**Input Vectors**

* ~~Attribute Types (multiple fields in raw\_data)~~
* ~~Attribute Names (one field in raw\_data; will be an IDENTIFIER)~~ (WONT WORK BECAUSE WE MAY NEED TO CALL attribute\_list() MULTIPLE TIMES BEFORE USING VALUES)
* New Attribute Names (one field in raw\_data)
* Keys (one field in raw\_data; will be an IDENTIFIER)
* Tuple (one field in raw\_data; will be INTEGER or LITERAL)
* All located in Parser.h for use in all Parser functions

**Queries**

*Select*

In Parser: selection()

1. Parse condition()
2. Get relation or view name returned from atomic\_expr()
3. Run condition tree on all tuples, find all indices that satisfy condition
4. Db.select() for each valid index ); name the new view with some undetermined naming scheme
5. Return new view’s name

In Database: select()

1. Throw error if no such table exists
2. Throw error if a tuple with identical key already exists

*Project*

In parser: projection()

1. Get attribute names vector returned from attribute\_list()
2. Get relation or view name returned from atomic\_expr()
3. Call db.project() with arguments from (1) and (2); name the new view with some undetermined naming scheme
4. Return new view’s name

In Database: select()

1. Accept a relation or a view (currently only accepts relation)
2. Throw error if no such argument
3. Throw error if no such relation or view (in\_table\_name)

Rename

In parser: renaming()

1. Get attribute names vector returned from attribute\_list()
2. Get relation or view name returned from atomic\_expr()
3. Call db.rename() with a rguments from (1) and (2); name the new view with some undetermined naming scheme
4. Return view’s name

In Database: rename()

1. Throw error if no such table
2. Throw error if incorrect number of attributes

Union (PLUS)

In parser: renaming()

1. Get relation or view name 1 from atomic\_expr()
2. Get relation or view name 2 from atomic\_expr()
3. Call db.

In Database: rename()

Difference (MINUS)

Product (TIMES)

Expression() must return strings (of the arbitrarily named created views)

Atomic\_expr() must return strings (will either be a relation name or a view returned from expression())