LECTURE 5 - CANONICAL COVERS

Canonincal Cover Minimal Cover
Schema Refinement = Can **FD**'s **F** be minimized/more compact?

> **G** is a *canonical/mnimal cover* of **F** when:
>
> 1. **G** is *equivalent* to **F** (**G** **F**)
> 2. **G** is *minimal* (if any part of G is removed G F)
> 3. Every **FD** in **G** has a single attribute on the RHS.

* (G F) if (F G) and (G F)
* multiple canonical covers can exists

A canonical cover **G** is minimal in two respects:

1. Every FD in **G** is "required" in order for G to be equivalent to F.
2. Every FD in **G** is as "small" as possible (only one attribute RHS)

Computing Canonical Cover

A FD in the set is *redundant* if it can be derived from the other FD's in the set.
ORDER MATTERS

1. Decompose all FDs in **standard form**
   - i.e. only one attribute on the RHS
2. Check LHS for **Redundant Attributes**:
   - Check FD's with attributes on the LHS fore redun
   - for each FD **AB** **C** in **G**, check if **A** or **B** on the LHS is redundant
   - Can **A** be removed from **AB** **C** ?
     - Check A
     - if **C** **A** then **A** is **Redundant**
     - then **A** can be removed from **AB** **C**

- Can **B** be removed from **AB** **C** ?
  - Check B
  - if **C** **B** then **A** is **Redundant**
  - then **B** can be removed from **AB** **C**
3. Remove **Redundant FD's**:
   - Remove each FD one at a time, and check fro the closure of F with it removed. If the result can be acheived without the FD it is redundant.
   - (G - {X A}) F ?
   - For every FD **X** **A** in **G**
     - Remove {**X** **A**} from **G**; call the result **G’**
     - Compute X under G’
     - If **A** **X** under **G’**, then **X** **A** is redundant and hence remove **X** **A** from **G**.

Example:

This example explains computing canonical covers well, just dont move to the next line until you know exactly whats going on first.
R = {A, B, C, D, E, F, G, H}

F = { CD A , EC H , GHB AB , C D , EG A , H B , BE CD , EC B }

Find a canonical cover for F ?

1. Decompose all FDs in standard form

   G = { CD A , EC H , GHB A , ~~GHB B~~ , C D , EG A , H B , BE C , BE D , EC B}

   Since, GHB B is trivial

2. Eliminate unnecessary attributes from LHS
   We need to check all the FDs that have more than one attribute on the LHS:

- **CD A**

  Can we remove D from CD A?

  C+ = CDA; YES (because, A∈ C+) [C D, CD A]

  Can we remove C from CD A?
  D+ = D; NO

- **EC H**

  Can we remove E from EC H?
  C+ = CDA; NO

  Can we remove C from EC H?
  E+ = E; NO

- **GHB A**
  We first check if we could remove 2 attributes from LHS:

  Can we remove GH from GHB A?
  B+ = B; NO

  Can we remove GB from GHB A?
  H+ = HB; NO

  Can we remove HB from GHB A? G+ = G; NO
  We now check if we could remove one single attribute from LHS:

  Can we remove G from GHB A?
  HB+ = HB; NO

  Can we remove H from GHB A?
  GB+ = GB; NO

  Can we remove B from GHB A?

  GH+ = GHBA; YES (because, A∈ GH+) [H B, GHB A]

- **EG A**

  Can we remove E from EG A?
  G+ = G; NO

  Can we remove G from EG A?
  E+ = E; NO

- **BE C**

  Can we remove B from BE C?
  E+ = E; NO

  Can we remove E from BE C?
  B+ = B; NO

- **BE D**

  Can we remove B from BE D?
  E+ = E; NO

  Can we remove E from BE D?
  B+ = B; NO

- **EC B**

  Can we remove E from EC B?
  C+ = CDA; NO

  Can we remove C from EC B?
  E+ = E; NO

- We now have:

  G' = { C A , EC H , GH A , C D , EG A , H B , BE C , BE D , EC B }

3. Remove redundant FD(s)

- **C A,**

  C+ = CD; so it’s not redundant.

- **EC H**

  EC+ = ECBDA; so it’s not redundant. [EC B, C D, C A]

- **GH A**

  GH+ = GHB; so it’s not redundant.

- **C D**

  C+ = C; so it’s not redundant.

- **EG A**

  EG+ = EG; so it’s not redundant.

- **H B**

  H+ = H; so it’s not redundant.

- **BE C**

  BE+ = BED; so it’s not redundant.

- **BE D**

  BE+ = BECD; [BE C, C D]

  Because D BE+, BE D is redundant. **Remove it**!

- **EC B**

  EC+ = ECHB; [EC H, H B]

  Because B D EC+, EC B is redundant. **Remove it**!

Therefore the canonical cover is:

F = { C A , EC H , GH A , C D , EG A , H B , BE C }

How To Deal With Redundancy

Decompose the relation into seperate relations:

Before:

Name	Address	RepresentingFirm	Spokesperson
Carrie Fisher	123 Maple	Eone	Joe Smith
Harrison Ford	789 Palmer	Eone	Joe Smith
Mark Hamil	456 Oak	LoserRus	Mary Johns

After:

Name	Address	RepresentingFirm
Carrie Fisher	123 Maple	Eone
Harrison Ford	789 Palmer	Eone
Mark Hamil	456 Oak	LoserRus

RepresentingFirm	Spokesperson
Eone	Joe Smith
LoserRus	Mary Johns

Properties of a decomposition?

1. **Lossless-join** (**a must**)
2. **Dependency-preserving** (desirable)

A	B	C
1	2	3
4	2	5

Decomposed into:

A	B
1	2
4	2

B	C
2	3
2	5

When joined:

A	B	C
1	2	3
4	2	5
4	2	3
1	2	5

Lossless-join:

A	B	C
1	2	3
4	2	5

* no new tuples 🙌

Dependency-preservation

F = { B C , B D , A D }

A	B	C	D
1	2	5	7
4	3	6	8

Decomposed into:

A	B
1	2
4	3

B	C	D

B	C	D
2	5	7
3	6	8

* Dependency-preservation would preserve A D