Functional dependency in relational databases

(adapted after M.Y.Vardi’s survey)

K.V.Iyer
Computer Science and Engineering
National Institute of Technology
Trichy – 620015
Contents

1 Introduction 1

2 Preliminaries 2
   2.1 Projection and join 2

3 Functional dependency 3
   3.1 Formal system for functional dependencies 5
   3.2 Computing the closure 7
   3.3 Covers 8

4 Database schema design 9
   4.1 BCNF 9
   4.2 Normalization via decomposition 10
      4.2.1 Checking for BCNF violations 11
   4.3 3NF and normalization via synthesis 13

5 Concluding remarks 14
**NOTE 1:** These are the first version of suggested lecture notes for a second level course on advanced topics in database systems for master’s students of Computer Science. A prerequisite in algorithms and an exposure to database systems are required. Additional reading may require exposure to mathematical logic which seems to be getting outdated going by newer editions of Computer Science theory books.

**NOTE 2:** These notes are from M.Y.Vardi’s survey listed as reference [1]. This select rewrite on functional dependency is intended to provide a few clarifications while avoiding topics relating to logic.
1 Introduction

Manipulation of a large store of structured information has been the fundamental requirement in many computer-based applications which has evolved into database systems and has promoted the associated technologies in the West. A database management system is now understood to be a computer-based system maintaining a large amount of permanent data appertaining to a real-world organization/institution together with mechanisms to search, add, update, delete data and with mechanisms for administrative control such as granting/revoking privileges, defining views for restricted access and archiving. In the business domain relational database story has been a success due to cost-effectiveness and the support of a sound formalism in organizing and managing structured data. IBM’s pioneering implementation during the 1970s of these concepts was System R. As apparent from the contemporary literature relational databases still form the core in most of the database management systems since their introduction more than three decades before.

Conceived in the late 1960’s, the relational model of databases views a database as a collection of relations where each relation is a set of well-defined tuples. A relation is synonymous with a table whose columns are named by attributes. This notion of databases apparent from the work of E.F.Codd in 1970’s is founded on the following two principles:

(a) All information pertaining to an application are captured as data values in relations or tables.

(b) No information is represented by ordering of columns or rows of any table.

Searching, adding, deleting and updating of data is effected by manipulations of relations by relational algebra having a procedural flavor or by relational calculus having a declarative flavor. Codd’s theorem states that any relational algebra expression can be converted efficiently to an equivalent relational calculus expression and vice versa. Here efficiency is interpreted to mean that there exists a conversion algorithm whose running time is bounded by a polynomial in the size of the input expression. The relational model is almost devoid of semantics. Therefore meaningful relations in a given context are understood by specifying semantic or integrity constraints. In particular the notion of functional dependency introduced by Codd in 1972 is of significance in practice and the related notion of implication apparent from the work of P.A.Bernstein in now considered fundamental. Subsequently in 1976 multivalued dependency was introduced by R. Fagin and C. Zaniolo independently, opening up the way for data dependency
analyses. Much of theory about the relational model during the 1970’s and first half of 1980’s focussed on query processing and optimization and database design including dependency analysis (see [9]). Another studied problem relating to integrity constraints that allow permissible data in relation instances is this: assuming that the current data values satisfy the constraints before an update, how to efficiently check (or decide that no checking is necessary) that the constraints hold after the update?

This condensed survey is essentially a select rewrite of M.Y.Vardi’s survey [7] incorporating more explanations as needed. Further pertinent technical details, associated concepts such as other types of dependencies and their significance in the real-world, intractability results, relationships to mathematical logic and combinatorial problems such as constraint satisfaction and many original references can be found in the database literature from the 1970’s (see for example [4, 7, 8, 9]).

2 Preliminaries

In the sequel, it is understood that the implicit context is a given real-world application. We use $I, J, K, \ldots$ to denote the different tables that together form a database. The set $X$ of all attributes in a relation $I$ is referred to as a relation scheme. We then say $I$ is defined over $X$. By convention we denote by $U$ the set of all attributes occurring across all tables comprising the database. The headers $A, B, C, \ldots$ denote attributes. The tailenders $R, S, \ldots, X, Y, Z$ denote sets of attributes. For the sake of convenience we make no distinction between $\{A\}$ and $A$. Given the attribute sets $X$ and $Y$, $XY$ will denote $X \cup Y$; $ACE$ is a shorthand for $\{A, C, E\}$. Associated with each column of a table is a domain of values from which the entries are taken in the column. That is, for each attribute we have a finite or infinite set $\text{Dom}(A)$. In the relational model the elements of $\text{Dom}(A)$ are assumed to be atomic in the usual sense. We denote $\text{DOM} = \bigcup_j \text{Dom}(A_j)$ where $U = \{A_1, \ldots, A_n\}$. We recall that by tuple we mean a row in a table, that specifies an appropriate value for each attribute. By $u, v, w, \ldots$ we denote tuples. Given an attribute set $X$, a tuple $u$ on $X$ is then a mapping $u : X \rightarrow \text{DOM}$ such that for each $A, u(A) \in \text{Dom}(A)$. $|X|$ can be as large as the arity or the total number of attributes in the considered relation. If $v$ is a tuple on $X$ then $v[Y]$ will mean the restriction of $v$ to $Y$ where $Y \subset X$. We take $v[X] = v$.

2.1 Projection and join

Let $X$ be a relation scheme and $I$ a relation on $X$. Given $Y \subset X$ we define the projection $\pi_Y(I)$, a relation as $\pi_Y(I) = \{w[Y] | w \in I\}$. Let $I_1, \ldots, I_k$ be relations on $X_1, \ldots, X_k$ and let $X = \bigcup_{j=1}^k X_j$. Then we define the join $I_1 \bowtie \ldots \bowtie I_k$ abbreviated to $\bowtie_{j=1}^k I_j$ as $\bowtie_{j=1}^k I_j = \{w[X] | w[X_j] \in I_j \forall j\ 1 \leq j \leq k\}$.

Projection builds a new relation from a given one by selecting one or more attributes. Join combines tuples from two or more relations when they agree on common columns.

Mar. 2011 2
Join is commutative as well as associative. In some sense projection and join are duals. Beginning with the following relations the examples below illustrate that projection and join cannot always be regarded as inverses.

<table>
<thead>
<tr>
<th>I:</th>
<th>J:</th>
<th>K:</th>
<th>L:</th>
<th>M:</th>
<th>N:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-------</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-------</td>
<td>----</td>
<td>----</td>
</tr>
</tbody>
</table>

We have \( \pi_{AB}(I) \bowtie \pi_{BC}(I) = I \) and \( \pi_{AB}(J) \bowtie \pi_{BC}(J) = I \). Also \( K \bowtie L = M \) while \( \pi_{AB}(M) = N \) and \( \pi_{BC}(M) = L \).

Generalization to the following lemma is immediate.

**Lemma 1:**

(a) Let \( I \) be a relation on \( X \). Let \( X_1, \ldots, X_m \) be attribute sets such that \( X = \bigcup_{j=1}^{m} X_j \). Then \( I \subseteq \bowtie_{j=1}^{m} \Pi_{X_j}(I) \).

The simultaneous projection of \( I \) onto \( X_1, \ldots, X_m \) is referred to as a *decomposition*. The decomposition is *lossless* when \( I = \bowtie_{j=1}^{m} \Pi_{X_j}(I) \); otherwise it is *lossy*.

(b) Let \( I_1, \ldots, I_m \) be relations on \( X_1, \ldots, X_m \) respectively. Then \( \Pi_{X_j}(\bowtie_{k=1}^{m} I_k) \subseteq I_j \).

**Remark:** It is appropriate to interpret lemma 1 for legal relations i.e., when the relations are meaningful w.r.t. a given real-world scenario. More details follow.

### 3 Functional dependency

The presence of redundancy and anomalies in instances of relations and the natural requirement to do away with them has motivated the dependency theory of relational databases and hence database design. We first recall what are commonly referred to as Codd’s anomalies in a relation by considering the following example.

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>DEPARTMENT</th>
<th>SUPERVISOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Cryptology</td>
<td>John</td>
</tr>
<tr>
<td>Bob</td>
<td>Cryptology</td>
<td>John</td>
</tr>
<tr>
<td>Carol</td>
<td>Graph Theory</td>
<td>Yohann</td>
</tr>
<tr>
<td>Darrel</td>
<td>Graph Theory</td>
<td>Yohann</td>
</tr>
<tr>
<td>Frank</td>
<td>Graph Theory</td>
<td>Yohann</td>
</tr>
</tbody>
</table>

The cited problems are:

Mar. 2011 3
(a) **Redundancy**: That John is a supervisor for Cryptology or Yohann is a supervisor for graph theory can get repeated in many tuples.

(b) **Potential inconsistency**: In the graph theory department if Carol gets a new supervisor does it mean that the department gets two supervisors or is the intended meaning to change the supervisor to the new supervisor for all students in graph theory department?

It stands to reason that there is a functional dependency between **DEPARTMENT** and **MANAGER**—this is a kind of semantic constraint on the data that comprise legal relations. In this case we say that **DEPARTMENT** determines **SUPERVISOR** and write $$\text{DEPARTMENT} \rightarrow \text{SUPERVISOR}$$.

In formal terms, for attribute sets $X, Y$, $X \rightarrow Y$ is a functional dependency (fd) over a relation scheme $R$ i.e., $XY \subseteq R$ if for all tuples $u, v \in I$ $u[X] = v[X] \implies u[Y] = v[Y]$, where $I$ is any legal relation on $R$. We say $I$ satisfies a set of fd’s $\Sigma$ if $I$ satisfies all fd’s in $\Sigma$.

Two notions namely equivalence and redundancy of fd’s are useful in the context of manipulating fd’s mechanically. Two sets of fd’s $\Delta$ and $\Sigma$ are equivalent written $\Sigma \equiv \Delta$ if they are precisely satisfied by the same set of legal relations. In other words $I$ satisfies $\Delta \iff I$ satisfies $\Sigma$. A set $\Sigma$ of fd’s is redundant written $\Sigma$ if there is a $\Delta \subset \Sigma$ such that $\Delta \equiv \Sigma$. These can be expressed by a more fundamental notion viz., implication. We write $\Sigma \models \sigma$ to say that a set $\Sigma$ of fd’s implies an fd $\sigma$. By this we mean that any relation that satisfies $\Sigma$ necessarily satisfies $\sigma$. For example $\{B \rightarrow C, A \rightarrow B\} \models A \rightarrow C$. The database implication (or inference) problem for fds is: given $\Sigma$ that necessarily holds for any legal instance of a database and given any $\sigma$ does $\Sigma \models \sigma$?

In terms of implication redundancy and equivalence can be stated as

- $\Sigma$ is redundant iff there is an fd $\sigma \in \Sigma$ such that $\Sigma - \{\sigma\} \models \sigma$.
- $\Delta \equiv \Sigma$ iff $\Delta \models \sigma$ for any $\sigma \in \Sigma$ and $\Sigma \models \delta$ for any $\delta \in \Delta$.

**Remarks:**

1. If $X \cap Y = \phi$ then the fd $X \rightarrow Y$ may be interpreted as a case of multivalued dependency (see [4] for example).
2. Let $I$ be any relation on $R$. Then an fd $X \rightarrow Y$ where $XY \subset R$ is satisfied by $I$ iff $\pi_{XY}(I)$ satisfies $X \rightarrow Y$.
3. It has been shown that fds can be interpreted as formulas in propositional calculus. To this end it is sufficient to interpret an fd like $A_1 \ldots A_k \rightarrow B$ as an equivalent Horn formula and then take note of the fact that for Horn formulas satisfiability can be tested in polynomial-time. In turn this implies the existence of a polynomial-time algorithm for the implication problem.

In the above finite as well as infinite relations are allowed though in practice we need to consider only finite relations. Written as $\Sigma \models_f \sigma$, $\Sigma$ finitely implies $\sigma$ if any finite
relation $I$ that satisfies $\Sigma$ satisfies $\sigma$ as well. Surely if $\Sigma \models \sigma$ then $\Sigma \models_f \sigma$.

**Fact 1:** Implication and finite implication coincide for fds. An fd $X \rightarrow Y$ is said to be *reduced* if there is no proper subset $W \subset X$ such that $\Sigma \models W \rightarrow Y$. We say $\Sigma$ is *reduced* if every fd in it is reduced. The algorithm that follows outputs a reduced equivalent to a given set $\Sigma$ of fds.

**Algorithm** $\text{REDUCED}(\Sigma)$

begin
$\Delta \leftarrow \Sigma$
for (each fd $X \rightarrow Y$ in $\Delta$) do
  for (each attribute $A$ in $X$) do
    if $\Delta \models X - A \rightarrow Y$
    then $X \leftarrow X - A$ in $X \rightarrow Y$
return $\Delta$
end

Algorithm $\text{REDUCED}(\Sigma)$ depends on a test for implication of fds which determines its complexity.

### 3.1 Formal system for functional dependencies

A formal system for fds comprises a set of axioms and inference rules – this was first studied by W.W. Armstrong in 1974 when the significance of implication was not apparent. Armstrong’s system denoted by $\text{F-A}$ consists of one axiom and three inference rules.

- **FDA0:** (Reflexivity) $\vdash X \rightarrow X$.
- **FDA1:** (Transitivity) $X \rightarrow Y, Y \rightarrow Z \vdash X \rightarrow Z$.
- **FDA2:** (Augmentation and projection) $X \rightarrow Y \vdash W \rightarrow Z$ if $X \subseteq W$ and $Z \subseteq Y$.
- **FDA4:** (Union) $X \rightarrow Y, Z \rightarrow W \vdash XZ \rightarrow YW$.

Describing $\text{F-A}$ requires the notion of a derivation – a derivation of $\sigma$ from $\Sigma$ is denoted by $\Sigma \vdash \sigma$. In a formal system such as $\text{F-A}$, given a set $\Sigma$ of fds and an fd $\sigma$, by a *derivation* of $\sigma = \sigma_1, \ldots, \sigma_n$ we mean: each $\sigma_i$ ($1 \leq i \leq n$) is either an instance of an axiom scheme or it follows from the preceding dependencies in the sequence by one of the inference rules. Soundness and completeness of any system like $\text{F-A}$ are expressed as

(1) $\text{F-A}$ is *sound* if $\Sigma \models \sigma$ is a necessary consequence of $\Sigma \vdash \sigma$.

(2) $\text{F-A}$ is *complete* if $\Sigma \vdash \sigma$ is a necessary consequence of $\Sigma \models \sigma$. 
The formal system **FD** given below is as in [7]. **FD** consists of FD1, FD2 and FD3.

- **FD1:** (Reflexivity) \( \vdash X \rightarrow \phi \).
- **FD2:** (Transitivity) \( X \rightarrow Y, Y \rightarrow Z \vdash X \rightarrow Z \).
- **FD3:** (Augmentation) \( X \rightarrow Y \vdash XZ \rightarrow YZ \).

**Theorem 1:** **FD** is sound and complete.

**Proof:** Soundness and completeness are proved separately as given below.

**Soundness:** As FD1 is vacuously true it is sufficient to show that individually FD2 and FD3 are sound. Let \( I \) be a relation on \( R \), let \( X, Y, Z \) be drawn from \( R \) and \( u, v \in I \) be any two tuples. First, considering FD2, assume that \( I \) satisfies \( X \rightarrow Y \) and \( Y \rightarrow Z \). Then (using the definition of fd) if \( u[X] = v[X] \) since \( u[Y] = v[Y] \) we have \( u[Z] = v[Z] \). That is \( I \) satisfies \( X \rightarrow Z \) and hence FD2 is sound. Next, considering FD3, assume that \( I \) satisfies \( X \rightarrow Y \). If \( u[XZ] = v[XZ] \) then \( u[X] = v[X] \) and \( u[Z] = v[Z] \). As \( I \) satisfies \( X \rightarrow Y \) we have \( u[Y] = v[Y] \) and so \( u[ZY] = v[ZY] \). In other words \( I \) satisfies \( XZ \rightarrow YZ \) and thus FD3 is sound.

**Completeness:** Let \( \Sigma \) be the set of fds that any legal relation \( I \) on \( R \) needs to satisfy. If \( \sigma = X \rightarrow Y \) be any given fd, completeness requires that if \( \Sigma \vdash \sigma \) then FD1, FD2 and FD3 are sufficient to conclude \( \Sigma \vdash \sigma \). This amounts to proving the contrapositive viz., if \( \Sigma \not\vdash \sigma \) then \( \Sigma \not\vdash \sigma \). In the context \( \Sigma \) we define the closure \( X^+ \) of \( X \) as \( X^+ = \{ A \mid \Sigma \vdash X \rightarrow A \} \). By FD1, \( \vdash X \rightarrow \phi \); now invoking FD2 and FD3 taking \( Z \) as any \( A \in X \) we have \( \vdash X \rightarrow A \). Hence \( X \subset X^+ \). If \( X^+ = X \) then any relation \( I \) that satisfies \( \Sigma \) clearly satisfies any given fd \( X \rightarrow Y \). Hence we need to consider the case where \( X \subset X^+ \). The following holds due to FD2 and FD3: \( W \rightarrow Z_1, W \rightarrow Z_2 \vdash W \rightarrow Z_1 Z_2 \). Repeated use of this yields \( \Sigma \vdash X \rightarrow X^+ \) since \( \Sigma \vdash X \rightarrow A \) by definition, for all \( A \in X^+ \) and since \( X \subset X^+ \). If as assumed \( \Sigma \not\vdash \sigma \) then we claim that \( Y \not\subset X^+ \). If possible let the contrary hold viz., \( Y \subset X^+ \). In such a case we can use FD1 and FD3 to show that \( \Sigma \vdash X^+ \rightarrow Y \). Combining this with the fact \( \Sigma \vdash X \rightarrow X^+ \) by virtue of FD2 we get \( \Sigma \vdash X \rightarrow Y \) which is a contradiction. Therefore there exists some \( B \in R \) such that \( B \in Y \) but \( B \not\in X^+ \). We construct a specific legal relation \( I \) consisting of two tuples \( u, v \). We prescribe that \( u[A] = v[A] \) iff \( A \in X^+ \). In more details let \( u[A] = \alpha \) for all \( A \in R \) and \( v[A] = \beta \) for any \( A \in R \) \( X^+ \). As \( X \subset X^+ \) \( u[X] = v[X] \) but by construction \( u[Y] \neq v[Y] \). So in \( I \) \( X \not\rightarrow Y \). We now show that \( I \) satisfies \( \Sigma \). Let \( S \rightarrow T \) be any fd in \( \Sigma \). Indeed if \( I \) is legal then if \( u[S] = v[S] \) then we should be able to show \( u[T] = v[T] \). So suppose that \( u[S] = v[S] \). Then by a previous argument \( S \subset X^+ \).

Using FD1 and FD3 \( S \rightarrow S \vdash S \). By the assumption \( S \rightarrow T \), with FD1 we can conclude \( S \rightarrow T \rightarrow X^+ \rightarrow T \). For any \( A \in T \) by FD1 \( T \rightarrow A \). Then by FD2 it follows that for all \( A \in T \) \( X^+ \rightarrow A \) which implies that \( \Sigma \vdash X^+ \rightarrow T \). Therefore \( T \subset X^+ \) and by construction \( u[T] = v[T] \). Therefore \( I \) satisfies \( \Sigma \). Thus \( I \) is one relation that satisfies \( \Sigma \) but it does not satisfy \( X \rightarrow Y \) i.e., \( \Sigma \not\vdash \sigma \).

**Remarks:**
1. The counter-example constructed in the proof above is finite. It then follows that in
the case of fds implication and finite implication coincide.

2. The polynomial-time algorithm for implication of fds due to C. Beeri and P. A. Bernstein depends on the efficient construction of the closure $X^+$ w.r.t. $\Sigma$ (otherwise read $\Sigma$-closure of $X$) also denoted by $cl_\Sigma(X)$.

**Lemma 2:** Let $\Sigma$ be a set of fds to be satisfied by all legal relation over $R$. Let $X, Y \in R$. Then the following holds.

$$\Sigma \models X \rightarrow Y \iff Y \subseteq cl_\Sigma(X).$$

Thus in order to test if a given fd $X \rightarrow Y$ is implied by $\Sigma$ it is sufficient to build $cl_\Sigma(X)$ and see if $Y \subseteq cl_\Sigma(X)$. We call an fd $X \rightarrow Y$ closed if $Y = cl_\Sigma(X)$. A set $\Delta$ of fds is closed if every fd in it is closed. The case $|Y| = 1$ is noteworthy. An fd $X \rightarrow Y$ is said to be in canonical form if $|Y| = 1$. Any fd $X \rightarrow Y$ can be converted to a set of fds in canonical form in view of the following lemma.

**Lemma 3:** For $1 \leq j \leq k$ and $Y = A_1 \ldots A_k$, $X \rightarrow Y \models X \rightarrow A_j$ and $\{X \rightarrow A_j\} \models X \rightarrow Y$.

### 3.2 Computing the closure

The following algorithm $CLOSURE(\Sigma, X)$ takes as input a set of fds $\Sigma$ and an attribute set $X$ and outputs $cl_\Sigma(X)$.

**Algorithm $CLOSURE(\Sigma, X)$**

begin

$Y \leftarrow X$

while (there exists an fd $S \rightarrow T$ such that $S \subseteq T$ and $T \not\subseteq Y$) do

$Y \leftarrow YT$

return $Y$

end

**Lemma 4:** Algorithm $CLOSURE(\Sigma, X)$ correctly outputs $cl_\Sigma(X)$.

**Proof:** The algorithm terminates after examining a finite number of fds. By induction on the steps of the algorithm we first claim that starting from initialization till termination $Y \subseteq cl_\Sigma(X)$ holds. As $Y$ is initialized to $X$ the claim is true initially since $\Sigma \models X \rightarrow A$ for all $A \in X$. Let the claim be true at some intermediate stage during the execution after which the algorithm considers an fd $S \rightarrow T$ from $\Sigma$ such that $S \subseteq Y$. Then $Y \rightarrow S$ and since $S \rightarrow T$ we have $Y \rightarrow T$. By induction hypothesis $X \rightarrow Y$ and so $X \rightarrow T$ and we have $X \rightarrow YT$. That is $\Sigma \models X \rightarrow YT$. Therefore the claim is true upon termination of the algorithm.

We further show that upon termination it is impossible to have $Y \subseteq cl_\Sigma(X)$. If possible let the contrary hold. That is let $B \in cl_\Sigma(X)$ but $B \not\subseteq Y$. Since $B \in cl_\Sigma(X)$ we have $\Sigma \models X \rightarrow B$. We now build a relation $I$ that is legal. Let $I$ consist of two tuples $u, v$ such that $u[A] = v[A]$ iff $A \in Y$. To show that $I$ satisfies $\Sigma$ we assume the contrary if possible. Let $S \rightarrow T$ be an fd causing violation. Then when this fd was considered
by the algorithm we should have had $S \subseteq Y$ but $T \not\subseteq Y$. However in such a case the algorithm should have been at some intermediate state of execution. Therefore $I$ satisfies $\Sigma$ but by construction $I$ does not satisfy $X \rightarrow B$. In symbols $\Sigma \not\models X \rightarrow B$ which is a contradiction. Hence it follows that $Y = cl_\Sigma(X)$.

**Remark.** It follows that $CLOSURE(\Sigma, X)$ can be implemented efficiently. A theorem due to C.Beeri and P.A.Bernstein asserts that the implication problem for fds is solvable in time linear in the length of the input.

### 3.3 Covers

Given two sets of fds $\Delta$ and $\Sigma$, we say one is a *cover* for the other if $\Delta \equiv \Sigma$. We can speak of minimal covers in the sense that for any other cover $\Gamma$ of $\Sigma$ a cover $\Delta$ is minimum if the number of fds in $\Delta$ is not greater than that in $\Gamma$. With an efficient algorithm for implication we can efficiently determine equivalence, redundancy and a non-redundant cover. Correctness of the following algorithm $NONREDUND(\Sigma)$ is evident.

**Algorithm** $NONREDUND(\Sigma)$

begin
  $\Delta \leftarrow \Sigma$
  for (each fd $\sigma$ in $\Delta$) do
    if $\Delta - \{\sigma\} \models \sigma$
      then $\Delta = \Delta - \{\sigma\}$
  return $\Delta$
end

The above algorithm does not necessarily find the minimum covers. Let $\Sigma$ be a set of fds. The following theorem from [6] (which has a stronger version) is stated without proof.

**Theorem 2:** Let $\Delta$ be a non-redundant cover for $\Sigma$. If $\Delta$ is closed then it is minimum.

The following simple but non-trivial algorithm $MINCOVER(\Sigma)$ takes as input a set $\Sigma$ of fds and outputs a minimum cover for $\Sigma$.

**Algorithm** $MINCOVER(\Sigma)$

begin
  $\Delta \leftarrow \Sigma$
  for each $\sigma = X \rightarrow Y \in \Sigma$ do
    begin
      $\Delta \leftarrow \Delta - \sigma$
      if $Y \not\subseteq CLOSURE(\Delta, X)$ then
        begin
          $Z \leftarrow CLOSURE(\Delta, Y)$
        end
    end
end
Algorithm $MINCOVER(\Sigma)$ scans through all the fds in $\Sigma$. On encountering each $\sigma$ the algorithm removes $\sigma$ from $\Delta$ and thus updates $\Delta$. It checks if the removal is safe in which case the updated $\Delta$ will be equivalent to $\Sigma$. Otherwise it updates $\Delta$ so that after the updation $\Delta \equiv \Sigma$. The later updation ensures that the newly added fd is closed. Theorem 2 assures the correctness of $MINCOVER(\Sigma)$. The time complexity of the algorithm can be estimated as $O(|\Sigma| \times \tau)$ where $\tau$ is the time to find closure.

Given a set $\Sigma$ of fds, by $CANONICAL(\Sigma)$ we denote a canonical cover of $\Sigma$ i.e., a cover for $\Sigma$ such that for any fd $\sigma \in CANONICAL(\Sigma)$, $\sigma$ is in canonical form.

## 4 Database schema design

The principal goal in database schema design is how to design a set of relation schemes to constitute a database and how to indicate their meaningfulness by specifying appropriate constraints such as fds. Intuitively it appears that there is a trade-off between updating a database versus querying it – smaller relational schemes are easier to update while queries on them them may be harder to process. Past research along these lines have focussed on arriving at acceptable ways of grouping attributes into tables and on obtaining normal forms [3]. The criteria for acceptance of a design is preservation of information and suggested dependencies and elimination of redundancy.

In formal terms, a relation scheme is a 2-tuple $(R, \Sigma)$ where $R \subseteq U$ and $\Sigma$ are respectively a relation scheme and an associated set of fds over $R$. Then a database schema $D$ is a set of relation schemes i.e., $D = \{(R_1, \Sigma_1), \cdots, (R_k, \Sigma_k)\}$ where $U = \bigcup_{j=1}^{k} \Sigma_j$. It is also convenient to define $\Sigma = \bigcup_{j=1}^{k} \Sigma_j$. Finally a database $B$ over $D$ is an assignment of a meaningful relation to each relation scheme in each 2-tuple in $D$. The primary objective in database schema design is that problems such as Codd’s anomalies should not exist during database operations w.r.t tuples like additions, deletions and updations - this concern has resulted in what are called as normal forms of database schemas.

### 4.1 BCNF

For a relation scheme $R$ and attribute set $X \subseteq R$ is a determinant of $R$ if there exists at least one attribute $A \in R - X$ such that $\Sigma \models X \rightarrow A$. If for all $A \in R$ we have $\Sigma \models X \rightarrow A$ then $X$ is called a superkey of $R$. If $X$ is a superkey and further if for any $B \in X$, $\Sigma \models X - B \not\rightarrow A$ then $X$ is called a key of $R$. Using an algorithm for implication, the following algorithm constructs keys for a relation scheme $(R, \Sigma)$.
Algorithm $KEY(R, \Sigma)$
begin
$X \leftarrow R$
for (each $A \in R$) do
if $\Sigma \models X - A \rightarrow R$
then $X = X - A$
return $X$
end

The $X$ returned by the algorithm depends on the sequence of testing for redundancy the different $A$’s.

**Remark:** The problem of finding all the keys for a given relation scheme is $NP$-Complete.

We now state the Boyce-Codd normal form ($BCNF$) probably what is referred to as the strongest of all normal forms sought after in database schema design. Intuitively the permitted fds in any legal relation in $BCNF$ are all due to keys.

**Definition of BCNF:** $\mathcal{D}$ satisfies $BCNF$ if whenever $X$ is a determinant of $R$ then $X$ is a superkey of $R$ where $R$ is a part of any relation scheme of $\mathcal{D}$.

### 4.2 Normalization via decomposition

Let $R = ABCDE$ be a relation scheme such that for any meaningful relation on $R$ the fd $E \rightarrow CD$ holds. Consider the decomposition of $R$ as $ABE$ and $CDE$. The following relation $I$ on $R$ is built by ensuring that $E \rightarrow CD$ holds and by randomly filling values for $A$ and $B$ from their underlying domains.

$$
\begin{array}{cccccc}
& A & B & C & D & E \\
\hline
0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 0 \\
1 & 1 & 1 & 1 & 0 \\
\end{array}
$$

We can check that $\pi_{ABE}(I) \ast \pi_{CDE}(I) = I$ is always true. This isn’t a coincidence – the presence of fds can guarantee non-lossy decompositions as asserted by the following theorem.

**Theorem 3:** Let $I$ be defined on $R = XYZ$ such that $I$ satisfies $X \rightarrow Y$. Then the decomposition of $I$ into $\pi_{XY}(I)$ and $\pi_{XZ}(I)$ is lossless.

**Proof.** Let $J = \pi_{XY}(I) \times \pi_{XZ}(I)$. In general $I \subseteq J$. It is therefore sufficient to prove $J \subseteq I$. This is done by showing that $u \in J \implies u \in I$ for any $u \in J$. By the definition of $J$, $u[XY] \in \pi_{XY}(I)$ and $u[XZ] \in \pi_{XZ}(I)$. As $\pi_{XY}(I)$ and $\pi_{XZ}(I)$ are obtained from $I$, there are two tuples $v, w \in I$ such that $u[XY] = v[XY]$ and $u[XZ] = w[XZ]$. 
Consequently $v[X] = w[X]$ and since $X \rightarrow Y$ it follows that $w[XY] = v[XY] = u[XY]$. We reason $w[XYZ] = u[XYZ]$ i.e., $w = u$.

To implement normalization via decomposition assume that $\mathcal{D}$ is not in BCNF where $\mathcal{D}$ is a database schema defined as above. W.l.o.g. assume that the relation scheme $(R_j, \Sigma_j)$ is the cause for violation. For an attribute set $X$ we define $\pi_X(\Sigma) = \{S \rightarrow T | S \rightarrow T \in X$ and $ST \subseteq X\}$. Since $(R_j, \Sigma_j)$ is not in BCNF there exists an $X$ that is a determinant of $R_j$ but that isn’t a superkey for $R_j$. Therefore there is an attribute $A \in R_j - X$ such that $\Sigma \models X \rightarrow A$ where $\Sigma = \bigcup_{j=1}^{k} \Sigma_j$. In the decomposition process we invoke theorem 3 and replace $(R_j, \Sigma_j)$ by $D_1 = (\pi_{X \rightarrow A}(R_j), \Sigma_1^j)$ and $D_2 = (\pi_{R_j \rightarrow A}(R_j), \Sigma_2^j)$ where $\Sigma_1^j = \pi_{X \rightarrow A}(\Sigma_j)$ and $\Sigma_2^j = \pi_{R_j \rightarrow A}(\Sigma_j)$ and continue further the process of decomposition if $D_2$ is not in BCNF.

Let $\mathcal{D}$ be as defined above and let $\mathcal{E} = (U, \Sigma)$, also referred to as a universal schema. As a part of the design process we would like to find when does it make a reasonable sense to say that $\mathcal{D}$ represents $\mathcal{E}$. It perhaps follows intuitively that we need these two conditions viz., (a) there should be no loss of information if relations are stored using schema $\mathcal{D}$ rather than as schema $\mathcal{E}$ and (b) all the $\Sigma$ should logically imply the fds in all the $\Sigma_j$’s and together the $\Sigma_j$’s should logically imply $\Sigma$. This amounts to the requirement that a decomposition from $\mathcal{E}$ to $\mathcal{D}$ should be lossless and dependency preserving. Formally $\mathcal{D}$ represents $\mathcal{E}$ if the following conditions hold together.

(i) Let $I$ be a relation on $U$ satisfying $\Sigma$. Then for $j = 1, \ldots k$ the decomposition of $I$ into $\pi_{R_j}(I)$’s is lossless.

(ii) Let $\Delta = \bigcup_{j=1}^{k} \Sigma_j$. Then $\Sigma \models \Delta$ and $\Delta \models \Sigma$.

The decomposition in theorem 3 is such that (i) is guaranteed but only half of (ii) is satisfied. Unfortunately in general it appears that it not possible to efficiently find a decomposition resulting in BCNF that satisfies both (i) and (ii) above. This is asserted by theorem 4 that follows.

### 4.2.1 Checking for BCNF violations

We begin with the hitting set problem which is $NP$–complete (problem [SP8] on p.222 of [5]).

**Hitting set:** Let $T = \{A_1, \ldots, A_n\}$ and let $B_j \subset T$ for $j = 1, \ldots, m$. The problem asks to find if possible a hitting set $W \subseteq T$ such that for each $j$, $|W \cap B_j| = \alpha$ where $\alpha \geq 1$.

In the following example $\alpha = 1$. For a generalized hitting set problem see [1]. The generalized version models known problems such as vertex cover.

Ex. Setting $T = \{p_1, \ldots, p_8\}$, let $B_1 = \{p_1, p_2, p_3\}$, $B_2 = \{p_2, p_3, p_4\}$, $B_3 = \{p_1, p_7, p_8\}$, $B_4 = \{p_5, p_6, p_7\}$. We interpret $T$ to be a set of persons and we set that a task $t_j$ requires for its completion skills available with any person in $B_j$. It is required to find a group of persons from $T$ who can complete all the tasks subject to the constraint that only one
person is selected from each $B_j$. We may require finding (i) a $W$ so as to maximize or minimize $|W|$ or (ii) $W_1$ and $W_2$ where possible so that $W_1 \cap W_2 = \phi$.

The problem of determining if a given database schema violates BCNF is a hard problem. The following proof relies on the hardness of Hitting set with $\alpha = 1$.

**Theorem 4:** [C.Beeri and P.A.Bernstein] Let $D$ be a given database schema. It is $NP$-complete to check if there is a BCNF violation in $D$.

**Proof:** That the problem belongs to the class $NP$ is clear from [7]. Following [7] we show that the problem is $NP$–hard. That is, we reduce the hitting set problem to the problem at hand. More specifically, the proof shows that each instance of the hitting set problem can be mapped in polynomial-time to a database schema such that there exists a hitting set iff the produced schema violates BCNF.

Retaining the notations of the hitting set problem let $U = \{A_1, \ldots , A_n, B_1, \ldots , B_m, C, D\}$. We build a database schema $D$ consisting of the following relation schemes.

D1. For every pair $A_i, B_j$ such that $A_i \in B_j$ we include in $D \{A_iB_j, (A_i \rightarrow B_j)\}$.

D2. We include in $D \{B_1 \ldots B_mCD, (B_1 \ldots B_m \rightarrow C)\}$.

D3. Finally we include in $D \{A_1 \ldots A_nCD, \{CD \rightarrow A_1 \ldots A_n\} \cup \{A_iA_j \rightarrow CD \text{ if } i \neq j \text{ and both } A_i, A_j \text{ belong to } B_k \text{ for some } k\}\}$.

It is not difficult to reason that with D1, D2 and D3, $D$ can be constructed in polynomial-time. Let $W = \{A_{\alpha_1}, \ldots , A_{\alpha_r}\} \subseteq T$ be a hitting set. Then for every $B_j$ $W \cap B_j = A_{\alpha_k}$, $1 \leq k \leq r$. By D1 this necessarily means $A_{\alpha_k} \rightarrow B_j$. We can conclude, for every $i$, $\Sigma \vdash W \rightarrow B_i$. Using lemma 4 with D2 we then have $\Sigma \vdash W \rightarrow C$. So in D3 $W$ is a determinant of $A_1 \ldots A_nCD$. We now make the following claim 4.1 which implies that $W$ is not a superkey for the relational scheme in D3.

Claim 4.1: $cl_\Sigma(W) = WB_1 \ldots B_mC$ where $\Sigma$ is the set of all fds in $D$.

To establish the claim we note that from $CLOSURE(\Sigma, X)$ it follows that it is sufficient to show that for every fd $S \rightarrow T \in \Sigma$ either $S \notin cl_\Sigma(W)$ or $T \subseteq cl_\Sigma(W)$ holds. This follows by considering D1, D2 and D3 separately.

Conversely let $D$ be not in BCNF. Then one or more of D1, D2, D3 should contain a violation of BCNF – we find that only D3 can cause a violation. From D3 we first observe that for any $i, j$ ($i \neq j$) if $A_i \in B_k$ and $A_j \in B_k$ then $A_iA_j \rightarrow A_1 \ldots A_n$. Let $W \subseteq A_1 \ldots A_nCD$ be a determinant but not a superkey. Clearly $C, D \notin W$. So let $W \subseteq A_1 \ldots A_n$. Because of the observation above $W$ cannot contain two distinct $A_i, A_j$ belonging to some $B_k$. That is $W$ consists of $A_i$’s such that every such $A_i \in B_j$ and there exists no other $A_j \in B_j$. Let $W = \{A_{\alpha_1}, \ldots , A_{\alpha_s}\}$ so that each $A_{\alpha_j}$ embraces some $B_p$. If all the $B_i$’s are not embraced by our choice of $W$ let there be a $B_i$ such that there exists $A_q \in B_i$ as $B_i \subseteq T$ but $A_q \notin W$. We then update the current $W$ by including $A_q$ and do not further reckon any $B_s$ if $A_q \in B_s$. This way we can expand $W$ embracing all possibly left out $B_i$’s so that it becomes a hitting set.  

\[\blacksquare\]
4.3 3NF and normalization via synthesis

We assume $\mathcal{D}$ is a database schema as defined above. Starting from $\mathcal{D}$ instead of obtaining a desirable database schema in $BCNF$ representing $\mathcal{D}$ which may not be feasible in some cases it is possible to settle for a weaker normal form referred to as the third normal form ($3NF$). The definition of $3NF$ can be given in terms of a strong determinant. An attribute $A \in R_j$ is called prime if there is a key $Z$ of $R_j$ such that $A \in Z$. $Z$ is a strong determinant of $R_j$ if $Z \subseteq R_j$ and there exists a nonprime $A \in R_j - Z$ such that $\Sigma \vdash Z \rightarrow A$. Here it is not necessary for $Z$ to be a superkey.

Modern definition of $3NF$: $\mathcal{D}$ is in $3NF$ if whenever $Z$ is a strong determinant for any $R_j$ in $\mathcal{D}$ then $Z$ is a superkey of $R_j$.

The problem of determining whether a given relation scheme $(R_j, \Sigma_j)$ is in $3NF$ can be done in one of the following two ways.

1. Show that for all $fd X \rightarrow A \in \Delta_j$, $X$ is a superkey or $A$ is prime, where $\Delta_j = CANONICAL(\Sigma_j)$. Conclude that $3NF$ is not violated.
2. Show that there exists an $fd X \rightarrow A \in \Delta_j$ such that $X$ is not a superkey and $A$ isn’t prime, where $\Delta_j$ is as in 1. Conclude that $3NF$ is violated.

Remarks:
1. First we note that $BCNF$ implies $3NF$. Let $(R_j, \Sigma_j)$ be in $\mathcal{D}$ and let $X \rightarrow A \in \Sigma_j$. If $A \in R_j$ is prime then there is at least one key $Y \subseteq R_j$ such that $A \in Y$. Assume that we decompose $R_j$ as $\pi_{XA}(R_j)$ and $\pi_{R_j-A}(R_j)$. Then the $fd Y \rightarrow R_j$ is lost since $Y$ is a key and $A \in Y$. Therefore the resulting schema does not represent $\mathcal{D}$. Such a problem will not arise if every $A \in R_j$ is nonprime.
2. The problem of determining whether a given attribute is prime is $NP$–Complete. (problem [SR28] on p.232 of [5])
3. [8] presents an algorithm to check whether or not a relation scheme is in $3NF$.

Normalization to $3NF$ via decomposition is not practical but fact 2 gives the good news.

Fact 2: For any universal schema $\mathcal{E}$ there exists a database schema $\mathcal{D}$ representing $\mathcal{E}$. Moreover normalization through synthesis finds $\mathcal{D}$ efficiently as shown by P.A.Bernstein around 1976.

The $3NF$ synthesis algorithm can be described along the following lines.

Algorithm $3NF(U, \Sigma)$

begin
$\mathcal{D} \leftarrow \phi$
$\Delta \leftarrow REDUCED(NONREDUND(CANONICAL(U, \Sigma)))$
$X \leftarrow KEY(U, \Delta)$
for (each $fd Y \leftarrow A \in \Delta$) do
$\mathcal{D} \leftarrow \mathcal{D} \cup (YA, \pi_{YA}(\Delta))$
$\mathcal{D} \leftarrow \mathcal{D} \cup (X, \phi)$
end
Algorithm $3NF(U, \Sigma)$ first finds a reduced nonredundant canonical cover for $\Sigma$. Then for every fd it a relation scheme is created. Finally a key for $U$ is added. It can be seen that the algorithm is efficient. The following theorem is stated without proof. 

**Theorem 5:** Algorithm $3NF(U, \Sigma)$ is correct i.e., $D$ outputted by the algorithm represents the input schema $(U, \Sigma)$.

### 5 Concluding remarks

Database theoreticians have pointed out that there is not much literature on the properties of sets of fds. We refer to [2] for some modern approaches concerning the presentation of sets of fds based on which full knowledge on the validity of fds w.r.t. a stated context may be extracted. It is known that redundancy and potential inconsistency can also be present in certain relations in the absence of fds. A *multivalued dependency* (mvd) occurs when an attribute set conditionally defines a set of other attributes. Formally if $XY \subseteq R$, $X$ multidetermines $Y$ i.e., $X \rightarrow Y$ if for every relation $I$ on $R$, for all tuples $u, v \in I$ if $u[X] = v[X]$ then there exists a tuple $w \in I$ such that $w[X] = u[X] = v[X]$, $w[Y] = u[Y]$, $w[R - XY] = v[R - XY]$. The notion of mvds is closely related to join. The fourth normal form $(4NF)$, a generalization of $BCNF$ requires that every mvd is due to keys. Let $R$ be a relation scheme and $X, Y \subseteq R$ ($X \neq \phi, Y \neq \phi$) and let $\Sigma$ be a set of fds and mvds that need to satisfied on legal relation on $R$. $R$ is in $4NF$ if for every mvd $X \rightarrow Y$ that is to hold on legal relations over $R$ either the mvd is trivial i.e., $Y \subseteq X$ or $XY = R$ or $X$ is a super key in the sense defined before. A theorem states that if $R$ obeys only those fds and mvds that are logical consequences of a set of fds then $4NF$ coincides with $BCNF$. The interaction between fds and mvds has been studied under a sound and complete formal system.
References


