DB DESIGN THEORY

Chapter 10 (part 1):

FUNCTIONAL DEPENDENCIES

Outline of Chapter 11

1 Introduction.

2 Informal Design Guidelines for Relational Databases.

3 Functional Dependencies (FDs)

4 General Normal Form Definitions

1. Introduction (1)

- We have seen how to map EER diagrams to relational database schemas.
- However, we still need some formal measure of why one grouping of attributes into a relation schema may be better than another.
- There are two levels at which we can discuss the "goodness" of relation schemas:
  - The logical (or conceptual) level – how users interpret the relation schemas and the meaning of their attributes, and
  - The implementation (or storage) level – how the tuples in a base relation are stored and updated.

1. Introduction (2)

- We first informally discuss some criteria for good and bad relation schemas.
- We define the concepts of Functional Dependency, (a formal constraint among attributes), and of other dependencies. Functional dependencies is the main tool for formally measuring the appropriateness of attribute grouping into relation schemas.
- We show how dependencies can be used to group attributes in relation schemas that are in a normal form.

  A relation schema is in a normal form when it satisfies certain desirable properties. Higher normal forms meet increasingly more stringent requirements.
- The process of normalization consists of analyzing relations to relations in higher normal forms.
- We discuss properties of sets of relation schemas that form a relational database schema: lossless join and dependency preservation.

2. Informal Design Guidelines for Relational Databases

We present informal measures of quality of a relation schema design based on:

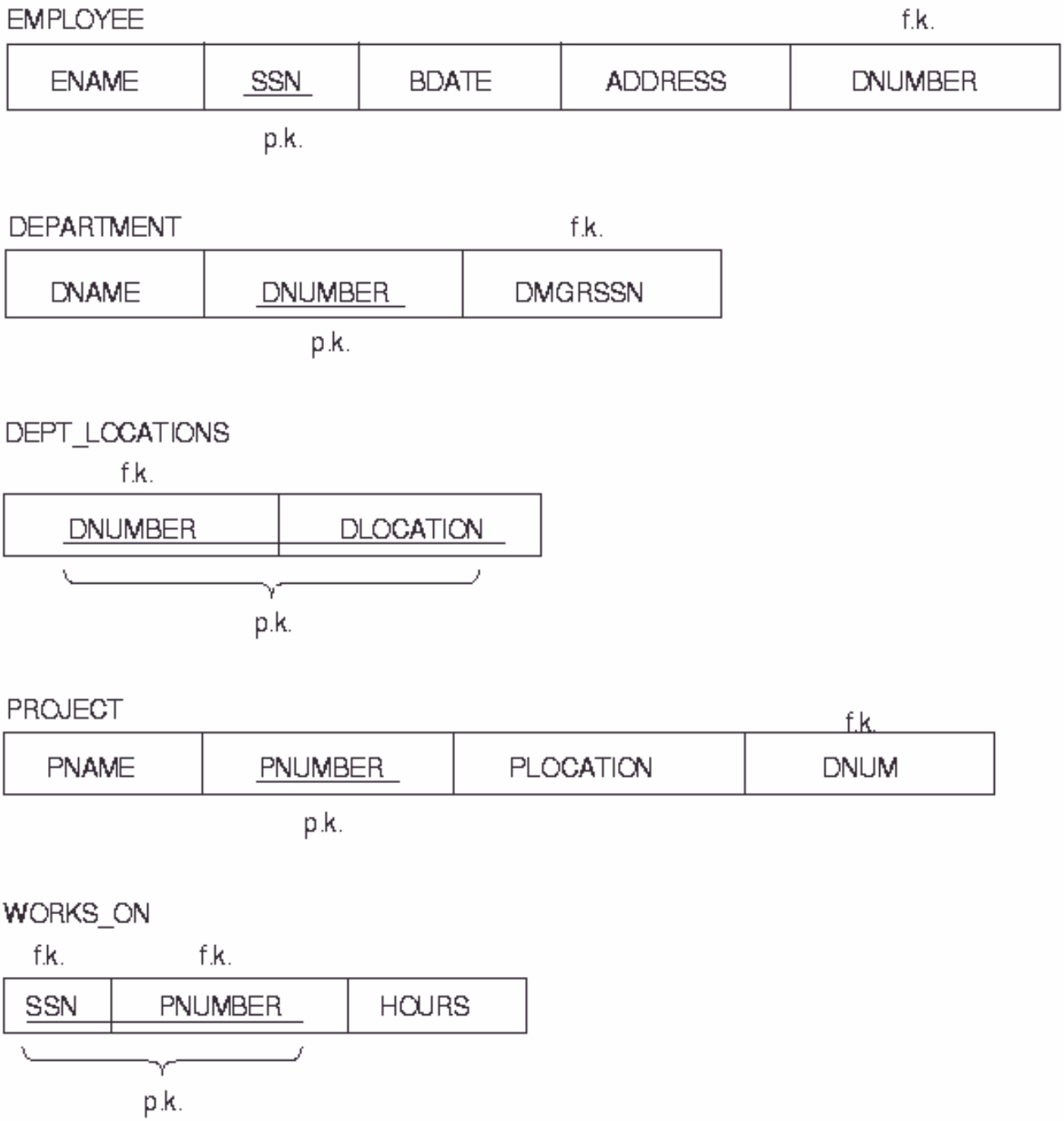
1. Semantics of attributes.

2. Null values in tuples.

Later we will see formal measures of quality of a relation schema based on normal forms.

2.1 Semantics of Relation Attributes (1)

Figure 14.1 Simplified version of the COMPANY relational database schema.



2.1 Semantics of Relation Attributes (2)

- **GUIDELINE:** Design a schema that can be explained easily relation by relation. The semantics of attributes should be easy to interpret.

2.2 Null values in tuples (1)

- Many NULL values for an attribute in a relation pose problems:
  - space waste at the storage level
  - not clear meaning for the attribute
    (There are different reasons for nulls:
      a. attribute not applicable
      b. attribute value unknown (may exist)
      c. value known to exist, but unavailable)
  - difficulty in accounting for NULLs when aggregate functions such as COUNT or SUM are applied.

- **GUIDELINE:** Relations should be designed such that their tuples will have as few NULL values as possible.

2.2 Null values in tuples (2)

- Attributes that are NULL frequently could be placed in separate relations (with the primary key)

- Example:

  If only 10% of the employees have individual offices, instead of including attribute OFFICE_NUMBER in the EMPLOYEE relation, the relation

  EMP_OFFICES(ESSN, OFFICE_NUMBER) can be created.

3. Functional dependencies

- The concept of the *Functional Dependency (FD)* is the single most important concept in relational schema design.

- A FD is a *constraint* between two sets of attributes from the database.

- FDs (and keys) are used to define *normal forms* for relations.

3.1. Definition of FDs (1)

- Consider a "universal relation" R that contains all the attributes in a database. Let $X, Y \subseteq R$ (X and Y are subsets of R).

  A **Functional Dependency (FD)**, denoted $X \rightarrow Y$, in R specifies a *constraint* on all relation instances r(R).

- An instance r(R) **satisfies** the FD $X \rightarrow Y$ iff any two tuples t_1 and t_2 in r that have $t_1[X] = t_2[X]$, they also have $t_1[Y] = t_2[Y]$.

- That is, the values of the Y component of a tuple in r *depend on*, or *are determined by*, the values of the X component.

  Alternately, the values of the X component of a tuple *uniquely, (or functionally) determine* the values of the Y component.

- An instance r(R) **violates** the FD $X \rightarrow Y$ if and only if there are two tuples t_1 and t_2 in r that have $t_1[X] = t_2[X]$, and $t_1[Y] \neq t_2[Y]$.

3.1. Definition of FDs (2)

- We say that a FD $X \rightarrow Y$ **holds** on a relation schema R, if it is specified on this schema.

  In this case we also say that **there is a FD from X to Y in R**, or that **Y is functionally dependent on X in R**.

- An instance of a relation is called **legal** if it satisfies *all the FDs that hold on this schema*.

- Remark:

  A FD is a property of a relation schema.
  It is not a property of a particular relation instance.

  A FD *cannot be inferred from a given relation instance*. It is *derived from the semantics or meaning of the attributes*.

3.1. Definition of FDs (3)

Examples of FD constraints:

- *Social security number* determines *employee name* in rel. schema EMPLOYEE

  SSN $\rightarrow$ ENAME

- *Project number* determines *project name and location* in rel. schema PROJECT

  PNUMBER $\rightarrow$ {PNAME, PLOCATION}

- *Employee ssn and project number* determine *the hours per week that the employee works on the project* in rel. schema WORKS_ON.

  {SSN, PNUMBER} $\rightarrow$ HOURS

3.1. Definition of FDs (4)

- Example:

TEACH

TEACHER	COURSE	TEXT
Smith	Data Structures	Bartram
Smith	Data Management	Al-Nour
Hall	Compilers	Hoffman
Brown	Data Structures	Augenthaler

- This relation instance satisfies the FDs

  {TEACHER, TEXT} → COURSE
  TEXT → TEACHER

  However, we cannot deduce that these FDs hold on TEACH.

- This relation instance violates the FDs

  TEACHER → COURSE
  COURSE → TEXT
  TEACHER → TEXT

  (If this is a legal instance of relation TEACH, we can deduce that TEACHER → COURSE does not hold on TEACH.

3.1. Definition of FDs (5)

- If K is a *(candidate or super-) key* of R, then K functionally determines all attributes in R (since we never have two distinct tuples t_1, t_2 in R with $t_1[K] = t_2[K]$).

  That is, for every $X \subseteq R$, $K \rightarrow X$

- $X \rightarrow Y$ holding in R, does not necessarily mean that $Y \rightarrow X$ also holds in R.

3.2. Inference rules for FDs (1)

- Given a set of FDs F, we can *infer* additional FDs that hold whenever the FDs in F hold.

- Example:
  Let F = {
SSN → {ENAME,BDATE,ADDRESS, DNUMBER},
DNUMBER → {DNAME,MGRSSN} }

  We can infer the following additional FDs from F:
  SSN → SSN
  DNUMBER → DNAME
  SSN → {DNAME,MGRSSN}

- A FD X → Y **is inferred from** a set of FDs F specified on R iff X → Y is satisfied by every instance that satisfies the FDs in F.

- If X → Y is inferred from F, we write F |= X → Y

- The *set of all FDs that can be inferred from F*, denoted **F^+**, is called **closure** of F.

- That is, $X \rightarrow Y \in F^+$ iff $F \models X \rightarrow Y$

3.2. Inference rules for FDs (2)

- To determine a systematic way to infer dependencies, we use a set of inference rules to infer new FDs from a given set of FDs.

Inference rules:

IR1. (Reflexive) If $X \supseteq Y$, then $X \rightarrow Y$

IR2. (Augmentation) $X \rightarrow Y \models XZ \rightarrow YZ$

IR3. (Transitive) $\{ X \rightarrow Y, Y \rightarrow Z \} \models X \rightarrow Z$

IR4. (Decomposition) $\{ X \rightarrow YZ \} \models X \rightarrow Y$

IR5. (Union) $\{ X \rightarrow Y, X \rightarrow Z \} \models X \rightarrow YZ$

IR6. (Pseudotransitivity) $\{ X \rightarrow Y, WY \rightarrow Z \} \models WX \rightarrow Z$

- IR1 generates dependencies that are always true.
  A FD $X \rightarrow Y$ is **trivial** if $X \supseteq Y$.
  Otherwise, it is **non-trivial**.

- IR4 can be used to decompose the FD
  $X \rightarrow A_1 \ldots A_n$ into a set of FDs
  $\{X \rightarrow A_1, \ldots, X \rightarrow A_n\}$.

- IR5 can be used to combine a set of FDs
  $\{X \rightarrow A_1, \ldots, X \rightarrow A_n\}$ into a single FD
  $X \rightarrow A_1 \ldots A_n$.

3.2. Inference rules for FDs (3)

- Rules IR1, IR2, IR3 are known as **Armstrongs Inference Rules.**

- Armstrong showed that these rules are **sound** and **complete**.

- **Sound** means that, given a set of FDs F specified on a relational schema R, any FD that can be produced from F by using IR1, IR2 and IR3 is also satisfied by every state r of R that satisfies the FDs in F.

- **Complete** means that any FD that can be inferred from F can also be produced using the inference rules IR1, IR2 and IR3.

- IR1, IR2, and IR3 can be proved from the definition of FDs, either by direct proof of by contradiction.
  IR4, IR5, IR6 (and other inference rules) can be proved using IR1, IR2, and IR3 (completeness property).

3.3. Checking inference of a FD (1)

- In order to provide a systematic way to check FD inference we introduce the concept of *closure of a set of attributes*.

- The **closure of a set of attributes X with respect to F** is the set X^+ of all attributes that are functionally determined by X.

- **Algorithm**. Determining X^+ under F.
  Input: X, F.
  Output: X^+ under F

  $X^+ := X$;
  repeat
  $oldX^+ := X^+$;
  for each FD $Y \rightarrow Z$ in F do
  if $Y \subseteq X^+$ then $X^+ := X^+ \cup Z$;
  until $(X^+ = oldX^+)$;

3.3. Checking inference of a FD (2)

- Example:

$F = \{ SSN \rightarrow ENAME,$
$PNUMBER \rightarrow \{PNAME, PLOCATION\},$
$\{SSN, PNUMBER\} \rightarrow HOURS \}$

Using the previous algorithm,

$\{SSN\}^+ = \{ SSN, ENAME\},$
$\{PNUMBER\}^+ = \{PNUMBER, PNAME, PLOCATION\}$
$\{SSN, PNUMBER\}^+ = \{SSN, PNUMBER, ENAME, PNAME, PLOCATION, HOURS\}$

3.3. Checking inference of a FD (3)

- **Proposition:** $F \models X \rightarrow Y$ iff $Y \subseteq X^+$.

  We can use the previous proposition to check FD inference from a set of FDs.

- Example (cont.):

  Is the FD SSN, PNUMBER $\rightarrow$ ENAME inferred from F?

  YES since {ENAME} $\subseteq$ {SSN, PNUMBER}$^+$

  Similarly, we can see that

  $F \models$ SSN, PNUMBER $\rightarrow$ SSN, PNUMBER

- **Corollary:** If $X \rightarrow A$ is a non-trivial FD and A does not appear in the RHS of any FD in F, $F \not\models X \rightarrow A$.

3.4. Equivalent sets of FDs (1)

- F and G are sets of FDs.

- F is **covered by** G – or alternatively,
  G covers F – iff
  for every FD $X \rightarrow Y$ in F, G $\models X \rightarrow Y$.

- If F is covered by G, we write $G \models F$.

- Clearly, $G \models F$ iff $G^+ \supseteq F^+$

- F and G are **equivalent** iff
  G covers F, and F covers G

- If F and G are equivalent, we write $G \equiv F$.

- Clearly, $G \equiv F$ iff $G \models F$ and $F \models G$ iff $G^+ = F^+$

3.4. Equivalent sets of FDs (2)

- We can determine whether F |= G by computing X^+ w.r.t. F for each FD $X \rightarrow Y$ in G, and then checking whether $Y \subseteq X^+$.

- We can determine whether F and G are equivalent by checking whether F |= G and G |= F.

3.5. Minimal sets of FDs (1)

- A set of FDs is **minimal** if it satisfies the following conditions:
  (1) Every FD in F has a single attribute in its right-hand side.
  (2) We cannot replace any FD $X \rightarrow A$ in F with a FD $Y \rightarrow A$, where Y is a proper-subset-of X ($Y \subset X$) and still have a set of FDs that is equivalent to F.
  (3) We cannot remove any FD from F and have a set of FDs that is equivalent to F.

- A minimal set of FDs is a set of FDs in *standard*, or *canonical* form and has no *redundancies*.

- Every set of FDs has an equivalent minimal set.

- A **minimal cover** of a set F of FDs is a minimal set of FDs that is equivalent to F.

- There can be several minimal covers of a set of FDs.

3.5. Minimal sets of FDs (2)

- The following algorithm computes a minimal cover of a set of FDs.

Algorithm

Input: a set of FDs F

Output: a minimal cover G for F.

1. Set G := F
2. Replace each FD $X \rightarrow \{A_1, \ldots, A_n\}$ in G by the n FDs $X \rightarrow A_1, \ldots, X \rightarrow A_n$.
3. For each FD $X \rightarrow A$ in G
   for each attribute B in X
   if $(G - \{X \rightarrow A\}) \cup \{(X - \{B\}) \rightarrow A\} \equiv G$
   then replace $X \rightarrow A$ with $(X - \{B\}) \rightarrow A$ in G.
4. For each remaining FD $X \rightarrow A$ in G,
   if $G - \{X \rightarrow A\} \equiv G$
   then remove $X \rightarrow A$ from G.

One can see that:

$(G - \{X \rightarrow A\}) \cup \{(X - \{B\}) \rightarrow A\} \equiv G$ iff
$G \models (X - \{B\}) \rightarrow A$, and

$G - \{X \rightarrow A\} \equiv G$ iff
$G - \{X \rightarrow A\} \models X \rightarrow A$

3.6. Keys and FDs (1)

- Given a set F of FDs on a relation schema R, a set of attributes $X \subseteq R$ is a **superkey of R w.r.t. F** iff $F \models X \rightarrow R$.

  X is a **(candidate) key of R w.r.t. F** if it is a superkey of R and no *proper subset* of X is a superkey of R w.r.t. F.

- When F is well understood we do not mention it.

3.6. Keys and FDs (2)

- The previous definitions suggest a straightforward algorithm for determining whether a set of attributes is a key of a relation w.r.t. a set of FDs.

- **Algorithm**

  Input: a rel. schema R, a set of FDs F on R, a set of attributes X.

  Output: a decision as to whether X is a key of R (w.r.t. F).

  if $F \not\models X \rightarrow R$ then stop (return X is not a key)
  else
  for every $Y = X - A, A \in X$ do
  if $F \models Y \rightarrow R$ then stop (return X is not a key);
  return X is a key;

- Notice that we do not need to check whether all the proper subsets of X are superkeys of R. It suffices to check only the sets $Y = X - A, A \in X$. Why?

3.6. Keys and FDs (3)

- Example
  R(A, B, C, D, E).
  F ={ AB→ C, CD→ E, DE→ B }

- Is AB a key of R?

  NO since F|≠ AB → R. (Thus, AB is not even a superkey of R)

- Is ABCD a key of R?

  ABCD is a superkey of R since F |= ABCD → R.
  However, it is *not* a key of R since
  F |= ABD → R, and ABD = ABCD – C

- Is ABD a key of R?

  YES since F |= ABD → R
  F |≠ AB → R (AB = ABD – D)
  F |≠ AD → R (AD = ABD – B)
  F |≠ BD → R (BD = ABD – A)

  Notice that we do not need to consider the other proper subsets of ABD: A, B, D.

3.6. Keys and FDs (4)

- The following algorithm computes *one key (out of the possibly many candidate keys) for R*.

- **Algorithm**
  Input: a rel. schema R, a set of FDs F on R
  Output: a key K of R w.r.t. F.

  set K := R;
  for each attribute A in K
    {compute $(K - A)^+$ w.r.t. F;
     if $(K - A)^+ = R$, then set K := K – {A}};

- An improvement of this algo can be obtained by observing that: *all the attributes of R that are not in the RHS of any non-trivial FD in F are part of any key of R* (why?).
  Therefore, we don't need to check in the algorithm whether these attributes can be removed from K (they cannot).

3.6. Keys and FDs (5)

- Example

  R(A, B, C, D, E).

  F ={ AB→ C, CD→ E, DE→ B }

  Apply the previous algorithm to find *a* key of R.

  We consider the attributes in the order A,B,C,D,E.

  Initially, K = ABCDE.

  We try to remove A from K:

  $(BCDE)^+ = BCDE \neq R$. We cannot remove A.

  We try to remove B from K:

  $(ACDE)^+ = R$. We can remove B.

  K := K – B =ACDE.

  We try to remove C from K:

  $(ADE)^+ = R$. We can remove C.

  K := K – C =ADE.

  We try to remove D from K:

  $(AE)^+ = AE \neq R$. We cannot remove D.

  We try to remove E from K:

  $(AD)^+ = AD \neq R$. We cannot remove E.

  Therefore, ADE is a key of R.

3.6. Keys and FDs (6)

- For computing *all the keys of R* w.r. t. a set of FDs F observe that:

  1. All the attributes of R that are not in the RHS of any non-trivial FD in F are part of any key of R. Let X be the set of these attributes (clearly, this set can be empty).

  2. The set of attributes in R – X that are functionally determined by X do not appear in any key of R.

  3. If X is a superkey, it is the unique key of R.

  4. In general, if a set of attributes Y is a key of R, we do not need to examine if a proper superset of Y is a key of R (it is not).

3.6. Keys and FDs (7)

- Example

  R(A, B, C, D, E).

  F ={ AB→ C, CD→ E, DE→ B }

  Find *all* the keys of R (w.r.t. F).

  Attributes A and D do not appear in the RHS of any FD. Every key of R contains AD.

  We need to examine if the following sets are keys.

  AD

  ABD ACD ADE

  ABCD ABDE ACDE

  ABCDE

  We first examine if AD is a key of R:

  $(AD)^+ = AD \neq R$. Therefore, AD is not a key of R.

  ~~AD~~

  ABD ACD ADE

  ABCD ABDE ACDE

  ABCDE

3.6. Keys and FDs (8)

- Example (cont.)
  We examine if the sets ABD, ACD, and ADE (obtained by adding one attribute to AD) are keys:

  (ABD)+ = R. Therefore, ABD is a key. We do not need to examine the sets ABCD, ABDE, and ABCDE: they are proper supersets of ABD and thus are not keys.

  ~~AD~~
  ABD ACD ADE
  ~~ABCD~~ ~~ABDE~~ ACDE
  ~~ABCDE~~

  (ACD)+ = R. Thus, ACD is a key of R. We do not need to examine the set ACDE. It is a proper superset of ACD and thus is not a key.

  ~~AD~~
  ABD **ACD** ADE
  ~~ABCD~~ ~~ABDE~~ ~~ACDE~~
  ~~ABCDE~~

3.6. Keys and FDs (9)

- Example (cont.)

  (ADE)+ = R. Therefore, ADE is a key.

  ~~AD~~
  ABD **ACD** **ADE**
  ~~ABCD~~ ~~ABDE~~ ~~ACDE~~
  ~~ABCDE~~

  We conclude that the keys of R are ABD, ACD, and ADE.