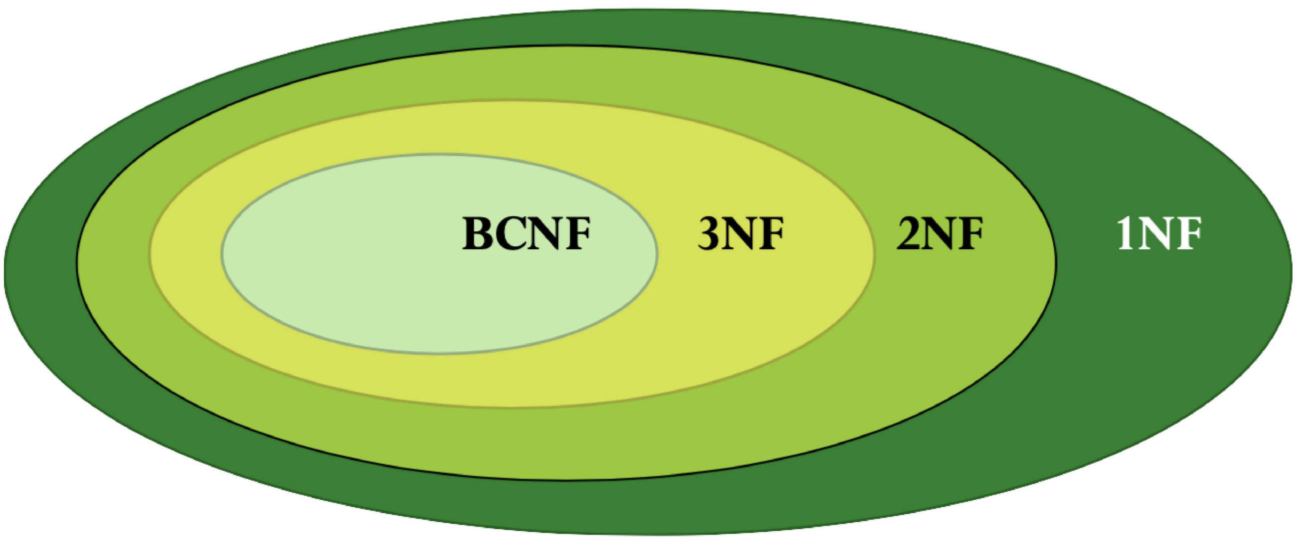
LECTURE 6 - NORMALIZATION

Normal Forms

We normalize to get rid of “extra” functional dependencies that causes redundancy and anomalies.
Normals Forms:

- **First normal form (1NF)**: components of tuples must be atomic
- **Second normal form (2NF)**: transitive FD’s still allowed, no FD whose LHS is a proper subset of a key
- **Third normal form (3NF)**: LHS is superkey or RHS key subset
- **Boyce-Codd normal form (BCNF)**: LHS is superkey



Each normal form gets a little stricter and gets rid of more anomalies.

1NF

1. No attribute is allowed to be composite or multi valued

Example:
The following relation is **not** in 1NF: Student (SID, SName, {(CourseId, CouseName, Grade)})

2NF

1. It is in 1NF
2. Every non-prime attribute of relation is fully functionally dependent on the primary key

For each non-key attribute, ask:
If I knew the value for part of the Primary-Key, could I tell what the value for a non-key attribute would be?

Example:
Inventory (Item, Supplier, Cost, SupplierAddress)
If I know just Item, can I find out SupplierAddress? NO
If I know just Supplier, can I find out SupplierAddress? YES

SupplierAddress is NOT fully functionally dependent upon the ENTIRE Primary-Key **NOT 2NF**

3NF

One of the 3 must be met for every FD **X** **A**:
Need to compute cnadidate keys in order to check!

1. **X** **A** is a **trivial** FD (**A** **X** ex: **AB** **A**)
2. **LHS** is a superkey (i.e. a key is contained in **LHS**)
3. **RHS** is part of any key of **R**

BCNF

One of the 3 must be met for every FD **X** **A**:

1. **X** **A** is a **trivial** FD (**A** **X** ex: **AB** **A**)
2. **X** is a superkey
3. ~~**RHS** is part of any key of **R**~~

BCNF can always obtain **lossless-join** decomposition
BCNF is not always **dependency-preserving**

Synthetic 3NF Decompostion

1. Compute the canonical cover **F**
2. Create relations
3. Check if at least one of the keys exists in the above relations
4. Add an extra relation containing those attributes that form any key of **R**

Example:

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 }

Candidate Keys = {BEFG, CEFG, EFGH}

1. F = { C → AD , EC → H , GH → A , EG → A, H → B , BE → C }

2. Create the relations:

R	F
R = {A, C, D}	F = {C → AD}
R = {E, C, H}	F = {EC → H}
R = {A, G, H}	F = {GH → A}
R = {A, E, G}	F = {EG → A}
R = {B, H}	F = {H → B}
R = {B, C, E}	F = {BE → C}

3. Check if at least one of the keys {BEFG, CEFG, EFGH} exists in the above relations.
   Since none of these keys is in the relations, this decomposition is **not lossless**.

4. add an extra relation containing those attributes that form any key of R:

R	F
R = {B, E, F, G}	F$$-7 = {}

The relation schema **is** now **lossles** as well as **dependency preserving**.

Is R in 3NF or BCNF ?

R = { A , B , C , D , E , F , G , H }

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

Is Dependency is Preserved ?

Given a decomposed set of relations and FD's R_1 , R_2 ...R_n and F_1 , F_2 ...F_n .

Compute {**X**}$^+$ for the origial **X** $\rightarrow$ **A** originals FD's under the decomposed FD's and check if **A** $\in$ {**X**}$^+$

Example:
R = (A , B , C , D)

F = { A $\rightarrow$ B , B $\rightarrow$ C , C $\rightarrow$ D }

Decomposed into:

R	F
R_1 = {A, B}	F_1 = A $\rightarrow$ B
R_2 = {A, C, D}	F_2 = C $\rightarrow$ D, A $\rightarrow$ D, A $\rightarrow$ C

Is the decomposition R = {R1, R2} dependency-preserving?

- Check if A $\rightarrow$ B is preserved

  Compute A+ under { A $\rightarrow$ B } U { C $\rightarrow$ D, A $\rightarrow$ D, A $\rightarrow$ C }
  A+ = { A, B, C, D }
  B $\in$ A^+ ? Yes

  A $\rightarrow$ B is **preserved**

- Check if B $\rightarrow$ C is preserved

  Compute B+ under { B $\rightarrow$ C } U { C $\rightarrow$ D, A $\rightarrow$ D, A $\rightarrow$ B }
  B+ = { B }

C B ? NO

B C is not preserved

Lossy Decomposition ?

The decompositio of relation R into R1 and R2 is lossy when the join of R1 and R2 does not yield the same relation as in R.