- Relation schema T with attributes $Attr(T)$

  Decomposed into relation schemas as follows:

  - T_1 with attributes $Attr(T_1)$
  - T_2 with attributes $Attr(T_2)$
  - $Attr(T) = Attr(T_1) \cup Attr(T_2)$

- <u>Theorem:</u> Assume from the FDs for T we can derive:

  $$Attr(T_1) \cap Attr(T_2) \rightarrow Attr(T_1), \quad \text{or}$$

  $$Attr(T_1) \cap Attr(T_2) \rightarrow Attr(T_2),$$

  That is, $Attr(T_1) \cap Attr(T_2)$ is superkey for T_1 or T_2

  Then, the decomposition of T into T_1 and T_2 is lossless

  (for instances of T that satisfy its FDs)

- A useful criterion that depends on the schema and its dependencies, not on potential instances

- The attributes in the FDs above appear all in T's schema
  Then, FD derivation refers to a single schema

- Example on page 90 revisited: $T(A, B, C)$: $B \to C$

  $$Attr(T_1(A, B)) \cap Attr(T_2(B, C)) = \{B\}$$

  Does any of these hold?

  - $B \to C \;\vdash\; B \to AB$? or NO!
  - $B \to C \;\vdash\; B \to BC$? YES!

  Then, the decomposition is lossless for every instance of T that satisfies the FD $B \to C$

  Notice: $B \to C$ is an implied FD for subschema $T_2(B, C)$

- We can decompose a table through several decomposition steps of two tables (obtaining implied FDs for the new tables)

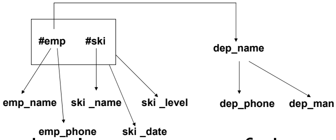
- We can use the theorem with the (derived) FDs to verify that each of the subsequent decompositions is lossless

Back to 2NF:

- Example: (cont.) We had:

  emp_info(#emp, emp_name, emp_phone, dep_name, dep_phone, dep_man, #skill, skill_name, skill_date, skill_level)

- The only candidate key is: {#emp, #skill}



- Relation schema not in 2NF

  Several attributes functionally depend only on a part of the key, e.g. emp_name, ski_level

- Bring the schema into 2NF through a decomposition

- The resulting relations schemas should be in 2NF

- We want a lossless decomposition

  This is not a requirement for 2NF (which is checked on a single relation schema)

emp_info(#emp, emp_name, emp_phone, dep_name, dep_phone, dep_man,
#skill, skill_name, skill_date, skill_level)

- Decomposition 1: Attributes emp_name, emp_phone, dep_name, dep_phone, dep_man depend only partially on the primary key

  And they do not belong to any candidate key

- The latter condition should be checked, considering all possible candidate keys

  In this example there is only a single candidate key

- Break the partial dependency! A possible decomposition is:

  emp1(#emp, emp_name, emp_phone, dep_name, dep_man, dep_phone)

  emp_skill1(#emp,#skill, skill_name, skill_date, skill_level)

- Lossless? Each relation schema in 2NF?

```
emp1(#emp, emp_name, emp_phone, dep_name, dep_man, dep_phone)
emp_skill1(#emp,#skill, skill_name, skill_date, skill_level)
```

- For emp_skill1 we derive that {#emp, #skill} is a key
- In emp_skill1 attribute skill_name partially depends on the key {#emp, #skill} (not belonging to any candidate key)
- Lossless decomposition

  To schema emp_info the theorem can be applied:

  $$Attr(\text{emp1}) \cap Attr(\text{emp\_skill1}) = \{\#\text{emp}\} \longrightarrow Attr(\text{emp1})$$

  Yes! (see page 93)
- First relation schema in 2NF? Yes!
- Second relation schema in 2NF? No! (see above)
- Decomposition 2: To break partial dependency

```
emp1(#emp, emp_name, emp_phone, dep_name, dep_man, dep_phone)
skill(#skill, skill_name)
emp_skill(#emp,#skill, skill_date, skill_level)
```

- One can verify that the decomposition of emp_skill1 into emp_skill and skill is lossless
- Then, the decomposition of emp_info into {emp1, skill, emp_skill} is lossless
- The three resulting relation schemas are now all in 2NF

  And the new resulting relational schema is in 2NF
- Is 2NF good enough?

- Let us see what we have with what we obtained:

  `emp1(#emp, emp_name, emp_phone, dep_name, dep_man, dep_phone)`
  `skill(#skill, skill_name)`

  `emp_skill(#emp,#skill, skill_date, skill_level)`

- Plus the derived FD for `emp1`:

  `dept_name` → `dept_phone`, `dept_man`

  In `emp1` attributes `dep_name`, `dep_man`, `dep_phone` (among others) do not belong to any key for the relation

- For `emp1` there is a transitive FD:

  `#emp` → `dep_name` → `dep_man`, `dep_phone`

  The last two attributes transitively depend on `#emp`

- Any problems with this relation schema?

- An instance could have redundant information: the same dept phone and manager repeated several times

  They have to be the same for each value of `dept_name` `emp1(#emp, emp_name, emp_phone, dep_name, dep_man, dep_phone)`

- Update anomalies?
- What happens if a new department is created?
- It would have no employees (yet)
  We would need values for `#emp`
- We do not want null values in key attributes, `#emp` here
- Better keep info about departments separate
- These issues (redundancy, update anomalies) are related to the issues we uncovered in the previous slide (on dependencies and keys) ...
- To avoid these problems we move on to the 3NF

3NF: First an important notion ...

Definition: Attribute B of relation schema $R(\mathbf{A})$ is non-prime
iff for every candidate key $\mathbf{A'} \subseteq \mathbf{A}$ for R: $B \notin \mathbf{A'}$

- In other words, a non-prime attribute does not belong no any candidate key
- Contrapositively, a prime attribute of a relation belongs to some candidate key for the relation
- We can reformulate 2NF as follows:

  $R(\mathbf{A})$ is in 2NF if, for every non-prime attribute B, there is no candidate key $\mathbf{A'}$ and $\mathbf{A''} \subsetneq \mathbf{A'}$, such that $\mathbf{A''} \rightarrow B$
- `emp1(#emp, emp_name, emp_phone, dep_name, dep_man, dep_phone)`

  with `#emp` $\rightarrow$ `dep_name` $\rightarrow$ `dep_man, dep_phone`

  `dep_name, dep_man, dep_phone` are non-prime

  A transitive dependency involving a non-prime attribute ...

- Informally, a relation schema is in 3NF if it is in 2NF and there are no transitive functional dependencies involving non-prime attributes

- More formally: (A) A relation schema (with explicit and implicit FDs) is in 3NF when the following combination is forbidden:

  1. $\mathbf{A} \to \mathbf{B} \to \mathtt{C}$ ($\mathbf{A}, \mathbf{B}, \mathtt{C}$ any attributes)
  2. $\mathbf{A}$ a candidate key
  3. $\mathbf{B} \not\to \mathbf{A}$
  4. The FDs in 1. are non-trivial (trivial as in Armstrong's first rule)
  5. $\mathtt{C}$ is a non-prime attribute

  (B) Equivalently, the schema is in 3NF when, for every (explicit or implicit) FD $\mathbf{D} \to \mathbf{E}$, at least one of the following holds:

  1. The dependency is trivial, or
  2. $\mathbf{D}$ is a superkey for the relation, or
  3. Each attribute in $\mathbf{E} \setminus \mathbf{D}$ is prime

- Exercise: (a) Verify the equivalence of (A) and (B) above.
  (b) Verify that in both cases the schema is also in 2NF

- The example on page 99 with condition (A):

  emp1(#emp, emp_name, emp_phone, dep_name, dep_man, dep_phone)

  Not in 3NF: $\{\#\texttt{emp}\} \to \{\texttt{dep\_name}\} \to \texttt{dep\_man}$
  is FD of the form $\mathbf{A} \to \mathbf{B} \to \texttt{C}$ and satisfies the combination

- Same example with condition (B):

  emp1(#emp, emp_name, emp_phone, dep_name, dep_man, dep_phone)

  Not in 3NF: $\{\texttt{dept\_name}\} \to \{\texttt{dep\_man}\}$ is FD of the form
  $\mathbf{D} \to \mathbf{E}$, and none of 1.-3. holds
  It is not trivial, dept_name is not a superkey, and
  $\mathbf{E} \setminus \mathbf{D} = \{\texttt{dep\_man}\}$ is non-prime

- Having already 2NF, 3NF becomes easier to verify
- The relation schema n previous example:

  emp1(#emp, emp_name, emp_phone, dep_name, dep_man, dep_phone)

  has transitive dependency: #emp $\rightarrow$ dep_name $\rightarrow$ dep_man, dep_phone

- Non-prime attributes dep_man and dep_phone functionally depend upon non-prime attribute dep_name

  The latter strictly depends on a key attribute

  Then, the schema is not in 3NF

- To break the transitive dependency, decompose via dep_name
- Additional decomposition:

  emp(#emp, emp_name, emp_phone, dep_name)

  dept(dep_name, dep_man, dep_phone)

  skill(#skill, skill_name)

  emp_skill(#emp, #skill, skill_date, skill_level)

$\texttt{emp}(\underline{\texttt{\#emp}}, \texttt{emp\_name}, \texttt{emp\_phone}, \texttt{dep\_name})$

$\texttt{dept}(\underline{\texttt{dep\_name}}, \texttt{dep\_man}, \texttt{dep\_phone})$

$\texttt{skill}(\underline{\texttt{\#skill}}, \texttt{skill\_name})$

$\texttt{emp\_skill}(\underline{\texttt{\#emp}}, \underline{\texttt{\#skill}}, \texttt{skill\_date}, \texttt{skill\_level})$

- Each table does not contain any transitive FD involving non-prime attributes
- New relational schema is in 3NF
- The problem of redundancy disappeared

  We can create new departments without worrying about employees
- Is the decomposition of the first table `emp1` lossless?

  Yes! Apply "the theorem" to `emp1`:

  $\mathit{Attr}(\texttt{emp}) \cap \mathit{Attr}(\texttt{dept}) = \{\texttt{dep\_name}\} \to \mathit{Attr}(\texttt{dept})$

$\texttt{emp}(\underline{\#\texttt{emp}}, \texttt{emp\_name}, \texttt{emp\_phone}, \texttt{dep\_name})$

$\texttt{dept}(\underline{\texttt{dep\_name}}, \texttt{dep\_man}, \texttt{dep\_phone})$

- We have key dependencies for the new relation squemas

  $\texttt{emp}: \ \#\texttt{emp} \ \rightarrow \ \texttt{emp\_name}, \texttt{emp\_phone}, \texttt{dep\_name}$

  $\texttt{dept}: \ \texttt{dept\_name} \ \rightarrow \ \texttt{dept\_phone}, \texttt{dept\_man}$

- They imply the FDs for the original relation `emp1`
- The FDs for `emp1` imply the new key dependencies
- This reasoning about implication of FDs can be made relative to the original schema `emp1`

  Where it makes sense to analyze all these dependencies
- Can we always achieve this preservation of FDs?
- This is a question about preservation of the data model semantics

- There is a polynomial-time algorithm to decompose a relation schema into a set of relation schemas that are in 3NF w.r.t. their derived FDs

  P. A. Bernstein. "Synthesizing Third Normal Form Relations from Functional Dependencies". *ACM Trans. Database Syst.*, 1976, 1(4):277-298

- The example shows the idea: break transitive dependencies
- With the following guaranteed properties:
  - The decomposition is lossless
  - The FDs are preserved
- First property enforced by construction
- About the second property:

  The set $\mathcal{F}_0$ of FDs for the original schema is logically equivalent to the union $\mathcal{F}'$ of the sets of derived (or "projected") FDs for the relation schemas in the decomposition (in the semantic or symbolic sense)

- The algorithm does not check 3NF, but enforces 3NF by construction

- Actually, deciding if a schema in in 3NF is NP-complete!

  J. H. Jou, P. C. Fischer. "The Complexity of Recognizing 3NF Relation Schemes". *Inf. Process. Lett.*, 1982, 14(4):187-190

- The source of complexity of deciding 3NF is non-primality checking

  It appeals to finding all candidate keys, i.e. minimal superkeys

- In general, checking minimality under set inclusion is a source of complexity

- Is 3NF enough?
- To avoid redundancy and update anomalies?

Example: Relation schema $ZipCodes(City, Street, Zip)$

FDs: $City, Street \rightarrow Zip$ and $Zip \rightarrow City$

- Redundant information is likely: with the same $(City, Zip)$ subtuples
- Is this schema in 3NF?

Non-prime attributes?

Candidate keys: $\{City, Street\}$ and $\{Zip, Street\}$

- All attributes belong to some candidate key

No non-prime attributes

- Relation schema is trivially in 3NF
- Can we improve on this?

The Boyce-Codd Normal Form (BCNF)

- A relation schema is in BCNF w.r.t. its FDs if no attribute transitively depends upon a candidate key
- In previous example: $\mathit{City}, \mathit{Street} \rightarrow \mathit{Zip} \rightarrow \mathit{City}$ (*)

  The schema is not in BCNF
- More precisely, the following combination cannot happen:

  (from the schema with derived FDs)
  ($\mathbf{A}$, $\mathbf{B}$, $\mathtt{C}$ any attributes)

  1. $\mathbf{A} \rightarrow \mathbf{B} \rightarrow \mathtt{C}$
  2. $\mathbf{A}$ candidate key, $\mathtt{C}$ an attribute
  3. The FDs in 1. are non-trivial
  4. $\mathbf{B} \not\rightarrow \mathbf{A}$ (in particular, $\mathbf{B}$ cannot be superkey)
- With (*) this does happen!
- These conditions are similar to those for 3NF

  3NF only for non-prime attributes: BCNF more demanding
- Then trivially: BCNF $\Rightarrow$ 3NF

  By counterexample above: 3NF $\not\Rightarrow$ BCNF

- An equivalent condition for BCNF: (check it!)

  BCNF $\iff$ For attributes $D \notin \mathbf{X}$: (**)

  If $\mathbf{X} \rightarrow D$, then $\mathbf{X}$ is a superkey

- Another general fact that can be proved:

  If a relation schema in 3NF is not in BCNF, then it has at least two intersecting candidate keys that are composite (have more than one attribute)

- If this does not happen, then BCNF and 3NF coincide

  In this case: 3NF $\Rightarrow$ BCNF

- Exercise: Check that the last schema for our running example is in BCNF (see page 103)

- Natural questions:

  1. How expensive is to decide BCNF?
  2. Is there a decomposition algorithm?
  3. If yes, with what nice or bad properties?

- Deciding if a relation schema with FDs is in BCNF?
- Deciding BCNF: With initial set $\mathcal{F}$ of explicit FDs
  1. Compute deductive closure $\mathcal{F}^+$ (derived FDs) $\mathcal{F} \subseteq \mathcal{F}^+$

     This can be done in linear time (see reference on page 71)
  2. Non-trivial FDs are of the form $\mathbf{X} \rightarrow D$ with $D \notin \mathbf{X}$

     Check for them if $\mathbf{X}$ is superkey (using (**) on page 109)
- Checking superkey could be easier that checking key

  The latter includes checking minimality (something that usually adds complexity)
- Apparently easier than deciding 3NF
- However, deciding if a relation schema with FDs is in BCNF is provably difficult

  Actually, coNP-hard (see paper by Beeri and Bernstein, 1979, op. cit.)

- Decompositions to reach BCNF?
  With good properties?

- Example: Schema $\mathit{Wine}(\mathit{Vineyard}, \mathit{Region}, \mathit{Country})$
  FDs: $\mathit{Vineyard}, \mathit{Country} \rightarrow \mathit{Region}$ & $\mathit{Region} \rightarrow \mathit{Country}$
  (similar to preceding example)

  $\mathit{Wine}(\mathit{Vineyard}, \mathit{Region}, \mathit{Country}) \rightsquigarrow$
  $\mathit{Wine1}(\mathit{Vineyard}, \mathit{Region})$, $\mathit{Wine2}(\underline{\mathit{Region}}, \mathit{Country})$

- The last two are in BCNF (trivially, no transitivity)

- The decomposition is lossless, by "the theorem"
  For intersection attribute *Region*, in *Wine*2: $\mathit{Region} \rightarrow \mathit{Country}$
  (A reason, together with breaking the partial dependency $\mathit{Region} \rightarrow \mathit{Country}$, for attempting the decomposition above)

- However, The FDs are not be preserved: No way derive that *Vineyard* is a key for *Wine1* (it has trivial key $\{\mathit{Vineyard}, \mathit{Region}\}$)