Normal Forms

- Restrictions on the database schema
  - precludes certain undesirable properties from DB
  - NFs represent good DB design to be specified

- First normal form (1NF)
  - R is in 1NF if values in dom(A) are atomic for every attribute A in R: not multi-valued
  - accurate expressions of FDs
  - non-1NF relations force a complex DML to manipulate
  - is it reasonable to require 1NF?

- Transforming the document schema into 1NF can be done, but the results would not be user friendly.

First Normal Form

- Normalization of document relation into 1NF

- document-1NF(title, author-list, date, keywords)

- (salesplan, Smith, 1, 4, 93, profit)
- (salesplan, Smith, 1, 4, 93, sales)
- (salesplan, Jones, 1, 4, 93, profit)
- (salesplan, Jones, 1, 4, 93, sales)

- Fourth normal form (4NF) will remove some of the awkwardness of such 1NF schema, but still complicates user interaction with the system

  - title →→ author
  - title →→ keyword

- For new applications of relational databases that would benefit from non-1NF representation, new NFs and criteria for good design are necessary.

Second Normal Form

- 2NF and 3NF arise from trying to avoid update anomalies and redundancy

- 2NF: Assign (Flight, Day, Pilot, Gate)

- Solution: decomposition

Second Normal Form

- In the example above, (Flight#, Gate#) is replicated in the relation, making the information redundant, and update complicated.

- Solution: decomposition

- (Passign, (Flight, Day, Pilot))

- (Gassign, (Flight, Gate))

- 112 1 May Jones
- 112 2 May Cook
- 203 2 May Jones
Second Normal Forms

- Full dependency
  - Given a set of FDs F and a FD $X \rightarrow Y$ in $F^+$, Y is fully dependent on X if Y is not functionally dependent on any proper subset of X.
  - Otherwise, Y is partially dependent on X.

<ex> FD = \{Flight Day $\rightarrow$ Pilot Gate, Flight $\rightarrow$ Gate\}
  - Gate is partially dependent on (Flight Day)
  - Pilot is fully dependent on (Flight Day)

- Prime
  - Attribute A is prime in relation schema R if A is a member of some candidate key of R.
  - Otherwise, A is non-prime in R.

Transitive Dependency and 3NF

- Why those problems are still there?
  - Intuitively, the chair is an attribute of a department entity, not of a student entity.
  - We have S# $\rightarrow$ Major-dept $\rightarrow$ Chair

- Transitive dependency
  - An attribute A is transitively dependent on X if for some attribute Y, $X \rightarrow Y \wedge Y \rightarrow A \wedge A \not\subseteq XY$

<ex> R = (Flight, Day, Pilot-id, Name)
  FD = \{FD $\rightarrow$ P, P $\rightarrow$ N, N $\rightarrow$ P\}
  Name is transitively dependent on (Flight, Day) because FD $\rightarrow$ P, P $\rightarrow$ FD, P $\rightarrow$ N, N $\not\subseteq$ FDP

- 3NF
  - R is in 3NF if no non-prime attribute of R is transitively dependent on any key of R.
  - No transitive dependency

Third Normal Form

In the example above, R is not in 3NF because of a transitive dependency between Name and (Flight Day)

Solution: decomposition
  - $R_1 = (\text{Flight, Day, Pilot-id, Name})$
  - $R_2 = (\text{Pilot-id, Name})$
  - Then $R = \{R_1, R_2\}$ is in 3NF w.r.t. F

<Theorem> If R is in 3NF, then R is in 2NF.

Proof: If R is in 3NF but not in 2NF, then there must be a partial dependency $X \rightarrow A$, where A is a non-prime and X is a proper subset of some key Y.

Then $Y \rightarrow X$ (Y is a key), $X \rightarrow A$ (given), $X \rightarrow Y$ ($X \subseteq Y$), and $A \not\subseteq X$ (A is non-prime)

$\Rightarrow$ A is transitively dependent on Y

R is not in 3NF, a contradiction.
Notes on 3NF

- 3NF allows two FDs \( X \rightarrow Y \) and \( Y \rightarrow A \) in the same relation only if \( X \rightarrow Y \) and \( Y \rightarrow X \)
  --- both \( X \) and \( Y \) are keys
- It leads to decomposing independent FDs into different relation schema, where they may be updated independently
- This suggests that more joins will be performed at run-time
  - system performance for such joins can be improved by physically storing such relations near to each other
- 1NF, 2NF, and 3NF are very popular
  - other normal forms and more theory exist, but only those simpler ones have achieved widespread acceptance

Good Database Design

- What is a good design?
  A 3NF decomposition that satisfies lossless join and dependency preservation
  --- certainly necessary, although not sufficient
- Constructing a 3NF database through decomposition
  - always possible to begin with any relation schema not in 3NF w.r.t. \( F \), and decompose it into a schema that is in 3NF w.r.t. \( F \)
  - decomposing \( R \) means breaking it into \( R_1 \) and \( R_2 \) such that \( \Pi_{R_1}(r) \upharpoonright \Pi_{R_2}(r) = r \)
    --- lossless join decomposition
  - decompose again repeatedly if either \( R_1 \) or \( R_2 \) is not in 3NF, until all relations are in 3NF w.r.t. \( F \)

Decomposition into 3NF

Given \( R \) and \( F \)
1. If there are no transitive dependencies in \( R \), we are done
2. Otherwise, we have a transitive dependency on a key in \( R \\
   \text{Let } K \text{ be a key, } Y \subseteq R, A \text{ a non-prime attribute in } R, \text{ such that } K \rightarrow Y, Y \rightarrow K, Y \rightarrow A, \text{ and } A \notin KY
3. Let \( R_1=R-A \) and \( R_2=YA \)
   \( K \) is a key for \( R_1 \), \( Y \) is a key for \( R_2 \\
   --- one transitive dependency is removed from \( R \)
4. Other attributes that are non-prime and \( \in R -(KY) \)
   that are dependent upon \( Y \) can be placed in \( R_2 \) at the same time
   --- speed up

Notes on Decomposition

- \( \Pi_{R_1}(r) \upharpoonright \Pi_{R_2}(r) = r \) (lossless join)
  Let \( R_1 \) and \( R_2 \) be relation schema with \( R_1 \cap R_2 = X \)
  where \( X \rightarrow R_1 \) (or \( R_2 \))
  Then for any relation \( r \) with schema \( R_1 R_2 \)
  \( r = \Pi_{R_1}(r) \upharpoonright \Pi_{R_2}(r) \)
In step 3 of decomposition, \( R_1=R-A, R_2=YA \)
\( R_1 \cap R_2 = Y \)
\( Y \rightarrow A \) (given)
\( Y \rightarrow YA \) (augmentation)
Hence \( Y \rightarrow R_2 \)
- Decomposition cannot go on forever, since each time we decompose \( R \), resulting schema \( R_1 \) and \( R_2 \) are smaller, and no transitive dependency can exist with only two attributes
- Decomposition is not unique