**Functional Dependency**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Graphical user interface  Description automatically generated with medium confidence | |  |  |  |  |  | | --- | --- | --- | --- | --- | | a | b | c | d | e | | C | 3 | X | 722 | 112 | | A | 1 | X | 289 | 553 | | A | 1 | Y | 189 | 583 | | B | 1 | X | 289 | 513 | | C | 1 | X | 289 | 553 | |
| Graphical user interface  Description automatically generated with medium confidence | **Level 1:**  E: {a, b, c, d, e} –> (LHS)  C: {a: abcde, b: abcde, c: abcde, d: abcde, e: abcde} -> (RHS)  F = E |
| Graphical user interface  Description automatically generated with medium confidence | **Level 2:**  **Step 1**  E: {ab, ac, ad, ae, bc, bd, be, cd, ce, de} –> (LHS)  C: {ab: abcde, ac: abcde, ad: abcde, ae: abcde,  bc: abcde, bd: abcde, be: abcde, cd: abcde, ce: abcde, de: abcde} |
|  | **Step 2**  Traverse de LHS, for each attribute of LHS if in RHS, **check FD**\*:  If X\A ->A (Determinant -> dependant)  Check ab\a->a, ab\b->b, ac\a->a...)  **FDs: bd\b -> b, be\b -> b,**  **cd\c -> c, ce\c -> c, de\d -> d**  **C1: remove a from RHS**  C: {ab: abcde, ac: abcde, ad: abcde, ae: abcde,  bc: abcde, bd: a~~b~~cde, be: a~~b~~cde, cd: ab~~c~~de, ce: ab~~c~~de, de: abc~~d~~e}  **C2: remove all R\X if there is a X\B->B**  C: {ab: abcde, ac: abcde, ad: abcde, ae: abcde,  bc: abcde, bd: ~~ac~~d~~e~~, be: ~~acd~~e, cd: ~~ab~~d~~e~~, ce: ~~abd~~e, de: ~~abc~~e}  C: {ab: abcde, ac: abcde, ad: abcde, ae: abcde,  bc: abcde, bd: d, be: e, cd: d, ce: e, de: e} |
|  | **Step 3**  Look for (super)key  ad is (super)key  ae is (super)key  For each (super)key, traverse the RHS C(X)\X  if RHS in all C of augmented LHS subsets:  FD (X->A) is minimal  **ad->e is minimal**  key pruning, removing LHS from the prefix tree.  F: {ab, ac, ~~ad~~, ~~ae~~, bc, bd, be, cd, ce, de}  FDs (acum): {d->b, e->b, d->c, e->c, e->d, ad->e}  Keys (acum): {ad, ae} |
|  |  |
| Graphical user interface  Description automatically generated with medium confidence | **Level 3:**  **Step 1**  E: {abc, bcd, bce, bde, cde} –> (LHS)  C: {abc: abcde, bcd: d, bce: e, bde: ‘’ , cde: ‘’} |
|  | **Step 2**  Traverse de LHS, for each attribute of LHS if in RHS, **check FD**\*:  If X\A ->A (Determinant -> dependant)  **FDs: bcd\d -> d**  **C1: remove a from RHS**  C: {abc: abcde, bcd: ~~d~~, bce: e, bde: ‘’, cde: ‘’}  **C2: remove all R\X if there is a X\B->B**  C: {abc: abcde, bcd: ‘’, bce: e, bde: ‘’, cde: ‘’}  **Nothing to remove**  C: {abc: abcde, bcd: ‘’, bce: e, bde: ‘’, cde: ‘’} |
|  | **Step 3**  Look for (super)key  abc is (super)key  For each (super)key, traverse the RHS C(X)\X  if RHS in all C of augmented LHS subsets (C3):  FD (X->A) is minimal  **There is no minimal FD**  key pruning, removing LHS from the prefix tree.  F: {~~abc~~, ~~bcd~~, bce, ~~bde, cde~~}  FDs (acum): {d->b, e->b, d->c, e->c, e->d, ad->e, bc->d}  Keys (acum): {ad, ae, abc} |
|  | **Level 4:**  **Step 1**  E: { } –> (LHS)  C: { }  F: { }  FDs (acum): {d->b, e->b, d->c, e->c, e->d, ad->e, bc->d}  Keys (acum): {ad, ae, abc} |

**Explanations**

|  |
| --- |
| **Concepts**  C(X): candidate sets for each node X. These sets store the RHS attribute (consequent) of a potential FD having the generating node as LHS (dependant) |
| **Check for valid functional dependencies (X\A ->A)**  (Determinant -> dependent)  Use key-error and pli to infer FDs (key error can be used with pli and stripped-pli):    X->y  pli(X)=pli(XY)  cr(X)=Cr(XY)    Therefore:  X\A->A  Pli (X\A) = pli (X\A+A) = pli (X)  Cr(X\A) = Cr(X) |

|  |
| --- |
| **C1**  C1 = {A ⋲ X | X\A->A holds}  C1 removes the dependent for the other levels that contain X (all the next levels will have FDs, but they will not be minimal)  Removing A from the RHS, avoid generating the same FD with the dependent term in the other levels.  E.g.:  d->b  X=bd  A=b  Graphical user interface, diagram  Description automatically generated  If d->b:  In the next level (level 3):  abd (ab->d, ~~ad->b~~, bd->a)  bcd (bc->d, bd->c, ~~cd->b~~)  bde (bd->e, be->d, ~~de->b~~)  Remembering that in the algorithm it is not checked all these edges…one to one (e.g. picture below), but these edges are removed from the candidates (b is removed from the candidates for ‘bd’). Therefore, for the next level ‘b’ will not be considered in the RHS for the LHS ‘abd’, ‘bcd’, ‘bde’). It will be removed when do the intersection of the RHS**.**  Diagram, schematic  Description automatically generated |

|  |
| --- |
| **C2**  C2 = {R\X} if ⱻB ⋲ X: X\B->B}  Removes all R\X if there is a X\B->B  C2 removes supersets of a FD (all the next levels will have FDs, but they will not be minimal)  E.g.:  Graphical user interface, diagram  Description automatically generated  **C3**  C3 = {A ⋲ X | ⱻB ⋲ X\A : X\{A, B}->B holds}  Same idea than B2, but for subsets  Assume X has proper subset Y (X ⊃ Y) such that Y\B -> B  Then we can remove from C(X) all A ⋲ X\Y  same approach than B2, but top-down – opposite  **C3 is explained in C1.**  same picture and explanation that is in the example.    **(Other example seeing in class)** |
| **If key or superkey**  check if is a (super)key using key-error and pli  If is a (super)key all rows will be singletons and (key-error =0)    number of values in PLI minus number of groups  if X is (super) key: X -> R\X  remove from the LHS |

|  |
| --- |
| **If key or superkey,**  **check if FD is minimal**  Graphical user interface, diagram  Description automatically generated  X=ad  ad->e is minimal  If X is (super) key  ad is (super) key  traverse the RHS C(X)\X  C: {ab: abcde, ac: abcde, ad: abcde, ae: abcde, bc: abcde, bd: d, be: e, cd: d, ce: e, de: e}  ... in this case C(‘ad’) = 'abcde' – ‘ad’ = ‘bce’  For each A in ‘bce’  if RHS in all C of augmented LHS subsets:  Look the values for C[X+A-B]... B⋲X…  In this case:  when A=b, iterate B... [ad +b -a], [ad +b -d]  …[bd], [ab]  Check if A=b belongs to every candidate sets for [bd], [ab]  (false # in this case b is not in RHS of ‘bd'!)  when A=c, iterate B... [ad +c -a], [ad +c -d]  …[cd], [ac]  Check if A=c belongs to every candidate sets for [cd], [ac]  (false # in this case c is not in RHS of ‘cd'!)  when A=e, iterate B... [ad +e -a], [ad +e -d]  …[de], [ae]  (true in this case e is in RHS of ‘de’ and ‘ae’!)    FD (X->A) is minimal  **ad->e is minimal** |

Comments:

|  |
| --- |
| X\A means A is inside X  X -> A means A is in R\X |

**Similar results to the Metanome:**

Graphical user interface, application

Description automatically generated

Metanome has one FD more: abc->e, but it is not minimal.

Using the results of the developed algorithm, we can find the same FD, since bc->d and ad->e. Therefore, abc->e.