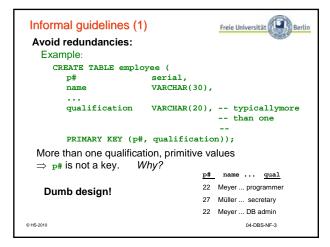
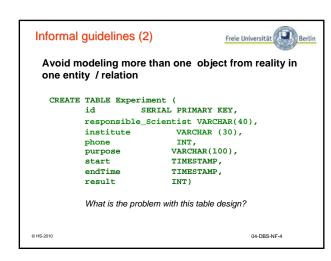
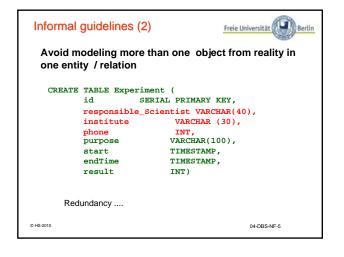
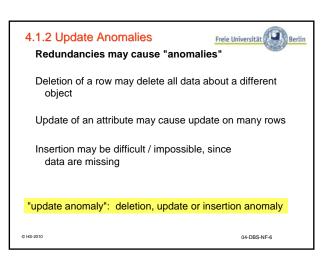
4. Normalization: Quality of relational designs 4.1 Functional Dependencies 4.1.1 Design quality 4.1.2 Update anomalies 4.1.3 Functional Dependencies: definition 4.1.4 Properties of Functional Dependencies 4.2 Normal forms 4.2.1 Informal introduction 4.2.2 Normal Forms and FDs 4.2.3 Normal forms (2NF, 3NF, BCNF, MV NF) 4.2.4 Lossless join and dependency preservation 4.2.5 Multivalued dependencies and 4NF 4.3 Algorithms for finding Normal Forms 4.4 Normal Forms: Critical review Kemper/Eickler: chap 6; Garcia-Molina/Ullman/Widom: chap 3.4 ff.; Litt Elmasr/Navathe: chap 14 , Kifer et al.: chap. 6

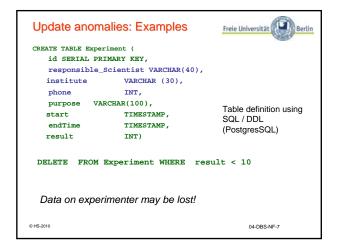


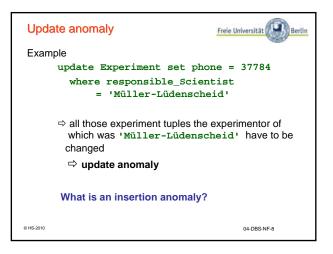


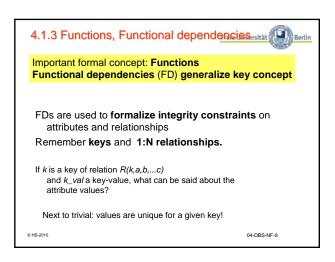


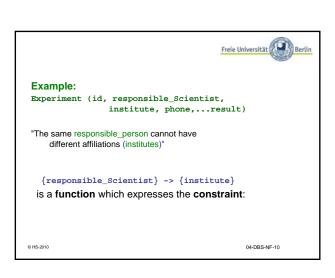


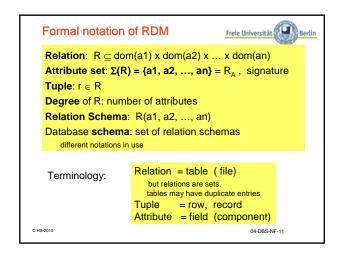


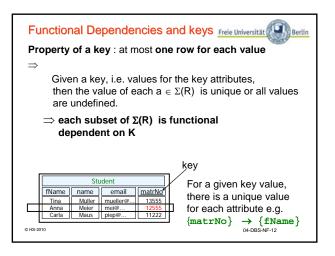












Functional Dependencies



Functional and key dependencies are

constraints (invariants) of the application domain

"Functional dependency" constraints have to be identified during requirements engineering - like all constraints.

Ultimate goal: DBS monitors compliance with DB state.

Example:

Experiment (id, responsible_Scientist, institute, phone ... result)

What has to be done, when a new experiment is inserted?

04-DBS-NF-13

Motivation for Normal Forms



Suppose we can find a relational schema which has only keyinduced functional dependencies (FD) (and "trivial" ones like {a,b} -> {b})

How can we efficiently check the DB state after an update with respect to FD? Do they still hold?

A "good" schema avoids

Update anomalies

Costly check of functional dependencies after update

© HS-2010 04-DBS-NF-14

Functional Dependency: Definition Freie Universität Parlin Berlin



Def.: Functional Dependencies (FDs)

Let $A = \Sigma(R)^* = \{a,b,c,...a_i,...\}$ be the attribute set of a relation Rand X, $Y \subseteq A$, $r, r' \in R, r \neq r'$ Y is functionally dependent on X (written: $X \rightarrow Y$)

:⇔ $(\forall xi \in X) r.xi = r'.xi \Rightarrow (\forall yi \in Y) r.yi = r'.yi$

· As we know: invariants are independent of the particular

- database state • They must hold at all times,
 - i.e. they restrict the valid states of the database

 $_{_{\odot\, HS\text{-}2010}}^{\quad \ \, \star}\Sigma(R)$: Attribute set of relation R

04-DBS-NF-15

4.1.4 FD Properties



Trivial functional dependency

$$X \subseteq Y \Rightarrow Y \to X$$

Augmentation

$$Z \subseteq A=\Sigma(R), X \to Y \Rightarrow XZ \to YZ$$

Transitivity

$$X,Y,Z \subset A=\Sigma(R), X \to Y, Y \to Z \Rightarrow X \to Z$$

Proof?

Notation XY -> Z means X ∪ Y -> Z

04-DBS-NF-16

Implied and inferred FD



A functional dependency $Y \rightarrow Z$ is called <u>implied</u> by a set $F = \{F1, ..., Fn\}$ of functional dependencies, if $Y \rightarrow Z$ can be proven from F.

A functional dependency $Y \rightarrow Z$ can be <u>inferred</u> (\vdash) by a set of inference rules R = {r1,...rm} from set

 $F = \{F1, ..., Fn\}$ of functional dependencies

if $Y \rightarrow Z$ can be constructed by a finite number of **syntactic** transformations of F according to rules ri

04-DBS-NF-17

Armstrong inference rules



Given a set of FDs, find all implied FD's

A sound, complete, minimal set (Armstrong axioms):

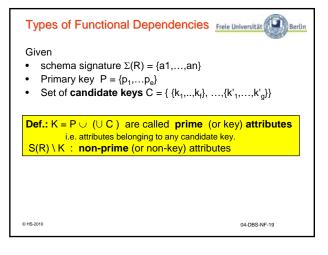
 $Y \subseteq X \vdash X \rightarrow Y$ (I: inclusion) $\{X \to Y , Y \to Z\} \vdash X \to Z$ (T: transitivity) $\{X \rightarrow Y\} \vdash XZ \rightarrow YZ$ (A: augmentation)

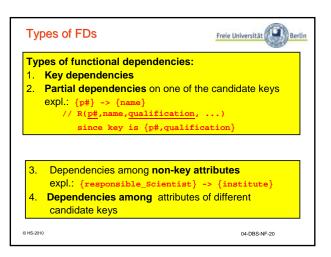
Only implied FDs are constructed by the inference rules

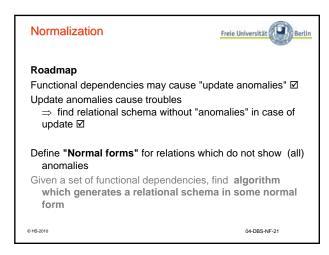
Complete:

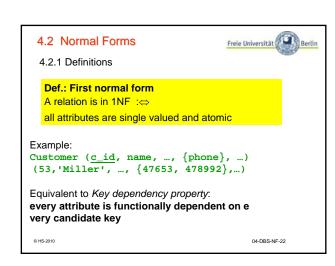
Every implied FD will be produced by a finite number of inferences

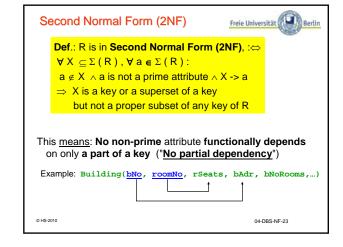
04-DBS-NF-18

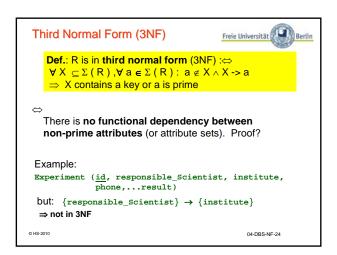




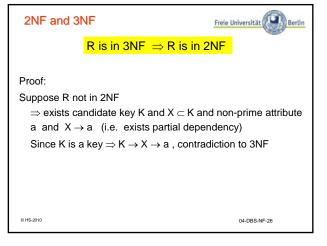


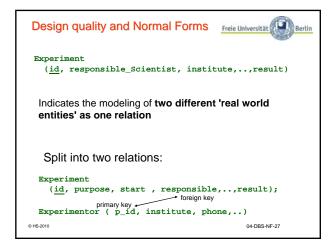


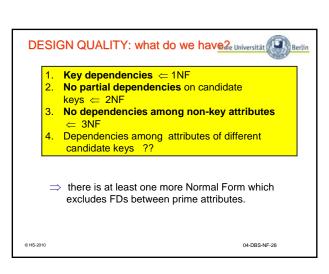


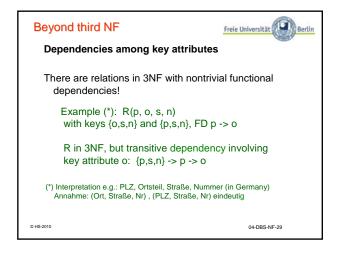


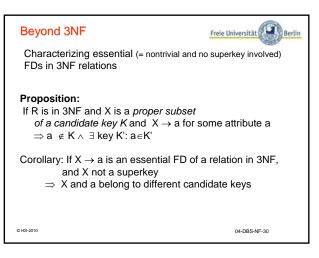
More on 3NF Equivalent definition: R is in 3NF: no non-prime attribute depends transitively on a key. A non-prime attribute y is transitively dependent on a key K, if K → X and X → y and not X → K Notation: K → X → y Experiment (id, responsible_Scientist, institute,..,result)

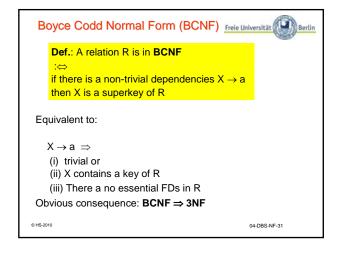


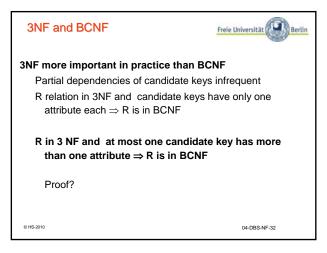




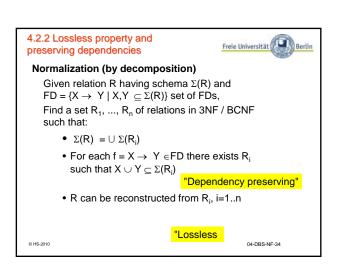


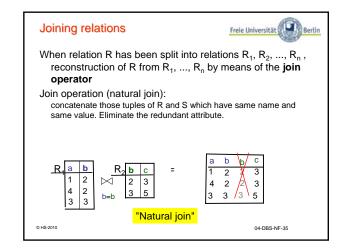


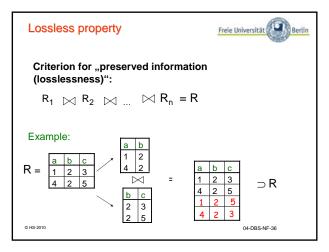


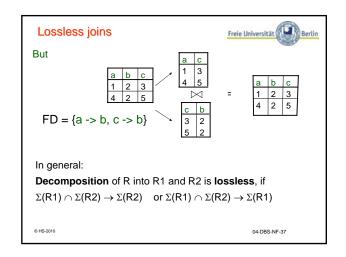


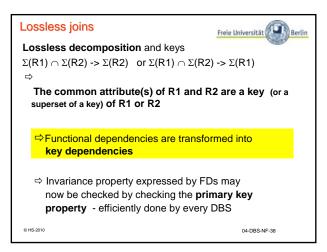


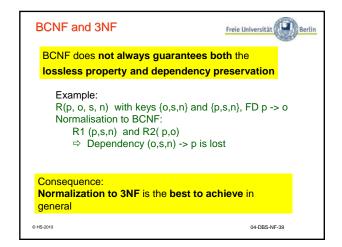


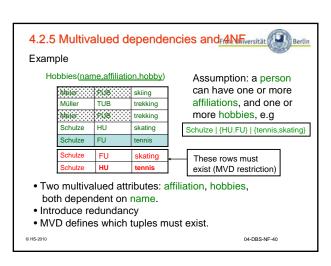


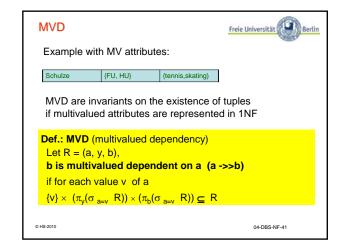


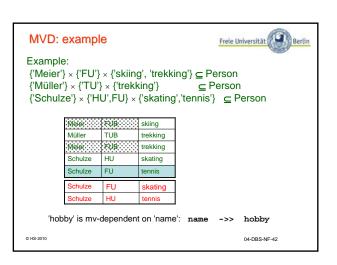












Fourth Normal Form



Def.: Let A, B $\subseteq \Sigma(R)$; R is in **Fourth Normal Form** if for every MVD A ->> B
(i) B \subseteq A or (ii) B = $\Sigma(R) \setminus A$ or (iii) A contains a key

Example not in 4NF, check

Normalized representation:

Müller	TUB
Meier	FUB
Schulze	HU
Schulze	FU

Müller	trekking
Meier	trekking
Meier	skiing
Schulze	skating
Schulze	tennis

© HS-2010 04-DBS-NF-43

Normal forms: summary



Normal forms are quality criteria for database design.

Important: 1NF - 3NF

Exotics: BCNF, 4NF (and higher!)

2NF / 3NF formalize the basic design principle:

"Never mix up different real world entities into a single design object (e.g. entity)"

2NF / 3NF already defined for ERM, since FDs are given (result of requirement analysis, just like key dependencies).

HS-2010 04-DBS-NF-44

4.3 Finding Normal Forms



Invariants of application domain have to be made explicit during requirements analysis

- e.g. "A scientist has at most one affiliation her institute"
 - " A region-id is unique within a country"
 - "A person has exactly one date of birth"

Formalization ⇒ Functional Dependencies
Wanted: algorithm producing "good" relational schemas from the set DEP of all FDs

© HS-2010 04-DBS-NF-48

FDs and Normal Forms



Given a set of dependencies DEP there are two approaches:

Synthesis

Set up relations in such a way, that

- All attributes are consumed
- The relations are in normal form

• Decomposition

For a given set of relations find those which are not normalized with respect to DEP and decompose them into normalized relations

© HS-2010 04-DBS-NF-46

Decomposition: eliminate FDs



Given $\Sigma(R) = U$ and DEP the set of FDs

Algorithm DECOMP(R):

(i) Find the set of keys K:

 $K \rightarrow U \in DEP \text{ or } K \rightarrow U \in DEP^+$ (DEP+ set of all implied dependencies)

(ii) Eliminate all transitive dependencies by splitting recursively: $\{ \text{if } K \to Y \text{ --> a is a transitive FD in } R_k, \text{ split } R_k \text{ into } R_i, R_j \\ \Sigma(R_i) = \Sigma(R_k) \ \setminus \ \{a\}, \ \Sigma(R_j) = Y \cup \{a\}$

} If no more relations l

(iii) If no more relations R_k with transitive dependency exit else for all R_k DECOMP($\!R_k\!)$

© HS-2010 04-DBS-NF-47

Synthesis



Disadvantage of decomposition:

inefficient (e.g. determination of keys) produces more relations than necessary

Synthesis

Given relation R and set of FDs DEP

Find a canonical set MIN of FDs which "covers" DEP and is minimal.

Construct normalized Relations R_k from MIN with $\bigcup \Sigma(R_k) = \Sigma(R)$

10 04-DBS-NF-48

Finding a canonical set of FDs



Given a set of FDs DEP and a relational schema R

- Find a minimal set MIN such that DEP ⊆ MIN+
- Find a relational schema in 3NF, from which R can be losslessly reconstructed

MIN is called minimal cover of DEP

Definitions

$$\begin{split} X \to Y \in \textbf{MIN}^+ :&\Leftrightarrow X \to Y \text{ can be proven from} \\ \text{the FDs} \in \text{MIN} \\ \text{e.g. } \{a \to b, \, b \to c\} + = \{a \to b, \, b \to c, \, a \to c \, \} \\ \text{MIN is } \textbf{minimal} :&\Leftrightarrow \text{for every FD f} \quad (\text{MIN} \setminus f)^+ \neq \text{MIN}^+ \end{split}$$

© HS-2010 04-DBS-NF-49

Finding a minimal cover (1)



Example:

 $\{ab\to d,\,b\to c,\,dc\to e\,\}+\ =\{ab\to d,\,b\to c,\,a\to dc,\,dc\to e,\,ab\to e\}\,\}$ Important first step:

Given a set of attributes X, determine all attributes (closure of X) which can be functionally determined by X?

Def.: Closure of attribute set X with respect to the set DEP of FDs is the largest set Y of attributes such that $X \rightarrow Y \in DEP^+$

© HS-2010 04-DBS-NF-50

Closure of attribute set X



Rule used: X -> YZ and Z -> W then X -> YZW Proof?

© HS-2010 04-DBS-NF-5

Finding a canonical set



Algorithm for determining a <u>minimal cover</u> in polynomial time

Step 1: Normalization

Replace each FD $X \rightarrow Y$ of DEP in which Y contains more than one attribute, by FDs with one attribute on the right hand side

Example:

 $DEP = \{ab -> cd, a -> e\} \rightarrow \{ab -> c, ab -> d, a -> e\}$

© HS-2010 04-DBS-NF-52

Finding a canonical set (2)



Step 2: Remove redundant FDs

f is redundant, if (DEP \ {f}) $^+$ = DEP $^+$ For each f = X \rightarrow Y \in DEP: if Y \subseteq (X+) using only FDs \in DEP \{f} then f is redundant else not redundant

$$\begin{split} \text{Example: } \{b \to d, \, d \to e, \, & \text{ef} \to a, \, c \to f, \, bc \to a\} \\ bc+= \{b, d, e, f, a\} \supseteq \{a\} \\ \Rightarrow FD \, f = bc - > a \, \text{is redundant} \\ \text{Explicit derivation not using } f: \\ b - > d, \, d - > e \Rightarrow b - > e, \, c - > f \Rightarrow bc - > ef \, , \, ef - > a \Rightarrow bc - > a \, \\ \end{split}$$

© HS-2010 04-DBS-NF-53

Finding a canonical set (3)



Step 3: Minimize left hand side

For of each FD $\ f = X \to Y \in DEP, \ a \in X$ DEP'= {DEP\f} $\ \cup \{X \setminus \{a\} \to Y\}$ if {DEP}+ = (DEP')+ then replace DEP by DEP' end_for

If a FD f has been minimized repeat step 2

Example:

 $\{bcd -> a, c -> e, e -> b\}^+ = \{cd -> a, c -> e, e -> b\}^+$

010 04-DBS-NF-54

Finding a canonical set (4)



Step 4: Unify left hand side

Applying the union rule $"X \to Y \land X \to Z \Rightarrow X \to YZ"$

Example: {cd -> a, cd -> e, d -> f} unified: {cd -> ae, d -> f}

The remaining set of Functional Dependencies is the minimal cover of the original set DEP of FDs

© HS-2010 04-DBS-NF-55

Synthesis algorithm



Normalization problem:

Given a relation R in 1NF and a set DEP of FD Find a lossless, dependency preserving decomposition $R_1,...,R_k$, all of them in 3NF

Synthesis Algorithm

Find minimal cover MIN of DEP;

For all $X \to Y$ in MIN define a relation

RX with schema $\Sigma(RX) = X \cup Y$ Assign all FDs $X' \to Y'$ with $X' \cup Y' \subseteq \Sigma(RX)$ to RX

If none of the synthesized relations RX contains a candidate key of R then introduce a relation Rkey which contains a candidate key of R Remove relations RY where: $\Sigma(\text{RY}) \subseteq \Sigma(\text{RX})$

© HS-2010 04-DBS-NF-56

Normal Forms: Critical review



Should relations be always normalized?

Yes: makes invariant checking easy, and no "update

No: Why should we normalize if there are no updates?

Example: Customer(cu_ld, name, fname, zipCode, city, street, no) No reason to normalize if only one address per customer and updates are infrequent

Consider cost of joins / updates

- How expensive are selects which need joins because of normalization?
- · Updates which cause anomalies?

04-DBS-NF-57

ER modeling and Normal Forms



ER and Normal Forms:

Two different mechanisms to design a database scheme ER more intuitive, NF uses algorithms

ER-models often already in 3NF

Use normalization as a complementary design method

- Set up ER model
- Transform to relations
- Normalize each non normalized relation if the tradeoff of join processing and updating redundant data suggests to do so

04-DBS-NF-58