Relational Database's Beta ranges over names
\tau :\beta := T_{1} as N_{1},...,T_{k} as N_{k}\; for \;k>0,\tau=(\tau _{1}to\tau_{k}),\beta =(N_{1}toN_{k})

Relational Database's Alpha ranges over terms
Each Entry has k terms:
\alpha :{\beta}' := t_{1}as{N_{1}}',...,t_{m}as{N_{k}}' \; for: \;m>0,\alpha=(\tau _{1}to\tau_{m}), {\beta}' =({N_{1}}'to{N_{k}}')

n*m*k values associated this way
Of course none of these values can be zero for a proper database to form.

One of the problems with SQL Standard is that the semantics of queries is non com-positional, semantically, a query can behave differently depending on the context in which it occurs.

Conditions for our SQL Query language are important to outline:
Our conditions all boil down to boolean states, a conditional is either True,False,or a predicate that evaluates to a boolean True,False.
\theta := True|False|P(t_{1} to t_{k}), \; Where \; P \; is \; a \; collection \; of \; predicates

The predicate states are typical in any boolean system. Membership vs. negated membership,
|t \; is \; [NOT]\; NULL

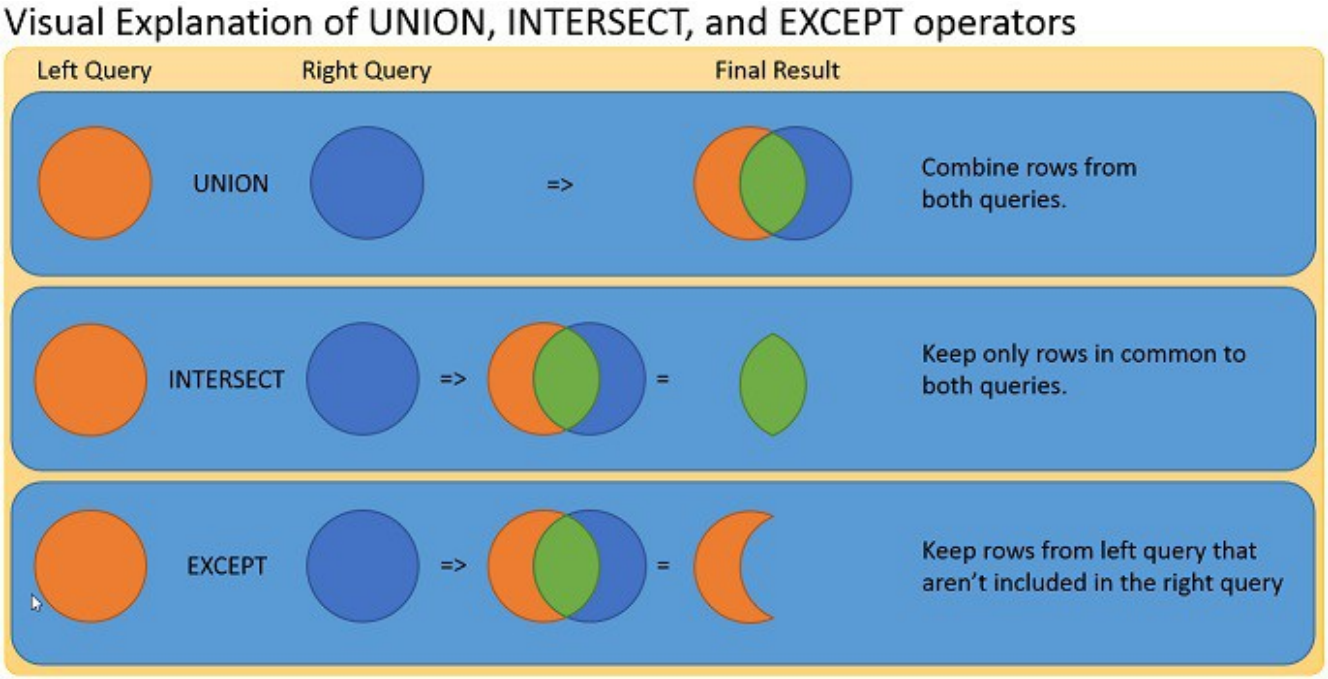
A querry evaluation where (assuming t was a member) t-bar represents the conglomeration of values from t. So our value either, is/is not in Q, this assumes Q exists,
|\bar{t}\; is \; [NOT] \; in \; Q|Exists \; Q

Theta has union, intersection, and negation as part of it's conditional functionality,
|\theta [AND][OR]\theta
|[NOT]\theta

For SQL Queries, it is different from what we have seen before:
Q:= SELECT[DISTINCT]\alpha :{\beta}' from \tau :\beta where \Theta
Given our conditional theta, we search the entries terms (Alpha:Beta') from the full database entries (Tau:Beta). This can either be distinct or non-distinct selection.
Then,
|SELECT[DISTINCT]*from(\tau :\beta) where \;\Theta
In this usage, '*' simply implies the return of all the entries terms  from the full database, so if you grabbed a dog → You'd get Color, Name, Size or whatever your terms happened to be.

As mentioned before with the conditionals, SQL queries have specific clauses for different combinational structures:
Q(UNION|INTERCEPT|EXCEPT)[ALL]Q
The SQL Union operator returns rows from both databases. If used by itself, UNION returns a distinct list of rows. Using UNION ALL, returns all rows from both databases
The SQL Intercept operator returns rows that the two databases share in common, as well as the ones that are unique in between them.
The SQL Except operator returns rows from the first databases that aren't also in the second database.

Quick Visualization I found:




Source: https://277dfx2bm2883ohl6u2g3l59-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/UnionInsersectExcept.jpg

Small Steps:

If Q steps to some E, then Q exists, and the E it steps to is non-empty
If \; Q\overset{n}{\rightarrow}E \; | EXISTS \; Q \overset{n}{\rightarrow} \; NOT \; EMPTY(E)

It is possible if our base value, say b is already boolean, T or F, then b steps to b in any number of steps
b\overset{n}{\rightarrow}b|b = TRUE \; or \; FALSE

If t IS NULL, it steps to null t-hat if t stepped to t hat before:
t \; IS \; NULL \overset{n}{\rightarrow}null(\hat{t}) |IF \; t\overset{n}{\rightarrow}\hat{t}

If t IS NOT NULL, it steps to null t-hat if t stepped to t hat before:
t \; IS \; NOT \; NULL \overset{n}{\rightarrow}NOT \;null(\hat{t}) |IF \; t\overset{n}{\rightarrow}\hat{t}

We can step the whole predicate base to t-hat if entry I steps to I-hat and entry I' steps to I'-hat:
P(t1,...,ti)\overset{n}{\rightarrow}P(\hat{t1},...,\hat{ti})|IF \; ti\overset{n}{\rightarrow}\hat{ti},ti'\overset{n}{\rightarrow}\hat{ti}'

While in the state with conditionals (theta) evaluating to theta' we have:
|For \; \Theta_{1}\overset{n}{\rightarrow}\Theta '_{1},\Theta_{2}\overset{n}{\rightarrow}\Theta_{2}'|
|\Theta_{1}\;AND\;\Theta_{2}\overset{n}{\rightarrow}\Theta_{1}'\wedge \Theta_{2}'|
|\Theta_{1}\;OR\;\Theta_{2}\overset{n}{\rightarrow}\Theta_{1}'\vee \Theta_{2}'|
|NOT \; \Theta_{1}\overset{n}{\rightarrow}NOT \; \Theta_{1}'|

Such that the union of theta_1 and theta_2 can evaluate to the union of theta_1' and theta_2' over n steps. The same applies for intersection and NOT, but not EXCEPT