) since C# doesn't match   Consider ST# 200: the query must be true for ALL enrollments   * 1st row thru 4th row: left part is TRUE since ST# <> 200. * 5th row: left part is FALSE and right part is FALSE, reject!!   Consider ST# 300: the query must be true for ALL enrollments   * First 6 rows: TRUE since ST# is not 300 * 7th row: left part is FALSE and right part is TRUE, therefore TRUE * Do we have to keep looking at the remaining rows? ?? * Last 2 rows: ??. * Since it was TRUE for all rows, 300 JR is in our result.   …  Would student 400 be in the result? ??. |
| **Using Implication**  Recall a → b = ~a ∨ b  Using implication helps the query read better. | **Example 11v3:** Find the ST# and Classif of students who **are only** enrolled in CS 1713.  RANGE Enrollment e  RANGE Student s  Get W (s.ST#, s.Classif): ∀e(e.ST# = s.ST# → e.C#="CS1713")  Consider a student. Include its S# and classif in the result if for **all** enrollments for that student, the C# is CS1713. |
| **Not Any of Something**  Examine the query on the right. What are we universally quantifying? Enrollment  Enrollment   |  |  |  |  | | --- | --- | --- | --- | | **ST#** | **C#** | **Midterm** | **Final** | | 100 | CS 1713 | 95 | 80 | | 100 | MAT 1224 | 100 | 100 | | 100 | ENG 1043 | 85 | 85 | | 100 | HIS 1033 | 80 | 85 | | 200 | CS 1713 | 91 | 90 | | 200 | MAT 1224 | 50 | 80 | | 300 | CS 1713 | 65 | 80 | | 500 | CS 1713 | 100 | 80 | | 500 | HIS 1033 | 95 | 90 |   Student   |  |  | | --- | --- | | **ST#** | **Classif** | | 100 | FR | | 200 | SO | | 300 | JR | | 400 | SR | | 500 | FR | | **Example 12:**  Find the ST# and Classif of students who **are not** enrolled in "HIS 1033".  RANGE Enrollment e  RANGE Student s  Get W (s.ST#, s.Classif): ∀e(e.ST# = s.ST# → e.C# ≠"HIS1033")  Consider a student. Include its ST# and classif in the result if for **all** enrollments for that student, the C# is not HIS1033.  Result:   |  |  | | --- | --- | | **ST#** | **Classif** | | 200 | SO | | 300 | JR | | 400 | SR |   Why would student 400 be included in the result?  Let’s use the V form instead of the 🡪 to make this clearer:  Get W (s.ST#, s.Classif): ∀e(e.ST# ≠ s.ST# V e.C# ≠"HIS1033")  ??. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Supplier Parts DB Example**  We will use this classical schema in MANY examples.  SUPPLIER(S#, SNAME, STATUS, CITY)  S# - supplier number  SNAME - supplier name(e.g., SMITH, JONES),  STATUS - a number representing the quality of this supplier with the higher number representing a better quality of service (e.g., 10, 20)  PART(P#, PNAME, COLOR, WEIGHT)  P# - part number  PNAME - part name (e.g., NUT)  COLOR - color of the part (e.g., RED)  WEIGHT - weight of the part in grams  SHIPMENT(S#, P#, QUANTITY)  S# - supplier number  P# - part number  QUANTITY - the number of this part shipped by this supplier  **Constraints**: keys are underlined | **Supplier Parts DB sample Data**   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **SUPPLIER**   |  |  |  |  | | --- | --- | --- | --- | | **S#** | **SNAME** | **STATUS** | **CITY** | | S1 | SMITH | 20 | LONDON | | S2 | JONES | 10 | PARIS | | S3 | BLAKE | 30 | PARIS | | S4 | FORD | 20 | LONDON | | S5 | ADAMS | 30 | ATHENS |   **PART**   |  |  |  |  | | --- | --- | --- | --- | | **P#** | **PNAME** | **COLOR** | **WEIGHT** | | P1 | NUT | RED | 12 | | P2 | BOLT | GREEN | 17 | | P3 | SCREW | BLUE | 17 | | P4 | SCREW | RED | 14 | | P5 | CAM | BLUE | 12 | | P6 | COG | RED | 19 | | **SHIPMENT**   |  |  |  | | --- | --- | --- | | **S#** | **P#** | **QUANTITY** | | S1 | P1 | 300 | | S1 | P2 | 200 | | S1 | P3 | 400 | | S1 | P4 | 200 | | S1 | P5 | 100 | | S1 | P6 | 100 | | S2 | P1 | 300 | | S2 | P2 | 400 | | S3 | P2 | 200 | | S4 | P2 | 200 | | S4 | P4 | 300 | | S4 | P5 | 400 | | |
| **Simple Retrievals (no qualifying expression)** | **Example SP-1:** Get the part numbers for each of the parts.  RANGE PART p  GET W (p.P#)  This returns a subset of the columns -- just the part number.  Result: W   |  | | --- | | **P#** | | P1 | | P2 | | P3 | | P4 | | P5 | | P6 |   **Example SP-2:**  Get the supplier number, name, and city for each of the suppliers  RANGE SUPPLIER s  GET W (s.S#, s.SNAME, s.CITY)  Result: W   |  |  |  | | --- | --- | --- | | S# | SNAME | CITY | | S1 | SMITH | LONDON | | S2 | JONES | PARIS | | S3 | BLAKE | PARIS | | S4 | FORD | LONDON | | S5 | ADAMS | ATHENS | |
| **Simple Qualified Retrievals** | **Example SP-3:**  Get the supplier number of suppliers in Paris with a status > 20.  RANGE SUPPLIER s  GET W (s.S#): s.CITY = 'PARIS' ^ s.STATUS > 20.  In other words, consider a supplier. Include it in the result if its city is Paris and its status is greater than 20.  Result: W   |  | | --- | | S# | | S3 | |
|  | **Example SP-4**: Get the supplier number and name of suppliers in London.  RANGE SUPPLIER s  ??  Result: W   |  |  | | --- | --- | | **S#** | **SNAME** | | S1 | SMITH | | S4 | FORD | |
|  | **Example SP-5**: Get the supplier number of suppliers who supply part P2.  RANGE SHIPMENT h  GET W (h.S#) : h.P# = 'P2'  Result: W   |  | | --- | | **S#** | | S1 | | S2 | | S3 | | S4 | |
|  |  |
|  | **Example SP-6**: Get the supplier name of suppliers of part P2.  RANGE SUPPLIER s  RANGE SHIPMENT h  GET W (s.SNAME): ∃h (h.P# = 'P2' ^ h.S# = s.S#)  In other words, consider a supplier. Include it in the result if there exists a shipment of part P2 by this supplier.  Result: W   |  | | --- | | SNAME | | SMITH | | JONES | | BLAKE | | FORD |   **Example SP-7:**  Get the part name of parts supplied by supplier S2.  ?? |
|  | **Example SP-8:**  Get the city of suppliers who supply part P2.  Partially complete query:  RANGE SUPPLIER s  RANGE ??  GET w (s.CITY): ??  Consider a supplier include its city in the result if there exists at least one shipment by this supplier of part P2. |
|  | **Example SP-9:**  Get the supplier number of suppliers who supply at least one red part.  RANGE SHIPMENT h  RANGE PART p  GET W9 (h.S#): ∃p (p.COLOR = 'RED' ^ p.P# = h.P#)  In other words, consider a shipment. Include its S# in the result if there exists a RED part with the shipment's P#.  Result: W9   |  | | --- | | S# | | S1 | | S2 | | S4 |   What happens if a supplier supplies multiple red parts? S1 supplies P1 and P6 which are both red. However, with Alpha, duplicate rows are eliminated in the result. SQL doesn't eliminate duplicates unless you specify DISTINCT. |
|  | **Example SP-10:**  Get the name of suppliers who supply at least one red part.  PART[COLOR = 'RED'][P# = P#] SHIPMENT[S# = S#] SUPPLIER [SNAME]  Approaches:   1. Use our result from SP-9 since W9 is the S# of suppliers that supply red parts.   RANGE w9 r  RANGE SUPPLIER s  GET W (s.SNAME): ∃r(r.S# = s.S#)   1. An approach which doesn't use w9.   RANGE SHIPMENT h  RANGE PART p  RANGE SUPPLIER s  GET W (s.SNAME): ∃h(h.S# = s.S#  ^ ∃p (p.COLOR = 'RED' ^ p.P# = h.P#))  Consider a supplier. If there exists a shipment for this supplier such that there exists a part for this shipment which is red, include the supplier's name in the result.  Since there is only one part for a shipment, this makes the query appear more complicated than necessary.  For the same RANGE statements, the following would also be correct:   1. GET w (s.SNAME): ∃h∃p (h.S# = s.S# ^ p.COLOR = 'RED' ^ p.P# = h.P#)) 2. GET w (s.SNAME): ∃p∃h (h.S# = s.S# ^ p.COLOR = 'RED' ^ p.P# = h.P#))   However, the following is **not** correct. Why?  GET w (s.SNAME): ∃h(h.S# = s.S#)  ^ ∃p (p.COLOR = 'RED' ^ p.P# = h.P#)  ?? |
| Some queries require multiple references to the same relation.  The query could be done in steps:  Step 1. Get the P# of parts supplied by S2  RANGE SHIPMENT h  GET W11\_1 (h.P#): h.S#='S2'  Result: W11\_1   |  | | --- | | P# | | P1 | | P2 |   Step 2. Get the S# of suppliers of at least one of those parts.  RANGE W11\_1 hs2  RANGE SHIPMENT h  GET W11\_2 (h.S#): ∃hs2 (h.P# = hs2.P#)  Result: W11\_2   |  | | --- | | S# | | S1 | | S2 | | S3 | | S4 |   Step 3. Get the name of those suppliers.  RANGE W11\_2 n  RANGE SUPPLIER s  GET W11\_3 (s.S#, s.SNAME): ∃n (n.S# = s.S#)  Result: W11\_3   |  | | --- | | SNAME | | SMITH | | JONES | | BLAKE | | FORD | | **Example SP-11:**  Get the S# and names of suppliers who supply at least one part supplied by supplier S2.  Instead of three separate steps, how can we do this as one GET?  The next query is **incorrect**:  RANGE SUPPLIER s  RANGE SHIPMENT h  GET w (s.S#, s.SNAME) : ∃h (h.S# = s.S# ^ h.S# = 'S2')  Why is it incorrect? It would only return S2 JONES.  What is it asking? Consider a supplier, include it in the result if there exists a shipment by this supplier and the S# is 'S2'.  The next query is a solution involving only one get.  RANGE SUPPLIER s  RANGE SHIPMENT hs2  RANGE SHIPMENT h  GET w (s.SNAME) : ∃h (h.S# = s.S#  ^ ∃hs2 (h.P# = hs2.P# ^ hs2.S# = 'S2'))   |  |  | | --- | --- | | **S#** | **SNAME** | | S1 | SMITH | | S2 | JONES | | S3 | BLAKE | | S4 | FORD |   Note that S2 JONES is in the result. How can we eliminate S2 Jones?  RANGE SUPPLIER s  RANGE SHIPMENT hs2  RANGE SHIPMENT h  GET w (s.SNAME) : **s.S# ≠ 'S2'** ^ ∃h (h.S# = s.S# ^  ^ ∃hs2 (h.P# = hs2.P# ^ hs2.S# = 'S2')) |
| **Universal Quantification**  What does the query on the right mean?  A supplier should be included in the result if **none of its** shipments are for part P1.  **SHIPMENT**   |  |  |  | | --- | --- | --- | | **S#** | **P#** | **QUANTITY** | | S1 | P1 | 300 | | S1 | P2 | 200 | | S1 | P3 | 400 | | S1 | P4 | 200 | | S1 | P5 | 100 | | S1 | P6 | 100 | | S2 | P1 | 300 | | S2 | P2 | 400 | | S3 | P2 | 200 | | S4 | P2 | 200 | | S4 | P4 | 300 | | S4 | P5 | 400 | | **Example SP-12:**  Get the supplier names of suppliers who do **not** supply part P1.  An incorrect answer. Why?  RANGE SUPPLIER s  RANGE SHIPMENT h  Get w (s.SNAME): ∀h (s.S# = h.S# ^ h.P# ≠ 'P1')  What does the qualifier mean? To be true, it must be true for **all** shipment tuples. This can only be true if all the shipments are by **the same** supplier.  A solution:  RANGE SUPPLIER s  RANGE SHIPMENT h  Get w (s.SNAME): ∀h (s.S# ≠ h.S# ∨ h.P# ≠ 'P1')  Consider a supplier. For all shipment tuples, either the supplier isn't the one we are currently considering or the part number is not P1.  Another solution:  RANGE SUPPLIER s  RANGE SHIPMENT h  Get w (s.SNAME): ∀h (s.S# = h.S# → h.P# ≠ 'P1')  Consider a supplier. Include its name in the result if for all shipments by that supplier the P# is not P1.  Recall a → b = ~a ∨ b |

|  |  |
| --- | --- |
| **ALL** | **Example SP-13:**  Get the names of suppliers who supply **all** of the parts.  Consider a supplier. For all the parts, there must be a shipment of the part by the supplier.  RANGE SUPPLIER s  RANGE SHIPMENT h  RANGE PART p  Get w (s.SNAME): ∀p ∃h (s.S# = h.S# ^ h.P# = p.P#) |
|  | **Example SP-14:**  Get the supplier numbers of suppliers who supply all the parts.  Correct Answer:  RANGE SUPPLIER s  RANGE SHIPMENT h  RANGE PART p  Get w (s.S#): ∀p ∃h (s.S# = h.S# ^ h.P# = p.P#)  Incorrect answer A:  RANGE SHIPMENT h  RANGE PART p  Get w (h.S#): ∀p ∃h (h.P# = p.P#)  This is invalid since a tuple variable cannot be redefined within its scope. A variable in the target list cannot be quantified.  Incorrect answer B:  RANGE SHIPMENT h  RANGE PART p  Get w (h.S#): ∀p (h.P# = p.P#)  Why is this incorrect?  ?? |

|  |  |
| --- | --- |
| **ONLY** | **Example SP-15:** Get the names of suppliers who **only** supply part P2.  Consider a supplier. Include its name in the result if for all shipments by that supplier the P# is P2.  RANGE SUPPLIER s  RANGE SHIPMENT h  Get w (s.SNAME): ∀h (s.S# = h.S# → h.P# = 'P2').  Did that include supplier S5 in the answer although S5 doesn't have shipments?  ??  How can we eliminate suppliers who didn't have a shipment?  RANGE SUPPLIER s  RANGE SHIPMENT h  Get w (s.SNAME): ∀h (s.S# = h.S# → h.P# = 'P2') ^  ∃h (s.S# = h.S#) |
| **ALL of a subset**  First decide what we want all of …  For all red parts, the supplier must have a shipment. | **Example SP-16:** Get the names of suppliers who supply **all** of the red parts.  Incorrect answer:  RANGE SUPPLIER s  RANGE SHIPMENT h  RANGE PART p  Get w (s.SNAME): ∀p (p.COLOR = 'RED'  ^ ∃h (h.P# = p.P# ^ s.S# = h.S#))  What could cause the qualifier to always be false?  ??  Correct answer:  RANGE SUPPLIER s  RANGE SHIPMENT h  RANGE PART p  Get w (s.SNAME): ∀p (p.COLOR = 'RED'  → ∃h (h.P# = p.P# ^ s.S# = h.S#))  Consider a supplier. Include its name in the result if for all red parts, there exists a shipment of the part by this supplier. |
| **ONLY**  **Supplier Parts DB sample Data**   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **SUPPLIER**   |  |  |  |  | | --- | --- | --- | --- | | **S#** | **SNAME** | **STATUS** | **CITY** | | S1 | SMITH | 20 | LONDON | | S2 | JONES | 10 | PARIS | | S3 | BLAKE | 30 | PARIS | | S4 | FORD | 20 | LONDON | | S5 | ADAMS | 30 | ATHENS |   **Parts supplied by suppliers not in London**   |  | | --- | | **P#** | | P1 | | P2 | | P3 | | P4 | | P5 | | P6 | | **SHIPMENT**   |  |  |  | | --- | --- | --- | | **S#** | **P#** | **QTY** | | S1 | P1 | 300 | | S1 | P2 | 200 | | S1 | P3 | 400 | | S1 | P4 | 200 | | S1 | P5 | 100 | | S1 | P6 | 100 | | S2 | P1 | 300 | | S2 | P2 | 400 | | S3 | P2 | 200 | | S4 | P2 | 200 | | S4 | P4 | 300 | | S4 | P5 | 400 | | | **Example SP-17:** Get the name and part number of parts **only** supplied by suppliers in  London.  Incorrect answer:  RANGE SUPPLIER s  RANGE SHIPMENT h  RANGE PART p  Get W17\_A (p.P#, p.PNAME): ∀s (s.CITY = 'London'  → ∃h (h.P# = p.P# ^ s.S# = h.S#))  This solves something else. What did this return?  Result: W17\_A   |  |  | | --- | --- | | P# | PNAME | | P2 | BOLT | | P4 | SCREW | | P5 | CAM |   What did it mean?  Consider a part. Include its P# and PNAME in the result if for all suppliers in London, there is a shipment of the part.  This is giving us the parts that are supplied by ALL of the suppliers in London.  Correct answer:  RANGE SUPPLIER s  RANGE SHIPMENT h  RANGE PART p  Get W17 (p.P#, p.PNAME): ∀h (h.P# = p.P#  → ∃s (s.S# = h.S# ^ s.CITY = 'London'))  Which parts are in the result?  **Result:**   |  |  | | --- | --- | | **P#** | **PNAME** | | P3 | SCREW | | P4 | SCREW | | P5 | CAM | | P6 | COG |   What did it mean?  Consider a part. Include its P# and PNAME in the result if for all shipments of this part, the supplier is in London.  This is giving us the parts that are only supplied by suppliers in London. |
| **Selection from multiple relations with multiple criteria** | **Example SP18:** For shipments with a quantity < 500 by suppliers with a status greater than 10 of a part with a weight greater than 12, get the part#.  RANGE ??  ??  Result:   |  | | --- | | P# | | P2 | | P3 | | P4 | | P6 | |
| **Selection from multiple relations with multiple criteria** | **Example SP19:** For shipments with a quantity < 500 by suppliers with a status greater than 10 of a part with a weight greater than 12, get the part number, part name, supplier number, and supplier name.  RANGE ??  ??  Result:   |  |  |  |  | | --- | --- | --- | --- | | P# | PNAME | S# | SNAME | | P2 | BOLT | S1 | SMITH | | P2 | BOLT | S3 | BLAKE | | P2 | BOLT | S4 | FORD | | P3 | SCREW | S1 | SMITH | | P4 | SCREW | S1 | SMITH | | P4 | SCREW | S4 | FORD | | P6 | COG | S1 | SMITH | |
|  |  |
|  |  |

**©2020 Larry W. Clark,** UTSA CS students may make copies for their personal use