Abstract

To do : A lot :) (SAT/SMT SOLVER ???) to evaluate what we loose ?

1 Preliminaries

Definition 1. *We denote the Set of well formed select query without agregation, full join and null test by* $[\![SQL]\!]$
We denote the Set of well formed select query without agregation and full join by $[\![SQL]\!]_\perp$

Definition 2. *Let's a Select query* $Q \in [\![SQL]\!]$ *a tuple* (Σ, R, H, P) *such that*

$\Sigma \subseteq \{r_i.a_i | \exists r_i \in R \wedge \exists a_i \in r_i\}$
$P \subseteq \{p_i = r_j.a_j | r_j \notin R\}$ *a set of external parameter.*
R *a set of relation.*
H *belongs to the following grammar*

$$\begin{aligned} H ::= & \ r_i.a_i = c_i \mid r_i.a_i \neq c_j \mid r_i.a_i = r_j.a_j \mid r_i.a_i \neq r_j.a_j \mid r_i.a_i = p_i \mid \\ & exists(Q) \mid notexists(Q) \mid in(r_i.a_i, Q) \mid notin(r_i.a_i, Q) \mid \\ & H \wedge H \mid H \vee H \end{aligned}$$

Definition 3. *Let's a Select query* $Q \in [\![SQL]\!]_\perp$ *a tuple* (Σ, R, H, P) *such that*

$\Sigma \subseteq \{r_i.a_i | \exists r_i \in R \wedge \exists a_i \in r_i\}$
$P \subseteq \{p_i = r_j.a_j | r_j \notin R\}$ *a set of external parameter.*
R *a set of relation.*
$H_\perp$ *belongs to the following grammar*

$$
\begin{aligned}
H_\perp ::= {} & r_i.a_i = c_i \mid r_i.a_i \neq c_j \mid r_i.a_i = r_j.a_j \mid r_i.a_i \neq r_j.a_j \mid \\
& r_i.a_i = p_i \mid null(r_i.a_i) \mid const(r_i.a_i) \\
& exists(Q_\perp) \mid notexists(Q_\perp) \mid in(r_i.a_i, Q_\perp) \mid notin(r_i.a_i, Q_\perp) \mid \\
& H_\perp \wedge H_\perp \mid H_\perp \vee H_\perp
\end{aligned}
$$

We denote $(\Sigma, R, H, P)[x]$ the query $(\Sigma, R, H, P \cup x)$
We denote $(\Sigma, R, H, P)_*$ the query $(*, R, H, P)$

Proposition 1.

$$[\![SQL]\!] \subset [\![SQL]\!]_\perp$$

Definition 4. *We call a bag B a function $D \to \mathbb{N}$ such that $B(x)$ represents the multiplicity of x in the bag B.*

Definition 5.

$$
\begin{aligned}
\forall x, \emptyset(x) &= 0 \\
\forall x, (B_1 \cap B_2)(x) &= min(B_1(x), B_2(x)) \\
\forall x, (B_1 \cup B_2)(x) &= max(B_1(x), B_2(x)) \\
\forall x, (B_1 \uplus B_2)(x) &= B_1(x) + B_2(x) \\
\forall x, (B_1 \setminus B_2)(x) &= max(0, B_1(x) - B_2(x)) \\
\forall x, [\![a]\!](x) &= \begin{cases} 1 \ \textit{if } x = a \\ 0 \ \textit{otherwise} \end{cases} \\
\forall x, [\![a^n]\!](x) &= \begin{cases} n \ \textit{if } x = a \\ 0 \ \textit{otherwise} \end{cases} \\
\forall x, [\![y^n | P(y,n)]\!](x) &= max(\{i | P(x,i)\}) \\
x \in B &\iff B(x) \geq 1 \\
x \in^n B &\iff B(x) \geq n \\
x \notin B &\iff B(x) = 0 \\
B_1 = B_2 &\iff \forall x, B_1(x) = B_2(x) \\
B_1 \subseteq B_2 &\iff \forall x, B_1(x) \leq B_2(x) \\
\{B\} &= \{x | B(x) \geq 1\}
\end{aligned}
$$

2 Semantics

Definition 6. *Let Q be a query (Σ, R, H, P) then $R^\times$ denotethe bag coming from the carthesian product or R.*

$$R^\times = \prod_{r \in R} r$$

Definition 7.

$$\sigma_\Sigma(x) = (x[r_i.a_i] | r_i.a_i \in \Sigma)$$
$$\sigma_*(x) = x$$

Definition 8.

$$\sigma_\Sigma(B) = [\![y^n | n = \sum_{x \in \{z | z \in \{B\} \wedge \sigma_\Sigma(z) = y\}} B(x)]\!]$$

Definition 9.

$$\begin{aligned}
Eval_{SQL}((\Sigma, R, H_1 \wedge H_2, P), D) &= \sigma_\Sigma(Eval_{SQL}((*, R, H_1, P), D) \cap Eval_{SQL}((*, R, H_2, P), D)) \\
Eval_{SQL}((\Sigma, R, H_1 \vee H_2, P), D) &= \sigma_\Sigma(Eval_{SQL}((*, R, H_1, P), D) \cup Eval_{SQL}((*, R, H_2, P), D)) \\
Eval_{SQL}((\Sigma, R, null(r_i.a_i), P), D) &= \sigma_\Sigma([\![x^n | R^\times(x) = n \wedge x[r_i.a_i] = \bot]\!]) \\
Eval_{SQL}((\Sigma, R, const(r_i.a_i), P), D) &= \sigma_\Sigma([\![x^n | R^\times(x) = n \wedge x[r_i.a_i] \neq \bot]\!]) \\
Eval_{SQL}((\Sigma, R, r_i.a_i = c_i, P), D) &= \sigma_\Sigma([\![x^n | R^\times(x) = n \wedge x[r_i.a_i] = c_i]\!]) \\
Eval_{SQL}((\Sigma, R, r_i.a_i = r_j.a_j, P), D) &= \sigma_\Sigma([\![x^n | R^\times(x) = n \wedge x[r_i.a_i] = x[r_j.a_j] \wedge x[r_i.a_i] \neq \bot \wedge x[r_j.a_j] \neq \bot]\!]) \\
Eval_{SQL}((\Sigma, R, r_i.a_i = p_i, P), D) &= \sigma_\Sigma([\![x^n | R^\times(x) = n \wedge x[r_i.a_i] = P[p_i] \wedge x[r_i.a_i] \neq \bot \wedge P[p_i] \neq \bot]\!]) \\
Eval_{SQL}((\Sigma, R, r_i.a_i \neq c_i, P), D) &= \sigma_\Sigma([\![x^n | R^\times(x) = n \wedge x[r_i.a_i] \neq c_i \wedge x[r_i.a_i] \neq \bot]\!]) \\
Eval_{SQL}((\Sigma, R, r_i.a_i \neq r_j.a_j, P), D) &= \sigma_\Sigma([\![x^n | R^\times(x) = n \wedge x[r_i.a_i] \neq x[r_j.a_j] \wedge x[r_i.a_i] \neq \bot \wedge x[r_j.a_j] \neq \bot]\!]) \\
Eval_{SQL}((\Sigma, R, r_i.a_i \neq p_i, P), D) &= \sigma_\Sigma([\![x^n | R^\times(x) = n \wedge x[r_i.a_i] \neq P[p_i] \wedge x[r_i.a_i] \neq \bot \wedge P[p_i] \neq \bot]\!]) \\
Eval_{SQL}((\Sigma, R, exists(Q), P), D) &= \sigma_\Sigma([\![x^n | R^\times(x) = n \wedge Eval_{SQL}(Q[x], D) \neq \emptyset]\!]) \\
Eval_{SQL}((\Sigma, R, notexists(Q), P), D) &= \sigma_\Sigma([\![x^n | R^\times(x) = n \wedge Eval_{SQL}(Q[x], D) = \emptyset]\!]) \\
Eval_{SQL}((\Sigma, R, in(\Sigma_1, Q), P), D) &= \sigma_\Sigma([\![x^n | R^\times(x) = n \\
&\qquad \wedge \forall y \in Eval_{SQL}(Q, D), \forall i \in [\![1; |\Sigma_1|]\!], \sigma_{\Sigma_1}(x)[i] = y[i] \\
&\qquad\quad \wedge \sigma_{\Sigma_1}(x)[i] \neq \bot \wedge y[i] \neq \bot]\!]) \\
Eval_{SQL}((\Sigma, R, notin(\Sigma_1, Q), P), D) &= \sigma_\Sigma([\![x^n | R(x) = n \\
&\qquad \wedge \forall y \in Eval_{SQL}(Q, D), \forall i \in [\![1; |\Sigma_1|]\!], \sigma_{\Sigma_1}(x)[i] \neq y[i] \\
&\qquad\quad \vee \sigma_{\Sigma_1}(x)[i] = \bot \vee y[i] = \bot]\!])
\end{aligned}$$

Definition 10. *With marked nulls we have:*

$$cert_\bot(Q, D) = [\![x^n | \forall h, h(x) \in^n Eval_{SQL}(Q, h(D))]\!]$$

Proposition 2. *With SQL nulls we have:*

$$\forall Q \in [\![SQL]\!], cert_\bot((\Sigma, R, H, P), D) = \sigma_\Sigma([\![x^n | R(x) = n \wedge \forall h, h(x) \in \{Eval_{SQL}((*, R, H, h(P)), h(D))\}]\!])$$

Proof. Dont even know where to begin. □

Definition 11.

$$\forall Q \in [\![SQL]\!], posi_\bot((\Sigma, R, H, P), D) = \sigma_\Sigma([\![x^n | R(x) = n \wedge \exists h, h(x) \in \{Eval_{SQL}((*, R, H, h(P)), h(D))\}]\!])$$

3 Translation

$$(\Sigma, R, H, P)^+ \to (\Sigma, R, H^*, P)$$
$$(\Sigma, R, H, P)^? \to (\Sigma, R, H^{**}, P)$$

$$\begin{aligned}
(H_1 \wedge H_2)^* &\to H_1^* \wedge H_2^* \\
(H_1 \vee H_2)^* &\to H_1^* \vee H_2^* \\
(r_i.a_i = c_i)^* &\to r_i.a_i = c_i \\
(r_i.a_i \neq c_i)^* &\to r_i.a_i \neq c_i \\
(r_i.a_i = r_j.a_j)^* &\to r_i.a_i = r_j.a_j \\
(r_i.a_i \neq r_j.a_j)^* &\to r_i.a_i \neq r_j.a_j \\
null(r_i.a_i)^* &\to null(r_i.a_i) \\
const(r_i.a_i)^* &\to const(r_i.a_i) \\
exists(Q)^* &\to exists(Q^+) \\
notexists(Q)^* &\to notexists(Q^?) \\
in(\Sigma_1, Q)^* &\to in(\Sigma, Q^+) \\
notin(\Sigma_1, (\Sigma, R, H, P))^* &\to notexists(\Sigma, R, H \wedge (\Sigma = \Sigma_1), P \cup \Sigma_1)^*
\end{aligned}$$

$$\begin{aligned}
(H_1 \wedge H_2)^{**} &\to H_1^{**} \wedge H_2^{**} \\
(H_1 \vee H_2)^{**} &\to H_1^{**} \vee H_2^{**} \\
(r_i.a_i = c_i)^{**} &\to r_i.a_i = c_i \vee null(r_i.a_i) \\
(r_i.a_i \neq c_i)^{**} &\to r_i.a_i \neq c_i \vee null(r_i.a_i) \\
(r_i.a_i = r_j.a_j)^{**} &\to r_i.a_i = r_j.a_j \vee null(r_i.a_i) \vee null(r_j.a_j) \\
(r_i.a_i \neq r_j.a_j)^{**} &\to r_i.a_i \neq r_j.a_j \vee null(r_i.a_i) \vee null(r_j.a_j) \\
null(r_i.a_i)^{**} &\to null(r_i.a_i) \\
const(r_i.a_i)^{**} &\to const(r_i.a_i) \\
exists(Q)^{**} &\to exists(Q^?) \\
notexists(Q)^{**} &\to notexists(Q^+) \\
in(r_i.a_i, Q)^{**} &\to in(r_i.a_i, Q^?) \\
notin(\Sigma_1, (\Sigma, R, H, P))^{**} &\to notexists(\Sigma, R, H \wedge (\Sigma = \Sigma_1), P \cup \Sigma_1)^{**}
\end{aligned}$$

Proposition 3.

$$\forall Q \in [\![SQL]\!], Eval_{SQL}(Q^+, D) \subseteq cert_\perp(Q, D)$$

Proposition 4.

$$\forall Q \in [\![SQL]\!], posi_\perp(Q, D) \subseteq Eval_{SQL}(Q^?, D)$$

Proof. Assume (5).
By induction :

$$
\begin{aligned}
&Q = (\Sigma, R, H_1 \wedge H_2, P)\\
&x \in^n Eval_{SQL}(Q^+, D) \Rightarrow \sum_{z \in \{y | y \in Eval_{SQL}(Q_*^+, D) \wedge \sigma_\Sigma(y) = x\}} Eval_{SQL}(Q_*^+, D)(z) \geq n\\
&\qquad\qquad Moreover\\
&Eval_{SQL}(Q_*^+, D)(z) = k \Rightarrow Eval_{SQL}((*, R, H_1^* \wedge H_2^*, P), D)(z) = k\\
&\qquad\qquad \Rightarrow (Eval_{SQL}((*, R, H_1^*, P), D) \cap Eval_{SQL}((*, R, H_2^*, P), D))(z) = k\\
&\qquad\qquad \Rightarrow (Eval_{SQL}((*, R, H_1, P)^+, D) \cap Eval_{SQL}((*, R, H_2, P)^+, D))(z) = k\\
&\qquad\qquad \Rightarrow (Eval_{SQL}((*, R, H_1, P)^+, D))(x) \geq k \wedge (Eval_{SQL}((*, R, H_2, P)^+, D))(z) \geq k\\
&\qquad\qquad \Rightarrow (cert_\perp((*, R, H_1, P), D))(x) \geq k \wedge (cert_\perp((*, R, H_2, P), D))(z) \geq k\\
&\qquad\qquad \Rightarrow (cert_\perp((*, R, H_1, P), D) \cap cert_\perp((*, R, H_2, P), D))(z) \geq k\\
&\qquad\qquad \Rightarrow (cert_\perp((*, R, H_1 \wedge H_2, P), D))(z) \geq k\\
&\qquad\qquad \Rightarrow (cert_\perp(Q_*, D))(z) \geq k\\
&\qquad\qquad Then\\
&x \in^n Eval_{SQL}(Q^+, D) \Rightarrow \sum_{z \in \{y | y \in Eval_{SQL}(Q_*^+, D) \wedge \sigma_\Sigma(y) = x\}} cert_\perp(Q_*, D)(z) \geq n\\
&\qquad\qquad \Rightarrow x \in^n cert_\perp(Q, D)
\end{aligned}
$$

$$
\begin{aligned}
&Q = (\Sigma, R, H_1 \vee H_2, P)\\
&x \in^n Eval_{SQL}(Q^+, D) \Rightarrow \sum_{z \in \{y | y \in Eval_{SQL}(Q_*^+, D) \wedge \sigma_\Sigma(y) = x\}} Eval_{SQL}(Q_*^+, D)(z) \geq n\\
&\qquad\qquad Moreover\\
&Eval_{SQL}(Q_*^+, D)(z) = k \Rightarrow Eval_{SQL}((*, R, H_1^* \vee H_2^*, P), D)(z) = k\\
&\qquad\qquad \Rightarrow (Eval_{SQL}((*, R, H_1^*, P), D) \cup Eval_{SQL}((*, R, H_2^*, P), D))(z) = k\\
&\qquad\qquad \Rightarrow (Eval_{SQL}((*, R, H_1, P)^+, D) \cup Eval_{SQL}((*, R, H_2, P)^+, D))(z) = k\\
&\qquad\qquad \Rightarrow (Eval_{SQL}((*, R, H_1, P)^+, D))(x) \geq k \vee (Eval_{SQL}((*, R, H_2, P)^+, D))(z) \geq k\\
&\qquad\qquad \Rightarrow (cert_\perp((*, R, H_1, P), D))(x) \geq k \vee (cert_\perp((*, R, H_2, P), D))(z) \geq k\\
&\qquad\qquad \Rightarrow (cert_\perp((*, R, H_1, P), D) \cup cert_\perp((*, R, H_2, P), D))(z) \geq k\\
&\qquad\qquad \Rightarrow (cert_\perp((*, R, H_1 \vee H_2, P), D))(z) \geq k\\
&\qquad\qquad \Rightarrow (cert_\perp(Q_*, D))(z) \geq k\\
&\qquad\qquad Then\\
&x \in^n Eval_{SQL}(Q^+, D) \Rightarrow \sum_{z \in \{y | y \in Eval_{SQL}(Q_*^+, D) \wedge \sigma_\Sigma(y) = x\}} cert_\perp(Q_*, D)(z) \geq n\\
&\qquad\qquad \Rightarrow x \in^n cert_\perp(Q, D)
\end{aligned}
$$

$$
\begin{aligned}
&Q = (\Sigma, R, notexists(Q'), P)\\
&\quad x \in^n Eval_{SQL}(Q^+, D) \Rightarrow \sum_{z \in \{y | y \in Eval_{SQL}(Q_*^+, D) \wedge \sigma_\Sigma(y) = x\}} Eval_{SQL}(Q_*^+, D)(z) \geq n\\
&\qquad\qquad Moreover\\
&\quad Eval_{SQL}(Q_*^+, D)(z) = k \Rightarrow Eval_{SQL}((*, R, notexists(Q')^*, P), D)(z) = k\\
&\qquad\qquad \Rightarrow Eval_{SQL}((*, R, notexists(Q'^?), P), D)(z) = k\\
&\qquad\qquad \Rightarrow R(z) = k \wedge Eval_{SQL}(Q'[z]^?, D) = \emptyset\\
&\qquad\qquad \Rightarrow R(z) = k \wedge posi_\perp(Q'[z], D) = \emptyset\\
&\qquad\qquad \Rightarrow R(z) = k \wedge \forall h, \forall w, h(w) \notin Eval_{SQL}(Q'[h(z)], h(D))\\
&\qquad\qquad \Rightarrow R(z) = k \wedge \forall h, Eval_{SQL}(Q'[h(z)], h(D)) = \emptyset\\
&\qquad\qquad \Rightarrow R(z) = k \wedge \forall h, h(z) \in Eval_{SQL}((*, R, notexists(Q'), h(P)), h(D))\\
&\qquad\qquad \Rightarrow cert_\perp(Q_*, D)(z) = k\\
&\qquad\qquad Then\\
&\quad x \in^n Eval_{SQL}(Q^+, D) \Rightarrow \sum_{z \in \{y | y \in Eval_{SQL}(Q_*^+, D) \wedge \sigma_\Sigma(y) = x\}} cert_\perp(Q_*, D)(z) \geq n\\
&\qquad\qquad \Rightarrow x \in^n cert_\perp(Q, D)
\end{aligned}
$$

$$
\begin{aligned}
&Q = (\Sigma, R, r_i.a_i \neq p_i, P)\\
&\quad x \in^n Eval_{SQL}(Q^+, D) \Rightarrow \sum_{z \in \{y \mid y \in Eval_{SQL}(Q_*^+, D) \wedge \sigma_\Sigma(y) = x\}} Eval_{SQL}(Q_*^+, D)(z) \geq n\\
&\qquad\qquad Moreover\\
&Eval_{SQL}(Q_*^+, D)(z) = k \Rightarrow Eval_{SQL}((*, R, (r_i.a_i \neq p_i)^*, P), D)(z) = k\\
&\qquad\qquad \Rightarrow Eval_{SQL}((*, R, r_i.a_i \neq p_i, P), D)(z) = k\\
&\qquad\qquad \Rightarrow R(x) = k \wedge x[r_i.a_i] \neq P[p_i] \wedge x[r_i.a_i] \neq \bot \wedge P[p_i] \neq \bot\\
&\qquad\qquad Moreover\\
&x[r_i.a_i] \neq \bot \wedge P[p_i] \neq \bot \Rightarrow \forall h, h(x)[r_i.a_i] = x[r_i.a_i] \wedge h(P)[p_i] = P[p_i]\\
&\qquad\qquad Then\\
&Eval_{SQL}(Q_*^+, D)(z) = k \Rightarrow R(x) = k \wedge \forall h, h(x)[r_i.a_i] \neq h(P)[p_i] \wedge h(x)[r_i.a_i] \neq \bot \wedge h(P)[p_i] \neq \bot\\
&\qquad\qquad \Rightarrow cert_\bot(Q_*, D)(z) = k\\
&\qquad\qquad Then\\
&\quad x \in^n Eval_{SQL}(Q^+, D) \Rightarrow \sum_{z \in \{y \mid y \in Eval_{SQL}(Q_*^+, D) \wedge \sigma_\Sigma(y) = x\}} cert_\bot(Q_*, D)(z) \geq n\\
&\qquad\qquad \Rightarrow x \in^n cert_\bot(Q, D)
\end{aligned}
$$

□

Proof. Assume (4).

By induction ...

$$
\begin{aligned}
&Q = (\Sigma, R, H_1 \wedge H_2, P) \\
&\qquad x \in^n posi_\perp(Q, D) \Rightarrow \sum_{z \in \{y | y \in posi_\perp(Q_*, D) \wedge \sigma_\Sigma(y) = x\}} posi_\perp(Q_*, D)(z) \geq n \\
&\qquad\qquad Moreover \\
&posi_\perp(Q_*, D)(z) = k \Rightarrow R(z) = k \wedge \exists h, h(z) \in Eval_{SQL}((*, R, H_1 \wedge H_2, h(P)), h(D)) \\
&\qquad\qquad \Rightarrow R(z) = k \wedge \exists h, h(z) \in Eval_{SQL}((*, R, H_1, h(P)), h(D)) \wedge h(z) \in Eval_{SQL}((*, R, H_1 \\
&\qquad\qquad \Rightarrow posi_\perp((*, R, H_1, P), D)(z) \geq k \wedge posi_\perp((*, R, H_2, P), D)(z) \geq k \\
&\qquad\qquad \Rightarrow Eval_{SQL}((*, R, H_1, P)^?, D)(z) \geq k \wedge Eval_{SQL}((*, R, H_2, P)^?, D)(z) \geq k \\
&\qquad\qquad \Rightarrow Eval_{SQL}((*, R, H_1^{**}, P), D)(z) \geq k \wedge Eval_{SQL}((*, R, H_2^{**}, P), D)(z) \geq k \\
&\qquad\qquad \Rightarrow Eval_{SQL}((*, R, H_1^{**} \wedge H_2^{**}, P), D)(z) \geq k \\
&\qquad\qquad \Rightarrow Eval_{SQL}((*, R, H_1 \wedge H_2, P)^?, D)(z) \geq k \\
&\qquad\qquad \Rightarrow Eval_{SQL}(Q_*^?, D)(z) \geq k \\
&\qquad\qquad Then \\
&\qquad x \in^n posi_\perp(Q, D) \Rightarrow \sum_{z \in \{y | y \in posi_\perp(Q_*, D) \wedge \sigma_\Sigma(y) = x\}} Eval_{SQL}(Q_*^?, D)(z) \geq n \\
&\qquad\qquad \Rightarrow x \in^n Eval_{SQL}(Q^?, D)
\end{aligned}
$$

$Q = (\Sigma, R, notexists(Q'), P)$

$$x \in^n posi_\perp(Q, D) \Rightarrow \sum_{z \in \{y | y \in posi_\perp(Q_*, D) \wedge \sigma_\Sigma(y) = x\}} posi_\perp(Q_*, D)(z) \geq n$$

$Moreover$

$$\begin{aligned}
posi_\perp(Q_*, D)(z) = k &\Rightarrow R(z) = k \wedge \exists h, h(z) \in Eval_{SQL}((*, R, notexists(Q'), h(P)), h(D)) \\
&\Rightarrow R(z) = k \wedge \exists h, Eval_{SQL}(Q'[h(z)], h(D)) = \emptyset \\
&\Rightarrow R(z) = k \wedge [\![w^n | \forall g, g(w) \in^n Eval_{SQL}((Q'[g(z)]), g(D))]\!] = \emptyset \\
&\Rightarrow R(z) = k \wedge cert_\perp(Q'[z], D) = \emptyset \\
&\Rightarrow R(z) = k \wedge Eval(Q'[z]^+, D) = \emptyset \\
&\Rightarrow R(z) = k \wedge z \in Eval_{SQL}((*, R, notexists(Q'^+), P), D) \\
&\Rightarrow R(z) = k \wedge z \in Eval_{SQL}((*, R, notexists(Q')^{**}, P), D) \\
&\Rightarrow Eval_{SQL}((*, R, notexists(Q'), P)^?, D)(z) = k
\end{aligned}$$

$Then$

$$\begin{aligned}
x \in^n posi_\perp(Q, D) &\Rightarrow \sum_{z \in \{y | y \in posi_\perp(Q_*, D) \wedge \sigma_\Sigma(y) = x\}} Eval_{SQL}(Q_*^?, D)(z) \geq n \\
&\Rightarrow x \in^n Eval_{SQL}(Q^?, D)
\end{aligned}$$

$Q = (\Sigma, R, r_i.a_i = r_j.a_j, P)$

$$x \in^n posi_\perp(Q, D) \Rightarrow \sum_{z \in \{y | y \in posi_\perp(Q_*, D) \wedge \sigma_\Sigma(y) = x\}} posi_\perp(Q_*, D)(z) \geq n$$

$$Moreover$$

$$posi_\perp(Q_*, D)(z) = k \Rightarrow R(z) = k \wedge \exists h, h(z) \in Eval_{SQL}((*, R, r_i.a_i = r_j.a_j, h(P)), h(D))$$
$$\Rightarrow R(z) = k \wedge \exists h, h(z)[r_i.a_i] = h(z)[r_j.a_j] \wedge h(z)[r_i.a_i] \neq \perp \wedge h(z)[r_j.a_j] \neq \perp$$

$$Moreover$$

$$\exists h, h(z)[r_i.a_i] = h(z)[r_j.a_j] \Rightarrow z[r_i.a_i] = z[r_j.a_j] \vee z[r_i.a_i] = \perp \vee z[r_j.a_j] = \perp$$
$$\forall g, g(z)[r_i.a_i] \neq \perp \Rightarrow \text{ TRUE}$$

$$Then$$

$$posi_\perp(Q_*, D)(z) = k \Rightarrow R(z) = k \wedge z[r_i.a_i] = z[r_j.a_j] \vee z[r_i.a_i] = \perp \vee z[r_j.a_j] = \perp$$
$$\Rightarrow R(z) = k \wedge z \in Eval_{SQL}((*, R, (r_i.a_i = r_j.a_j \vee null(r_i.a_i) \vee null(r_j.a_j)), P), D)$$
$$\Rightarrow R(z) = k \wedge z \in Eval_{SQL}((*, R, (r_i.a_i = r_j.a_j)^{**}, P), D)$$
$$\Rightarrow Eval_{SQL}((*, R, r_i.a_i = r_j.a_j, P)^?, D)(z) = k$$

$$Then$$

$$x \in^n posi_\perp(Q, D) \Rightarrow \sum_{z \in \{y | y \in posi_\perp(Q_*, D) \wedge \sigma_\Sigma(y) = x\}} Eval_{SQL}(Q_*^?, D)(z) \geq n$$
$$\Rightarrow x \in^n Eval_{SQL}(Q^?, D)$$

□

4 Removing useless null check

The translation $Q \rightarrow Q^+$ has an heavy cost has explained in the paper ..., in order to remove some useless null test we offer an optimization.

Definition 12. *For each Query and nested Query we maintain a set of attributes that have to be not null in order for the query to be true resp. false we denote this set* $\perp_Q^T$ *resp.*$\perp_Q^F$.

Definition 13.

$$
\begin{aligned}
\bot^T_{(\Sigma,R,H_1\wedge H_2,P)} &= \bot^T_{(\Sigma,R,H_1,P)} \cup \bot^T_{(\Sigma,R,H_2,P)} \\
\bot^T_{(\Sigma,R,H_1\vee H_2,P)} &= \bot^T_{(\Sigma,R,H_1,P)} \cap \bot^T_{(\Sigma,R,H_2,P)} \\
\bot^T_{(\Sigma,R,r_i.a_i=c_i,P)} &= \{r_i.a_i\} \\
\bot^T_{(\Sigma,R,r_i.a_i\neq c_i,P)} &= \{r_i.a_i\} \\
\bot^T_{(\Sigma,R,r_i.a_i=r_j.a_j,P)} &= \{r_i.a_i, r_j.a_j\} \\
\bot^T_{(\Sigma,R,r_i.a_i\neq r_j.a_j,P)} &= \{r_i.a_i, r_j.a_j\} \\
\bot^T_{(\Sigma,R,r_i.a_i=p_i,P)} &= \{r_i.a_i, p_i\} \\
\bot^T_{(\Sigma,R,r_i.a_i\neq p_i,P)} &= \{r_i.a_i, p_i\} \\
\bot^T_{(\Sigma,R,null(r_i.a_i),P)} &= \emptyset \\
\bot^T_{(\Sigma,R,const(r_i.a_i),P)} &= \{r_i.a_i\} \\
\bot^T_{(\Sigma,R,exists(Q),P)} &= \bot^T_{Q[?]} \\
\bot^T_{(\Sigma,R,notexists(Q),P)} &= \bot^F_{Q[?]}
\end{aligned}
$$

$$
\begin{aligned}
\bot^F_{(\Sigma,R,H_1\wedge H_2,P)} &= \bot^F_{(\Sigma,R,H_1,P)} \cap \bot^F_{(\Sigma,R,H_2,P)} \\
\bot^F_{(\Sigma,R,H_1\vee H_2,P)} &= \bot^F_{(\Sigma,R,H_1,P)} \cup \bot^F_{(\Sigma,R,H_2,P)} \\
\bot^F_{(\Sigma,R,r_i.a_i=c_i,P)} &= \emptyset \\
\bot^F_{(\Sigma,R,r_i.a_i\neq c_i,P)} &= \emptyset \\
\bot^F_{(\Sigma,R,r_i.a_i=r_j.a_j,P)} &= \emptyset \\
\bot^F_{(\Sigma,R,r_i.a_i\neq r_j.a_j,P)} &= \emptyset \\
\bot^F_{(\Sigma,R,r_i.a_i=p_i,P)} &= \emptyset \\
\bot^F_{(\Sigma,R,r_i.a_i\neq p_i,P)} &= \emptyset \\
\bot^F_{(\Sigma,R,null(r_i.a_i),P)} &= \emptyset \\
\bot^F_{(\Sigma,R,const(r_i.a_i),P)} &= \emptyset \\
\bot^F_{(\Sigma,R,exists(Q),P)} &= \bot^F_{Q[?]} \\
\bot^F_{(\Sigma,R,notexists(Q),P)} &= \bot^T_{Q[?]}
\end{aligned}
$$

Proposition 5.

$$x \in Eval_{SQL}(Q_*, D) \Rightarrow \forall r_i.a_i \in \bot^T_Q, x[r_i.a_i] \neq \bot$$

$$x \notin Eval_{SQL}(Q_*, D) \Rightarrow \forall r_i.a_i \in \bot^F_Q, x[r_i.a_i] \neq \bot$$

Proof. □

Definition 14.

$$
\begin{aligned}
nested^+(H_1 \wedge H_2) &= \{H_1 \wedge H_2\} \cup nested^+(H_1) \cup nested^+(H_2) \\
nested^+(H_1 \vee H_2) &= \{H_1 \vee H_2\} \cup nested^+(H_1) \cup nested^+(H_2) \\
nested^+(r_i.a_i = c_i) &= \{r_i.a_i = c_i\} \\
nested^+(r_i.a_i \neq c_i) &= \{r_i.a_i \neq c_i\} \\
nested^+(r_i.a_i = r_j.a_j) &= \{r_i.a_i = r_j.a_j\} \\
nested^+(null(r_i.a_i)) &= \{null(r_i.a_i)\} \\
nested^+(const(r_i.a_i)) &= \{const(r_i.a_i)\} \\
nested^+(exists(Q)) &= \{exists(Q)\} \cup nested^+(Q) \\
nested^+(notexists(Q)) &= \{notexists(Q)\} \cup nested^-(Q)
\end{aligned}
$$

$$
\begin{aligned}
nested^-(H_1 \wedge H_2) &= nested^-(H_1) \cup nested^-(H_2) \\
nested^-(H_1 \vee H_2) &= nested^-(H_1) \cup nested^-(H_2) \\
nested^-(r_i.a_i = c_i) &= \emptyset \\
nested^-(r_i.a_i \neq c_i) &= \emptyset \\
nested^-(r_i.a_i = r_j.a_j) &= \emptyset \\
nested^-(null(r_i.a_i)) &= \emptyset \\
nested^-(const(r_i.a_i)) &= \emptyset \\
nested^-(exists(Q)) &= nested^-(Q) \\
nested^-(notexists(Q)) &= nested^+(Q)
\end{aligned}
$$

$$nested(Q) = nested^-(Q) \cup nested^+(Q)$$

Definition 15.

$$
\begin{aligned}
notexists(Q')^{\perp}_{Q} &\rightarrow notexists(Q'^{\perp}_{Q}) \\
exists(Q')^{\perp}_{Q} &\rightarrow exists(Q'^{\perp}_{Q})
\end{aligned}
$$

Definition 16.

$$
\begin{aligned}
(\Sigma, R, H \vee null(r_i.a_i), P)^{\perp}_{Q} \rightarrow (\Sigma, R, H^{\perp}_{Q}, P) \\
\text{if } \exists Q' \in nested^+(Q), (H \vee null(r_i.a_i)) \in nested(Q'), r_i.a_i \in \perp^T_{Q'} \\
\text{if } \exists Q' \in nested^-(Q), (H \vee null(r_i.a_i)) \in nested(Q'), r_i.a_i \in \perp^F_{Q'}
\end{aligned}
$$

Proposition 6.

$$Eval_{SQL}(Q, D) = Eval_{SQL}(Q^{\perp}_{Q}, D)$$

Proof.

Definition 17.

$$(\Sigma, R, H \vee null(r_i.a_i), P)^{\perp^F}_{Q} \rightarrow (\Sigma, R, H^{\perp^F}_{Q}, P)$$
$$\text{if } \exists Q' \in nested^{-}(Q), (H \vee null(r_i.a_i)) \in nested(Q'), r_i.a_i \in \perp^{T}_{Q'}$$
$$\text{if } \exists Q' \in nested^{+}(Q), (H \vee null(r_i.a_i)) \in nested(Q'), r_i.a_i \in \perp^{F}_{Q'}$$

Proposition 7.

$$\forall Q' \in nested^{+}(Q), Eval(Q, D) = Eval(Q^{\perp}_{Q'}, D)$$

Proposition 8.

$$\forall Q' \in nested^{-}(Q), Eval(Q, D) = Eval(Q^{\perp^F}_{Q'}, D)$$

Proof. Assume (10).
By Induction ...
$H_1 \wedge H_2 \in nested^{+}(Q)$
$\exists r_i.a_i \in (\perp^{T}_{H_1} \setminus \perp^{T}_{H_2}) \wedge (H \vee null(r_i.a_i) \in nested(H_2)$
Then if $x[r_i.a_i] = \perp$, $Eval(H^{T}_{2,H_2}, D, x)$ might be different from $Eval(H^{T}_{2,H_1 \wedge H_2}, D, x)$ □

□

Proposition 9.

5 Moving around null check

Definition 18. *For each Query and nested Query we maintain a set of attribute that have to be not null in order for the query to be true resp. false we denote this set* $\perp^{T}_{Q}$ *resp.*$\perp^{F}_{Q}$.

Definition 19.

$$
\begin{aligned}
\bot^T_{(\Sigma,R,H_1 \wedge H_2,P)} &= \bot^T_{(\Sigma,R,H_1,P)} \cup \bot^T_{(\Sigma,R,H_2,P)} \\
\bot^T_{(\Sigma,R,H_1 \vee H_2,P)} &= \bot^T_{(\Sigma,R,H_1,P)} \cap \bot^T_{(\Sigma,R,H_2,P)} \\
\bot^T_{(\Sigma,R,r_i.a_i=c_i,P)} &= \emptyset \\
\bot^T_{(\Sigma,R,r_i.a_i \neq c_i,P)} &= \{r_i.a_i\} \\
\bot^T_{(\Sigma,R,r_i.a_i > c_i,P)} &= \emptyset \\
\bot^T_{(\Sigma,R,r_i.a_i < c_i,P)} &= \emptyset \\
\bot^T_{(\Sigma,R,r_i.a_i = r_j.a_j,P)} &= \emptyset \\
\bot^T_{(\Sigma,R,r_i.a_i \neq r_j.a_j,P)} &= \{r_i.a_i, r_j.a_j\} \\
\bot^T_{(\Sigma,R,r_i.a_i > r_j.a_j,P)} &= \emptyset \\
\bot^T_{(\Sigma,R,r_i.a_i < r_j.a_j,P)} &= \emptyset \\
\bot^T_{(\Sigma,R,r_i.a_i = p_i,P)} &= \emptyset \\
\bot^T_{(\Sigma,R,r_i.a_i \neq p_i,P)} &= \{r_i.a_i, p_i\} \\
\bot^T_{(\Sigma,R,r_i.a_i > p_i,P)} &= \emptyset \\
\bot^T_{(\Sigma,R,r_i.a_i < p_i,P)} &= \emptyset \\
\bot^T_{(\Sigma,R,const(r_i.a_i),P)} &= \{r_i.a_i\} \\
\bot^T_{(\Sigma,R,const(p_i),P)} &= \{p_i\} \\
\bot^T_{(\Sigma,R,null(r_i.a_i),P)} &= \emptyset \\
\bot^T_{(\Sigma,R,null(p_i),P)} &= \emptyset \\
\bot^T_{(\Sigma,R,exists(Q),P)} &= \bot^T_{Q[?]} \\
\bot^T_{(\Sigma,R,notexists(Q),P)} &= \bot^F_{Q[?]}
\end{aligned}
$$

$$
\begin{aligned}
\bot^F_{(\Sigma,R,H_1 \wedge H_2,P)} &= \bot^F_{(\Sigma,R,H_1,P)} \cap \bot^F_{(\Sigma,R,H_2,P)} \\
\bot^F_{(\Sigma,R,H_1 \vee H_2,P)} &= \bot^F_{(\Sigma,R,H_1,P)} \cup \bot^F_{(\Sigma,R,H_2,P)} \\
\bot^F_{(\Sigma,R,r_i.a_i=c_i,P)} &= \emptyset \\
\bot^F_{(\Sigma,R,r_i.a_i \neq c_i,P)} &= \emptyset \\
\bot^F_{(\Sigma,R,r_i.a_i > c_i,P)} &= \emptyset \\
\bot^F_{(\Sigma,R,r_i.a_i < c_i,P)} &= \emptyset \\
\bot^F_{(\Sigma,R,r_i.a_i = r_j.a_j,P)} &= \emptyset \\
\bot^F_{(\Sigma,R,r_i.a_i \neq r_j.a_j,P)} &= \emptyset \\
\bot^F_{(\Sigma,R,r_i.a_i > r_j.a_j,P)} &= \emptyset \\
\bot^F_{(\Sigma,R,r_i.a_i < r_j.a_j,P)} &= \emptyset \\
\bot^F_{(\Sigma,R,r_i.a_i = p_i,P)} &= \emptyset \\
\bot^F_{(\Sigma,R,r_i.a_i \neq p_i,P)} &= \emptyset \\
\bot^F_{(\Sigma,R,r_i.a_i > p_i,P)} &= \emptyset \\
\bot^F_{(\Sigma,R,r_i.a_i < p_i,P)} &= \emptyset \\
\bot^F_{(\Sigma,R,const(r_i.a_i),P)} &= \emptyset \\
\bot^F_{(\Sigma,R,const(p_i),P)} &= \emptyset \\
\bot^F_{(\Sigma,R,null(r_i.a_i),P)} &= \{r_i.a_i\} \\
\bot^F_{(\Sigma,R,null(p_i),P)} &= \{p_i\} \\
\bot^F_{(\Sigma,R,exists(Q),P)} &= \bot^F_{Q[?]} \\
\bot^F_{(\Sigma,R,notexists(Q),P)} &= \bot^T_{Q[?]}
\end{aligned}
$$

Definition 20.

$$
\begin{aligned}
(\Sigma,R,H,P)^{up} &\rightarrow (\Sigma,R,(H \setminus \{null(p_i)\}) \wedge const(p_i),P) \\
&\quad \textit{if } p_i \in \bot^T_{(\Sigma,R,H,P)} \\
(\Sigma,R,H,P)^{up} &\rightarrow (\Sigma,R,(H \setminus \{null(r_i.a_i)\}) \wedge notexists((*,R,null(r_i.a_i),P)),P) \\
&\quad \textit{if } r_i.a_i \in \bot^T_{(\Sigma,R,H,P)}
\end{aligned}
$$

Not well written, actually false (notexists Query is different than that) but intuitive.

Proposition 10.

$$\forall Q \in [\![SQL]\!]_\bot Eval_{SQL}(Q,D) = Eval_{SQL}(Q^{up},D)$$

6 Optimizer

We can read in postgres source code : "We stop as soon as we hit a non-AND item." Indeed whenever postgres try to optimize join thanks to index or hash function it stop when a disjunction appears. Moreover as seen in previous section our translation involve adding disjunction in sub-queries, meaning the optimizer will choose to do a nested-loop in order to compute the answer, resulting in a blow up in time. In order to keep the computation time reasonable we offer an rewritting of the query which might seem counter-intuitive but allow the optimizer to use hash and index in order to perform join.

First of all please notice that the only disjunction that might appear in our rewriting are of the form of $r_i.a = r_j.b \vee null(r_i.a) \vee null(r_j.b)$ then we will focus on the case even if most of the optimization might be applicable in more general case (ie. if the null check doesn't concern the same attribute then we have to care when performing $UNIONALL$, if the first literal of the disjunction isn't an equality optimization is useless because the optimizer won't be able to do a hash or index join.) Moreover if the primary query already contain disjunction we won't optimize it then we can consider that our query after translation is in CNF.

6.1 Compute possible answer

Proposition 11.

$$Eval_{SQL}(\Sigma, R, (r_1.a = c \vee null(r_1.a)) \wedge H, P) = Eval_{SQL}(\Sigma, (R \backslash \{r_1\}) \cup \{r_1^{\vee}\}, H, P)$$

$$r_1^{\vee} = Eval_{SQL}(*, \{r1\}, r_1.a = c \vee null(r_1.a), \emptyset)$$

Proof. $Q = (\Sigma, R(, r_1.a = c \vee null(r_1.a)) \wedge H, P)$

$$\begin{aligned} x \in^n Eval_{SQL}(Q*, D) &\Leftrightarrow x \in^n Eval_{SQL}(*, R, r_1.a = c \vee null(r_1.a), P) \cap Eval_{SQL}(*, R, H, P) \\ &\Leftrightarrow x \in^n Eval_{SQL}(*, R, r_1.a = c \vee null(r_1.a), P) \wedge x \in^n Eval_{SQL}(*, R, H, P) \\ &\Leftrightarrow R^{\times}(x) \geq n \wedge (x[r_1.a] = c \vee x[r_1.a] = \bot) \wedge x \in^n Eval_{SQL}(*, R, H, P) \\ &... \end{aligned}$$

□

Proposition 12.

$$Eval_{SQL}(\Sigma, R, ((r_1.a = r_2.b \vee null(r_1.a) \vee null(r_2.b)) \wedge H, P) = Eval_{SQL}(\Sigma, (R \backslash \{r_1, r_2\}) \cup \{r_{1 \vee 2}\}, H, P)$$

$$r_{1 \vee 2} = Eval_{SQL}(*, \{r_1, r_2\}, r_1.a = r_2.b, P) \uplus Eval_{SQL}(*, \{r_1, r_2^+\}, null(r_1.a), P) \uplus Eval_{SQL}(*, \{r_1, r_2\}, null(r_2.b),$$

$$r_2^+ = EvalSQL(*, \{r_2\}, const(r_2.b), H, P)$$

6.2 Compute certain answer

In order to compute certain answer we do not necessarily have to compute possible answer, indeed we only have to compute a set representing the possible answer which interest us.

Proposition 13.

$(\Sigma, R, notexists((*, R', H \wedge (r_1.a = r_2.b \vee null(r_1.a)), P')), P) = (\Sigma, R, notexists((*, R', H \wedge r_1.a = r_2.b, P')) \wedge not$