Esercizio 1

Esprimere la semantica operazionale dell'operatore relazionale di selezione mediante una notazione imperativa:

S := **select** [*p*] R

Esercizio 1

Esprimere la semantica operazionale dell'operatore relazionale di selezione mediante una notazione imperativa:

```
S := select [ p ] R
```

Operatore relazionale	*Semantica operazionale*

`S := select [ p ] R`	`S := {};`
	`n := length(R);`
	`for i := 1 to n do`
	`begin`
	`  t := R[i];`
	`  ok := p(t);`
	`  if ok = true then`
	`  insert(t, S)`
	`end.`

Esercizio 2

Specificare la semantica operazionale dell'operatore relazionale di unione mediante una notazione imperativa:

```
R := X union Y
```

Stabilendo che le tuple di **`X`** ed **`Y`** debbano avere identico schema, si richiede inoltre di gestire gli eventuali errori semantici. Si assume che il linguaggio di specifica fornisca tipi di dati atomici, record e vettori (ma non insiemi).

Esercizio 2

Specificare la semantica operazionale dell'operatore relazionale di unione mediante una notazione imperativa:

```
R := X union Y
```

Stabilendo che le tuple di **X** ed **Y** debbano avere identico schema, si richiede inoltre di gestire gli eventuali errori semantici. Si assume che il linguaggio di specifica fornisca tipi di dati atomici, record e vettori (ma non insiemi).

```
var R: schema(X);

x := schema(X);
y := schema(Y);
if(x != y) errore();

R := X;
for i:=1 to length(Y) do
  if not member(Y[i], R) then
    insert(Y[i], R)
end.
```

Esercizio 3

Esprimere la semantica operazionale dell'operatore relazionale di intersezione mediante una notazione imperativa:

```
T := R  intersect S
```

Stabilendo che le tuple di `R` ed `S` debbano essere compatibili per struttura, si richiede inoltre di gestire gli eventuali errori semantici.

Esercizio 3

Esprimere la semantica operazionale dell'operatore relazionale di intersezione mediante una notazione imperativa:

```
T := R  intersect S
```

Stabilendo che le tuple di `R` ed `S` debbano essere compatibili per struttura, si richiede inoltre di gestire gli eventuali errori semantici.

```
var T: schema(R);

T := {};
r := schema(R);
s := schema(S);
if length(r) != length(s) then
  errore();
for i := 1 to length(r) do
  if domain(r[i]) != domain(s[i]) then
    errore()
end-for;
for i := 1 to length(R) do
  if R[i] in S then
    insert(R[i], T)
end-for.
```

Esercizio 4

Esprimere la semantica operazionale dell'operatore relazionale di join mediante una notazione imperativa:

T := R **Join** [p] S

Esercizio 4

Esprimere la semantica operazionale dell'operatore relazionale di join mediante una notazione imperativa:

T := R **Join** [*p*] S

```
var T: schema(R) || schema(S);
T := {};
nr := length(R);
ns := length(S);
for i := 1 to nr do
  r := R[i];
  for j := 1 to ns do
    s := S[j];
    t := r || s;
    ok := p(t);
    if ok = true then
      insert(t, T)
  end-for
end-for.
```

Esercizio 5

Specificare la semantica operazionale della seguente istruzione SQL-like di aggiornamento di una tabella:

update T
set a **=** E
where p

(*esempio*)

```
update Impiegati
set stipendio = stipendio + 0.2 * stipendio
where categoria <= 3
```

in cui *T* è il nome della tabella operando, *a* è il nome dell'attributo da aggiornare, *E* è l'espressione di aggiornamento e *p* il predicato di selezione delle righe da aggiornare. Si assume che il linguaggio di specifica della semantica sia imperativo.

Esercizio 5

Specificare la semantica operazionale della seguente istruzione SQL-like di aggiornamento di una tabella:

```
update T
set a = E
where p
```

(*esempio*)

```
update Impiegati
set stipendio = stipendio + 0.2 * stipendio
where categoria <= 3
```

in cui *T* è il nome della tabella operando, *a* è il nome dell'attributo da aggiornare, *E* è l'espressione di aggiornamento e *p* il predicato di selezione delle righe da aggiornare. Si assume che il linguaggio di specifica della semantica sia imperativo.

```
n := length(T);
for i=1 to n do
  if p(T[i]) then
    T[i].a := E(T[i]);
```

Esercizio 6

Specificare la semantica operazionale della seguente espressione relazionale di estensione di una tabella:

extend T
by a **=** E
where p

(*esempio di frase*)

```
extend Prodotti
by netto = lordo - tara
where costo > 100
```

Note:

- Il nome del nuovo attributo non può coincidere con il nome di un altro attributo;
- Il nuovo attributo viene inserito in coda allo schema della tabella;
- Le tuple che non soddisfano il predicato della clausola **`where`** vengono estese con il valore **`NULL`**;
- L'espressione di estensione non altera la tabella operando, ma genera una nuova tabella.

Esercizio 6

Specificare la semantica operazionale della seguente espressione relazionale di estensione di una tabella:

extend T
by a **=** E
where p

(*esempio di frase*)

```
extend Prodotti
by netto = lordo - tara
where costo > 100
```

Note:

- Il nome del nuovo attributo non può coincidere con il nome di un altro attributo;
- Il nuovo attributo viene inserito in coda allo schema della tabella;
- Le tuple che non soddisfano il predicato della clausola **`where`** vengono estese con il valore **`NULL`**;
- L'espressione di estensione non altera la tabella operando, ma genera una nuova tabella.

```
n := length(T);
t := schema(T);
if (a,_)∈t then errore();
r := t ∪ [(a,type(E)];
R := ∅;
for i=1 to n do
  newval := p(T[i]) ? E(T[i]) : NULL;
  newtup := append(T[i],newval);
  insert(R,newtup)
end-for;
return R.
```

Esercizio 7

Specificare la semantica operazionale dell'operatore relazionale di proiezione mediante una notazione imperativa:

```
Y := project [A] X
```

in cui `X` è l'operando della proiezione ed `A` la lista degli attributi di `X` su cui effettuare la proiezione. Ecco un esempio (in cui la parte in giallo è lo schema, mentre la parte in verde è l'istanza):

```
Y := project [a,c] X
```

(a:**integer**)	(b:**bool**)	(c:**string**)	(d:**integer**)
3	true	alfa	23
5	false	beta	12
20	true	gamma	5
3	false	alfa	8

⇨

(a:**integer**)	(c:**string**)
3	alfa
5	beta
20	gamma

In particolare, si richiede di computare le variabili **`sy`** ed **`iy`**, che rappresentano rispettivamente lo schema e l'istanza del risultato. Si richiede inoltre di verificare che la lista **`A`** non contenga nomi che non siano attributi dello schema dell'operando. Si assumono le seguenti funzioni ausiliarie (di cui non è richiesta la codifica):

- **`schema(X)`**: lista di coppie (nome, tipo) che definiscono lo schema di **`X`**;
- **`instance(X)`**: lista di tuple che definiscono l'istanza di **`X`**;
- **`attributes(A)`**: lista degli attributi di proiezione in **`A`**;
- **`error()`**: funzione di errore (chiamata in caso di errore semantico).

Esercizio 7

Specificare la semantica operazionale dell'operatore relazionale di proiezione mediante una notazione imperativa:

```
Y := project [A] X
```

in cui X è l'operando della proiezione ed A la lista degli attributi di X su cui effettuare la proiezione.
In particolare, si richiede di computare le variabili **sy** ed **iy**, che rappresentano rispettivamente lo schema e l'istanza del risultato. Si richiede inoltre di verificare che la lista **A** non contenga nomi che non siano attributi dello schema dell'operando. Si assumono le seguenti funzioni ausiliarie (di cui non è richiesta la codifica):

- **schema(X)**: lista di coppie (nome, tipo) che definiscono lo schema di **X**;
- **instance(X)**: lista di tuple che definiscono l'istanza di **X**;
- **attributes(A)**: lista degli attributi di proiezione in **A**;
- **error()**: funzione di errore.

check

```
foreach a in attributes(A) do
  if (a,_) not in schema(X) then
    errore()
  end-if
end-for;
```

schema

```
sy = [];
foreach (a,t) in schema(X) do
  if a in attributes(A) then
    sy += (a,t)
  end-if
end-for;
```

istanza

```
iy = {};
foreach x in instance(X) do
  y = [];
  foreach (a,_) in sy do
    y += x.a
  end-for;
  if y not in iy then
    iy += y
  end-if
end-for.
```

Esercizio 8

Specificare la semantica operazionale dell'operatore relazionale di differenza (insiemistica) di tabelle:

`X \ Y`

sulla base dei seguenti requisiti:

- È richiesta solo la specifica operazionale dell'istanza del risultato (quindi, non lo schema);
- Si assume che le variabili che rappresentano le tabelle siano definite ed abbiano un valore (anche vuoto);
- Gli schemi dei due operandi devono essere compatibili per struttura;
- Per la specifica, sono disponibili le seguenti funzioni ausiliarie (di cui non è richiesta l'implementazione):

  `schema(t)`: schema della tabella `t`, espresso come array di coppie `(attributo,tipo)`;

  `istanza(t)`: istanza della tabella `t`, espressa come array di tuple;

  `length(a)`: numero di elementi dell'array `a`;

  `member(elem,a)`: appartenenza di `elem` all'array `a`;

  `insert(elem,a)`: inserimento di `elem` nell'array `a` (in coda);

- Nel caso di errore semantico, il risultato della differenza è **`errore`**.

Esercizio 8

Specificare la semantica operazionale dell'operatore relazionale di differenza (insiemistica) di tabelle:

X \ Y

```
schemax = schema(X); nx = length(schemax);
schemay = schema(Y); ny = length(schemay);
if nx != ny then
  return errore
endif;
for i=1 to nx do
  if schemax[i].tipo != schemay[i].tipo then
    return errore
  endif
endfor;
instx = istanza(X);
insty = istanza(Y);
result = [];
for i=0 to length(instx) do
  if not member(instx[i], insty) then
    insert(instx[i] result)
  endif
endfor;
return result.
```

Esercizio 9

Specificare la semantica operazionale dell'operatore di contenimento (sottoinsieme) stretto tra tabelle (senza duplicati, per assunzione) mediante una notazione imperativa:

```
S ⊂ T
```

Si assumono le seguenti funzioni ausiliarie (di cui non è richiesta la specifica):

- `schema(X)`: lista di coppie (`nome`, `tipo`) che definiscono lo schema della tabella `X`;
- `istanza(X)`: lista di tuple che definiscono l'istanza della tabella `X`;
- `length(L)`: lunghezza della lista `L`.

Nel linguaggio di specifica è possibile esprimere l'operazione di appartenenza mediante il simbolo $\in$. Si richiede che le tabelle operando siano compatibili per struttura. Nel caso di errore semantico, il valore della espressione è `error`.

Esercizio 9

Specificare la semantica operazionale dell'operatore di contenimento (sottoinsieme) stretto tra tabelle (senza duplicati, per assunzione) mediante una notazione imperativa:

$S \subset T$

```
s = schema(S);
t = schema(T);
if length(s) != length(t) then
  return error
endif;
for i=0 to length(s) do
  if s[i].tipo != t[i].tipo then
    return error
  endif
endfor;
is = istanza(S);
it = istanza(T);
if length(is) >= length(it) then
  return false
endif;
foreach elem in is do
  if not (elem ∈ it) then
    return false
  endif
endfor;
return true.
```

check

Esercizio 10

Specificare la semantica operazionale dell'operatore di appartenenza di una tupla ad una tabella (senza duplicati, per assunzione) mediante una notazione imperativa:

```
t ∈ T
```

Si assumono le seguenti funzioni ausiliarie (di cui non è richiesta la specifica):

- `schema(X)`: lista di coppie (`nome`, `tipo`) che definisce lo schema della tupla o tabella `X`;
- `tupinst(t)`: record che definisce l'istanza della tupla `t`;
- `tabinst(T)`: lista di record che definisce l'istanza della tabella `T`;
- `length(L)`: lunghezza della lista `L`.

Si richiede che gli operandi siano compatibili per struttura. È possibile confrontare due tuple con l'operatore di uguaglianza. Nel caso di errore semantico, il valore della espressione è `errore`.

Esercizio 10

Specificare la semantica operazionale dell'operatore di appartenenza di una tupla ad una tabella (senza duplicati, per assunzione) mediante una notazione imperativa:

`t ∈ T`

```
st = schema(t);
sT = schema(T);
if length(st) != length(sT) then
  return errore
endif;
for i=0 to length(s)-1 do
  if st[i].tipo != sT[i].tipo then
    return errore
  endif
endfor;
it = tupinst(t);
iT = tabinst(T);
foreach tuple in iT do
  if it == tuple then
    return true
  endif
endfor;
return false.
```

Esercizio 11

Specificare la semantica operazionale dei seguenti operatori relazionali:

- `exists` [p] R : vero se e solo se esiste almeno una tupla in R che soddisfa il predicato p;
- `all` [p] R : vero se e solo se tutte le tuple di R soddisfano il predicato *p* oppure R è vuota.

Esercizio 11

Specificare la semantica operazionale dei seguenti operatori relazionali:

- `exists` [p] R : vero se e solo se esiste almeno una tupla in R che soddisfa il predicato p;
- `all` [p] R : vero se e solo se tutte le tuple di R soddisfano il predicato *p* oppure R è vuota.

```
// exists [ p ] R

foreach r in R do
  if p(r) then
    return true;
return false.
```

```
// all [ p ] R

foreach r in R do
  if not p(r) then
    return false;
return true.
```

Esercizio 12

Specificare la semantica operazionale della seguente espressione relazionale:

```
select [a > b and c = 10] R
```

sulla base dei seguenti requisiti:

- L'operatore logico di congiunzione è valutato in corto circuito;
- L'operando dela selezione è una tabella i cui attributi sono di tipo intero;
- Nel caso di errore semantico, si restituisce la chiamata della funzione `errore` il cui argomento è un messaggio di errore;
- È disponibile la funzione ausiliaria `schema(X)`, che restituisce la lista dei nomi degli attributi della tabella `X`.

Esercizio 12

Specificare la semantica operazionale della seguente espressione relazionale:

```
select [a > b and c = 10] R
```

```
var T: schema(R);
    ok: boolean;

T = [];

if a not in schema(R) or
   b not in schema(R) or
   c not in schema(R) then
  return errore("Attributo non definito");

foreach r in R do
  if r.a <= r.b then
    ok = false
  else
    ok = r.c == 10
  endif;
  if ok then
    insert(r, T)
  endif
endfor;

return T.
```

Esercizio 13

Specificare la semantica operazionale della seguente espressione relazionale:

```
join [x = y and z > 10] R S
```

(in cui `R` ed `S` sono gli operandi dell'operazione di join) sulla base dei seguenti requisiti:

- L'operatore logico di congiunzione è valutato in corto circuito;
- Gli attributi delle tabelle operando sono interi o stringhe;
- Nei confronti all'interno del predicato, nel caso di identificatore, si assume che il primo operando appartenga alla tabella `R` ed il secondo alla tabella `S`;
- Non sono ammesse operazioni miste;
- Nel caso di errore semantico, si invoca la funzione `error`, il cui argomento è un messaggio (pertinente) di errore, la quale termina l'esecuzione;
- Sono disponibili le seguenti funzioni ausiliarie:
  - `schema(T)`: restituisce la lista delle coppie (*attributo*, *tipo*) della tabella `T`,
  - `attributes(T)`: restituisce la lista degli attributi della tabella `T`,
  - `type(attr, schema)`: restituisce il tipo dell'attributo `attr` nello schema;
- È possibile utilizzare gli operatori insiemistici di appartenenza, unione ed intersezione applicati a liste.

Esercizio 13

Specificare la semantica operazionale della seguente espressione relazionale:

```
join [x = y and z > 10] R S
```

```
var T: schema(R) ∪ schema(S);
    ok: boolean;
T = [];
if attributes(R) ∩ attributes(S) ≠ ∅ then error("Repeated attribute name") endif;
if x ∉ attributes(R) or y ∉ attributes(S) then error("Undefined attribute") endif;
if type("x", schema(R)) ≠ type("y", schema(S)) then error("Illegal comparison") endif;
if z ∉ attributes(S) then error("Undefined attribute") endif;
if type("z", schema(S)) ≠ INTEGER then error("Illegal comparison") endif;
foreach r in R do
  foreach s in S do
    t =  r ∪ s;
    if t.x ≠ t.y then
      ok = false
    else
      ok = t.z > 10
    endif;
    if ok then
      insert(t, T)
    endif
  endfor
endfor;
return T.
```

Esercizio 14

Specificare la semantica operazionale della espressione relazionale corrispondente alla seconda istruzione nel seguente frammento di programma (in cui **`in`** denota l'operatore di appartenenza):

```
R: table(a: integer, b: integer, S: table(c: integer));
select [ a = b and b in S ] R;
```

sulla base dei seguenti requisiti:

- L'operatore di congiunzione (**`and`**) ha precedenza minima ed è valutato in corto circuito;
- L'ordine di valutazione degli operandi è da sinistra a destra;
- Il linguaggio di specifica non dispone di un operatore di appartenenza;
- Il linguaggio di specifica dispone però della struttura di controllo **`foreach x in X do ...`** (per iterare su tutti gli elementi `x` di un insieme `X`).

Esercizio 14

Specificare la semantica operazionale della espressione relazionale corrispondente alla seconda istruzione nel seguente frammento di programma (in cui **in** denota l'operatore di appartenenza):

```
R: table(a: integer, b: integer, S: table(c: integer));
select [ a = b and b in S ] R;
```

```
var T: table(a: integer, b: integer, S: table(c: integer));
    ok: boolean;

T := [];
foreach r in R do
  ok := false;
  if r.a = r.b then
    foreach s in r.S do
      if r.b == s then
        ok := true;
        break
      endif
    endfor
  endif;
  if ok then
    insert(r, T)
  endif
endfor;
return T.
```

Esercizio 15

Specificare la semantica operazionale della seguente espressione di selezione:

```
select [ select [a > b] X = select [c > d] Y ] R;
```

assumendo che `R` sia una tabella complessa e non ordinata così definita:

```
R: table(X: table(a: integer, b: integer), Y: table(c: integer, d: integer));
```

La selezione restituisce le tuple di `R` che soddisfano l'uguaglianza delle selezioni interne sugli attributi complessi `X` ed `Y`. Il linguaggio di specifica operazionale fornisce l'operatore polimorfo di uguaglianza (==), applicabile a qualsiasi tipo, e gli operatori di confronto (<, >), applicabili solo agli interi.

Esercizio 15

Specificare la semantica operazionale della seguente espressione di selezione:

```
select [ select [a > b] X = select [c > d] Y ] R;
```

assumendo che R sia una tabella complessa e non ordinata così definita:

```
R: table(X: table(a: integer, b: integer), Y: table(c: integer, d: integer));
```

La selezione restituisce le tuple di R che soddisfano l'uguaglianza delle selezioni interne sugli attributi complessi X ed Y. Il linguaggio di specifica operazionale fornisce l'operatore polimorfo di uguaglianza (==), applicabile a qualsiasi tipo, e gli operatori di confronto (<, >), applicabili solo agli interi.

```
R': table(X: table(a: integer, b: integer), Y: table(c: integer, d: integer));

R' = {};
foreach r in R do
  X' = {};
  foreach x in r.X do
    if x.a > x.b then insert(x, X') endif
  endfor;
  Y' = {};
  foreach y in r.Y do
    if y.c > y.d then insert(y, Y') endif
  endfor;
  if X' == Y' then insert(r, R') endif
endfor;
return R'.
```

Esercizio 16

Specificare la semantica operazionale della seguente espressione di selezione:

```
select [ a > 0 and (select [ b > c ] X == select [ d > e ] Y) ] R
```

assumendo che R sia una tabella complessa e non ordinata così definita:

```
R: table(a: integer,
         X: table(b: integer, c: integer),
         Y: table(d: integer, e: integer));
```

L'operatore logico di congiunzione (**and**) è valutato in corto circuito. Il linguaggio di specifica operazionale fornisce l'operatore polimorfo di uguaglianza (==), applicabile a qualsiasi tipo, e gli operatori di confronto (<, >), applicabili solo agli interi.

Esercizio 16

Specificare la semantica operazionale della seguente espressione di selezione:

```
select [ a > 0 and (select [ b > c ] X == select [ d > e ] Y) ] R
```

assumendo che R sia una tabella complessa e non ordinata così definita:

```
R: table(a: integer,
         X: table(b: integer, c: integer),
         Y: table(d: integer, e: integer));
```

```
R': table (a: integer,
           X: table (b: integer, c: integer),
           Y: table (d: integer, e: integer));
R' := {};
foreach r in R do
  if a > 0 then
    X' := {};
    foreach x in r.X do
      if x.b > x.c then insert(x, X') endif
    endfor;
    Y' := {};
    foreach y in r.Y do
      if y.d > y.e then insert(y, Y') endif
    endfor;
    if X' == Y' then insert(r, R') endif
  endif
endfor;
return R'.
```

Esercizio 17

Specificare la semantica operazionale della seguente espressione di selezione:

```
select [ X ⊆ Y or w = z ] R
```

assumendo che `R` sia una tabella complessa e non ordinata, così definita:

```
R: table(X: table(a: integer, b: integer),
         Y: table(c: integer, d: integer),
         w: integer,
         z: integer);
```

L'operatore logico **`or`** è valutato in corto circuito, da destra a sinistra. Il linguaggio di specifica operazionale fornisce l'operatore di uguaglianza scalare (==) e gli operatori insiemistici di appartenenza (∈) e non-appartenenza (∉).

Esercizio 17

Specificare la semantica operazionale della seguente espressione di selezione:

```
select [ X ⊆ Y or w = z ] R
```

assumendo che `R` sia una tabella complessa e non ordinata, così definita:

```
R: table(X: table(a: integer, b: integer),
         Y: table(c: integer, d: integer),
         w: integer,
         z: integer);
```

```
R': table (X: table (a: integer, b: integer),
           Y: table (c: integer, d: integer)
           w: integer, z: integer);
R' := {};
foreach r in R do
  if w == z then
    ok := true
  else
    ok := true;
    foreach x in r.X do
      if x ∉ r.Y then ok := false; break endif
    endfor
  endif;
  if ok then insert(r, R') endif
endfor;
return R'.
```