Deductive Databases
Summer Term 2020
Prof. Dr. W. May

1. Unit: Relational Calculus

Self-contained subformulas (i.e., formulas in RANF) will also be needed for translating complex queries into Datalog programs. The transformation into RANF works in the same intuitive way as done in the DB lecture before knowing the relational calculus at all. For complex cases (and for the computer), a systematic formal procedure helps.

Exercise 1 (Kalkül: Sichere, Wertebereichsunabhängige Anfragen) Check for the following queries whether they are in SRNF (give $rr(G)$ for each of their subformulas).

For the formulas that are in RANF:

- check whether the formulas are in RANF. If not, give an equivalent formula in RANF. In such cases, give a small data example that illustrates the problem.
- Give equivalent expressions in the relational algebra and in SQL (develop the SQL expressions both from the original formula and from the RANF formula).

a) $F(X,Y,Z) = p(X,Y) \wedge (q(Y) \vee r(Z))$,
b) $F(X,Y) = p(X,Y) \wedge (q(Y) \vee r(X))$,
c) $F(X,Y) = p(X,Y) \wedge \neg\exists Z : r(Y,Z)$,
d) $F(X) = p(X) \wedge \exists Y : (q(Y) \wedge \neg r(X,Y))$,
e) $F(X) = p(X) \wedge \neg\exists Y : (q(Y) \wedge \neg r(X,Y))$
f) $F(X,Y) = \exists V : (r(V,X) \wedge \neg s(V,X,Y)) \wedge \exists W : (r(W,Y) \wedge \neg s(W,Y,X))$

---

a) $p(X,Y) \wedge (q(Y) \vee r(Z))$:

G	$rr(G)$
$p(x,y)$	X,Y
$q(Y)$	Y
$r(Z)$	Z
$q(Y) \vee r(Z)$	$\{Y\} \cap \{Z\} = \emptyset$
$p(X,Y) \wedge (q(Y) \vee r(Z))$	$\{X,Y\} \cup \emptyset = \{X,Y\}$

Since $free(F) = \{X,Y,Z\} \neq \{X,Y\} = rr(F)$, F is not in SRNF (and thus also not in RANF). F is not domain-independent: for $\mathcal{S}$ with $\mathcal{S}(p) = \{1,a\}$ and $\mathcal{S}(q) = \{(a)\}$ and $\mathcal{S}(r) = \emptyset$ and domain $\mathcal{D}$ is the answer set $\{X \mapsto 1, Y \mapsto a, Z \mapsto d | d \in \mathcal{D}\}$.

b) $p(X,Y) \wedge (q(Y) \vee r(X))$:

G	$rr(G)$
$p(x,y)$	X,Y
$q(Y)$	Y
$r(X)$	X
$q(Y) \vee r(X)$	$\{Y\} \cap \{X\} = \emptyset$
$p(X,Y) \wedge (q(Y) \vee r(X))$	$\{X,Y\} \cup \emptyset = \{X,Y\}$

Since $free(F) = \{X,Y\} = rr(F)$, F is in SRNF.

F is not in RANF since the disjunction $q(Y) \vee r(X)$ is not self-contained.
F can easily be expressed in SQL (with $P(P_1, P_2), Q(Q_1), R(R_1)$):

```
SELECT P1,P2
FROM P
WHERE P2 in (SELECT Q1 FROM Q)
  OR P1 in (SELECT R1 FROM R)
```

The equivalent expression in the relational algebra is
$(P \bowtie_{P_2=Q_1} Q) \cup (P \bowtie_{P_1=R_1} R)$.
This is also obtained when translating from SRNF to RANF with "push-into-or":
$F'(X,Y) = p(X,Y) \wedge (p(X,Y) \wedge q(Y)) \vee (p(X,Y) \wedge r(Z))$
and then translates as usual to the relational algebra.
The problem can be illustrated with the following database instances:

(i)

p	
1	2

q
2

r

(ii)

p	
1	2
3	4

q
2

r
3

Example (i): For the answer bindings of the subformula $H(X,Y) = q(Y) \vee r(X)$, only $\{X/??, Y/1\}$ can be given, which is not allowed. Nevertheless, $\{X/2, Y/1\}$ is an answer to $F(X,Y)$.
Example (ii) is even more mean: For $H(X,Y) = q(Y) \vee r(X)$, a "total" answer can be given as $\{Y/1, X = 3\}$, which is not an answer for the whole formula $F(X,Y)$, and one is tempted to conclude that the answer set to $F(X,Y)$ is empty. The correct set of answers to $F(X,Y)$ is $\{\{X/1, Y/2\}, \{X/3, Y/4\}\}$.

c) $F(X,Y) = p(X,Y) \wedge \neg\exists Z : r(Y,Z)$:

G	$rr(G)$
$p(X,Y)$	X,Y
$r(Y,Z)$	Y,Z
$\exists Z : r(Y,Z)$	Y
$\neg\exists Z : r(Y,Z)$	$\emptyset$
$p(X,Y) \wedge \neg\exists Z : r(Y,Z)$	X,Y

Since $free(F) = \{X\} = rr(F)$, F is in SRNF.
All subformulas are self-contained.
F can easily be expressed in SQL (with $P(P_1, P_2), R(R_1, R_2)$):

```
SELECT P1,P2
FROM P
WHERE P2 NOT IN (SELECT R2 FROM R)
```

The equivalent expression in the relational algebra is
$P \bowtie (\pi[P_2](P) - \pi[R_1](R))$.
The standard translation that uses the enumeration formula for the active domain (here: those that occur in P and R) reads as:
$P \bowtie ((\pi[P_1](P) \cup \pi[P_2](P) \cup \pi[R_1](R) \cup \pi[R_2](R)) - \pi[R_1](R))$.

d) $F(X) = p(X) \wedge \exists Y : (q(Y) \wedge \neg r(X,Y))$:

G	$rr(G)$
$p(X)$	X
$q(Y)$	Y
$r(X,Y)$	X,Y
$\neg r(X,Y)$	$\emptyset$
$q(Y) \wedge \neg r(X,Y)$	Y
$\exists Y : q(Y) \wedge \neg r(X,Y)$	$\emptyset$
$p(X) \wedge \exists Y : q(Y) \wedge \neg r(X,Y)$	X

Since $free(F) = \{X\} = rr(F)$, F is in SRNF.
F is not in RANF since the subformula $G = \exists Y : q(Y) \wedge \neg r(X,Y)$ is not self-contained: for the body $H = q(Y) \wedge \neg r(X,Y)$ there is $free(H) = \{X,Y\} \supsetneq \{Y\} = rr(H)$ (note that the SAFE criterion from the lecture would already detect H as the problem).
The problem can be illustrated with the following database instance:

push-into:

$$\exists Y : p(X) \wedge q(Y) \wedge \neg r(X,Y)$$

p
1
2
3

q
4
5

r	
1	4
1	5
2	5

The set of answers to $H(X,Y) = q(Y) \wedge \neg r(X,Y)$ can be constrained as a set as $\{(X,Y)|(Y = 4 \wedge X! = 1) \vee (Y = 5 \wedge X! = 1 \wedge X! = 2)\}$, but this is not a positive characterization as a finite set of tuples, since the possible values for X are not known. ~~nicht kompt~~ For X, all values from the domain $\mathcal{D}$ can be taken. If it is considered that the next step is $G(X) = \exists Y : H(X,Y)$, a possible characterization is $\{X|X \in \mathcal{D} \backslash \{1\}\}$, but this is not algebraic evaluation, but reasoning.
F can easily be expressed in SQL (with $P(P_1), Q(Q_1), R(R_1,R_2)$):

```
SELECT P1
FROM P
WHERE EXISTS (SELECT Q1
              FROM Q
              WHERE (P1,Q1) NOT IN (SELECT R1,R2 FROM R))
```

The equivalent expression in the relational algebra is
$\pi[P_1]((P \times Q) - \rho[R_1 \to P_1, R_2 \to P_2]R$.
This is also obtained when translating from SRNF to RANF with "push-into-exist":
$F'(X) = p(X) \wedge \exists Y : (p(X) \wedge q(Y) \wedge \neg r(X,Y))$,
and then translates as usual to the relational algebra.
This corresponds to the (simpler) SQL query

```
SELECT P1
FROM P, Q
WHERE (P1,Q1) NOT IN (SELECT R1,R2 FROM R)
```

e) This formula is the pattern of the relational division, $r \div q$. It is equivalent with $F(X) = p(X) \wedge \forall Y : (q(Y) \to r(X,Y))$.

$F(X) = p(X) \wedge \neg\exists Y : (q(Y) \wedge \neg r(X,Y))$,

G	$rr(G)$
$p(X)$	X
$q(Y)$	Y
$r(X,Y)$	X,Y
$\neg r(X,Y)$	$\emptyset$
$q(Y) \wedge \neg r(X,Y)$	Y
$\exists Y : q(Y) \wedge \neg r(X,Y)$	$\emptyset$
$\neg\exists Y : q(Y) \wedge \neg r(X,Y)$	$\emptyset$
$p(X) \wedge \neg\exists Y : q(Y) \wedge \neg r(X,Y)$	X

Since $free(F) = \{X\} = rr(F)$, F is in SRNF.
F is –as in (d)– not in RANF since the subformula $G = \exists Y : q(Y) \wedge \neg r(X,Y)$ is not self-contained.
The example from (d) can be reused. With the mentioned argument that in the next step, $G(X) = \neg\exists Y : H(X,Y)$ is asked for, it is clear that the answer set to $G(X) = \{\{X/1\}\}$. This shows that for such a *relational division*, the answer can be restricted to $\pi[\$1](r)$ independently from the surrounding formula (if the "required" set of combinations with q is not empty).
F can easily be expressed in SQL (with $P(P_1), Q(Q_1), R(R_1,R_2)$):

```
SELECT P1
FROM P
WHERE NOT EXISTS (SELECT Q1
                  FROM Q
                  WHERE (P1,Q1) NOT IN (SELECT R1,R2 FROM R))
```

The equivalent expression in the relational algebra is
$P - \pi[P_1]((P \times Q) - \rho[R_1 \to P_1, R_2 \to P_2](R))$.
the rel. div from the lecture
This is also obtained when translating from SRNF to RANF with "push-into-not-exist":
$F'(X) = p(X) \wedge \neg\exists Y : (p(X) \wedge q(Y) \wedge \neg r(X,Y))$,
and then translates as usual to the relational algebra.

f) This is an example for a conjunction, where none of the conjuncts is self-contained:
$F(X,Y) = \exists V : (r(V,X) \wedge \neg s(V,X,Y)) \wedge \exists W : (r(W,Y) \wedge \neg s(W,Y,X))$

G	$rr(G)$
$r(V,X)$	X,V
$s(V,X,Y)$	V,X,Y
$\neg s(V,X,Y)$	$\emptyset$
$r(V,X) \wedge \neg s(V,X,Y)$	X,V
$\exists V : (r(V,X) \wedge \neg s(V,X,Y))$	X
$r(W,Y)$	W,Y
$s(W,Y,X)$	W,Y,X
$\neg s(W,Y,X)$	$\emptyset$
$r(W,Y) \wedge \neg s(W,Y,X)$	W,Y
$\exists W : (r(W,Y) \wedge \neg s(W,Y,X))$	Y
$(\ldots) \wedge (\ldots)$	X,Y

Since $free(F) = \{X,Y\} = rr(F)$, F is in SRNF.
F is not in RANF since the subformulas $\exists V : (r(V,X) \wedge \neg s(V,X,Y))$ and $\exists W : (r(W,Y) \wedge \neg s(W,Y,X))$ are not self-contained (again, the problem is located inside each of the subformulas, as SAFE would complain about).
F can easily be expressed in SQL (with $P(P_1), Q(Q_1), R(R_1,R_2)$):

```
SELECT rv.R2, rv.R2
FROM R rv, R rw
WHERE NOT EXISTS (SELECT * FROM S
                  WHERE S1=rv.R1 and S2=rv.R2 and S3=rw.R2)
  AND NOT EXISTS (SELECT * FROM S
                  WHERE S1=rw.R1 and S2=rw.R2 and S3=rv.R2)
```

oder

```
SELECT rv.R2, rw.R2
FROM R rv, R rw
WHERE NOT (rv.R1, rv.R2, rw.R2 IN (SELECT * FROM S))
  AND NOT (rw.R1, rw.R2, rv.R2 IN (SELECT * FROM S))
```

The equivalent expression in the relational algebra is ... not that easy.

Thus, F has to be transformed from SRNF to RANF by moving the first conjunct into the second by "push-into-exists" and vice versa:

$$F' = \exists V \exists W : \varphi \wedge \exists W \exists V : \varphi$$

where (after reordering the atoms, which are the same in both subformulas) $\varphi = r(V, X) \wedge r(W, Y) \wedge \neg s(V, X, Y)) \wedge \neg s(W, Y, X)$. The formula F' is self-contained since $free(\varphi) = \{V, W, X, Y\} = rr(\varphi)$ and $free(F') = \{X, Y\} = rr(F')$.

The human "solver" sees that the formula is redundant, and can be reduced to

$$F' = \exists V \exists W : \varphi \ .$$

According to the RANF-to-algebra transformation algorithm given in the lecture, the following has to be done:

- build the (VXY) component of φ, subtract s,
- in parallel build the (WYX) component of φ, subtract s,
- these are the triples of bindings that "survive",
- join them on XY,
- and project to XY:

$$\begin{aligned}\pi[X,Y](\quad & (\pi[V,X,Y](\rho[R_1 \to V, R_2 \to X](r) \times \rho[R_1 \to W, R_2 \to Y](r)) \\ & -\rho[R_2 \to V, S_1 \to X, S_2 \to Y](s)) \\ \bowtie \ & (\pi[X,Y,W](\rho[R_1 \to V, R_2 \to X](r) \times \rho[R_1 \to W, R_2 \to Y](r)) \\ & -\rho[R_2 \to W, S_1 \to Y, S_2 \to X](s)))\end{aligned}$$

Note: the pure algorithmic procedure would maybe not detect that redundancy, and translate it straightforwardly in a redundant tree. Database-internal algebraic optimization would then remove the redundancy.

Note: another approach, also using a typical formula transformation, would go another way from the beginning:

- pull out the $\exists$ quantifiers (since $A(...) \wedge \exists x : B(x, ...) \equiv \exists x : (A(...) \wedge B(x, ...))$ if A does not contain x),
- obtain immediately $F'' = \exists V, W : \varphi$.
- F'' is in **prenex form**, i.e., all quantifiers are pulled in front. Here, it is even in **prenex literal normal form**, i.e. negations are pushed down until only literals are negated. The example formula is even only conjunctive, which will be required later on for Datalog.

The problem can be illustrated with the following database instance, which supports locking scheduler in a datbase system: unterstützt: $r(T_i, x)$ means that transaction T_i asks for an object

x to lock it for using it, and $s(T_i, x, y)$ means that T_i, before it will again unlock x, requires object y. Thus $F(X, Y)$ means that these objects are currently not in danger of a deadlock, and can be locked.

r	
T_1	b
T_2	a
T_3	c
T_4	d

s		
T_1	b	a
T_2	a	b
T_2	c	a
T_2	c	b

Die Antwort zu $G(X, Y) = \exists V : (r(V, X) \wedge \neg s(V, X, Y))$ sind also alle Paare (x, y) von Objekten, so dass es keine Transaktion V gibt, die X hält, und Y benötigen würde, bevor sie X wieder freigeben kann. Dies läßt sich nicht beantworten, welche Objekte Y überhaupt zur Zeit existieren, bzw. relevant sind.

Die Einschränkung $\exists V, W : r(V, X) \wedge r(W, Y)$ hingegen erlaubt es, die Betrachtung auf alle Objekte einzuschränken, die derzeit im Besitz irgendeiner Transaktion sind.

Exercise 2 (Relationale Anfragen an Mondial: Schweizer Sprachen) Give expressions in the relational calculus for the following queries against the Mondial database. Compare with the same queries in the relational Algebra and in SQL.

a) All codes of countries, in which some languages is spoken that is also spoken in Switzerland.
b) All codes of countries, in which only languages are spoken that are not spoken in Switzerland.
c) All codes of countries, in which only languages are spoken that are also spoken in Switzerland.
d) All codes of countries in which all languages that are spoken in Switzerland are also spoken.

german,french, italian, romansch

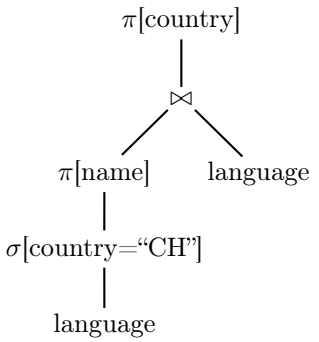
---

a) $F(C) = \exists L, Perc1, Perc2 : (\text{language}('CH', L, Perc1) \wedge \text{language}(C, L, Perc2))$

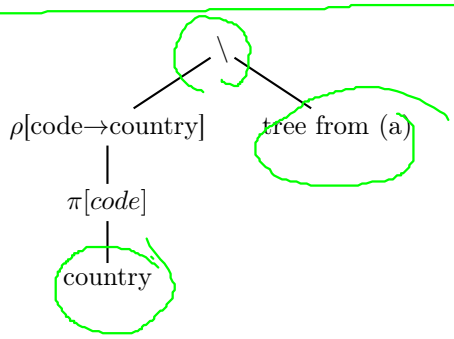
b) $F(C) = \exists CN, A, Pop, Cap, CapP :$
$(\text{country}(CN, C, A, Pop, Cap, CapP) \wedge$
$\neg \exists L, Perc1, Perc2 : (\text{language}('CH', L, Perc1) \wedge \text{language}(C, L, Perc2)))$

Algebra:

a)



(b)



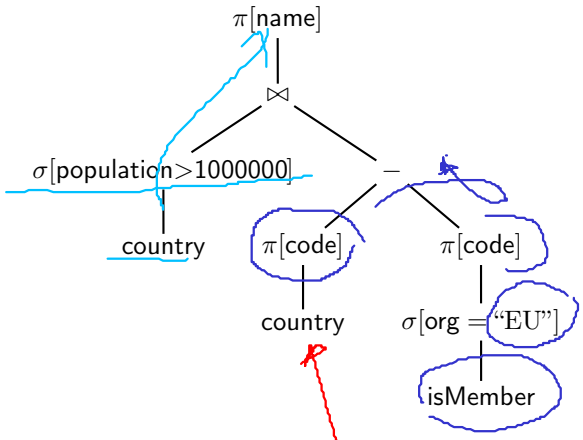
c) $F(C) = (\exists CN, A, Pop, Cap, CapP : \text{country}(CN, C, A, Pop, Cap, CapP)) \wedge$
$\neg \exists L, Perc1 : (\text{language}(C, L, Perc1) \wedge \neg \exists Perc1 : \text{language}('CH', L, Perc2))$

d) $F(C) = (\exists CN, A, Pop, Cap, CapP : \text{country}(CN, C, A, Pop, Cap, CapP)) \wedge$
$\forall L : ((\exists Perc1 : \text{language}('CH', L, Perc1)) \rightarrow (\exists Perc2 : \text{language}(C, L, Perc2)))$

Exercise 3 (RANF to Algebra – Minus) Give expressions in the relational algebra and in the relational calculus for the query *"Full names of all countries that have more than 1000000 inhabitants and are not member of the EU"*.

Check whether the calculus expression is in SRNF and RANF, and transform it into the relational algebra. Compare the result with the algebra expression.

A straightforward algebra expression is



The calculus expression is

$$F(N) = \exists C, Cap, CapProv, A, Pop : (\mathsf{country}(N, C, Cap, CapProv, A, Pop) \wedge Pop > 1000000 \wedge \neg\exists T : \mathsf{isMember}(C, \text{“EU”}, T)) .$$

It is in SRNF, it is safe range, and it is in RANF. Recall that for the subformula $\neg\exists T : \mathsf{isMember}(C, \text{“EU”}, T)$, RANF requires $rr(\exists T : \mathsf{isMember}(C, \text{“EU”}, T)) = free(\exists T : \mathsf{isMember}(C, \text{“EU”}, T)) = \{C\}$ which is the case.

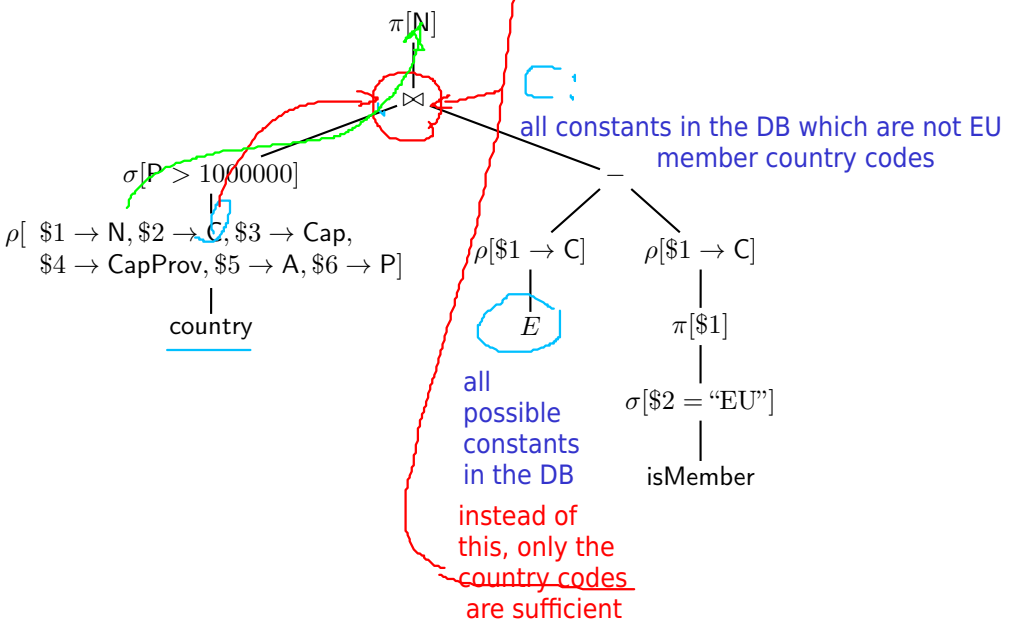
For the relational algebra,

$$\begin{array}{lcl} \mathsf{isMember}(C, \text{“EU”}, T) & \Rightarrow & \rho[\$1 \to C, \$3 \to T](\pi[\$1, \$3](\sigma[\$2 = \text{“EU”}](\mathsf{isMember}))) \\ \exists T : \mathsf{isMember}(C, \text{“EU”}, T) & \Rightarrow & \pi[\$1](\rho[\$1 \to C, \$3 \to T](\pi[\$1, \$3](\sigma[\$2 = \text{“EU”}](\mathsf{isMember})))) \\ & = & \rho[\$1 \to C](\pi[\$1](\sigma[\$2 = \text{“EU”}](\mathsf{isMember}))) \end{array}$$

For $\neg\exists T : \mathsf{isMember}(C, \text{“EU”}, T)$, let the expression E denote the algebra expression that enumerates all values of the active domain. With this,

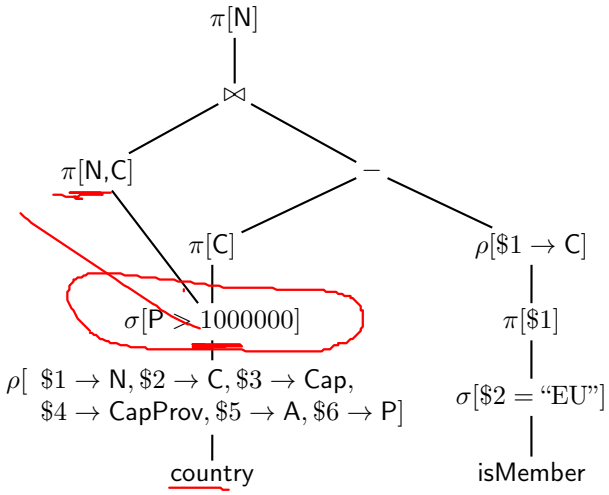
$$\neg\exists T : \mathsf{isMember}(C, \text{“EU”}, T) \Rightarrow \rho[\$1 \to C](E) - \rho[\$1 \to C](\pi[\$1](\sigma[\$2 = \text{“EU”}](\mathsf{isMember})))$$

Altogether, the whole query translates to

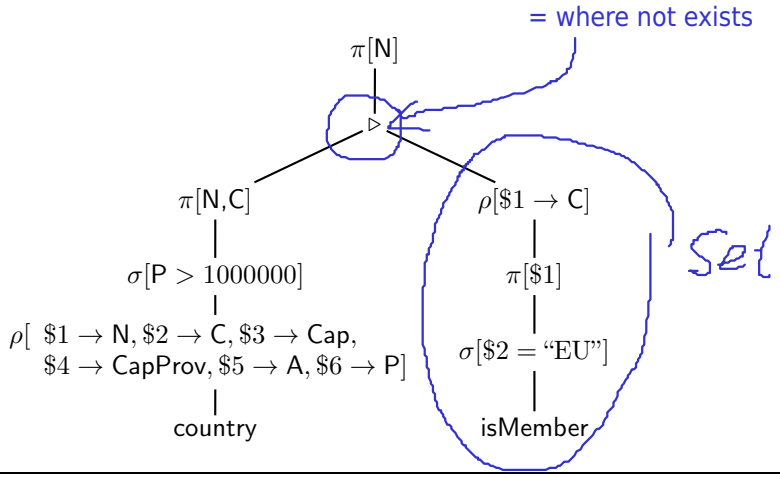


Obviously, the term $\rho[\$1 \to \mathsf{C}](E)$ can be replaced by $\rho[\$2 \to \mathsf{C}](\pi[\$2](\mathsf{country}))$ which enumerates a superset of all values of C that can result from the left subtree.

Instead, also $\rho[\$2 \to \mathsf{C}](\pi[\$2](\sigma[\$6 > 1000000](\mathsf{country})))$ is sufficient, which makes the left subtree (nearly) unnecessary. From it, only the full name must still be obtained.



Another possibility is the anti-join ▷ (which is one of the built-in operators of internal algebras):



Exercise 4 (RANF to Algebra – Minus) Transfer the notion of *domain-independence* to SQL and Relational Algebra expressions. Is there any danger, ~~or why not?~~

SQL:

- SFW queries: The FROM clause (and WHERE join conditions) specify a conjunctive query that acts as an upperbound and uses only tuples from the database.
- Don't forget the binary operations:
  - INTERSECT, MINUS: the first subquery acts as a restricted upper bound.
  - UNION: both subqueries are ~~SFW queries~~. The condition that their number of columns (and datatypes) must match guarantees that nothing can be undefined/unrestricted.
- Note that the outer join ~~could be a risk~~, similar to the ∨ in the calculus. The outer join specifies to fill the "open" positions with NULL values.

Algebra: structural induction over the set of top operators of a tree of height n and subtrees of size $n-1$.

- π, σ, $\bowtie$, ρ, $\times$: always applied to a safe subtree by induction hypothesis.
- $\setminus$: The left subtree acts as an upper bound.
- *union*: the formats of both subtrees must fit, so there is no "open" position as in the $\vee$ in the calculus. Both subtrees are domain-independent by induction hypothesis.
- Note: the outer join is covered as a derived operator consisting of $\cup$ and $\times$. Furthermore, when the outer join's definition is considered alone, the "completion" with null values for the "open" positions guarantees domain-independence.

Exercise 5 (Division: Äquivalenz von Algebra und Kalkül) For the relational algebra, the division operator has been introduced as a derived operator (cf. lecture "Databases"). Consider the relation schemata $r(A,B)$ and $s(B)$.

$$r \div s = \{\mu \in Tup(A) \mid \{\mu\} \times s \subseteq r\} = \pi[A](r) \setminus \pi[A]((\pi[A](r) \times s) \setminus r).$$

Derive a query in the relational calculus from the left-hand side, and prove the equivalence with the right-hand side.

The left-hand side expression: the specification states that X must be an A-component in R, and for all values Y in S, the combination of X and Y must be in R:

$$F(X) = \exists Y : r(X,Y) \wedge \forall Z : (s(Z) \rightarrow r(X,Z)) \ .$$

The query is (syntactically) not in SRNF, and is equivalently rewritten into

$$F'(X) = \exists Y : r(X,Y) \wedge \neg\exists Z : (s(Z) \wedge \neg r(X,Z)) \ ,$$

which is in SRNF (thus, domain-independent), but not in RANF.

Transformation to RANF ("push-into-not-exists"):

$$F''(X) = \exists Y : r(X,Y) \wedge \neg\exists Z : (\exists Y_2 : r(X,Y_2) \wedge s(Z) \wedge \neg r(X,Z))$$

Derivation of the algebra expression:

query Q	Algebra
$Q(X,Z) = (\exists Y_2 : r(X,Y_2)) \wedge s(Z) \wedge \neg r(X,Z)$	$(\pi[A](r) \times s) \setminus r$
$Q(X) = \exists Z : (\exists Y_2 : r(X,Y_2)) \wedge s(Z) \wedge \neg r(X,Z)$	$\pi[A]((\pi[A](r) \times s) \setminus r)$ (the expression has the format A)
$Q(X) = \exists Y : r(X,Y)$	$\pi[A](r)$ (has again the format A)
$F''(X)$ as above	$\pi[A](r) \setminus \pi[A]((\pi[A](r) \times s) \setminus r)$

... is exactly the right-hand side.

Exercise 6 (Kalkül: Gruppierung und Aggregation) Define a syntactical extension for the relational calulus, that allows to use aggregate functions similar to the `GROUP BY` functionality of SQL.

For this, consider only aggregate functions as simple applications over single attributes like max(population), but not more complex expressions like max(population/area).

- What is the result of an aggregate function, and how can it be used in the calculus?
- Which inputs does an aggregate function have?
- how can this input be obtained from the database?

Give a calculus expression for the query "For each country give the name and the total number of people living in its cities".

The result is a number. It can be bound to a variable or it can be used in a comparison. Thus, the aggregate function is to be considered as a term (whose evaluation yields a value).

The immediate input to an aggregate function is a set/list of values, over which the aggregate is computed (sum, count, ...).

This list can be obtained as results of a (sub)formula (similar to a correlated subquery) with a free variable.

The results are grouped by zero, one or more free variables of the subquery. Usually, these also occur in other literals outside the aggregation.

$$X = \textit{agg-op}\{\textit{var}\,[\textit{group-by-vars}]; \textit{subq-fml}\}$$

where in *subq-fml* the *group-by-vars* and *var* have free occurrences. E.g.,

$$\begin{aligned} &F(CN, SumCityPop) = \\ &\quad \exists C, A, P, Cap, CapProv : country(CN, C, A, P, Cap, CapProv) \wedge \\ &\quad SumCityPop = \mathsf{sum}\{CityPop\,[C]; \\ &\qquad\qquad \exists CtyN, CtyProv, L1, L2 : city(CtyN, CtyProv, C, CityPop, L1, L2)\} \end{aligned}$$

groups by C, computes the sum over $CityPop$ and binds the value to $SumCityPop$.

Comments:

- a similar syntax is used in F-Logic;
- the usage in XSB is similar, but the user has to program it more explicitly:
  - the list is created by the Prolog predicate "bagof";
  - the aggregation operation over the list must be programmed in the common Prolog style for handling a list.

Exercise 7 (Kalkül→Algebra) Consider the relation schemata $R(A, B), S(B, C)$ und $T(A, B, C)$.

a) Give an equivalent algebra expression for the following safe relational calculus expression:

$$F_1(X, Y) = T(Y, a, Y) \wedge (R(a, X) \vee S(X, c)) \wedge \neg T(a, X, Y)$$

Proceed as shown in the lecture for the equivalence proof.

b) Simplify the expression.

c) Extend the expression from (a) to

$$F_2(Y) = \exists X : (F_1(X, Y) \wedge X > 3)$$

---

a) First, consider each of the three conjuncts (denoted as F_2, F_1 and F_3) separately:
The literal $F_1(Y) = T(Y, a, Y)$ corresponds to the subexpression

$$E_1 = \rho[A \to Y](\pi[A](\sigma[(A = C) \wedge (B = a)](T))) \ .$$

The subformula $F_2(X) = R(a, X) \vee S(X, c)$ corresponds to the expression

$$E_2 = \rho[B \to X](\pi[B](\sigma[A = a](R))) \cup \rho[B \to X](\pi[B](\sigma[C = c](S))) \ .$$

The negated literal $F_3(X, Y) = \neg T(a, X, Y)$ corresponds to the expression

$$E_4 = \rho[B \to X, C \to Y](\pi[B, C](\sigma[A = a](T)))$$

The expression corresponding to $F_3(X, Y)$ is then

all ADOM pairs for (X,Y)

$$E_3 = \rho[\$1 \to X, \$2 \to Y](ADOM^2) - \rho[B \to X, C \to Y](\pi[B, C](\sigma[A = a](T)))$$

where $ADOM^2 = ((\pi[A](R) \cup \pi[B](R) \cup \pi[B](S) \cup \pi[C](S) \cup \pi[A](T) \cup \pi[B](T) \cup \pi[C](T)) \times (\pi[A](R) \cup \pi[B](R) \cup \pi[B](S) \cup \pi[C](S) \cup \pi[A](T) \cup \pi[B](T) \cup \pi[C](T)))$ contains all 2-tuples of values from the database.

Thus, $E = E_1 \bowtie E_2 \bowtie (ADOM^2 - E_4)$ is the complete algebra expression.

b) Simplify: E_1 and E_2 have no variable/column in common, thus it can be simplified as $(E_1 \times E_2) \bowtie (ADOM^2 - E_4)$. Both subterms bind X and Y, thus, $ADOM^2$ can be omitted, obtaining $E' = (E_1 \times E_2) - E_4$.

c) The additional comparison is expressed as a selection, and the $\exists X$ quantification is expressed as a projection to Y:

$$\pi[Y](\sigma[X > 3](E'))$$

---

or: do the x-selection already in E_1