**Overview**: *Rel* is an open source desktop database management system from Dave Voorhis that implements **Tutorial D**, a relational database language designed by Chris Date and Hugh Darwen. For more information, see <https://reldb.org>, <http://thethirdmanifesto.com> and <http://www.dcs.warwick.ac.uk/~hugh/TTM/Tutorial%20D%202016-09-22.pdf> **Tutorial D** is not SQL.

**Launching**: Download instructions are at <https://reldb.org/c/index.php/download/> Once downloaded, open the folder – or go to Applications on macOS – and run the *Rel* executable.

****

***Rel* command-line**: In the upper-right hand corner of the *Rel* window, there are these three icons. The left icon is for the main *Rel* user interface, the middle icon is the visual query editor, and the right icon is the command-line.

**Loading a database script**: Go to the command-line, select the Load File icon, load the file. Press F5 to execute.

**Evaluating expressions and statements**: Type the expression at the command-line and press F5. Statements always end with a semicolon; expressions do not.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Scalar expressions**  **3 + 4** 7  **3.4 + 5.6** 9.0  **3.4 > 5.6** false  **1.2 < 3.4** true  **"a" || "bcd"** abcd  **'a' || 'bcd'** abcd  **SIN(0.25)** 0.24740395925452294 | **Tuple expressions**  **TUPLE {x 1, y 2.3, z 'zap'}**   |  |  |  | | --- | --- | --- | | *x* 1 | *y* 2.3 | *z* zap |   **TUPLE {x 1, y 2.3, z 'zap'} JOIN TUPLE {p 1, q 4.3, r true}**   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | *x* 1 | *y* 2.3 | *z* zap | *p* 1 | *q* 4.3 | *r* true |   **TUPLE {x 1, y 2.3, z 'zap'} MINUS TUPLE {p 1, q 4.3, r true}**   |  |  |  | | --- | --- | --- | | *x* 1 | *y* 2.3 | *z* zap |   **TUPLE {x 1, y 2.3, z 'zap'} MINUS TUPLE {x 1, y 2.3}**   |  | | --- | | *z* zap |     **TUPLE {x 1, y 2.3, z 'zap'} UNION TUPLE {x 1, y 2.3}**   |  |  |  | | --- | --- | --- | | *x* 1 | *y* 2.3 | *z* zap |   **TUPLE {x 1, y 2.3, z 'zap'} UNION TUPLE {x 1, y 2.3, r 4.5}**   |  |  |  |  | | --- | --- | --- | --- | | *x* 1 | *y* 2.3 | *z* zap | *r* 4.5 | |
| **Relational expressions**  // Return value of relvar S  S  // Join S and P on common attributes  S JOIN P  // Return tuples of S that match tuples in P,  // based on common attributes  S MATCHING P  // Return tuples of S that do not match tuples in P,  // based on common attributes  S NOT MATCHING P  // Join S and P on common attributes;  // do not include common attributes  S COMPOSE P  // Return tuples of S where STATUS is greater than 10  S WHERE STATUS > 10  // Return tuples of S where SNAME equals  // NAME(‘Smith’). NAME is a user-defined type.  S WHERE SNAME = NAME('Smith')  // Return UNION of tuples of S where SNAME equals  // NAME(‘Smith’) with tuples  // of S where STATUS equals 30.  (S WHERE SNAME = NAME('Smith')) UNION (S WHERE STATUS = 30)  // Return tuples of S with S#, SNAME and STATUS  // attributes converted to a relation-valued attribute X.  S GROUP {S#, SNAME, STATUS} AS X  // Return tuples of S with S#, SNAME and STATUS  // attributes converted to a tuple-valued attribute X.  S WRAP {S#, SNAME, STATUS} AS X  // Get the single tuple from S WHERE STATUS = 10.  // Error if there isn’t exactly 1 tuple.  TUPLE FROM (S WHERE STATUS = 10)  // Get the SNAME attribute from the tuple from  // S WHERE STATUS = 10. Error if there isn’t 1.  SNAME FROM TUPLE FROM (S WHERE STATUS = 10)  // Project S on SNAME and STATUS.  S {SNAME, STATUS}  // Return value of S with SNAME renamed to NAME;  // STATUS renamed to STAT.  S RENAME {SNAME AS NAME, STATUS AS STAT}  // Return the scalar sum of the STATUS attribute of S.  SUM(S, STATUS)  // Return the scalar sum of an expression.  SUM(S, STATUS \* 2)  // Obtain total of STATUS attribute grouped by CITY  SUMMARIZE S BY {CITY}: {TOTAL := SUM(STATUS)}  // Obtain total of STATUS attribute, and count of tuples,  // grouped by CITY  SUMMARIZE S BY {CITY}:     {N := COUNT(), TOTAL := SUM(STATUS)}  // Obtain total of STATUS attribute times two,  // and count of tuples, grouped by CITY  SUMMARIZE S BY {CITY}:     {N := COUNT(), TOTAL := SUM(STATUS \* 2)}  // Calculate new attribute values from expressions.  EXTEND S: {BIGSTATUS := STATUS \* 10, R := 'Test'} | **Relvars**  **VAR myVariable REAL RELATION {x INT, y RATIONAL, z CHAR} KEY {x};**  Ok.  **INSERT myVariable RELATION {      TUPLE {x 1, y 2.3, z 'zap'},      TUPLE {x 2, y 3.4, z 'zot'},      TUPLE {x 3, y 4.2, z 'zaz'} };**  Ok.  **myVariable**   |  |  |  | | --- | --- | --- | | **x *INTEGER*** | **y *RATIONAL*** | **z *CHARACTER*** | | 1 | 2.3 | zap | | 2 | 3.4 | zot | | 3 | 4.2 | zaz |   **UPDATE myVariable WHERE x > 2: {y := y + 4.2, z := z || 'gurgle'};**  Ok.  **myVariable**   |  |  |  | | --- | --- | --- | | **x *INTEGER*** | **y *RATIONAL*** | **z *CHARACTER*** | | 1 | 2.3 | zap | | 2 | 3.4 | zot | | 3 | 8.4 | zazgurgle |   **DELETE myVariable WHERE x = 1;**  Ok.  **myVariable**   |  |  |  | | --- | --- | --- | | **x *INTEGER*** | **y *RATIONAL*** | **z *CHARACTER*** | | 2 | 3.4 | zot | | 3 | 8.4 | zazgurgle |   // Describe all relvars in the database  sys.Catalog  // Get the names of all relvars in the database  sys.Catalog {Name} |
| **User-defined operators and types**  // This is a user-defined operator  OPERATOR myOperator (x INTEGER, y INTEGER) RETURNS INTEGER;       RETURN x + y \* 2;  END OPERATOR;  // Evaluate  myOperator(3, 4) \* 2  // Execute  CALL myOperator(3, 4);  // Drop operator  DROP OPERATOR myOperator(INTEGER, INTEGER);  // User-defined type  TYPE myNewTYPE POSSREP {x INT, y CHAR};  // Value of type myNewTYPE  myNewTYPE(2, 'zot')  // Relvar using user-defined type  VAR myNewRelvar REAL RELATION {x INT, y myNewTYPE} KEY {x}; |