Normal Forms - Summary of Lectures given by Prof. J.Makowsky
Database theory course 236356 Winter 2001/2002 Semester

Shimon Golan 025429424
golans@cs.technion.ac.il
Department of Computer Science
Technion—Israel Institute of Technology
Technion City, Haifa 32000, Israel

April 8, 2002

Problem given the relation shcema $\mathcal{R}[x]$, where $x = x_1, x_2, \ldots, x_n$ and given a set of FD's (functional dependencies), F, the question that arises is :
should we decompose $\mathcal{R}$ *?*

Example given the schema $\mathcal{R}[c, s, z]$,where c stands for city, s stands for street, and z stands for zipcode, and having the following FD's, F :

1. $z \rightarrow c$
2. $c, s \rightarrow z$

should we decompose $\mathcal{R}$ *?*

1 Some background from the 1st database course

we shall recall the definition of functional dependencies. Syntactically, given a set of attributes,$U = \{A_1, A_2, \ldots, A_s\}$,and subsets $X, Y \subseteq U$, a functional dependency is of the form $X \rightarrow Y$.

Definition 1.1 *given a schema $\mathcal{R}[U] = \mathcal{R}[A_1, A_2, \ldots, \mathcal{A}_s]$, and given a relation,* $\mathbf{r}(\mathcal{R})$*, we say that* $\mathbf{r}$ *satisfies the functional dependency $f = X \rightarrow Y$, and denote it by* $\mathbf{r} \models f$*,if* for all *pairs of tuples $t1$ and $t2$ in* $\mathbf{r}$*, such that $t1[X] = t2[X]$,it it also the case that $t1[Y] = t2[Y]$.*

Definition 1.2 *given a schema $\mathcal{R}[U] = \mathcal{R}[A_1, A_2, \ldots, \mathcal{A}_s]$, and given a relation,* $\mathbf{r}$,$\mathbf{r}(\mathcal{R})$*, and a set of functional dependencies F, we say that* $\mathbf{r}$ *satisfies F,* $\mathbf{r} \models F$*,if* for each $f \in F$, $\mathbf{r} \models f$.

Definition 1.3 *given a schema $\mathcal{R}[U] = \mathcal{R}[A_1, A_2, \ldots, \mathcal{A}_s]$,a set of functional dependencies,F, and a single dependency f, we say that F logically implies f, denoted by $F \models f$, if* every *relation* $\mathbf{r}$*, above $\mathcal{R}$, s.t.* $\mathbf{r} \models F$*, also implies* $\mathbf{r} \models f$.

2 back to our discussion

Problem given a schema $\mathcal{R}[U] = \mathcal{R}[A_1, A_2, \ldots, \mathcal{A}_s]$, a set of functional dependencies,F, and a single dependency f, we would like to have an efficient method to check whether $F \models f$.

If we use only defintion 1.3, we could end up with an algorithm that never halts if the answer to the question whether $F \models f$ is true, since me must check every possible relation $\mathbf{r}$, abouve $\mathcal{R}$.
We would like to do better.

following is a set of 3 rules of inference for fd's

1. (reflexivity) if $Y \subseteq X$,then $X \rightarrow Y$.
2. (augmentation) if $X \rightarrow Y$,then $XZ \rightarrow YZ$
3. (transitivity) if $X \rightarrow Y$, and $Y \rightarrow Z$, then $X \rightarrow Z$

where X, Y, Z are sets of attributes over some schema $\mathcal{R}$.
These rules are used to form proofs about logical implication among fd's, similar to proofs in mathematical logic.

Definition 2.1 *We say that a fd f is* ***provable*** *from a set of FD's F, and denote it by $F \vdash f$, if there is a proof sequence of f from F,using the inference rules shown above.*

Theorem 2.2 *(Armstrong)The resulting proof system is sound and complete, i.e.* $F \models f$ *iff* $F \vdash f$*.*

Armstrong's theorem aids us in finding a solution to whether $F \models f$. We can find a proof sequence to $F \vdash f$ (in time $O(2^{2n})$, where n is the arity of $\mathcal{R}$, the relevant schema), and deduce from it that $F \models f$.

Definition 2.3 *given a schema* $\mathcal{R}[U] = \mathcal{R}[A_1, A_2, \ldots, \mathcal{A}_s]$*,a set of functional dependencies,*F*, and* $X \subseteq U$*, we say that* X *is a superkey relative to* F*, if* $F \models X \rightarrow U$*.* X *is said to be a minimal key, if* X *is a superkey and for all* $X_0 \subset X$*,* $F \not\models X_0 \rightarrow U$*.*

Example recall our first example of $\mathcal{R}[C, S, Z]$,where C stands for city, S stands for street, and Z stands for zipcode, with the following FD's :

1. $Z \rightarrow C$
2. $C, S \rightarrow Z$

These fd's descirbe the fact that given a zipcode, we can tell the city it belongs to (the first fd). However, since several streets, in a given city, can share a zipcode, we can not determine the street, given only the zipcode. Of course, once the city and street are at hand, the zipcode can be deduced. (the second fd).
According to definition 2.3, we see that C, S is a superkey (and also a minimal key). Also, using our inference rules, we can deduce that S, Z is a superkey.

Definition 2.4 *given a schema* $\mathcal{R}[U] = \mathcal{R}[A_1, A_2, \ldots, \mathcal{A}_s]$*, a* ***valid decomposition*** *of* $\mathcal{R}$ *is a vector of subschemas* $\rho = \{\mathcal{R}_1[U_1], \mathcal{R}_2[U_2], \ldots, \mathcal{R}_n[U_n]\}$*, s.t* $\bigcup_{i=1}^{n} = U$*.*

Definition 2.5 *given a schema* $\mathcal{R}[U] = \mathcal{R}[A_1, A_2, \ldots, \mathcal{A}_s]$*, a set of fd's* F*, and a valid decomposition of* $\mathcal{R}$*,* $\rho = \{\mathcal{R}_1[U_1], \mathcal{R}_2[U_2], \ldots, \mathcal{R}_n[U_n]\}$*,we say that* ρ *is a* ***lossless join*** *decomposition, if for every relation* $\mathbf{r}(\mathcal{R})$*,s.t* $\mathbf{r} \models F$*, it is also true that* $\bowtie_{i=1}^{n} \pi_{\mathcal{R}_i}\mathbf{r} = \mathbf{r}$*.*

Being loseless is of crucial importance to the decomposition. It means that upon decomposition, it is guaranteed that information will neither be lost, nor added from no where.

Definition 2.6 *Given a set of attributes* U*,a schema* $\mathcal{R}[U]$*,a set of fd's* F *over* U*,and a subset of attributes* V*,* $V \subset U$*, then the* ***projection*** *of* F *on* V*, denoted by* F_v*, is defined as :*

$$F_v = \{X \rightarrow Y : X, Y \subseteq V \, and \; F \models X \rightarrow Y\} = \{X \rightarrow Y \in F^+ : X, Y \subseteq V\}$$

i.e,F_v includes the projection on V of the relevant dependencies among $\mathbf{F}^+$.

Definition 2.7 *given a schema $\mathcal{R}[U] = \mathcal{R}[A1, A2, \ldots, A_s]$, a set of fd's F, and a valid decomposition of $\mathcal{R}$, $\rho = \{\mathcal{R}_1[U_1], \mathcal{R}_2[U_2], \ldots, \mathcal{R}_n[U_n]\}$,we say that ρ is a* ***dependency preserving*** *decomposition of $\mathcal{R}$, if $\bigcup_{i=1}^n F_{U_i} \models F$.*

Example Given $U = \{X, Y, Z\}$,$\mathcal{R}[U]$, $F = \{X \rightarrow Y\}$, and a decomposition $\rho_1 = \{\mathcal{R}_1[XY], \mathcal{R}_2[YZ]\}$, we see that

- $F_{U_1} = \{X \rightarrow Y\}$.
- $F_{U_2} = \phi$.
- $F_{U_1} \bigcup F_{U_2} = \{X \rightarrow Y\} = F$.

Thus, the decomposition is dependency preserving. On the other hand, ρ_1 is not a lossless join.

This can be shown by a simple example : let $\mathbf{r} =$

X	Y	Z
x1	y1	z1
x2	y1	z2

, we can see that $\mathbf{r} \models F$.

After the decomposition by ρ_1 we get :

$\mathcal{R}_1[U_1] =$

X	Y
x1	y1
x2	y1

and

$\mathcal{R}_2[U_2] =$

Y	Z
y1	z1
y1	z2

Taking their join yields :

X	Y	Z
x1	y1	z1
x1	y1	z2
x2	y1	z1
x2	y1	z2

which is not $\mathcal{R}$.

Theorem 2.8 *given a set of attributes U,a schema $\mathcal{R}[U]$, a set of fd's F, and a valid decomposition of $\mathcal{R}$ to 2 sub-schemas $\rho = \{\mathcal{R}_1[U_1], \mathcal{R}_2[U_2]\}$, then ρ is lossless iff the set of attributes $U_1 \bigcap U_2$ is a superkey in U_1, or U_2, or both.*

Theorem 2.9 *given a set of attributes U, a schema $\mathcal{R}[U]$, a set of fd's F, and a valid decomposition of $\mathcal{R}$, $\rho = \{\mathcal{R}_1[U_1], \mathcal{R}_2[U_2] \ldots \mathcal{R}_s[U_s]\}$, then ρ is lossless if there exists a sequence of decompositions, such that on each stage we carry out only a decomposition into 2 sub-schemas,which is lossless according to theorem 2.8.*

Example Returning to our initial example, with schema $\mathcal{R}[c, s, z]$, and FD's, F :

1. $Z \rightarrow C$
2. $C, S \rightarrow Z$

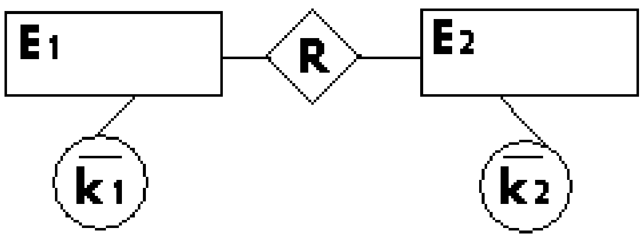
We notice that $\mathcal{R}$ does not have a decomposition under F, which is dependency preserving. This is so, since on each "real" decomposition no schema would include all the attributes, and therefore the dependency $C, S \rightarrow Z$ would not be inferred by the union of the projections of the dependencies on the sub-schemas.

3 Other dependencies

So far, we only treated functional dependencies. However, we would like to deal with other dependencies as well. Two questions arise :

1. What are feasible data dependencies ?
2. How to choose data shcemas ?

Other dependencies are *inclusion dependencies*. For example, given the following ERD diagram



we would like to have : $\pi_{k_1} R \subseteq \pi_{k_1} E_1$

In order to be able to treat all kind's of dependencies in a unified manner we use mathematical logic notation.
All the dependencies described thus far, and others not shown here (such as join dependencies), can be described as FOL (first order logic) sentences called FID (full implication dependencies), of the following form :

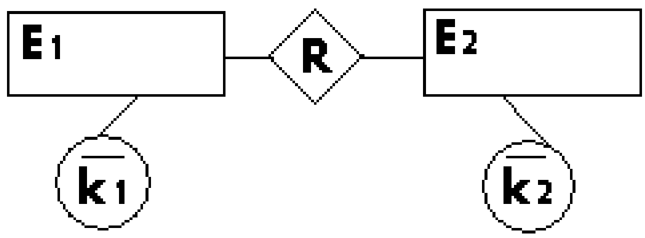
$$\forall \bar{x} (\bigwedge_{i=1}^{m} S_i \rightarrow S)$$

where S_i, S are atomic formulas.
Later variations of this sentence included : $\exists \bar{y} \bigwedge_{i=1}^{m} S_i$, and $\exists \bar{y} \bigvee_{i=1}^{m} S_i$ as the right hand side of $\rightarrow$.

Example Given the schema $\mathcal{R}[X, Y, Z]$ the functional dependency : $F = \{X \rightarrow YZ\}$, could be alternatively stated as :

$$\forall x_1, x_2, y_1, y_2, z_1, z_2((R(x_1, y_1, z_1) \wedge R(x_2, y_2, z_2) \wedge x_1 = x_2) \rightarrow (y_1 = y_2 \wedge z_1 = z_2))$$

Example Given the following ERD :



with the following schemas :

- $E_1(k_1, \_, \_)$
- $E_2(k_2, \_, \_)$
- $\mathcal{R}(k_1, k_2)$

We could state the inclusion dependency $\pi_{k_1} R \subseteq \pi_{k_1} E_1$ as :

$$\forall x_1, x_2 (R(x_1, x_2) \rightarrow \exists y_1, y_2, y_3 (E_1(y_1, y_2, y_3) \wedge x_1 = y_1))$$

The question now is : given a set of dependencies Σ, and given a single dependency φ, can we answer $\Sigma \models \varphi$?
We separate the problem to 2 classes of models :

1. **All models (including infinite)** In this case,we know, by Gödels Completeness theorem that $\Sigma \models_{All\ models} \varphi$ iff $\Sigma \vdash_{All\ models} \varphi$. However $\Sigma \vdash_{All\ models} \varphi$ is semi-computable (i.e. in RE).

2. **All finite models** In this case , $\Sigma \models_{finite\ models} \varphi$, is in $CO - RE$, since $\Sigma \not\models_{finite\ models} \varphi$ is in RE. (It suffices to find one finite model $\mathcal{M}$, which satisfies $\mathcal{M} \models \Sigma$, but not $\mathcal{M} \models \varphi$, thus a Turing machine that finds such a model halts and accepts since it only has to check all **finite** models.

Next, we prove a theorem that shows that on restricted versions of FOL formulas, it is possible to answer $\Sigma \models \varphi$ in exponential time.

Theorem 3.1 *(Bernays - Schönfinkel) : Let Σ, φ be of the form $\forall \bar{x} B(\bar{x})$, denoted as* ***universal formulas****, where $B(\bar{x})$ is a boolean combination of atomic formulas, over a dictionary τ, without function symbols, and no distinguished elements. Then, $\Sigma \models \varphi$ is computable in exponential time.*

in order to prove this theorem, we proceed with the following lemmas:

Lemma 3.2 *let τ be a dictionary without function symbols and distinguished elements. Also, let Σ be a set of universal formulas over τ.*
Let $\mathcal{A}, \mathcal{B}$ be τ-structures such that :

- $\mathcal{A} =< A, R_1^A, \cdots, R_I^A >$
- $\mathcal{B} =< B, R_1^B, \cdots, R_I^B >$
- $\forall i \in I, \rho(R_i^A) = \rho(R_i^B)$*, where ρ denotes arity.*
- $\forall i \in I, R_i^B = R_i^A \bigcap B^{\rho(R_i^A)}$

$\mathcal{B}$ is said to be a sub-structure of $\mathcal{A}$ over τ. It follows that if $\mathcal{A} \models \Sigma$, then $\mathcal{B} \models \Sigma$.

Proof The proof is by induction on the construction of the formulas comprising Σ.

- **Basis** : let $\Sigma = \{\varphi\}$, where φ is atomic. Then it follows by definition that if $\mathcal{A} \models \{\varphi\}$, then $\mathcal{B} \models \{\varphi\}$.

- **Step** : Without loss of generality, we can assume that φ is of the form: $\forall\bar{x} \bigwedge \bigvee literals$, where $literals$ are atomic formulas, or negations of atomic formulas. Now we separate the proof to 2 sub-cases :

  - **case 1 - φ is quantifier free -**
    We show that $\mathcal{A} \models \varphi$ iff $\mathcal{B} \models \varphi$.
    This is also proven by induction :
    basis - This is exactly by definition.
    step - let φ'_1, φ'_2 be quantifier free FOL formulas. By the induction hypothesis, $\mathcal{A} \models \varphi'_i$ iff $\mathcal{B} \models \varphi'_i$, where $i = 1, 2$. We have to show that :
    a) $\mathcal{A} \models \neg\varphi'_1$ iff $\mathcal{B} \models \neg\varphi'_1$.
    proof:
    $\Rightarrow$ if $\mathcal{A} \models \neg\varphi'_1$, then $\mathcal{A} \not\models \varphi'_1$, by the hypothesis it follows that $\mathcal{B} \not\models \varphi'_1$, and hence $\mathcal{B} \models \neg\varphi'_1$.

    $\Leftarrow$ Similiar to $\Rightarrow$.

    b) $\mathcal{A} \models \varphi'_1 \vee \varphi'_2$ iff $\mathcal{B} \models \varphi'_1 \vee \varphi'_2$
    proof:
    $\Rightarrow$ if $\mathcal{A} \models \varphi'_1 \vee \varphi'_2$, then by the truth table of $\vee$, it follows that either $\mathcal{A} \models \varphi'_1$ or $\mathcal{A} \models \varphi'_2$, or both. if $\mathcal{A} \models \varphi'_1$, then by the hypothesis it follows that $\mathcal{B} \models \varphi'_1$, and thus $\mathcal{B} \models \varphi'_1 \vee \varphi'_2$. Likewise, we reason when $\mathcal{A} \models \varphi'_2$.
    $\Leftarrow$ Similiar to $\Rightarrow$.

    c) $\mathcal{A} \models \varphi'_1 \wedge \varphi'_2$ iff $\mathcal{B} \models \varphi'_1 \vee \varphi'_2$
    proof:
    $\Rightarrow$ if $\mathcal{A} \models \varphi'_1 \wedge \varphi'_2$, then by the truth table of $\wedge$, it follows that $\mathcal{A} \models \varphi'_1$ and $\mathcal{A} \models \varphi'_2$.Then by the hypothesis it follows that $\mathcal{B} \models \varphi'_1$, and $\mathcal{B} \models \varphi'_2$. Thus $\mathcal{B} \models \varphi'_1 \wedge \varphi'_2$.
    $\Leftarrow$ Similiar to $\Rightarrow$.

  -

  - **case 2 - φ is a universal formula -**
    In order to complete the proof, we add to case 1 the option of having universal quantifiers in φ.
    We assume $\varphi(\bar{x})$ is a formula with free variables $\bar{x}$, and want to show that if $\mathcal{A} \models \forall_{\bar{x}}\varphi(\bar{x})$,then $\mathcal{B} \models \forall_{\bar{x}}\varphi(\bar{x})$.

    This is proven by negation: Assume that , $\mathcal{A} \models \forall_{\bar{x}}\varphi(\bar{x})$, but

$\mathcal{B} \not\models \forall_{\bar{x}}\varphi(\bar{x})$. This means that $\mathcal{B} \models \exists_{\bar{x}}\neg\varphi(\bar{x})$, which implies that $\mathcal{A} \models \exists_{\bar{x}}\neg\varphi(\bar{x})$,i.e., $\mathcal{A} \not\models \forall_{\bar{x}}\varphi(\bar{x}) \Rightarrow$ a contradiction. ∎

Lemma 3.3 *let φ be of the form $\exists x_1, x_2, \ldots, x_m\beta(\bar{x})$, where $\bar{x} = (x_1, x_2, \ldots, x_m)$, and $\beta(\bar{x})$ is a boolean combination of atomic formulas (called existential form). Then, if φ is satisfiable, it has a model of size m.*

Proof if φ is satisfiable, the let $\mathcal{A}$ be a τ-structure, s.t. $\mathcal{A} \models \exists x_1, x_2, \ldots, x_m\beta(\bar{x})$. Let $\mathcal{A} =< A, R_1^A, R_2^A, \cdots >$.
If $\mathcal{A} \models \exists x_1, x_2, \ldots, x_m\beta(\bar{x})$ then it follows that there exists at least one assignment $\bar{v} = \{v_1, v_2, \ldots, v_m\} \in A$ s.t. $\mathcal{A} \models \beta(\bar{v})$.
Now, let us extend τ to with m new constant symbols $c_1, c_2, \ldots, c_m$, such that $c_i = v_i, \forall i = \{1, 2, \ldots, m\}$. Call this extended structure τ':

$$\tau' = \tau \bigcup \{c_1, c_2, \ldots, c_m\}$$

Now, we define a new structure above τ' called $\mathcal{A}'$:

$$\mathcal{A}' =< A, R_1^A, R_2^A, \cdots, c_1, c_2, \ldots, c_m >$$

It follows that

$$\mathcal{A}' \models \beta(c_1, c_2, \ldots, c_m) \text{ iff } \mathcal{A} \models \exists x_1, x_2, \ldots, x_m\beta(\bar{x})$$

Hence, we showed that if φ is satisfiable, then an equivalent formula, which is quantifier free (and thus universal) is satisfiable as well.
Next, we show that there exists a τ-structure of size m that satisfies $\beta(c_1, c_2, \ldots, c_m)$. For this end, we choose the following sub-structure of $\mathcal{A}$:

$$\mathcal{B} =< B, R_1^B, R_2^B, \cdots > \quad \text{where} \quad B = \{c_1, c_2, \ldots, c_m\}$$

Since $\beta(c_1, c_2, \ldots, c_m)$ depends only upon $c_1, c_2, \ldots, c_m$, we can restrict our world to B above.
It follows from Lemma 3.2 that $\mathcal{B} \models \beta(c_1, c_2, \ldots, c_m)$, and thus $\mathcal{B} \models \exists\bar{x}\beta(\bar{x})$. The size of $\mathcal{B}$ is the size of $(c_1, c_2, \ldots, c_m)$, which is m.

Thus, we have shown that given τ,a formula in existential form with m quantifiers, and given that τ is satisfiable, then it follows that τ is satisfiable on a model of size m. ∎

Now we return to the proof of Theorem 3.1. Recall that it states that given Σ (a set of universal formulas) and φ, a universal formula, over a dictionary τ, without function symbols, and no distinguished elements, then, $\Sigma \models \varphi$ is computable in exponential time.
The proof is given by construction of an algorithm that computes $\Sigma \models \varphi$.

The input:

- Σ - a finite set of universal τ formulas.
- φ - a universal τ formula.
- we assume that $|\Sigma| + |\varphi| = n$.

The problems

1. $\Sigma \models \varphi$
2. $\Sigma \models_{finite\ models} \varphi$

$\Sigma \models \varphi$ iff $\Sigma \bigcup \{\neg\varphi\}$ is not satisfiable. This can happen, according to lemma 3.3, iff $\Sigma \bigcup \{\neg\varphi\}$ is not satisfiable in a model of size $\leq n$ (since the number of quantifiers in the formula must be smaller than the total size of the formula , which is n).

Thus, all that needs to be done is check all finite models of **size smaller than n**, whether they satisfy $\Sigma \bigcup \{\neg\varphi\}$. If neither satisfies, then we deduce that $\Sigma \models \varphi$ and $\Sigma \models_{finite\ models} \varphi$. Thus, the computation time complexity depends upon the number of τ-structures of size $\leq n$, which, in turn, is exponential in n. ∎

Now we know that the question $\Sigma \models \varphi$, where Σ is a finite set of universal τ-formulas, and φ is a universal τ-formula, is computable in exponential time. The question that arises is what happens when we allow Σ, φ be non-universal in order to describe other types of dependencies.

The following table summarizes results shown in literature :

Σ	φ	$\Sigma \models_{all\ models} \varphi$	$\Sigma \models_{finite\ models} \varphi$
fd	fd	computable	computable
universal	universal	computable	computable
fd + id	fd	not computable	not computable
fd + kid	fd + kid	computable	computable

The abbreviations used within the table are :

functional dependency	denotes	fd
universal formula	denotes	universal
(k)id	denotes	(key)inclusion dependency

4 Normal forms

Earlier, we discussed functional dependencies, and two types of schema decompositions, namely, lossless join and dependency preserving. However,we may wonder why is a decomposition desirable at all ? We illustrate the need shortly.

Assume we have a database schema $R[id, name, cid, faculty, grade]$, which describes the registration and achievement information about students in a given semester.

- *id* and *name* are the students id number and name , respectively.
- *cid* is the course number.
- $faculty$ is the faculty offering the course with number *cid*.
- *grade* is the grade of the student in course *cid*.

Obviously, schema R has the following functional dependencies :

$$F = \{id \rightarrow name, cid \rightarrow faculty, \{id, cid\} \rightarrow grade\}$$

We notice, given the above fd's , that this schema has a number of deficiencies :

- **Redundancy** - each student can take more than one course, and the same course can be taken by many students. However, we repeat the student's details (the name field), and the course's details (the faculty) for each occurrence.
- **Insertion and Update anomalies** - Suppose a course moves to a different faculty. Now, we have to find each occurrence of the course in the table and update it. If only some of the occurrences are updated, we obtain an inconsistent state. Another problem is that we have difficulties expressing some facts, such as a course that no student currently takes. In this case we have to insert rows to the table with nulls on some of the fields. Also, a student that takes no course would be represented in our database with null values in the fields $\{cid, faculty, grade\}$. This leads to many problems during updates and deletions. For example, if this student now registers to some course, than the previous record (with the nulls) has to be deleted. Also, before a student deletes the last registration to courses ,a new record must be inserted with his/ her *id*, and *name* , and nulls in the other fields, unless we wish to loose information about the student completely.

- **Storage minimization** Looking at the course registration database presented, we notice that avoiding redundancy would also result in storage minimization. For example, if we had not recorded in R on each occurrence of some course cid, the $faculty$ field, and saved it in a different table, would would have rendered our database smaller in storage.

Normal forms come in aid at this point and help us fix the above deficiencies. Normal forms are essentially conditions imposed on the relation schemas, such that if these conditions are satisfied, then the relation schemas exhibit desired properties, such as removal of update and insertion anomalies, redundancy avoidance, and storage minimization.

We shall first define $BCNF$. Next, we elaborate on the factors in the above list, and show how $BCNF$ handles them. Proofs are not always given, however, the interested reader can find them in relevant literature.

4.1 $BCNF$

The main idea behind $BCNF$ is that each fact should occur only once in the database.

Definition 4.1 *Given a set of fd's F over schema $\mathcal{R}[U]$ then we define F^{key} as*

$$F^{key} = \{X \rightarrow U \mid F \vdash X \rightarrow U \ \wedge \ X \text{ is minimal } \}$$

Definition 4.2 *Given a set of attributes U, a schema $\mathcal{R}[U]$, and a set of fd's F over $\mathcal{R}$ then* **R *is in* BCNF** *given F,* ***iff*** *$F^{key} \models F$. (or equivalently : for every non trivial functional dependency, $X \rightarrow Y \in F^+$, X is a superkey).*

- $\mathcal{R}[A1, A2]$ where $A1, A2$ are single attributes, is always in $BCNF$, for every F. This is so, because the only two possible non trivial fd's are $A1 \rightarrow A2$ and $A2 \rightarrow A1$, and in each case the left hand side is a superkey.
- $\mathcal{R}[A1, A2, A3]$ given fd $A2 \rightarrow A3$ is not in $BCNF$ since :

  $$F^{key} = \{A1, A2 \rightarrow A1, A2, A3\}$$

  and $F^{key} \not\models F$.

Now that we know what $BCNF$ means, we examine the various problems in database schema design introduced earlier, and show how $BCNF$ helps remedy them.

4.2 Redundancy

Definition 4.3 *Let $\mathcal{R}, F$ be a relation schema, and a set of functional dependencies, respectively.*
$\mathcal{R}$ is $F - redundant$ ($F^+ - redundant$) on XY if there exists a relation $\mathbf{r} \models F$ and a non-trivial FD $X \to Y \in F$ ($\in F^+$), and at least two distinct tuples $t_1, t_2 \in \mathbf{r}$ with $t_1[XY] = t_2[XY]$

Definition 4.4 *The set of attributes of the form XY*

- *with $X \to Y \in F$ and not trivial, are called* explicit facts.
- *with $X \to Y \in F^+$ and not trivial, are called* implicit facts.

As mentioned earlier, the idea behind redundancy is that each fact (either explicit or implicit) should only be stored once. Thus, if $\mathcal{R}$ is redundant on a fact, that fact should be stored in a different table.

Next we present a theorem, without a proof, that shows that $BCNF$ helps us avoid redundancy.

Theorem 4.5 ***(Bernstein, Goodman 1980; M.W.Vincent 1994)***
The following are equivalent :

1. *$\mathcal{R}, F$ is in $BCNF$.*
2. *$\mathcal{R}, F$ is not $F - redundant$.*
3. *$\mathcal{R}, F$ is not $F^+ - redundant$.*

4.3 Insertion anomalies

We next define **insertion violation**.
Given a relation schema $\mathcal{R}[U]$, and a set of functional dependencies F, suppose we have a set of candidate keys given by F_{key}.
Let $\mathbf{r}$ be a relation over $\mathcal{R}$ with $\mathbf{r} \models F$, and let $t[U]$ be a tuple we want to insert to $\mathbf{r}$.
Before insertion we check whether $\mathbf{r} \bigcup \{t[U]\} \models F_{key}$. If it does, then t is inserted to $\mathbf{r}$. Otherwise, it is rejected.

Definition 4.6 *If $t[U]$ was inserted to $\mathbf{r}$ but $\mathbf{r} \bigcup \{t[U]\} \not\models F$, we say that $t[U]$ is an insertion violation.*

Definition 4.7 *We say that $\mathcal{R}, F$ has an **insertion anomaly** if there exists* $\mathbf{r}$ *and $t[U]$, which is an insertion violation.*

Example Let us look at $\mathcal{R}[A, B, C]$ with $F = \{A \rightarrow B, B \rightarrow C\}$.

r		
A	B	C
a1	b1	c1
a2	b2	c2

We want to insert the tuple (a3,b1,c3). We have $F_{key} = \{A \rightarrow BC\}$ and thus this tuple is compatible with it. However, after the insertion we have a violation of $B \rightarrow C$, and thus, $\mathcal{R}, F$ has an insertion anomaly.

r		
A	B	C
a1	b1	c1
a2	b2	c2
a3	b1	c3

Next we present a theorem that shows how $BCNF$ avoids insertion anomalies.

Theorem 4.8 ***(R.Fagin,1979)***
*$\mathcal{R}, F$ is in $BCNF$ **iff** it has no insertion anomalies.*

proof

- $\Rightarrow$ If $\mathcal{R}, F$ is in $BCNF$ then $F_{key} \models F$, by definition. Also $\mathbf{r} \models F$,and now suppose we insert a tuple t which satisfies $\mathbf{r} \bigcup \{t\} \models F_{key}$. Thus it follows that $\mathbf{r} \bigcup \{t\} \models F$.

- $\Leftarrow$ the reader is referred to literature.

4.4 Deletion anomalies

We next define **deletion anomalies**.
Given a relation schema $\mathcal{R}[U]$, and a set of functional dependencies F, suppose we have a set of candidate keys given by F_{key}.
Let $\mathbf{r}$ be a relation over $\mathcal{R}$ with $\mathbf{r} \models F$, and let $t[U]$ be a tuple we want to delete from $\mathbf{r}$.
Before deletion we check whether $\mathbf{r} \backslash \{t[U]\} \models F_{key}$. If it does, then t is deleted from $\mathbf{r}$. Otherwise, the deletion is rejected.

Definition 4.9 *If $t[U]$ was deleted from $\mathbf{r}$ but $\mathbf{r} \backslash \{t[U]\} \not\models F$, we say that $t[U]$ is a deletion violation.*

Definition 4.10 *We say that $\mathcal{R}, F$ has an* ***deletion anomaly*** *if there exists $\mathbf{r}$ and $t[U] \in \mathbf{r}$, which make a deletion violation.*

Observation 4.11 *Let $\mathbf{r}$ be a relation for $\mathcal{R}$ and F be a set of function dependencies.*
Let $\mathbf{s} \subseteq \mathbf{r}$ be another relation for $\mathcal{R}$.
Then, if $\mathbf{r} \models F$, so also $\mathbf{s} \models F$.

conclusion

There are no deletion anomalies for functional dependencies.
Note: In the presence of Multivalued Dependencies (MVD's) there may occur deletion anomalies.

4.5 Modification anomalies

We next define **modification violation**.
Given is a relation schema $\mathcal{R}[U]$, a set of functional dependencies F, and a set of candidate keys F_{key}.

Let $\mathbf{r}$ be a relation over $\mathcal{R}$ with $\mathbf{r} \models F$, and let $t[U]$ be a tuple in $\mathbf{r}$, $t \in \mathbf{r}$.

Also let K_0 be a fixed candidate key for $\mathcal{R}$, and t' be a tuple such that $(\mathbf{r} \setminus t) \bigcup \{t'\} \models F_{key}$ and one of the followings :

1. $t[K] = t'[K]$ for some candidate key for F.
2. $t[K_0] = t'[K_0]$.
3. $t[K] = t'[K]$ for every candidate key for F.

but $(\mathbf{r} \setminus t) \bigcup \{t'\} \not\models F$.

Then $\mathbf{r}$ and t' exhibit a *modification anomaly* M_1, M_2, M_3, respectively.

Clearly, every M_3 anomaly is also an M_2 anomaly and every M_2 anomaly is also an M_1 anomaly.

Observation 4.12 *If $\mathcal{R}, F$ is in $BCNF$ then it has no modification anomaly M_1 (and hence neither M_2 nor M_3).*

Proof Follows directly from $BCNF$ definition in which $F_{key} \models F$.

Next we present a strong theorem that shows the equivalence between the absence of modification anomalies and $BCNF$:

Theorem 4.13 ***(M.W.Vincent, 1994)***
The following are equivalent :

1. *$\mathcal{R}, F$ is in $BCNF$*
2. *$\mathcal{R}, F$ has no modification anomaly M_1*
3. *$\mathcal{R}, F$ has no modification anomaly M_2*

Vincent also introduces a new normal form which is weaker than $BCNF$ but stronger than $3NF$, and is characterized by the absence of M_3 modification anomalies.

4.6 Unpredictable insertions

Let $\mathcal{R}[U]\,, F$ be a relation scheme, and a set of functional dependencies, respectively.

Definition 4.14 *An insertion of a tuple t into $\mathbf{r} \models F$ is said to be* $\mathbf{F-valid}$*, if* $\quad \mathbf{r} \bigcup \{t\} \models F$.

Definition 4.15 *A set of attributes $X \subseteq U$ is said to be* unaffected *by a valid insertion $\mathbf{r}' = \mathbf{r} \bigcup \{t\}$ iff* $\quad \Pi_X(\mathbf{r}) = \Pi_X(\mathbf{r}')$.

Definition 4.16 *A valid insertion is F-*unpredictable *(F^+-*unpredictable*) if there exists a non trivial dependency $X \rightarrow Y \in F$ ($X \rightarrow Y \in F^+$) such that XY is* unaffected *by it.*

Example Let us look at $\mathcal{R}[A, B, C]$ with $F = \{A \rightarrow B, BC \rightarrow A\}$, and examine the dependency $A \rightarrow B$:

r		
A	B	C
a1	b1	c1

after inserting $t = (a1, b1, c2)$ we get :

r		
A	B	C
a1	b1	c1
a1	b1	c2

Clearly, this is a valid insertion but AB is not affected. Thus, it is F-unpredictable, which implies it is also F^+-unpredictable.

Observation 4.17 *If $\mathcal{R}$, F has an F^+-unpredictable insertion, then it is not in BCNF.*

Proof Let $\mathbf{r}$ be a relation on $\mathcal{R}$ and with $t \in \mathbf{r}$, such that $\mathbf{r} \models F$, hence $\mathbf{r} \models F_{key}$.
Assume there is some non-trivial dependency $X \to Y \in F^+$, and a tuple t' is inserted such that $t' \neq t$ but $t[XY] = t'[XY]$.
Assume, by way of contradiction, that R, F is in $BCNF$. So, by definition, X must be a superkey for F. But $\mathbf{r} \bigcup \{t'\} \models F_{key}$, so $t = t' \quad \Rightarrow$ a contradiction. ∎

The following theorem shows how $BCNF$ helps us avoid unpredictable insertions :

Theorem 4.18 ***(Bernstein, Goodman, 1980)***
The following are equivalent :

1. *$\mathcal{R}, F$ is in $BCNF$*
2. *$\mathcal{R}, F$ has no F-unpredictable insertions.*
3. *$\mathcal{R}, F$ has no F^+-unpredictable insertions.*

4.7 Storage minimization

Let $\mathcal{R}[U]$, F be a relation scheme, and Π_{U_i} be an *information preserving decomposition*, i.e. $\bowtie_i \ R_i[U_i] = R \models F$.

Definition 4.19 *We say that the decomposition of $\mathcal{R}$ is* storage saving *if there are instances $\mathbf{r} = \bowtie_i \mathbf{r}_i$ such that $\sum_i |\mathbf{r}_i| \leq |\mathbf{r}|$.*

Next we show a theorem which shows, again, the strength of $BCNF$ in storage minimization.

Theorem 4.20 ***(Biskup; Vincent and Srinivasan)***
$\mathcal{R}$, F is in $BCNF$ iff it has no storage saving decomposition.

4.8 Can we always achieve $BCNF$?

It is known that there are relation schemes $\mathcal{R}[U]$ along with a set of functional dependencies F such that $\mathcal{R}$ is not in $BCNF$ and there *does not exist a decomposition of R via projections which is information preserving and dependency preserving.*

Example recall our first example of schema $\mathcal{R}[C, S, Z]$,where C stands for city, S stands for street, and Z stands for zipcode, with the following FD's F :

1. $Z \to C$
2. $C, S \to Z$

We note that $\mathcal{R}$ is not in $BCNF$, since in the first dependency, Z is not a superkey. However, in every decomposition of $\mathcal{R}$, the dependency $C, S \to Z$ would not be preserved, since no decomposition would contain more than 2 of the schema's attributes.

4.9 Achieving $BCNF$ via splitting attributes

As pointed out in the previous sub-section $BCNF$ can not always be achieved via mere decomposition of a schema to sub-schemas such that the decomposition is information and dependency preserving.

However, there are two solutions that can be implemented in order to achieve $BCNF$ even in such cases. We demonstrate them with our original schema $\mathcal{R}[C, S, Z]$.

1. Drop some of the functional dependencies. In our case, the dependency $Z \to C$ would be dropped. A side effect of this solution is that the character of postal distribution changes. Such a solution would not always be desirable.
2. Split some of the attributes.
   In our case we would split attribute Z, which stands for the zip code, to 2 attributes : Z_{city} and Z_{local}. The functional dependencies resulting from this change would be :
   (a) $CS \to Z_{city}, Z_{local}$
   (b) $Z_{city} \to C$

Now we can decompose $\mathcal{R}$ to $\mathcal{R}_1[C, Z_{city}]$ and $\mathcal{R}_2[C, S, Z_{local}]$ which would be an information and dependency preserving decomposition and each of the resulting sub schemas R_1 and R_2 would be in $BCNF$.

As a matter of fact, this solution is implemented in many countries.

Theorem 4.21 ***(Makowsky, Ravve 1998)***
Every relation scheme $\mathcal{R}, F$ can be modified preserving information and dependencies via decomposition and splitting attributes.

Furthermore, this modification can be computed using a combination of the synthesis algorithm of $3NF$ and splitting attributes.

5 Final remarks and conclusions

1. In the previous chapter we showed the use of normal forms, and specifically $BCNF$ as a means of achieving good database design.

   We have seen how $BCNF$ handles the various anomalies encountered in bad database design in situations of updates, modifications, and deletions. We also showed that $BCNF$ achieves minimum storage.

   We observed that decomposition alone can not always help us achieve $BCNF$ and introduced the notion of *attribute splitting* as a means of fixing the database design.

2. In chapter 3 we have examined other dependencies (besides functional dependencies) and showed how they could be formulated using mathematical logic notation with the introduction of FID (full implication dependencies) sentences.

   We have seen that implication among dependencies is computable with functional and key inclusion dependencies, but not with general inclusion dependencies. Hence, the latter should be avoid in database design.