Datenbanksysteme
SS 2025

Übungsblatt 3

Group:
Attila Haraszti (5602072),
Moritz Ruge (5600961),
Joshua Zahn (5600969).
John Jesus Llerena Kutscheid

Datum: Jun 2, 2025

Task 1: Functional Dependencies (10 %) A B C D

a_1	b_1	c_1	d_1
a_2	b_2	c_1	d_1
a_2	b_3	c_2	d_1
a_3	b_4	c_3	d_2
a_3	b_5	c_4	d_2
a_3	b_6	c_5	d_3

Table 1: Relation R_1(A, B, C, D)

1 **EN:** Which functional dependencies can be derived from the sample tuples of Relation R_1?
DE: Welche funktionalen Abhängigkeiten lassen sich aus den Tupeln der Relation R_1 ableiten?

- **Functional dependencies: A -> ?**
  - a1 - once (b1,c1,d1)
  - a2 - 2 times:
    - (b2,c1,d1)
    - (b3,c2,d1)
    - B and c are different, d is the same
  - a3 - 3 times:
    - B: b4,b5,b6
    - C: c3,c4,c5
    - D: d2,d2,d3
    - B and C and D are different
- A -> B is false
- A -> C is false
- A -> D is false

- **Functional dependencies: B -> ?**
    - b1 - once (a1,c1,d1)
    - b2 - once (a2,c1,d1)
    - b3 - once (a2,c2,d1)
    - b4 - once (a3,c3,d2)
    - b5 - once (a3,c4,d2)
    - b6 - once (a3,c5,d3)
    - B value appears only once:
        - B -> A,C,D
        - Because the values for B (b1,b2...) appear only once in every tuple, we can derive that B -> A,C,D
        - It is consistent with the data, but not derivable from them

- **Functional dependencies: C -> ?**
    - c1 - two times
        - (a1,b1,d1)
        - (a2,b2,d1)
            - D is the same
            - A & B are different
    - c2 - once
        - (a2,b3,d1)
    - c3 - once
        - (a3,b4,d2)
    - c4 - once
        - (a3,b5,d2)
    - c5 - once
        - (a3,b6,d3)
    - C -> A - false
    - C -> B - false
    - C -> D - **true**

Functional dependencies: D -> ?

- d1 - three times
    - a-b-c all different
- d2 - two times
    - a3 is the same
- d3 - once
- Because of the different values for d1, we have no functional dependencies for D
    - D -> A,B,C - **false**

Task 2: Keys and Covers (20 %)

- $R_2(A, B, C, D, E)$, $F\ D(R_2) = \{AE \rightarrow CD, CE \rightarrow B, BD \rightarrow A, C \rightarrow B, CE \rightarrow A\}$
- $R_3(A, B, C, D)$, $F\ D(R_3) = \{A \rightarrow CD, B \rightarrow D, BC \rightarrow A, AC \rightarrow B\}$
- $R_4(A, B, C, D, E, G)$, $F\ D(R_4) = \{AB \rightarrow C, C \rightarrow A, BC \rightarrow D, ACD \rightarrow B, D \rightarrow EG, BE \rightarrow C, CE \rightarrow AG, CG \rightarrow BD\}$

1 EN: What are candidate keys and how do they differ from superkeys?
DE: Was sind Schlüsselkandidaten und wie unterscheiden sich diese von Superschlüsseln?
Schlüsselkandidaten sind eine minimale Menge von Attributen mit der man alle anderen Attribute in einer Relation bestimmen kann. Es dürfen aber keine überflüssigen Attribute enthalten sein. Ein Schlüsselkandidat ist ein minimaler Superschlüssel. Denn bei Superschlüsseln können überflüssige Attribute vorkommen.

2 EN: When can you stop searching in the determination of candidate keys? **DE:** Wann kann man in der Bestimmung der Schlüsselkandidaten die Suche abbrechen?

Wenn alle Kandidaten gefunden wurden die die minimale Mengen sind, welche alle anderen Attribute der Relation in der Hülle enthalten sind.

3 EN: For R_2, determine all candidate keys.
DE: Bestimmen Sie für R_2 alle Schlüsselkandidaten.

CE
- ➔ A
- ➔ B
- ➔ D (CE->A AE->CD)
- ➔ C
- ➔ E

Also CE+ = {A,B,C,D,E}
CE ist minimal, denn C und E alleine können nicht alle Attribute darstellen.

4 EN: For R_3, determine all candidate keys. Based on your answer, list all prime and non-prime attributes.
DE: Bestimmen Sie für R_3 alle Schlüsselkandidaten. Listen Sie auf der Grundlage Ihrer Antwort alle Prime- und Non-Prime-Attribute auf.

BC
- ➔ A
- ➔ A-> CD
- ➔ BC

Also BC+ = {A,B,C,D}
BC ist minimal, denn B und C separat können nicht alle Attribute darstellen.

AC
- ➔ B

➔ A->CD

Also AC+ = {A,B,C,D}
AC ist minimal, da A und C separat nicht alle Attribute darstellen.

Prime-Attribute: A,B,C da alle Teil von Schlüsselkandidaten sind
Non-Prime-Attribute: D da D in keinem Schlüsselkandidaten vorkommt.

5 EN: For R_4, determine a minimal cover for the functional dependencies. **DE:** Bestimmen Sie für R_4 die minimal mögliche Abdeckung der funktionalen Abhängigkeiten.

D -> EG daraus folgt D->E, D->G

CE-> DG daraus folgt CE->A, CE->G

CG->BD daraus folgt CG->B, CG->D

Also ergibt sich die Menge { AB->C, C->A, BC->D, ACD->B, D->E, D->G, BE->C, CE->A, CE->G, CG->B, CG->D}

CE->A ist nicht notwendig, da C->A

CG->B ist nicht notwendig, da CG->D, C->A, ACD->B

Also ist die minimale Überdeckung: { AB->C, C->A, BC->D, ACD->B, D->E, D->G, BE->C, CE->G, CG->D}

Task 3: Armstrong Axioms (15 %)

▸ $R_5(A, B, C, D, E)$, $FD(R_5) = \{CE \rightarrow D, C \rightarrow BE, D \rightarrow A\}$

1 EN: Using Armstrong's axioms and their extensions, derive $C \rightarrow A$ from $FD(R_5)$. For each step, state the axiom used.
DE: Leiten Sie mit Hilfe der Armstrong-Axiome und ihrer Erweiterungen $C \rightarrow A$ aus $FD(R_5)$ ab. Geben Sie für jeden Schritt das verwendete Axiom an.

1. Decomposition rule (X->YZ then X->Y and X->Z):

   C->BE then C->B and **C->E**

+

2. Augmentation rule (X->Y, Z is attribute, XZ ->YZ):

   C->E (1), then CC->CE, CC = C by idempotence, **C->CE**

+

3. Transitivity rule (if X->Y and Y->Z then X-> Z):

   CE->D and D->A then **CE->A**

+

4. Transitivity rule (if X->Y and Y->Z then X-> Z):

   C->CE and CE->A then **C->A**

2 EN: Derive pseudo-transitivity using Armstrong's axioms (step by step, name each axiom).
DE: Leiten Sie die Pseudo-Transitivität mit Hilfe der Armstrong-Axiome ab (Schritt für Schritt, nennen Sie jedes Axiom).

Pseudotransitivity rule: If X->Y and ZY->T then XZ->T

X->Y then XZ->YZ	(augmentation rule)
YZ = ZY , XZ = ZX, XZ->YZ then ZX->ZY	(set commutativity)
ZX->ZY and ZY->T, then ZX->T	(transitivity rule)
ZX = XZ, then XZ->T	(set commutativity)

3 EN: Show that R_5 is not in 3NF (by contradiction).
DE: Zeigen Sie, dass R_5 nicht in 3NF ist (durch Widerspruch).

Zz: "A relation schema R is in 3NF if, whenever a FD (X->A) holds in R, either:
X is a superkey of R, or A is a prime attribute of R"

Presumptions: R_5(A, B, C, D, E) FD(R_5) = {CE->D, C->BE, D->A} and we have shown {C->A} in (1).

1. From the functional dependencies (FD) we know the keys are: CE, C, D
   C is **superkey**, because of C subset of CE, CE->D, C->BE and D->A. But we also know C->A, therefore D is not necessary.
2. We have shown C->A, which indicates C is a member of any key (A, B, C, D, E) of R, therefore C is also **the only prime attribute.**

If R_5 was in 3NF, D->A would imply that D is a superkey or A is a prime attribute. Conversely, if **A is not a prime attribute** (which is shown in 2), D *must* be a superkey in order for the 3NF property to hold. Because of $D^+ = \{A, D\} \neq \{A, B, C, D, E\}$ and also C->A, D->A, we know that **D is not a superkey.** This is a contradiction. ↯

Task 4: Normal Forms and Decomposition (20 %)

- $R_6(A, B, C, D, E)$, $FD(R_6) = \{A, B\} \rightarrow \{C\}, \{A\} \rightarrow \{B, E\}, \{C\} \rightarrow \{D\}$

- $R_7(E, F, G)$:

E	**F**	**G**
e_1	f_1	g_1
e_2	f_1	g_2
e_3	f_2	g_2
e_4	f_2	g_3

Table 2: Relation $R_7(E, F, G)$

- $R_8(A, B, C, D, E)$, $FD(R_8) = \{A, B, C\} \rightarrow \{E\}, \{B, E\} \rightarrow \{D\}, \{C\} \rightarrow \{A, B, E\}, \{D, E\} \rightarrow \{B, C\}, \{E\} \rightarrow \{C, D\}$

1 **EN:** Show that R_6 is not in 3NF. (Provide a justification.)
DE: Zeigen Sie, dass R_6 nicht in 3NF ist. (Begründen Sie dies.)

"A relation schema R is in 3NF if, whenever a FD (X->A) holds in R, either:
X is a superkey of R, or A is a prime attribute of R"

Here we see that C is not a superkey because it depends on A, B: {A,B}->{C}
But {D} is also dependent on a non-superkey {C}->{D}, which makes it a non-prime attribute, which is a contradiction of the 3NF definition ↯

2 **EN:** Show that R_7 (table below) is in BCNF.
DE: Zeigen Sie, dass R_7 (Tabelle unten) in BCNF ist.

"A relation is in BCNF if for each X → A in F+, X is a superkey"

e_1: appears once ($\mathbf{e_1}$, f_1, g_1)
e_2: appears once ($\mathbf{e_2}$, f_2, g_2)
e_3: appears once ($\mathbf{e_3}$, f_3, g_3)
e_4: appears once ($\mathbf{e_3}$, f_3, g_3)

f_1: appears twice ($\mathbf{e_1}$, f_1, g_1), ($\mathbf{e_2}$, f_1, g_2)
f_2: appears twice ($\mathbf{e_3}$, f_2, g_2), ($\mathbf{e_4}$, f_2, g_3)

g_1: appears once ($\mathbf{e_1}$, ...)
g_2: appears twice ($\mathbf{e_2}$), ($\mathbf{e_3}$)
g_3: appears once ($\mathbf{e_4}$)

We see that {E}->{E, F,G}, therefore {E} is superkey, which proves BCNF.

3 EN: Calculate a minimal cover for R_8.
DE: Bestimmen Sie die minimale Abdeckung für R_8.

- {E}->{C, D}: {E}->{C}, {E}->{D}
  - {B, E}->{D}: *covered by above* {E}->{D}
- {C}->{A, B, E}: {C->A, C->B, C->E}
  - {A, B, C}->{E}: *covered by above* C->E
- {D, E}->{B, C}: {D, E}->{B}, {D, E}->{C}

C->A, C->B, C->C, C->E->D: C->D, C->E *from* C->{A, B E}: *C is superkey*

From original set: C->A, C->B, C->E *are direct FDs and* E->C *and* E->D *are direct FDs, so all needed to provide minimal cover for* R_8(A, B, E, C, D) or R_8(A, B, C, D, E) (rearranged).

4 EN: Give a *good decomposition* of R_8 into two relations. Define "good decomposition" in this context, and prove your decomposition is lossless and dependency preserving.

DE: Geben Sie eine gute Zerlegung von R_8 in zwei Relationen an. Definieren Sie „gute Zer legung" in diesem Zusammenhang, und beweisen Sie, dass Ihre Zerlegung verlustfrei und ab hängigkeitserhaltend ist.

R_8(A, B, E, C, D) FD(R_8) = {A,B,C}->{E}, {B,E}->{D}, {C}->{A,B,E},{D, E}->{B, C}, {E}->{C, D}

R_1 = (A, B, E) F_{R1} = (C->A, C->B, C->E)

R_2 = (E, C, D) F_{R2}(E->C, E->D)

A good decomposition has lossless joins and preserves dependencies.

The decomposition is **lossless** because E is in R_1 and a E is a key in R_2.
The decomposition preserves dependencies: trivial plus {B,E}->{D} through transitivity {C->B, C->E}->{E->D}

Task 5: Indices (10 %)

1 **EN:** Explain the difference between a *Primary Index* and a *Secondary Index*. Give an example for each.
DE: Erklären Sie den Unterschied zwischen einem primären und einem sekundären Index. Geben Sie jeweils ein Beispiel an.

Der primäre Index bestimmt die Reihenfolge in einer sequentiell geordneten Datei (meistens der primäre Schlüssel). Beim sekundären Index geht es um eine Reihenfolge abseits der sequentiellen Reihenfolge.

Beispiel Primärer Index:

KundenID	Name	Stadt
1	Max	Potsdam
2	Lisa	Hamburg
3	Anna	Berlin
4	Alex	Köln

Die Tabelle ist nach der KundenID sequentiell sortiert.

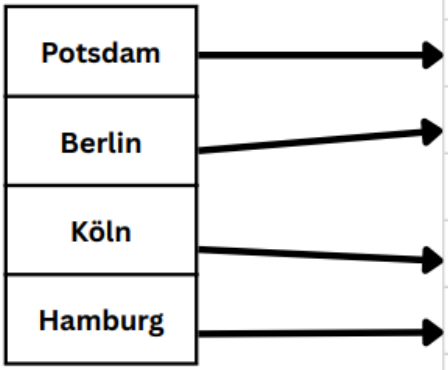
Beispiel Sekundärer Index: Wir verwenden die gleiche Tabelle wie bei dem primären Index. Nun wollen wir aber nach der Stadt suchen können.

Stadt	Stadt Index
Potsdam	1
Hamburg	2
Berlin	3
Köln	4

2 **EN:** Explain the difference between a *Dense Index* and a *Sparse Index*. Illustrate each with a small example sketch.
DE: Erklären Sie den Unterschied zwischen einem „Dense Index“ und einem „Sparse Index“. Veranschaulichen Sie beide mit einem kleinen beispielhaften Skizze.

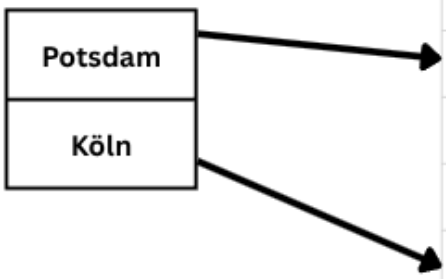
Beim Dense Index gibt es für jeden search-key Wert einen Index. Beim Sparse Index gibt es nur für ein paar search-key Werte einen Index. Dabei ist der Vorteil vom Denese Index die Geschwindigkeit und beim Sparse Index wird weniger Speicher verwendet.

Dense Index:



KundenID	Name	Stadt
1	Max	Potsdam
2	Lisa	Berlin
3	Anna	Berlin
4	Alex	Köln
5	Marie	Hamburg
6	Jared	Hamburg
7	Gregor	Hamburg

Sparse Index:



KundenID	Name	Stadt
1	Max	Potsdam
2	Lisa	Berlin
3	Anna	Berlin
4	Alex	Köln
5	Marie	Hamburg
6	Jared	Hamburg
7	Gregor	Hamburg

Task 6: Record Representation (15 %)

EN: The following data is to be stored: ”Me against the world” by artist ”Britney Ciccone” from ”Los Angeles”. ”A great big music” by artist ”Cristina Bocelli” from ”Miami”.

1 Give a record definition for the data in **Fixed-Length-Record format**, and present the corresponding table with entries.

```
type music_album = record
  title  : char(32);
  artist : char(16);
  city   : char(16);
end
```

	Title (CHAR 32)	Artist (CHAR 16)	City (CHAR 16)
0	“Me against the world ”	“Britney Ciccone “	“Los Angeles ”
1	”A great big music ”	“Cristina Bocelli”	“Miami ”

2 Do the same for Variable-Length-Record format.

A title has many words, an artist name has many words, city has many words etc.

```
type music_album = record
  title  : varchar();
  artist : varchar();
  city   : varchar();
end
```

	Title (string of word* sequences)					Artist (,,,)			City (...)		
0	“Me”	“against”	“the”	“word”	⊥	“Britney“	“Ciccone”	⊥	“Los”	“Angeles”	⊥
1	”A”	“great”	“big”	“music”	⊥	“Cristina”	“Bocelli”	⊥	“Miami”	⊥	⊥

*could be a symbol, fixed-length hash value or integer for a word table

3 Do the same for the Pointer format.

Here represented as a combination of both approaches to save space, pointers just for title:

	Title (one word + pointer)		Artist (CHAR 16)	City (CHAR 16)
0	“Me”	2	“Britney Ciccone ”	“Los Angeles ”
1	”A”	5	“Cristina Bocelli”	“Miami ”
2	“against”	3	⊥	⊥
3	“the”	4	⊥	⊥
4	“word”	5	⊥	⊥
5	“great”	6	⊥	⊥
6	“big”	7	⊥	⊥
7	“music”	⊥	⊥	⊥