Manipulating functional dependencies

This document contains detailed descriptions of the algorithms used to manipulate FDs.

Closure of an attribute set
Given relation R, attribute set A (subset of R), and an FD set FS over relation R, return A+, the set of columns entailed by A. Note that A may be the LHS of some F in FS, but this is not required. The computation proceeds as follows:

In plain English, every attribute set entails itself, and from there, we consider each FD in turn; for each F whose LHS is already in A+, we can add RHS(F) to A+ as well. Once we’ve used an FD in this way we never need to consider it again, but the change may mean that FDs we previously considered and discarded are now useful. This is why we have two loops; the outer loop stops if either we add all FDs into A+ or if we make a pass over all remaining FDs with no changes to A+.

Minimal basis
Given a relation R and FD set FS over R, we want to compute FS’, an FD set containing the fewest (and least complex) FDs that still allows recovery of FS by transitivity. FS’ is called the minimal basis of FS. The algorithm contains two pieces. The first part attempts to make a single change to the FD set:
The second calls the first until no more changes can be made:

```python
def minbase(FS):
    fprime = split FS into singleton FDs (e.g. |RHS|=1)
    tmp = one_change(fprime)
    while fprime != tmp:
        fprime = tmp
        tmp = one_change(fprime)
    return fprime
```

In plain English, we are interested in two kinds of simplifications:

1. Look for unnecessary FDs, for which all attributes in the RHS are entailed by other FDs. This can be detected by removing the FD from the FD set and testing whether LHS+ shrinks; a shrinking LHS+ means the FD was necessary because its RHS cannot be inferred from other FDs.

2. Look for FDs with unnecessary attributes in the LHS. This can be detected by removing the attribute from the LHS and then testing whether the new LHS+ is larger than the old one; a growing LHS+ means the attribute is a necessary restriction on the LHS, to prevent it from being too general.

Each change to the input set might allow further changes, so we start over and keep attempting to make changes until all rules have been applied unsuccessfully.

**Projection**

Given some relation R’ whose attributes are a subset of R, and FS a set of FDs over R, projection produces a new FS’ that contains as much of FS as possible but mentions only attributes in R’. The algorithm proceeds as follows:
def project(rprime, FS):
    ndfs = {}
    fprime = {f in minbase(FS) : LHS(f) subset-of R'}
    todos = {(f,fprime) : f in fprime }
    while todos not empty:
        f,fp = (!! pop f from todos having smallest |lhs(f)|
        if rhs(f) subset-of rhs(f'), where lhs(f)=lhs(f'):
            continue
        tmp = fp - f
        if lhs(f) already in some f' of nfds:
            union rhs(f) into rhs(f')
        else:
            add f to nfds
        for f' in fp:
            if lhs(f') subset-of lhs(f):
                continue # already seen
            if rhs(f') subset-of rhs(f):
                continue # redundant
            nlhs = lhs(f) | lhs(f')
            nrhs = closure(nlhs, FS)
            if nrhs != rhs(f) and nrhs != rprime:
                ! ! add ((nlhs,nrhs),tmp) to todos
        nfds = {(lhs(f), rhs(f)&rprime) : f in nfds}
    return minbase(nfds)

In plain English, we start with a list of FDs whose LHS is entirely contained in R', and add each of those (along with fprime) to a TODO list. We then examine FDs in the TODO list, smallest LHS first. For each F we examine, first check whether its RHS contains any new information (e.g. adds more columns to LHS+); if not, skip it. Otherwise, remove F from the fp we’re working with and union its RHS into LHS+ (or create a new LHS if it didn’t already exist). Finally, see this LHS+ is complete: if any lhs(f') in fp is not a subset of lhs(f), and if rhs(f') is not a subset of rhs(f), the union of lhs(f)|lhs(f') is non-trivial and we must process it. Compute what we already know of the closure for this new FD, then add it to the TODO list along with fp-f as its corresponding FD set (so we don’t re-examine FDs we already used). When the TODO list is empty, the rhs of various FD in nfds may refer to columns that are not in R', so we strip those out by intersecting them with R'. Finally, we compute the minimal basis of nfds and return that to the caller.