Part 10: Introduction to Relational Normal Forms

References:

- Elmasri/Navathe: Fundamentals of Database Systems, 3rd Ed., Ch. 14, "Functional Dependencies and Normalization for Relational Databases" Ch. 15, "Relational Database Design Algorithms and Further Dependencies"
- Silberschatz/Korth/Sudarshan: Database System Concepts, 3rd Ed., Ch. 7, "Relational Database Design"
- Ramakrishnan/Gehrke: Database Management Systems, 2nd Ed., Mc-Graw Hill, 2000. Ch. 15, "Schema Refinement and Normal Forms"
- Simsion/Witt: Data Modeling Essentials, 2nd Edition. Coriolis, 2001. Ch. 2: "Basic Normalization", Ch. 8: "Advanced Normalization".
- Kemper/Eickler: Datenbanksysteme (in German), Oldenbourg, 1997. Ch. 6, "Relationale Entwurfstheorie"
- Rauh/Stickel: Konzeptuelle Datenmodellierung (in German). Teubner, 1997.
- Kent: A Simple Guide to Five Normal Forms in Relational Database Theory. Communications of the ACM 26(2), 120–125, 1983.
- Thalheim: Dependencies in Relational Databases. Teubner, 1991.
- Lipeck: Skript zur Vorlesung Datenbanksysteme (in German), Univ. Hannover, 1996.



After completing this chapter, you should be able to:

- Detect bad relational database designs (that contain redundancies).
- Determine functional dependencies.
- Check whether a given table is in BCNF for given functional dependencies.
- Detect redundancy and normalization problems already during the conceptional design in the ERmodel.



1. Introduction (Anomalies)

2. Functional Dependencies

3. BCNF

Stefan Brass: Datenbanken I



- Relational database design theory is based mainly on a class of constraints called "Functional Dependencies" (FDs). FDs are a generalization of keys.
- This theory defines when a relation is in a certain normal form (e.g. Third Normal Form, 3NF) for a given set of FDs.
- It is usually bad if a schema contains relations that violate the conditions of a normal form.

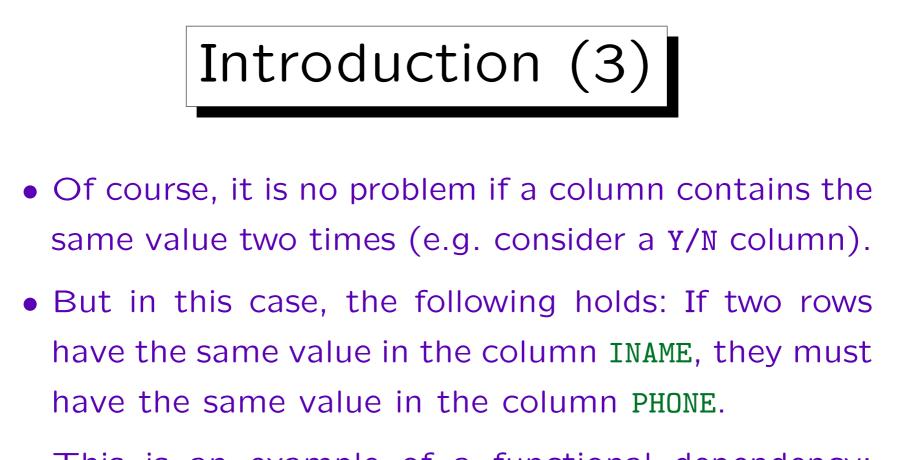
However, there are exceptions and tradeoffs.



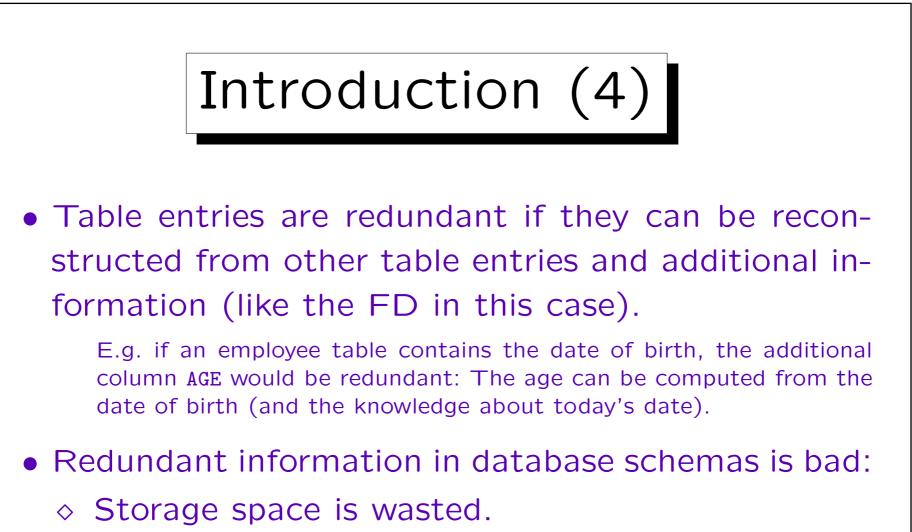
• If a normal form is violated, data is stored redundantly, and information about different concepts is intermixed. E.g. consider the following table:

COURSES			
CRN	TITLE	INAME	PHONE
22268	DB	Brass	9404
42232	DS	Brass	9404
31822	IS	Spring	9429

 The phone number of "Brass" is stored two times. In general, the phone number of an instructor will be stored once for every course he/she teaches.



- This is an example of a functional dependency: INAME \rightarrow PHONE.
- Because of this rule, one of the two PHONE entries for Brass is redundant.

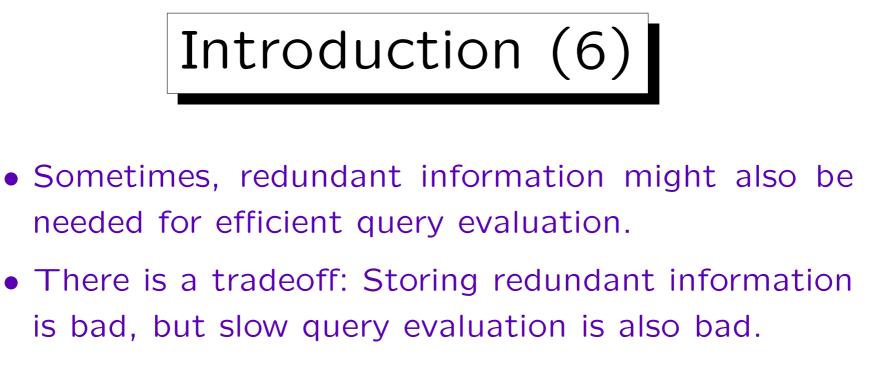


◊ If the information is updated, all redundant copies must be updated. If one is not careful, the copies become inconsistent (Update Anomaly).



- Redundant information is sometimes convenient for easy query formulation (e.g. a precomputed join).
- But in relational databases, one can define virtual tables (views) that are computed by a query.
- Since the contents of a view is not explicitly stored and not directly updated, redundant information is no problem for views.

The entire view is redundant, since it is computed from the stored relations.



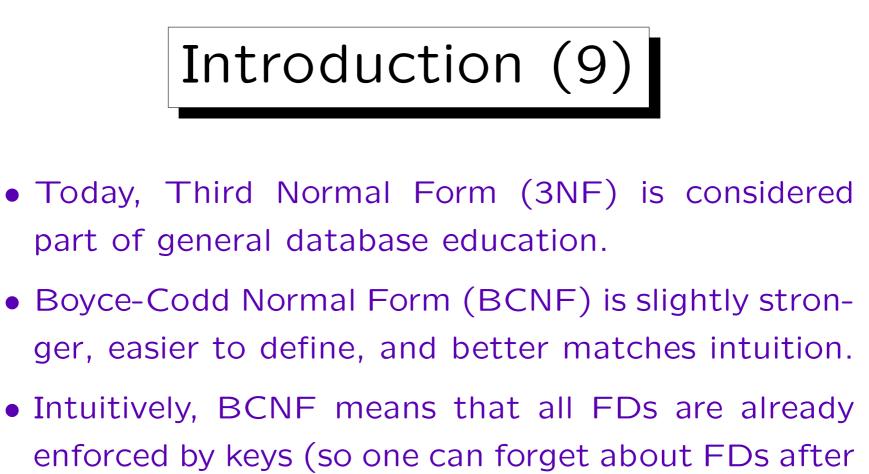
- But adding redundant information should only be discussed during physical design. There must be really good and quantifiable reasons.
- Avoid storing redundant data whenever you can! Many cases of redundant information can be detected by checking for normal forms.



- In the example, the information about the two concepts "Course" and "Instructor" are intermixed in one table. This is bad:
 - The phone number of a new faculty member can be stored in the table only together with a course (Insertion Anomaly).
 - Null values also do not help since the course reference number is the key of the table, and the key must be not null.
 - When the last course of a faculty member is deleted, his/her phone number is lost (Deletion Anomaly).



- If one does a good Entity-Relationship design and translates it into the relational model, all normal forms will be automatically satisfied.
- However, normal forms are generally accepted. If one should have to argue about design alternatives in a team, saying that one schema violates a normal form is a strong and formal reason against it.
- Normal forms give another possibility for checking a proposed schema. However, it is much better to detect the problems already on the ER-level.



- the normalization check).
- Only BCNF is defined here.
- If a table is in BCNF, it is automatically in 3NF.

10 - 12

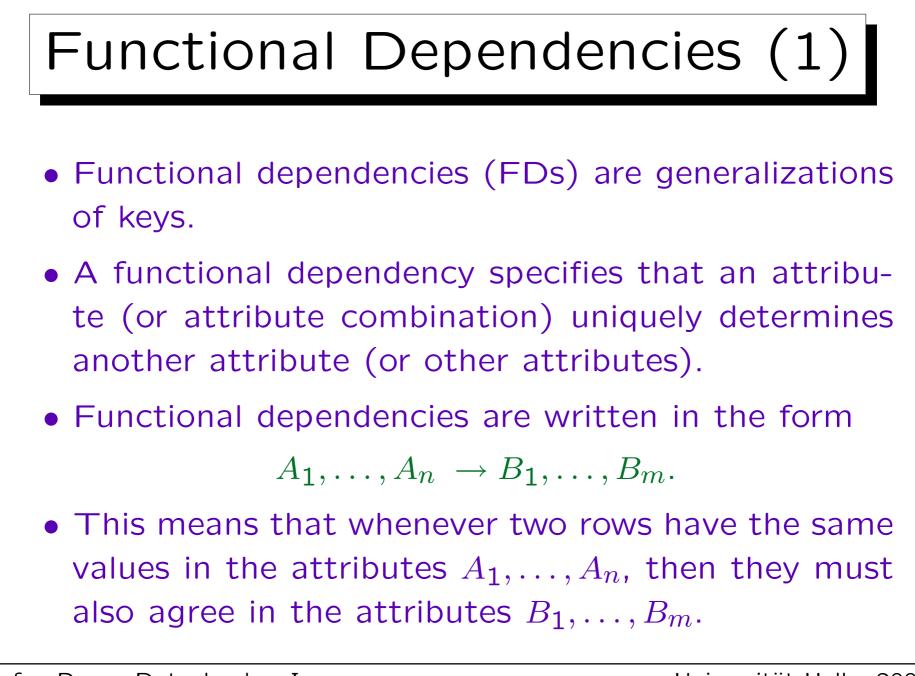


1. Introduction (Anomalies)

2. Functional Dependencies

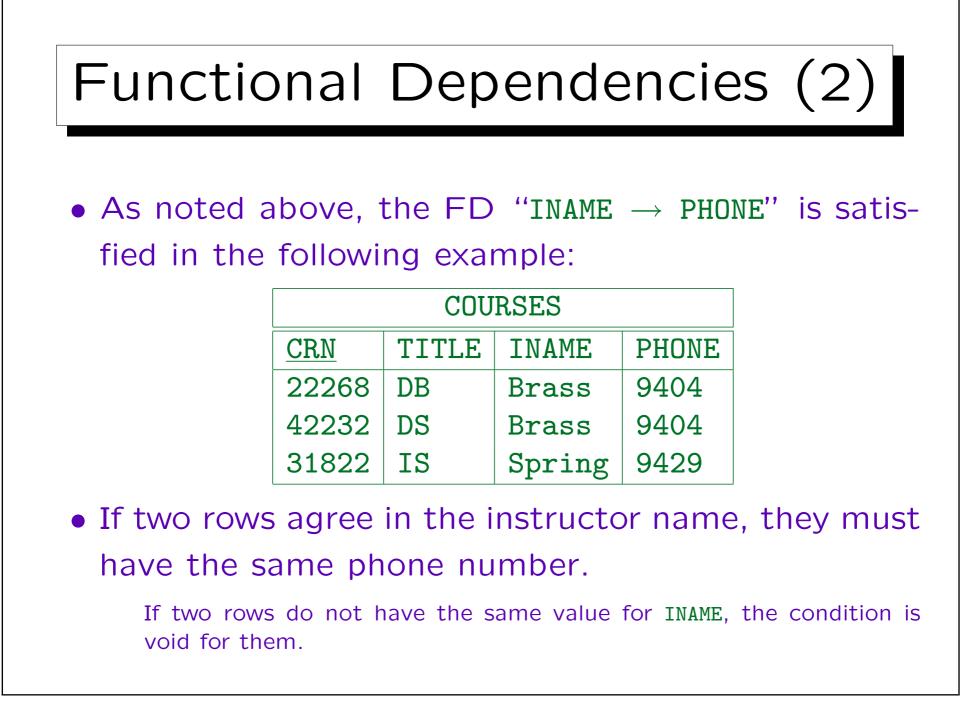
3. BCNF

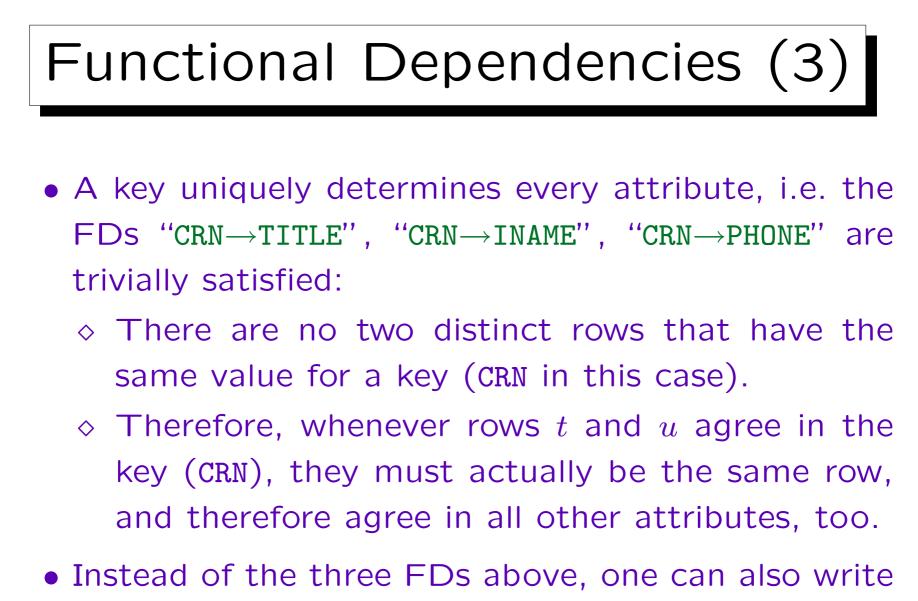
Stefan Brass: Datenbanken I

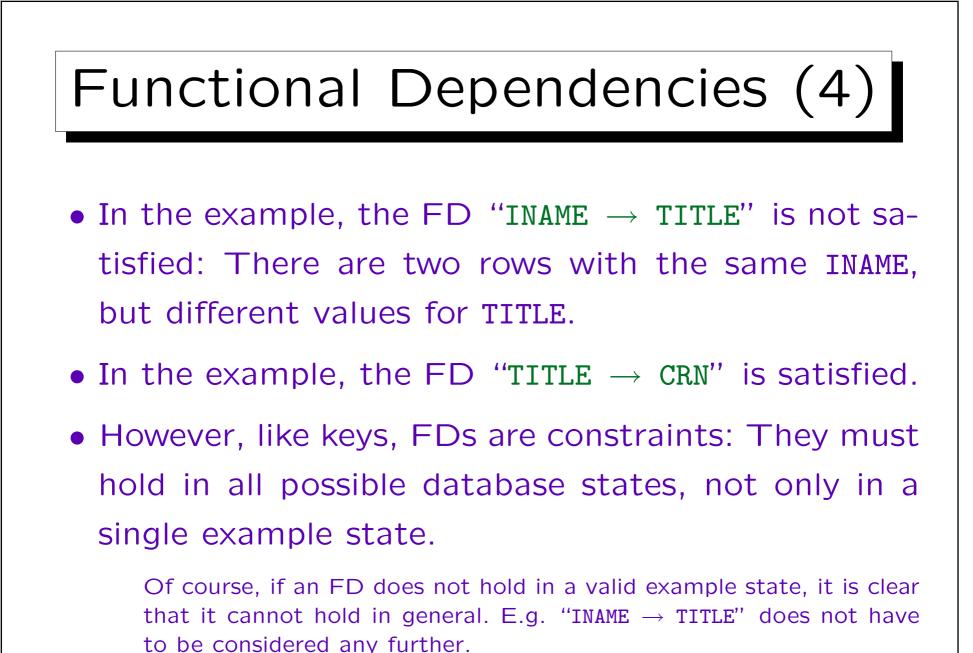


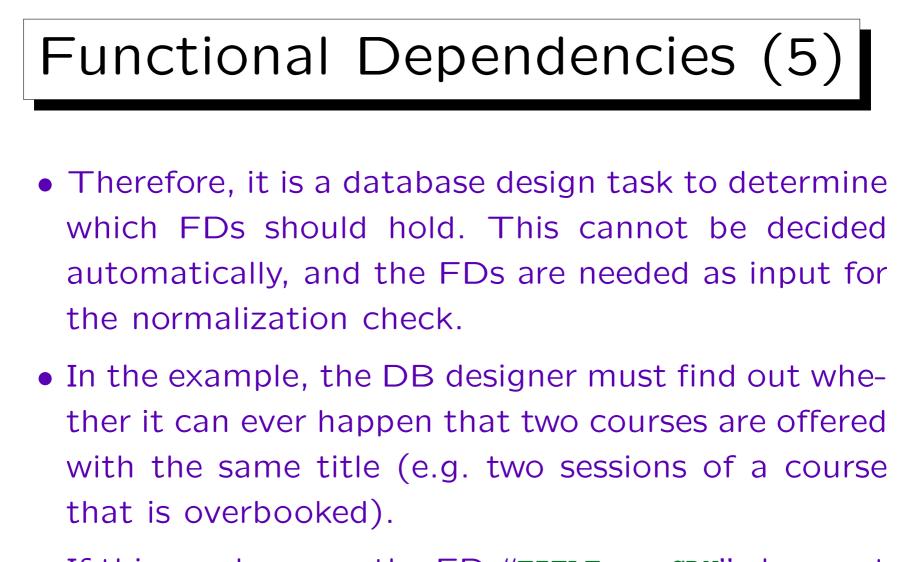
Stefan Brass: Datenbanken I

Universität Halle, 2004

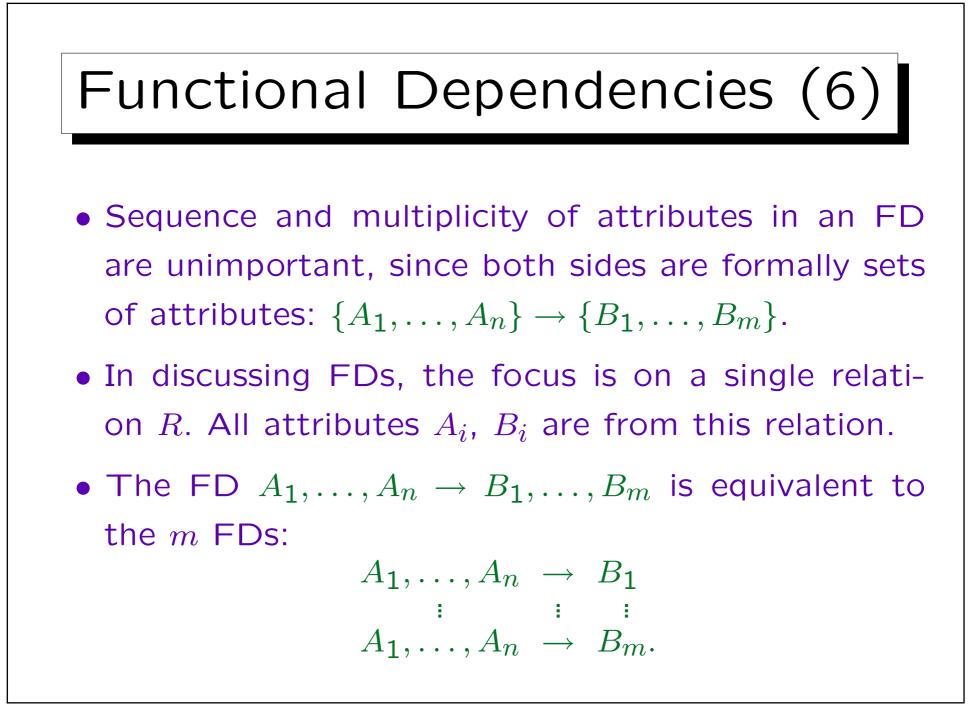








• If this can happen, the FD "TITLE \rightarrow CRN" does not hold in general.



FDs vs. Keys

• FDs are a generalization of keys: A_1, \ldots, A_n is a key of $R(A_1, \ldots, A_n, B_1, \ldots, B_m)$ if and only if the FD " $A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m$ " holds.

Under the assumption that there are no duplicate rows. Two distinct rows that are identical in every attribute would not violate the FD, but they would violate the key. In theory, this cannot happen, because relations are sets of tuples, and tuples are defined only by their attribute values. In practice, SQL permits two identical rows in a table as long as one did not define a key (therefore, always define a key).

• Given the FDs for a relation, one can compute a key by finding a set of attributes A_1, \ldots, A_n that functionally determines the other attributes.

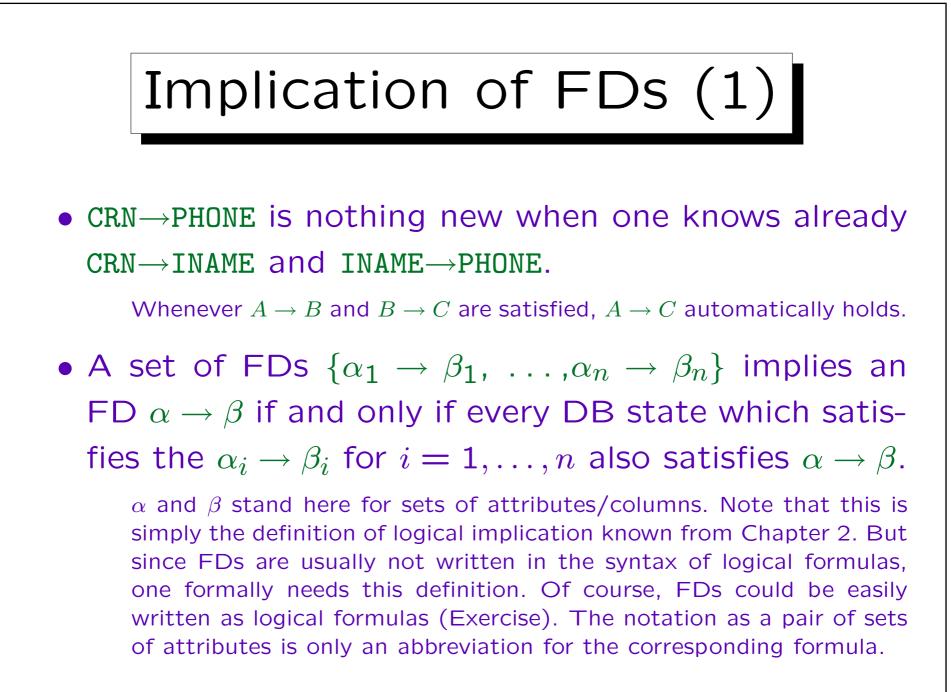


- A functional dependency $\alpha \to \beta$ such that $\beta \subseteq \alpha$ is called trivial.
- Examples are:
 - $\diamond \text{ TITLE } \rightarrow \text{ TITLE }$
 - \diamond INAME, PHONE \rightarrow PHONE
- Trivial functional dependencies are always satisfied (in every database state, no matter whether it satisfies other constraints or not).

In logic, they would be called a tautology.

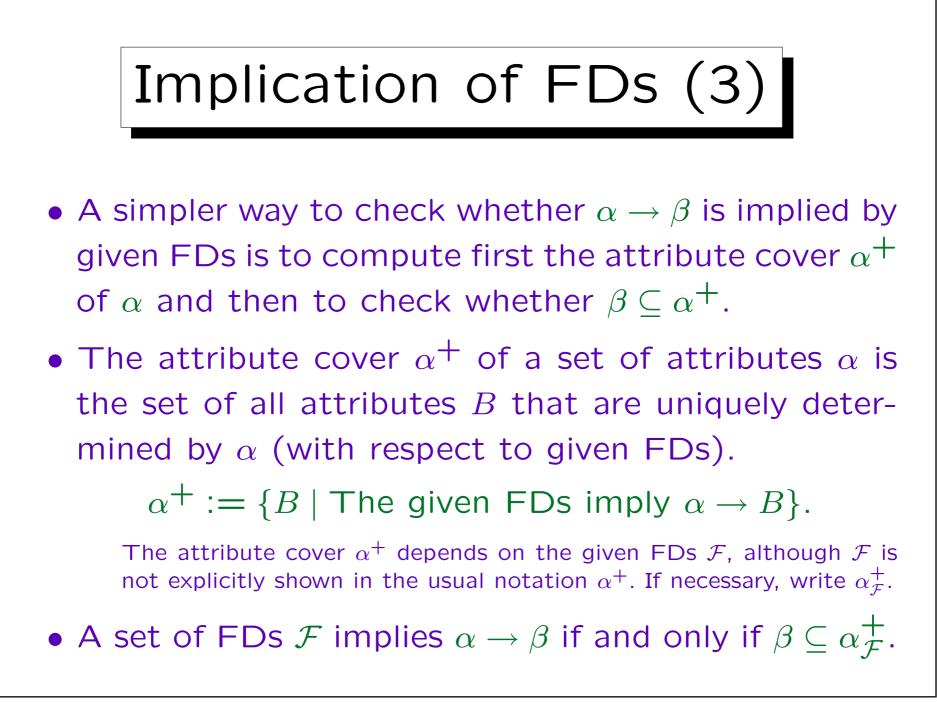
• Trivial FDs are not interesting.

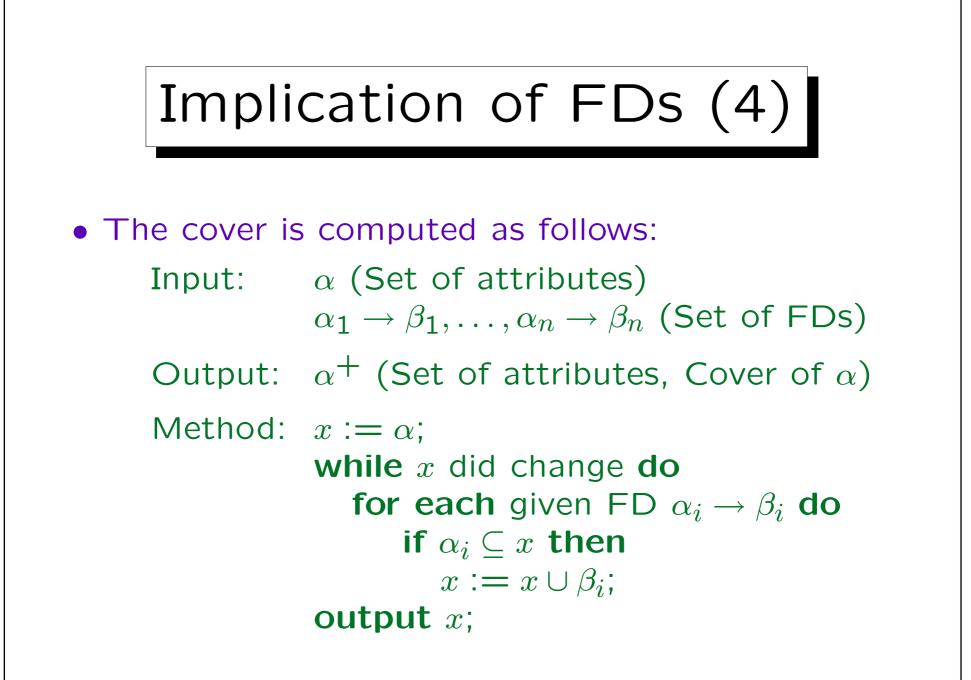
10 - 21

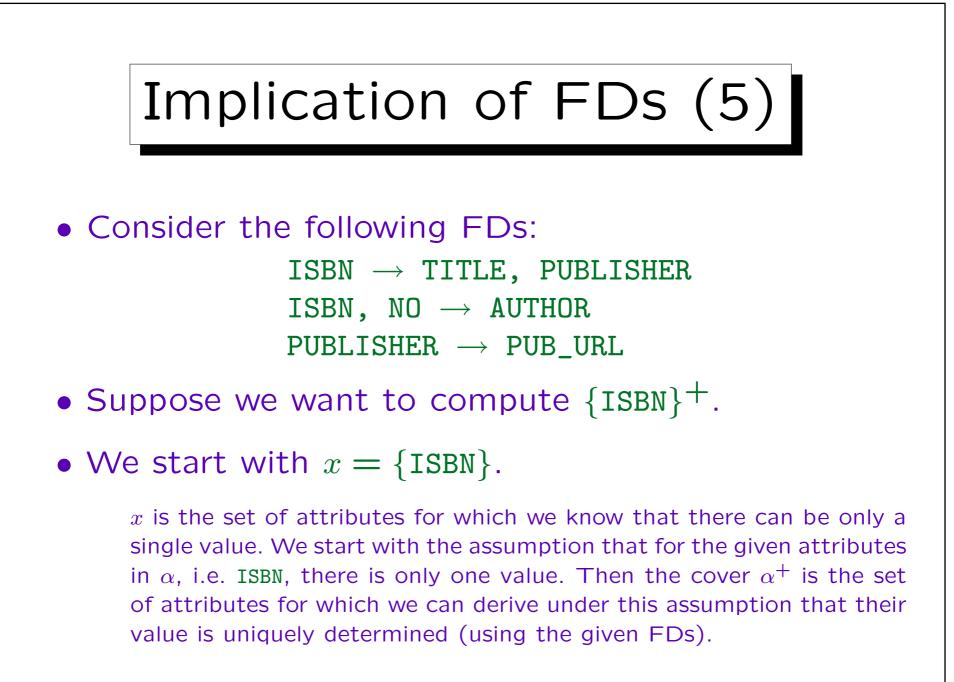


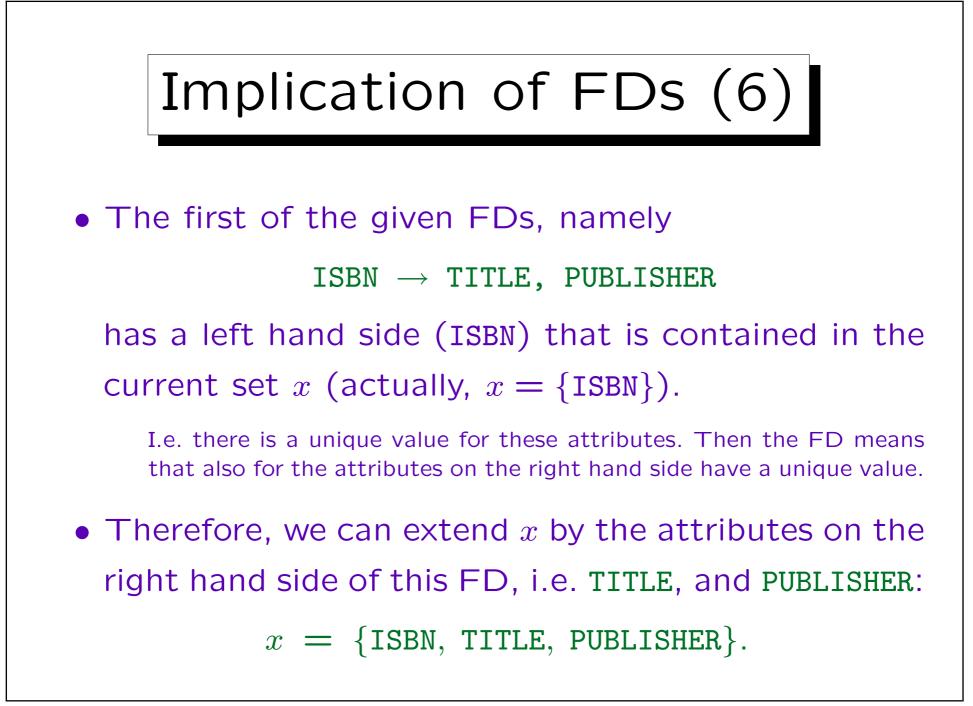


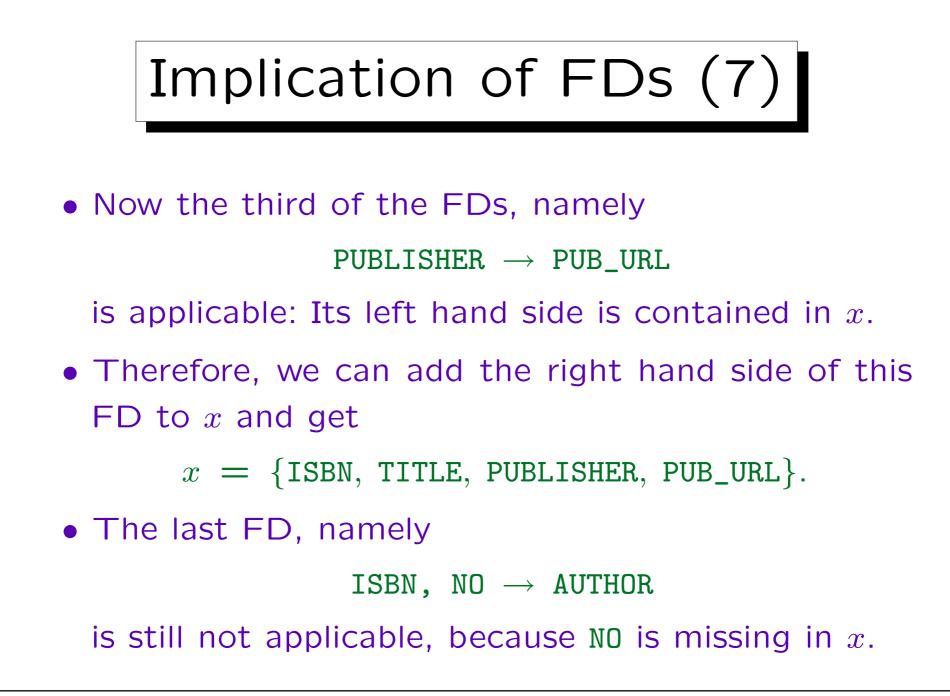
- One is normally not interested in all FDs which hold, but only in a representative set that implies all other FDs.
- Implied dependencies can be computed by applying the Armstrong Axioms:
 - ♦ If $\beta \subseteq \alpha$, then $\alpha \to \beta$ trivially holds (Reflexivity).
 - \diamond If $\alpha \to \beta$, then $\alpha \cup \gamma \to \beta \cup \gamma$ (Augmentation).
 - \diamond If $\alpha \rightarrow \beta$ and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$ (Transitivity).

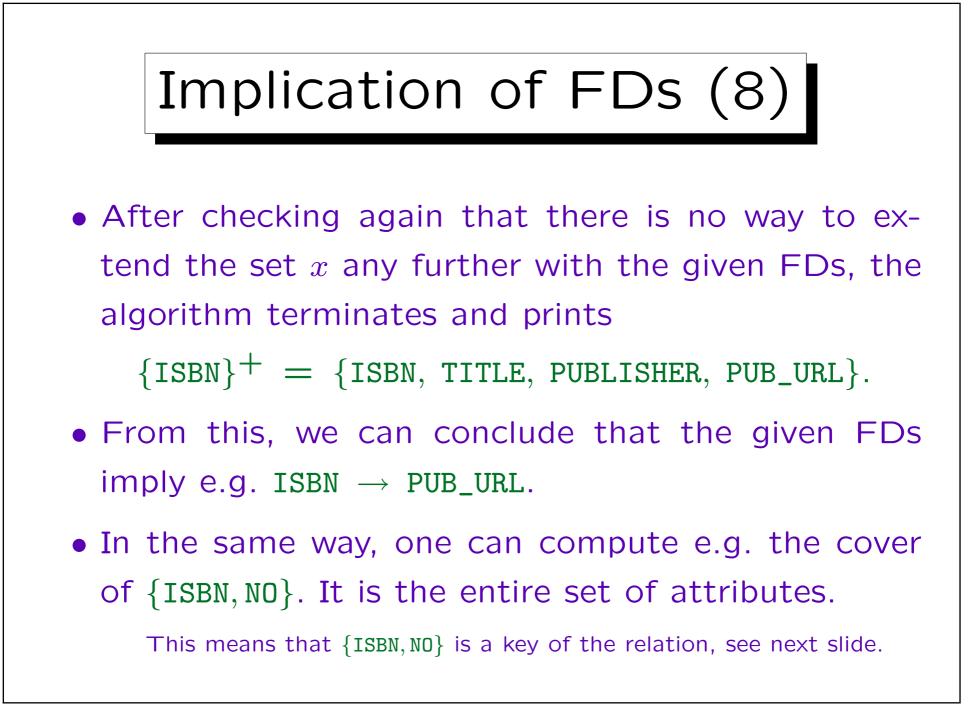


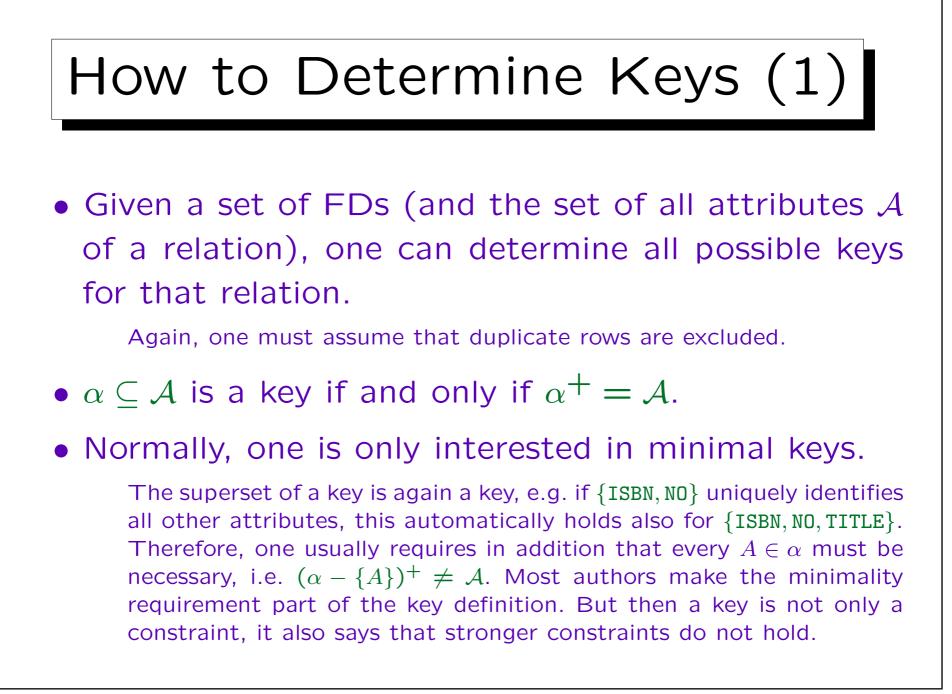


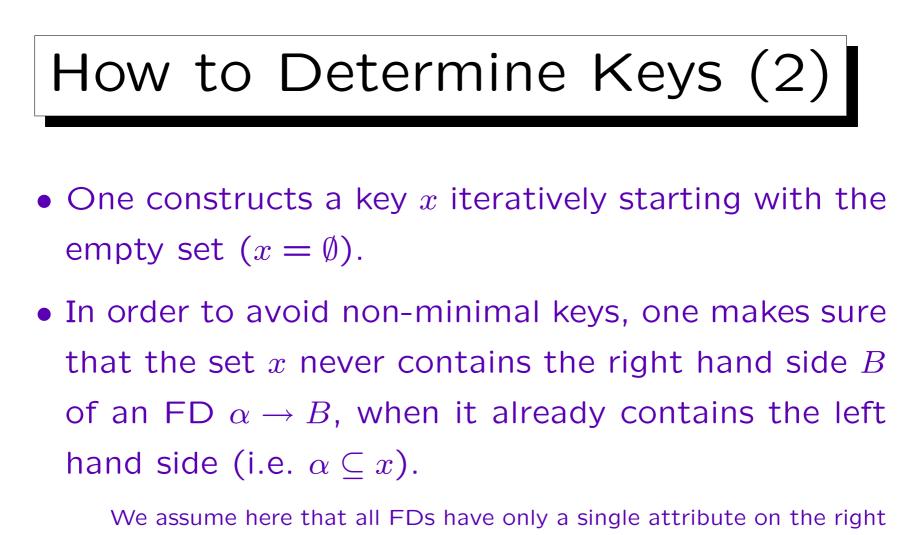




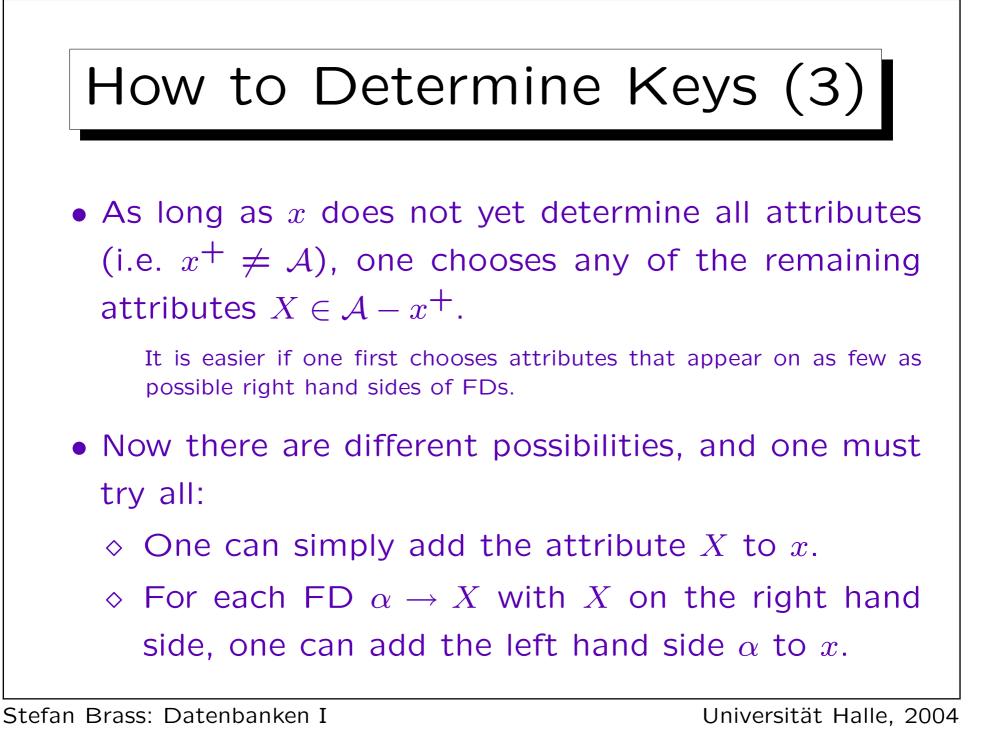


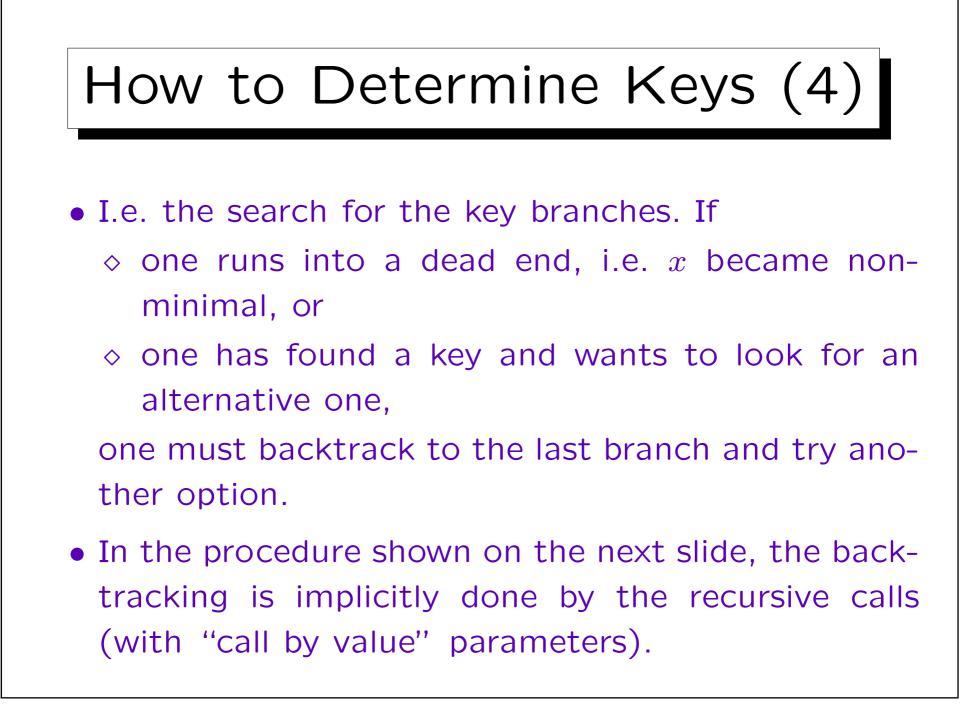


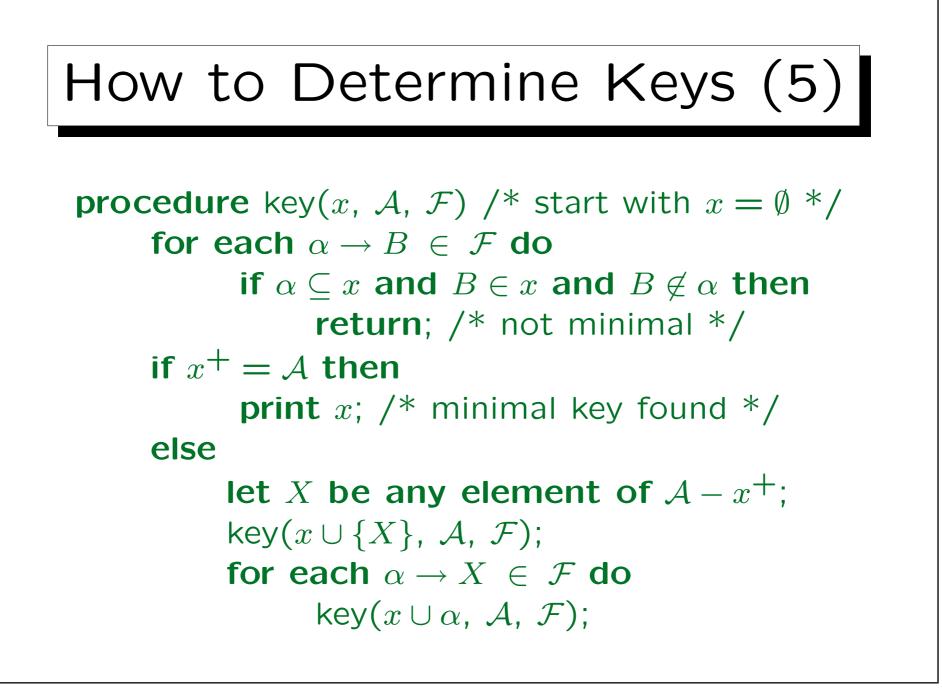


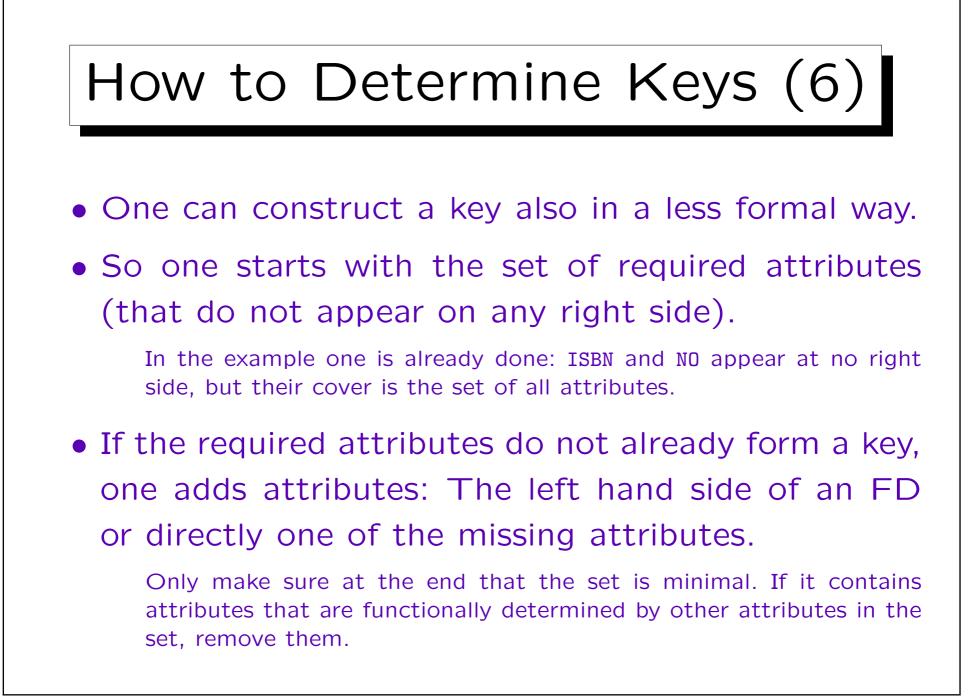


hand side, this is no restriction. Only FDs with $B \notin \alpha$ can destroy the minimality.











- The following relation is used for storing orders: ORDER(ORD_NO, DATE, CUST_NO, PROD_NO, QUANTITY)
- Please list FDs which hold for this relation:

One order can be about multiple products.

• Do these FDs imply the following FD?

ORD_NO, PROD_NO \rightarrow DATE

• Determine a key of the relation ORDER.



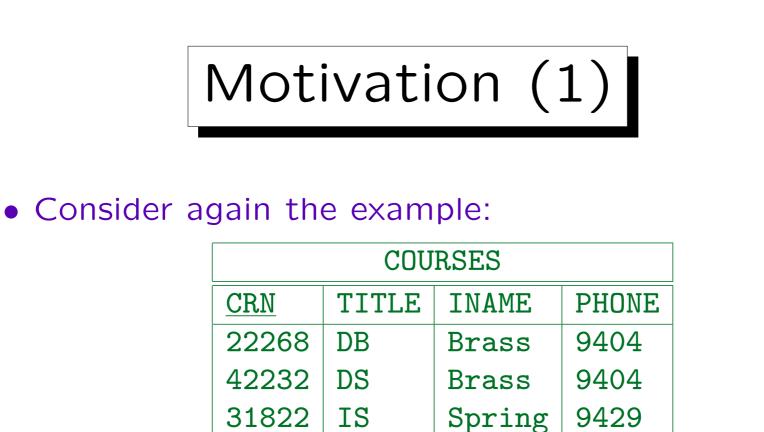
1. Introduction (Anomalies)

2. Functional Dependencies

3. BCNF

Stefan Brass: Datenbanken I

Universität Halle, 2004



 As noted above, the FD INAME→PHONE leads to problems, one of which is the redundant storage of certain facts (e.g. the phone number of "Brass").

Motivation (2)

• Actually, any FD $A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m$ will cause redundant storage unless A_1, \ldots, A_n is a key, so that each combination of attribute values for A_1, \ldots, A_n can occur only once.

Trivial constraints must be excluded here, i.e. at least one of the B_i should not appear among the A_j .

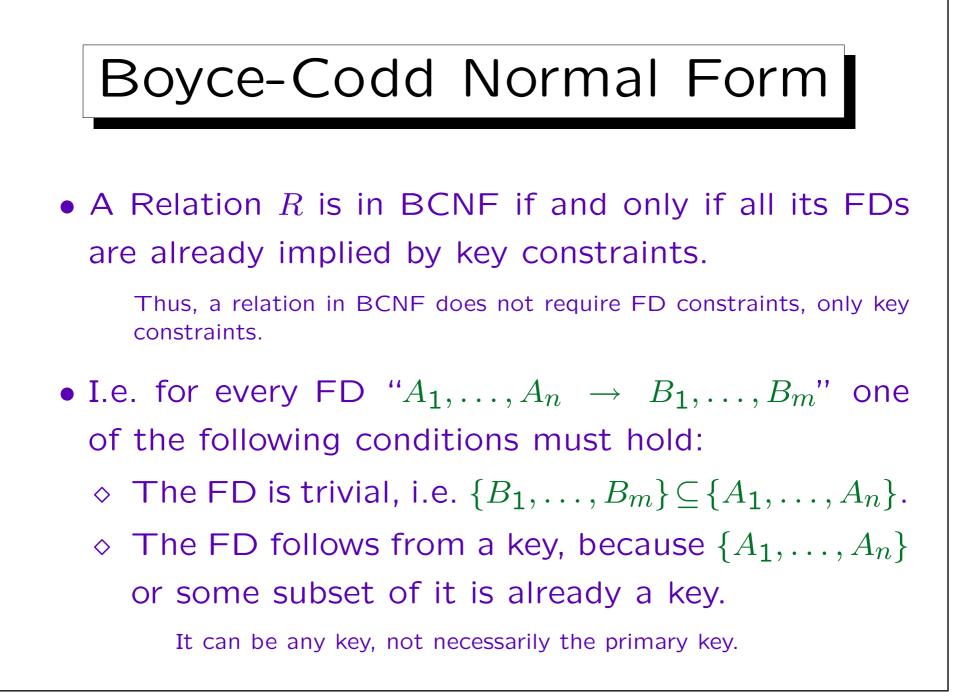
• In general, whenever one stores redundant data, one needs a constraint that ensures that the different copies of the same information remain consistent (i.e. do not contradict each other).

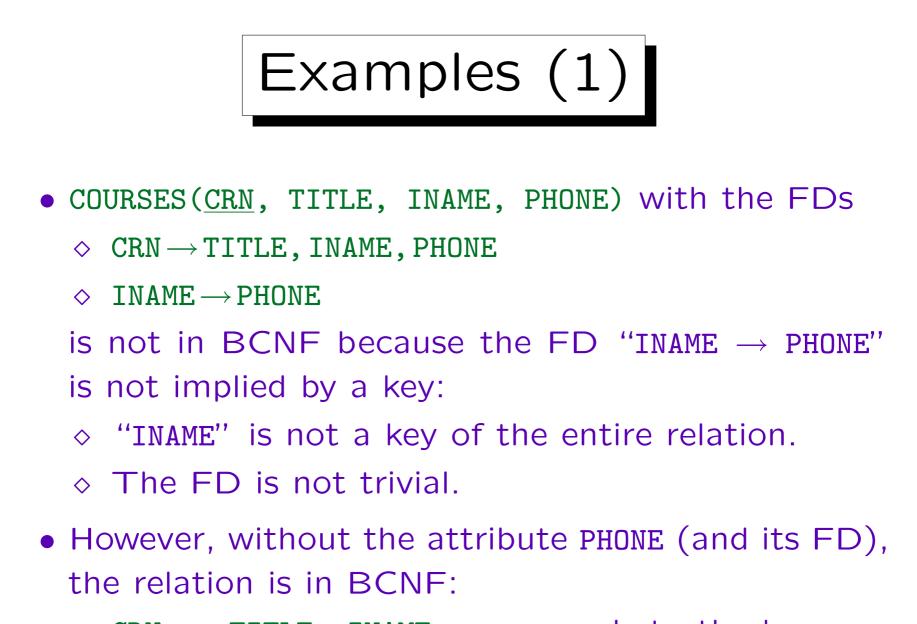
Motivation (3)

- In the cases of redundant data considered here, the constraints are precisely the FDs, e.g. INAME \rightarrow PHONE.
- But FDs are not one of the standard constraints of the relational model. They cannot be specified in the CREATE TABLE statement of current DBMSs.
- Only the special case of keys is supported.
- Thus: Avoid (proper) FDs by transforming them into key constraints. This is what normalization does.

Motivation (4)

- The problem in the example is also caused by the fact that information about different concepts is stored together (faculty members and courses).
- Formally, this follows also from "INAME \rightarrow PHONE":
 - ◇ INAME is like a key for only part of the attributes.
 - ◊ It identifies faculty members, and PHONE depends only on the faculty member, not on the course.
- Again: The left hand side of an FD should be a key. It is not a problem if a relation has two keys: Then there are only two ways to identify the same concept.





 \diamond CRN \rightarrow TITLE, INAME corresponds to the key.

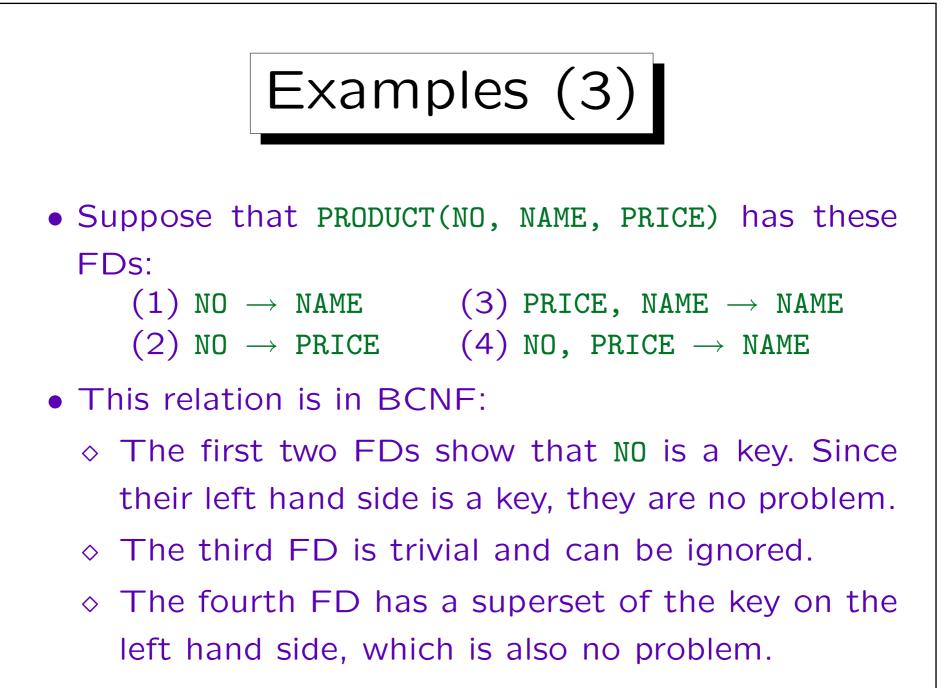


• Suppose that each course meets only once per week and that there are no cross-listed courses. Then

CLASS(CRN, TITLE, DAY, TIME, ROOM)

satisfies the following FDs (plus implied ones):

- \diamond CRN \rightarrow TITLE, DAY, TIME, ROOM
- \diamond DAY, TIME, ROOM \rightarrow CRN
- The keys are CRN and DAY, TIME, ROOM.
- Both FDs have a key on the left hand side, so the relation is in BCNF.





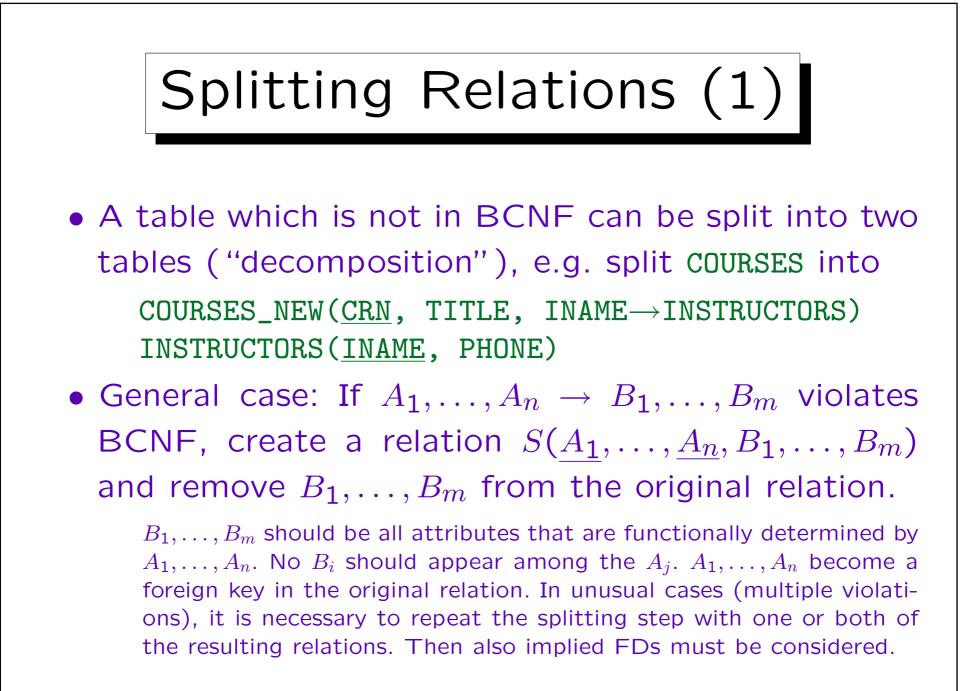
IS RESULTS(STUD_ID, EX_NO, POINTS, MAX_POINTS) with the following FDs in BCNF?

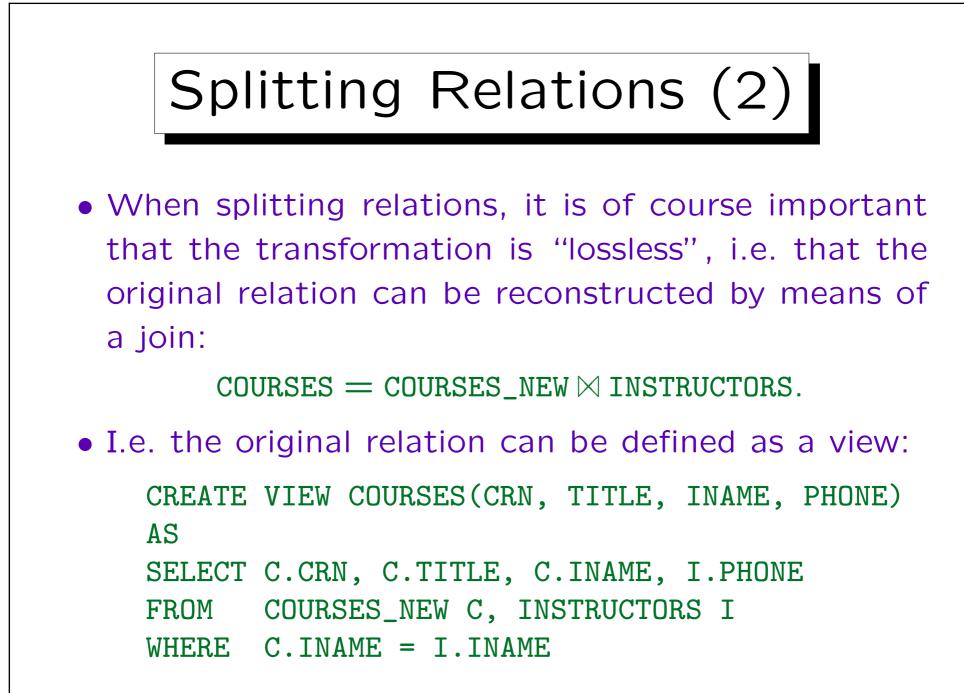
(1) STUD_ID, EX_NO → POINTS
(2) EX_NO → MAX_POINTS

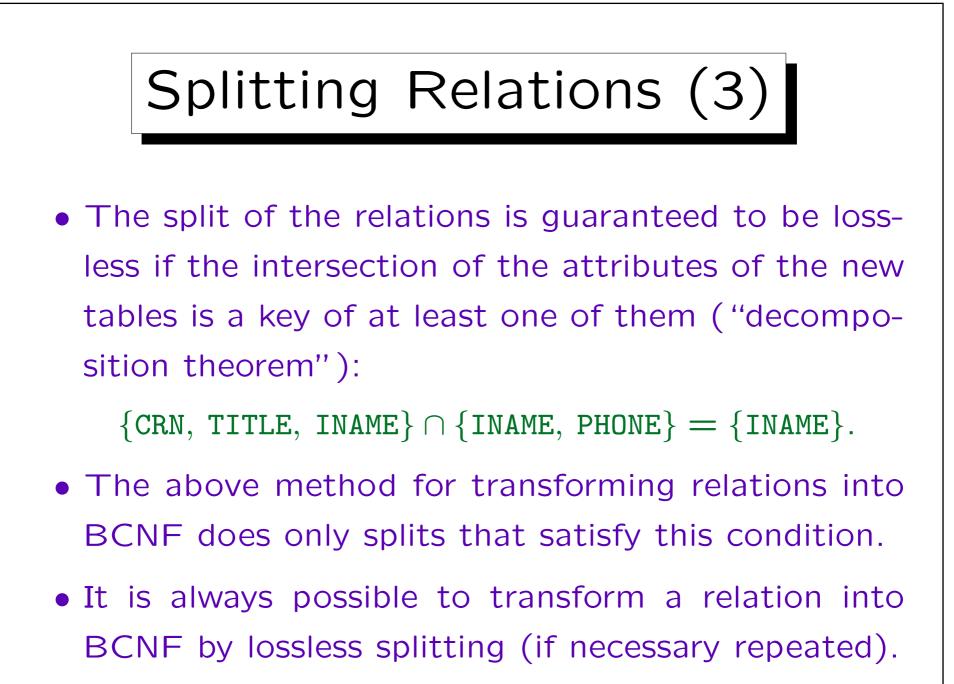
First determine all minimal keys (there is only one).

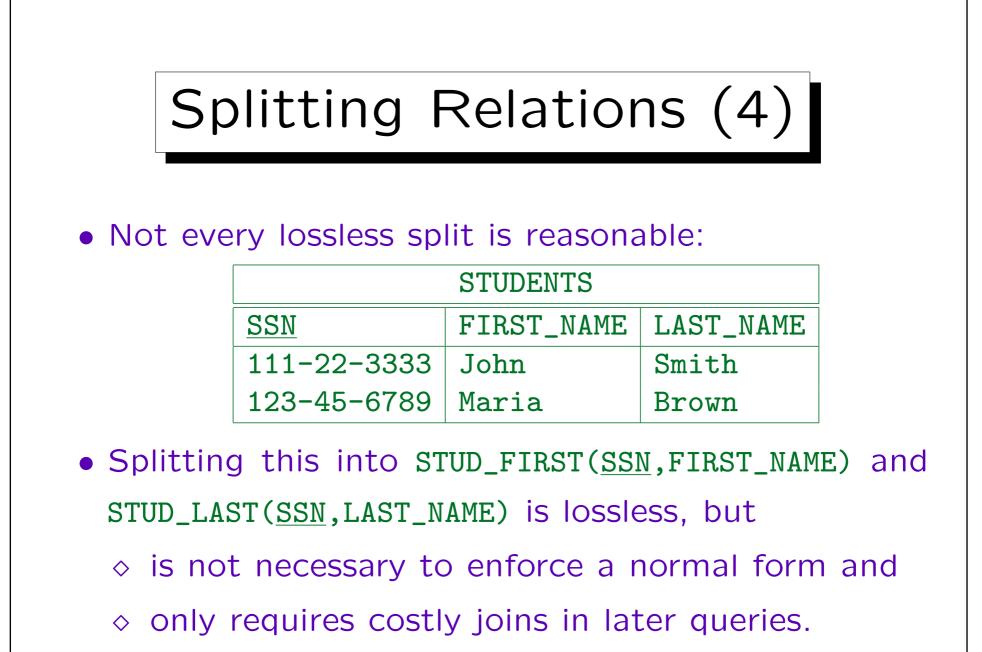
• Is the relation

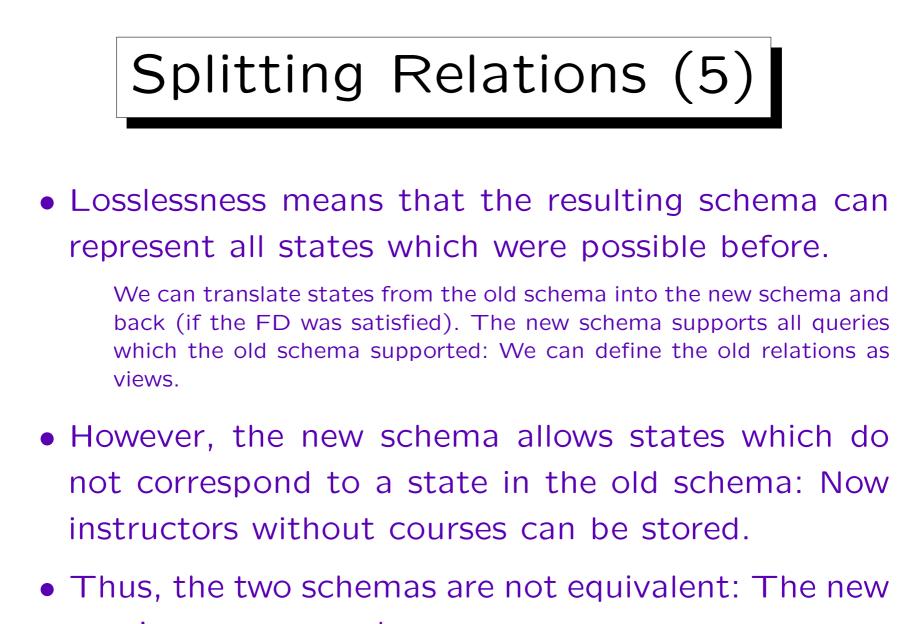
ORDER(ORD_NO, DATE, CUST_NO, PROD_NO, QUANTITY), for which you determined FDs above, in BCNF?













- If instructors without courses are possible in the real world, the decomposition removes a fault in the old schema (insertion and deletion anomaly).
- If they are not,
 - ◊ a new constraint is needed that is not necessarily easier to enforce than the FD, but at least

None of the two can be specified declaratively in the CREATE TABLE statement. Thus, nothing is gained or lost.

♦ the redundancy is avoided (update anomaly).