# Select statement planner specification

## Algorithm

### Get Range Table

Retrieve a list of all the relations referenced in the query **table\_ref\_list**[[1]](#footnote-1). Each list element returned must contain:

* The name of the relation (identifier)
* The alias of the relation (if provided)
* A query-scoped, generated **relation ID**
* A globally-scoped **object ID** for the relation obtained from the system catalogue

If the relation is not found in the system catalogue, an error is generated and the query fails.

### Identifier Resolution

For each IDENTIFIER in the **where\_clause**, we must determine:

* Whether it is a valid reference to a range table relation
* If so, *which* range table relation the identifier belongs to (ie, which table contains this column.)

#### Mini-algorithm for identifier resolution

For each IDENT node in the **where\_clause:**

1. Start
   1. If there is a relation reference included in the identifier, check the range table for a relation with the given reference. Otherwise obtain the base name of the relation (**table\_ref:value**) and go to (ii)
   2. If there is no relation with the given reference, generate an error[[2]](#footnote-2) and go to (iv)
   3. If there *is* a relation with the given reference, obtain the base name of that relation from the **range table** and go to (ii)
   4. Query the system catalogue to determine if a relation with the base name obtained in (i) exists. If not, generate an error and go to (iv)
   5. Query the system catalogue to determine if the attribute name given in the IDENT node is present for the relation. If it is not present, generate an error and go to (iv)
   6. Copy the following information to the IDENT node in the **where\_clause**:
      * The global object ID of the containing relation
      * The name of the containing relation
      * The data type of the *attribute*

### Scan Node Filter determination

The concept here is to determine the selection predicate for each relation scanned, and push those filter expressions down to the scan node in the execution plan.

The expression tree contained in the WHERE clause is broken into subtrees, each containing only references to attributes of the same relation. Implicit joins are identified by operator nodes linking those subtrees.

1. This actually refers to all relations referenced in the current **select\_statement**. Nested SELECTs, such as subqueries, are each planned seperately using this algorithm. [↑](#footnote-ref-1)
2. Compiling a list of error conditions and codes is a work-in-progress at the time of writing. Prototype will issue a generic ‘query failed’ error. [↑](#footnote-ref-2)