Normalisation

Informal guidelines
- Semantics of the attributes
- easy to explain relation
- doesn’t mix concepts
- Reducing the redundant values in tuples
- Choosing attribute domains that are atomic
- Reducing the null values in tuples
- Disallowing spurious tuples

Functional Dependency
- an attribute A is functionally dependent on a set of attributes X if and only if
  - value of A is determined solely by the values of X
  - values of X uniquely determine a value of A

  \[
  X \rightarrow A \\
  \text{child} \rightarrow \text{mother} \\
  \text{mother} \not\rightarrow \text{child}
  \]

  The value of child implies the value of mother
  Value of mother does NOT imply value of child
  Child is the determinant
  Mother is the dependent/determined

Our case study example

Use functional dependencies to …
check that a relation is legal or good. e.g keys

- K is a superkey of relation R if K \rightarrow R

  i.e.
  whenever 11[k] = 12[k]
  then 11[R] = 12[R]

  K functionally determines all attributes in a tuple in R

STUDENT(studno, name, hons, tutor, slot, year)

studno \rightarrow studno, name, hons, tutor, slot, year
Use functional dependencies to …
check that a relation is legal or good. e.g. remove redundancy
- Partial Dependency
  studno, courseno → subject
  (studno, courseno, subject)
- Transitive Dependency
  studno → yeartutor
  yeartutor → yeartutor
  studno → yeartutor
  (studno, yeartutor)
- Base functional dependencies F
- Set of logically implied functional dependencies_closure F+

Normalisation
Given a relation R with a set of functional dependencies F, and a key K
We must identify independent attributes
1. the key identifies all the attributes but…
2. ... if an attribute only depends on part of the key, then it is independent of the rest of it.
   Attribute is partially dependent on the key
3. ... if an attribute only depends on the key transitively, then it really depends directly on another attribute and is independent of the key.
   Attribute is transitively dependent on the key

Boyce-Codd Normal Form
A relation scheme R is in BCNF if, for all functional dependencies that hold on R of the form X → Y where R ⊇ X and R ⊇ Y at least one of the following holds
- X → Y is trivial
- X is a candidate key for the scheme R i.e. X → R
Every attribute must depend on the key, the whole key and nothing but the key
- Other Normal Forms: 1NF, 2NF and 3NF ... uses primary key only
- BCNF... generalised for candidate keys

Use functional dependencies to …
check constraints on the set of legal relations

Consequences of redundancy
- Wasted space
- Potential performance cost
- Potential inconsistency
- Inability to represent data

ReturnHistory
<table>
<thead>
<tr>
<th>readerid</th>
<th>bookid</th>
<th>date</th>
<th>fine</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Macbeth</td>
<td>7/6</td>
<td>0</td>
</tr>
<tr>
<td>123</td>
<td>Macbeth</td>
<td>7/6</td>
<td>0</td>
</tr>
<tr>
<td>123</td>
<td>Hamlet</td>
<td>1/6</td>
<td>0</td>
</tr>
<tr>
<td>456</td>
<td>Macbeth</td>
<td>8/9</td>
<td>2</td>
</tr>
</tbody>
</table>

Many:many relationships that could be weak entity types because they have hidden partial keys.
Using Functional Dependencies to check EER mappings

Attributes on wrong entities

- \( \text{STUDENT(studno, name, labmark)} \)
  - \( \text{studno} \rightarrow \text{name} \)
  - \( \text{studno} \rightarrow \text{labmark} ? \)
- \( \text{COURSE(courseno, subject, roomno)} \)
  - \( \text{courseno} \rightarrow \text{subject} \)
  - \( \text{courseno} \rightarrow \text{roomno} ? \)
- \( \text{STAFF(staffname, salary)} \)
  - \( \text{staffname} \rightarrow \text{salary} \)
  - \( \text{where is staffname} \rightarrow \text{roomno} ? \)

Wrong cardinalities on a relationship type

- \( \text{STUDENT(studno, name)} \)
  - \( \text{studno} \rightarrow \text{name} \)
- \( \text{COURSE(courseno, subject, studno)} \)
  - \( \text{courseno} \rightarrow \text{subject} \)
  - \( \text{courseno} \rightarrow \text{studno} ? \)

Functional Dependencies are hidden in EER Model

- \( \text{COURSE (courseno, subject, lecturer, roomno)} \)
  - \( \text{courseno} \rightarrow \text{subject} \)
  - \( \text{courseno} \rightarrow \text{ lecturer} ? \)
  - \( \text{courseno} \rightarrow \text{roomno} \)
  - \( \text{lecturer} \rightarrow \text{roomno} \)

Using the EER Model and Functional Dependencies

1. Draw EER model
2. Map EER schema to relational schema
3. For every relation
   - List the functional dependencies
   - what does determine every attribute?
   - Check that every relation is in BCNF
     - does the key really solely uniquely identify each attribute?
     - If its not in BCNF then why?
     - Fix the problem
     - normalize and/or
     - trace back to EER model
4. Are there any functional dependencies missing?
5. Optimise the relational schema

Database design

- Extended Entity Relationship
- Top Down
- Conceptual/Abstract View
- Functional Dependencies
- Bottom Up
- Implementation View
- The Determinancy Approach
- Synthesise relations
1. List all attributes
2. Consider the relationships between them
   - those which determine the values of others are entities
   - those whose values are determined by other items are attributes.

Using the EER Model and Functional Dependencies

- Extended Entity Relationship
- Top Down
- Conceptual/Abstract View
- Functional Dependencies
- Bottom Up
- Implementation View
- The Determinancy Approach
- Synthesise relations
1. List all attributes
2. Consider the relationships between them
   - those which determine the values of others are entities
   - those whose values are determined by other items are attributes.
Use functional dependencies to...Synthesise relations

STUDENT

studno, givenname, familyname, hons, tutor, slot, year

studno, courseno, labmark

ENROL(studno, courseno, labmark, exammark)

courseno, subject, equip

COURSE(courseno, subject, equip)

studno

STAFF(lecturer, roomno, appraiser)

lecture, lecturer

TEACH(courseno, lecturer)

courseno, lecturer, num_of_lectures

year, yeartutor

YEAR(year, yeartutor)

SCHOOL(.home, faculty)

Complementary Approaches

- Disadvantages of EER Top Down
  1. Not all entity types are represented by nouns or noun-phrases
     - association entity types
  2. Not all nouns and noun-phrases correspond to entities
     - single attribute entities
- Disadvantages of determinancy bottom-up
  1. Long-winded
  2. Hides overall picture of data model

The Steps of Normalisation

- Take one dependency at a time
- Treat each relation separately and independently
- Iterative process

Use functional dependencies to...

NORMALISE relations

- Systematically create legal relations
- Derive relations which avoid anomalies in
  - Insertion
  - Deletion
  - Modification
  - Accessing
- Ensure single valued-ness of facts represented in attributes in keyed relations
- Ensure the removal of redundancy in a relation

Normalisation

- Given
  - a universal relation that is unnormalised
  - a set of functional dependencies on the attributes in the relation
- produce a set of relations where each relation is normalised for the functional dependencies on the attributes in the relation
- Three approaches:
  1. Relational synthesis
  2. Step-wise normalisation
  3. Using BCNF decomposition
The Process of Normalisation

- Usually four steps giving rise to
  - First Normal Form (1NF)
  - Second Normal Form (2NF)
  - Third Normal Form (3NF)
  - Boyce-Codd Normal Form (BCNF)
  - Fourth Normal Form (4NF)

At each step we consider relationships between the functional dependencies of a relation’s attributes.
Normalisation is a:
- framework
- series of tests

## First Normal Form: Repeating Groups

### STUDENT

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>tutor</th>
<th>roomno</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Jones</td>
<td>Bush</td>
<td>2.26</td>
</tr>
<tr>
<td>2</td>
<td>Brown</td>
<td>Kahn</td>
<td>T206</td>
</tr>
</tbody>
</table>

### STUDENT_DETAILS

- STUDENT: name, tutor, roomno
  - STUDENT: name, tutor
  - STUDENT: roomno
  - STUDENT: tutor
   - ENROL: studno, courseno, subject, labmark
     - studno
     - courseno
     - subject
     - labmark

### ENROL

```
ENROL: studno, courseno, subject, labmark
studno: course

ENROL: studno, courseno, subject, labmark
studno: course
```

### Benefits from First Normal Form

- Any ‘hidden’ relations (entities) are identified.
- Process results in separation of different objects.
- BUT anomalies may still exist.

### Second Normal Form

- A relation is in 2NF if it is in 1NF and each non-identifying attribute depends upon the whole key (identifier).
- Any relation in 1NF is transformed to 2NF.
- Identify functional dependencies.
- Re-write relations so that each non-identifying attribute is functionally dependent on the whole key.
- Decompose ENROL into two relations:
  - ENROL: studno, courseno, subject, labmark
  - COURSE: courseno, subject

### Second Normal Form

```
STUDENT: studno, name, tutor, roomno

ENROL: studno, courseno, subject, labmark

COURSE: courseno, subject
```
Third Normal Form

- An relation is in 3NF if it is in 2NF and all non-identifying attributes are independent
- Any relation in 2NF is transformed in 3NF
- Determine functional dependencies between non-identifying attributes
- Decompose relation into new relations

\[
\begin{align*}
\text{STUDENT} & \quad \text{studno} \rightarrow \text{name, tutor, roomno} \\
\text{TUTOR} & \quad \text{tutor} \rightarrow \text{roomno} \\
& \quad \text{roomno} \rightarrow \text{tutor}
\end{align*}
\]

Student Relational Schema in 3NF

- \text{STUDENT (studno, name, tutor)}
  \quad \text{studno} \rightarrow \text{name, tutor}
- \text{TUTOR (tutor, roomno)}
  \quad \text{tutor} \rightarrow \text{roomno}
  \quad \text{roomno} \rightarrow \text{tutor}
- \text{ENROL (studno, courseno, labmark)}
  \quad \text{studno, courseno} \rightarrow \text{labmark}
- \text{COURSE (courseno, subject)}
  \quad \text{courseno} \rightarrow \text{subject}

Decomposition: Lossless or Non-additive Join

- R is a relational scheme, F is a set of functional dependencies on R. R1 and R2 form a decomposition of R.
- The decomposition of R is non-additive if at least one of the following functional dependencies are in \( F^+ \) of R:
  \[
  \begin{align*}
  R1 \cap R2 & \rightarrow R1 \\
  R1 \cap R2 & \rightarrow R2
  \end{align*}
  \]
- The decomposition of R is non-additive if for every state \( r \) of R that satisfies F:
  \[
  \pi_{R1} (r) \times \pi_{R2} (r) = r
  \]
  where \( \times \) condition is the natural join

Spurious Tuples Lossless or Non-additive Join

- \text{TEACH (courseno, lecturer, num_lecture)}
  \quad \text{courseno} \rightarrow \text{lecturer}
  \quad \text{courseno, lecturer} \rightarrow \text{num_lecture}
- \text{TEACH' (courseno, lecturer, num_lecture)}
  \quad \text{courseno} \rightarrow \text{lecturer}
  \quad \text{courseno, lecturer} \rightarrow \text{num_lecture}

Lossless or Non-additive Join

- \text{STUDENT (studno, name, tutor, roomno)}
  \quad \text{studno} \rightarrow \text{name, tutor, roomno}
  \quad \text{tutor} \rightarrow \text{roomno}
  \quad \text{roomno} \rightarrow \text{tutor}
- \text{STUDENT1 (tutor = tutor) TUTORS = STUDENT}
Decomposition Algorithm:
Decomposition D, relation R

- set D = \{ R \} ;
- while there is a relation schema Q in D that is not in BCNF do
  - begin
    - choose a relation schema Q in D that is not in BCNF;
    - find a functional dependency X \rightarrow Y in Q that violates BCNF;
    - violation means that (X)+ fails to find all of Q, so X can’t
      be a key.
    - replace Q in D by two schemas
      - R1 (Q - (Y)+ \cup X)
        - leave copy of X in relation to be the foreign key for R2
      and
      - R2 (X \cup (Y)+)
        - new relation for functional dependency and its closure, X
          will be the primary key
    - end;

Decomposition: Dependency Preservation

- When an update is made to a database, should be able to check that update satisfies all functional dependencies.
- It is desirable to allow validation of relational database schemes that allow update validation without the computation of joins.
- independent manipulation of relations.

Dependency Preservation

- The union of dependencies that hold on the individual relations in decomposition D must be equivalent to F.
- Given F on R, \Pi_r(R) where R_1 \subseteq R
  is the set of dependencies X \rightarrow Y in F
  such that the attributes in X \cup Y are all contained in R
- Decomposition D = \{R_1, R_2, ..., R_m\} of R is dependency preserving w.r.t. F if
  \( (\Pi_r(R_1)) \cup ... \cup (\Pi_r(R_m)) = F \)
- Given the restriction of functional dependencies to a
  relation is the fds that involve attributes of that relation F_i
  for R_i:
  \( \bigcup_{i=1}^{n} F_i \neq F \) possible, but...
  \( (\bigcup_{i=1}^{n} F_i)^+ = F^+ \)

Dependency Preservation

- STUDENT (studno, name, tutor, roomno, appraiser)
  - studno \rightarrow name, tutor
  - tutor \rightarrow roomno, appraiser
  - roomno \rightarrow tutor, appraiser

- STUDENT1 (studno, name, tutor)
  - studno \rightarrow name, tutor

- TUTOR (studno, roomno, appraiser)
  - studno \rightarrow roomno, appraiser

This is in Boyce-Codd Normal Form and is a lossless (nonadditive) join decomposition but we have lost....
- tutor \rightarrow roomno, appraiser
- roomno \rightarrow tutor, appraiser

Lossless or Non-additive Join

Dependency Preservation

- student \rightarrow name
- student \rightarrow tutor
- tutor \rightarrow roomno
- tutor \rightarrow appraiser
- roomno \rightarrow tutor
- roomno \rightarrow appraiser
- student \rightarrow appraiser
- student \rightarrow roomno

STUDENT

<table>
<thead>
<tr>
<th>studno</th>
<th>name</th>
<th>tutor</th>
<th>roomno</th>
<th>appraiser</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>Jones</td>
<td>Bush</td>
<td>2.26</td>
<td>Japan</td>
</tr>
<tr>
<td>s2</td>
<td>Brown</td>
<td>Kahn</td>
<td>IT206</td>
<td>Watson</td>
</tr>
<tr>
<td>s3</td>
<td>Smith</td>
<td>Gordon</td>
<td>2.82</td>
<td>Japan</td>
</tr>
<tr>
<td>s4</td>
<td>Bloggs</td>
<td>Goble</td>
<td>2.82</td>
<td>Japan</td>
</tr>
<tr>
<td>s5</td>
<td>Jones</td>
<td>Goble</td>
<td>2.34</td>
<td>Watson</td>
</tr>
<tr>
<td>s6</td>
<td>Peters</td>
<td>Kahn</td>
<td>IT206</td>
<td>Watson</td>
</tr>
</tbody>
</table>

TUTOR

<table>
<thead>
<tr>
<th>studno</th>
<th>name</th>
<th>tutor</th>
<th>roomno</th>
<th>appraiser</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>Jones</td>
<td>Bush</td>
<td>2.26</td>
<td>Japan</td>
</tr>
<tr>
<td>t2</td>
<td>Brown</td>
<td>Kahn</td>
<td>IT206</td>
<td>Watson</td>
</tr>
<tr>
<td>t3</td>
<td>Smith</td>
<td>Gordon</td>
<td>2.82</td>
<td>Japan</td>
</tr>
<tr>
<td>t4</td>
<td>Bloggs</td>
<td>Goble</td>
<td>2.82</td>
<td>Japan</td>
</tr>
<tr>
<td>t5</td>
<td>Jones</td>
<td>Goble</td>
<td>2.34</td>
<td>Watson</td>
</tr>
<tr>
<td>t6</td>
<td>Peters</td>
<td>Kahn</td>
<td>IT206</td>
<td>Watson</td>
</tr>
</tbody>
</table>
Designing a relational schema

- Build a relational database
- without redundancy
- normalization
- without loss of information or gain of data
- lossless join decomposition
- without losing dependency integrity
- dependency preservation

Multi-valued Dependencies

- a course has many lecturers
- a course has many texts
- lecturers and texts are independent
- a lecturer teaches many courses
- a text is used by many courses
- lecturer and text are independent sets
- for each course there is an associated set of lecturers
- for each course there is an associated set of texts
- the sets are independent

Multi-valued Dependencies and Fourth Normal Form

- Multi-valued Dependencies
- Course has many lecturers
- Course has many texts
- Lecturers and texts are independent
- A lecturer teaches many courses
- A text is used by many courses
- Lecturer and text are independent sets
- For each course there is an associated set of lecturers
- For each course there is an associated set of texts
- The sets are independent

Multi-valued Dependencies

- Each TEXT is associated with all the LECTURERS that teach a COURSE
- The attribute TEXT contains redundant values.
- If TEXT were deleted from rows 1, 2 & 3 the values could be deduced from rows 4, 5 & 6

Multi-valued Dependencies

- Course has many lecturers
- Course has many texts
- Lecturers and texts are independent
- A lecturer teaches many courses
- A text is used by many courses
- Lecturer and text are independent sets
- For each course there is an associated set of lecturers
- For each course there is an associated set of texts
- The sets are independent

Multi-valued Dependencies

- This is in BCNF
- Key is [courseno, lecturer, text]
- Trivial dependencies
- If (c, l, t) and (c, l', t) appear then
- Tuple (c, l', t) appears if c can be taught by l using text t
- For each course all possible combinations of lecturer and text appear

Multi-valued Dependencies

- If (c, l, t) and (c, l', t') appear then
- (c, l', t') and (c, l, t) appear also
- For each course all possible combinations of lecturer and text appear

<table>
<thead>
<tr>
<th>Coursesno</th>
<th>Lecturer</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lindsey</td>
<td>Intro to SML</td>
</tr>
<tr>
<td>2</td>
<td>Lindsey</td>
<td>SML for Beginners</td>
</tr>
<tr>
<td>3</td>
<td>Lindsey</td>
<td>More SML</td>
</tr>
<tr>
<td>4</td>
<td>Bush</td>
<td>Ray Tracing for Fun</td>
</tr>
<tr>
<td>5</td>
<td>Zobel</td>
<td>Chips with everything</td>
</tr>
<tr>
<td>6</td>
<td>Capon</td>
<td>Object Design</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Coursesno</th>
<th>Lecturer</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lindsey</td>
<td>Intro to SML</td>
</tr>
<tr>
<td>2</td>
<td>Lindsey</td>
<td>SML for Beginners</td>
</tr>
<tr>
<td>3</td>
<td>Lindsey</td>
<td>More SML</td>
</tr>
<tr>
<td>4</td>
<td>Bush</td>
<td>Ray Tracing for Fun</td>
</tr>
<tr>
<td>5</td>
<td>Zobel</td>
<td>Chips with everything</td>
</tr>
<tr>
<td>6</td>
<td>Capon</td>
<td>Object Design</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Coursesno</th>
<th>Lecturer</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lindsey</td>
<td>Intro to SML</td>
</tr>
<tr>
<td>2</td>
<td>Lindsey</td>
<td>SML for Beginners</td>
</tr>
<tr>
<td>3</td>
<td>Lindsey</td>
<td>More SML</td>
</tr>
<tr>
<td>4</td>
<td>Bush</td>
<td>Ray Tracing for Fun</td>
</tr>
<tr>
<td>5</td>
<td>Zobel</td>
<td>Chips with everything</td>
</tr>
<tr>
<td>6</td>
<td>Capon</td>
<td>Object Design</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Coursesno</th>
<th>Lecturer</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lindsey</td>
<td>Intro to SML</td>
</tr>
<tr>
<td>2</td>
<td>Lindsey</td>
<td>SML for Beginners</td>
</tr>
<tr>
<td>3</td>
<td>Lindsey</td>
<td>More SML</td>
</tr>
<tr>
<td>4</td>
<td>Bush</td>
<td>Ray Tracing for Fun</td>
</tr>
<tr>
<td>5</td>
<td>Zobel</td>
<td>Chips with everything</td>
</tr>
<tr>
<td>6</td>
<td>Capon</td>
<td>Object Design</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Coursesno</th>
<th>Lecturer</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lindsey</td>
<td>Intro to SML</td>
</tr>
<tr>
<td>2</td>
<td>Lindsey</td>
<td>SML for Beginners</td>
</tr>
<tr>
<td>3</td>
<td>Lindsey</td>
<td>More SML</td>
</tr>
<tr>
<td>4</td>
<td>Bush</td>
<td>Ray Tracing for Fun</td>
</tr>
<tr>
<td>5</td>
<td>Zobel</td>
<td>Chips with everything</td>
</tr>
<tr>
<td>6</td>
<td>Capon</td>
<td>Object Design</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Coursesno</th>
<th>Lecturer</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lindsey</td>
<td>Intro to SML</td>
</tr>
<tr>
<td>2</td>
<td>Lindsey</td>
<td>SML for Beginners</td>
</tr>
<tr>
<td>3</td>
<td>Lindsey</td>
<td>More SML</td>
</tr>
<tr>
<td>4</td>
<td>Bush</td>
<td>Ray Tracing for Fun</td>
</tr>
<tr>
<td>5</td>
<td>Zobel</td>
<td>Chips with everything</td>
</tr>
<tr>
<td>6</td>
<td>Capon</td>
<td>Object Design</td>
</tr>
</tbody>
</table>
Multi-Valued Dependencies

- Whenever \( X \rightarrow Y \) holds in \( R \) so does \( X \rightarrow (R - (XY)) \).
- A MVD is trivial if \( Y \subseteq X \) or \( X \cup Y = R \).
  \( i.e. \) the two attributes form the whole relation
- Non-trivial MV dependencies need at least 3 attributes.

Fourth Normal Form

- A relation \( R \) is in 4NF if it is in 3NF and there are no multi-valued dependencies between its attribute types.
- A relation \( R \) is in 4NF if whenever there exists a non-trivial multi-valued dependency in \( F^+ \) for \( R \)
  \( X \rightarrow Y \)
- \( X \) is a superkey for \( R \), i.e. all attributes are functionally dependent on \( X \).
- Any relation in 3NF is transformed in 4NF
  - Detect any multi-valued dependencies
  - Decompose relation

Fourth Normal Form EER modelling

- Leads to correctly normalised relational schema

Lossless join decomposition into 4NF

- Algorithm:
  Decomposition \( D \), relation \( R \)
  \begin{enumerate}
    \item set \( D \leftarrow \{ R \} \); \n    \item while there is a relation schema \( Q \) in \( D \) that is not in 4NF do \n      choose a relation schema \( Q \) in \( D \) that is not in 4NF; \n      find a non-trivial MVD \( X \rightarrow Y \) in \( Q \) that violates 4NF; \n      replace \( Q \) in \( D \) by two schemas \( (Q - Y) \) and \( (X \cup Y) \); \n  \end{enumerate}
  \nend;

Fourth Normal Form EER modelling

- Leads to relational schema that is not in 4NF
Conclusions

- Data Normalisation is a technique that ensures the basic properties of the relational model
- no duplicate tuples
- no nested relations
- Data normalisation is sometimes used as the only technique for database design—implementation view
- A more appropriate approach is to complement conceptual modelling with data normalisation

Lossless or Non-additive Join Algorithm

Decomposition D, relation R

1. set D := {R} ;
2. while there is a relation schema Q in D that is not in BCNF do begin
   choose a relation schema Q in D that is not in BCNF;
   find a functional dependency X→Y in Q that violates BCNF;
   replace Q in D by two schemas
   R1 (Q - Y) leave copy of X in relation to be foreign key for R2
   and
   R2 (X ∪ Y) new relation for functional dependency and its closure,
   X will be the primary key
end;