Databases

Lecture 6

Functional Dependencies. Normal Forms (III)

Relational Algebra

Sabina S. CS

- R[A] - a relation
- F - a set of functional dependencies
- α – a subset of attributes

- problems

I. compute the closure of F: F^+

II. compute the closure of a set of attributes under a set of functional dependencies, e.g., the closure of α under F: α^+

III. compute the minimal cover for a set of dependencies

P2. Relational schema $R[ABCDEF]$, set of functional dependencies $S = \{A \to B, A \to C, CD \to E, CD \to F, D \to E\}, \alpha = \{A, D\}$. Compute α^+.

$\alpha^+ = \{A, D\}$

- $A \to B \Rightarrow \alpha^+ = \{A, B, D\}$
- $A \to C \Rightarrow \alpha^+ = \{A, B, C, D\}$
- $CD \to E \Rightarrow \alpha^+ = \{A, B, C, D, E\}$
- $CD \to F \Rightarrow \alpha^+ = \{A, B, C, D, E, F\}$
- $D \to E$

Sabina S. CS

- R[A] - a relation
- F - a set of functional dependencies
- problems

III. compute the minimal cover for a set of dependencies

Definition: F, G - two sets of functional dependencies; F and G are equivalent (notation $F \equiv G$) if $F^+ = G^+$.

->

- R[A] - a relation
- F - a set of functional dependencies
- problems

III. compute the minimal cover for a set of dependencies

Definition: F - set of functional dependencies; a minimal cover for F is a set of functional dependencies F_M that satisfies the following conditions:

1. $F_M \equiv F$

2. the right side of every dependency in F_M has a single attribute;

3. the left side of every dependency in F_M is irreducible (i.e., no attribute can be removed from the determinant of a dependency in F_M without changing F_M's closure);

4. no dependency *f* in F_M is redundant (no dependency can be discarded without changing F_M's closure).

- see lecture problem
  - R - a relation, F - a set of functional dependencies
    - compute F_M - minimal cover for F

P3. Relational schema $R[ABCD]$, set of functional dependencies $S=\{A \rightarrow BC, B \rightarrow C, A \rightarrow B, AB \rightarrow C, AC \rightarrow D\}$. Compute a minimal cover for S.

$A \rightarrow BC \Rightarrow A \rightarrow B$, $A \rightarrow C$

$B \rightarrow C$

~~$A \rightarrow B$~~

$AB \rightarrow C$

$AC \rightarrow D$

$A \rightarrow C \Rightarrow AA \rightarrow AC$

$A\boxed{C} \rightarrow D$

redundant

$\Rightarrow A \rightarrow D \Rightarrow$

$A \rightarrow B$

$A \rightarrow C$

$B \rightarrow C$

$AB \rightarrow C \Rightarrow$ $A \rightarrow C$, $AB \rightarrow CB \Rightarrow$ $AB \rightarrow C$, $AB \rightarrow B$ red.

$A \rightarrow D$

$\Rightarrow$ $A \rightarrow B$, $A \rightarrow C$, $B \rightarrow C$, $A \rightarrow D$ — minimal cover

Sabina S. CS

Example 11. Consider relation DFM[Department, FacultyMembers, MeetingDates], with repeating attributes *FacultyMembers* and *MeetingDates*.

- a possible instance is given below:

Department	FacultyMembers	MeetingDates
Computer Science	FCS1 FCS2 … FCSm	DCS1 DCS2 … DCSn
Mathematics	FM1 FM2 … FMp	DM1 DM2 … DMq

- eliminate repeating attributes (such that the relation is at least in 1NF) - replace DFM by a relation DFM' in which *FacultyMember* and *MeetingDate* are scalar attributes:

Department	FacultyMember	MeetingDate
Computer Science	FCS1	DCS1
Computer Science	FCS1	DCS2
…	…	…
Computer Science	FCS1	DCSn
Computer Science	FCS2	DCS1
Computer Science	FCS2	DCS2
…	…	…
Mathematics	FM1	DM1
…	…	…
Mathematics	FMp	DMq

Department	FacultyMember	MeetingDate
Computer Science	FCS1	DCS1
Computer Science	FCS1	DCS2
…	…	…
Computer Science	FCS1	DCSn
Computer Science	FCS2	DCS1
Computer Science	FCS2	DCS2
…	…	…
Mathematics	FM1	DM1
…	…	…
Mathematics	FMp	DMq

- in this table, each faculty member has the same meeting dates
- therefore, when adding / changing / removing rows, additional checks must be carried out

- a simple functional dependency $\alpha \rightarrow \beta$ means, by definition, that every value u of α is associated with a unique value v for β

Definition. Let $R[A]$ be a relation, with the set of attributes $A = \alpha \cup \beta \cup \gamma$. The multi-valued dependency $\alpha \rightrightarrows \beta$ (read α *multi-determines* β) is said to hold over R if each value u of α is associated with a set of values v for β: $\beta(u) = \{v_1, v_2, \dots, v_n\}$, and this association holds regardless of the values of γ.

->

- obs. $\sigma_{\alpha=u}(R)$ produces a relation that contains the tuples of R where $\alpha = u$

- let $R[A]$ be a relation, $\alpha \rightrightarrows \beta$ a multi-valued dependency, and $A = \alpha \cup \beta \cup \gamma$, with γ a non-empty set
- the association among the values in $\beta(u)$ for β and the value u of α holds regardless of the values of γ (the context)
- i.e., these associations (between u and an element in $\beta(u)$) exist for any value w in γ:
  - $\forall w \in \Pi_\gamma\left(\sigma_{\alpha=u}(R)\right)$, $\exists r_1, r_2, \ldots, r_n$ such that $\Pi_\alpha(r_i) = u$, $\Pi_\beta(r_i) = v_i$, $\Pi_\gamma(r_i) = w$

Name	FavoriteCake	FavoriteBand
Tinker Bell	Diplomat cake	Phoenix
Tinker Bell	Diplomat cake	Inna
Tinker Bell	Tiramisu	Phoenix
Tinker Bell	Tiramisu	Inna
Tinker Bell	Porumbita	Phoenix
Tinker Bell	Porumbita	Inna
Galadriel	Frog cake	Iron Maiden
Galadriel	Frog cake	Metallica

- if $\alpha \rightrightarrows \beta$ and the following rows exist:

α	β	γ
u_1	v_1	w_1
u_1	v_2	w_2

then the following rows must exist as well:

α	β	γ
u_1	v_1	w_2
u_1	v_2	w_1

normal forms – 4NF

Fairies[Name, FavoriteCake, FavoriteBand]

- constraint:

Name	FavoriteCake	FavoriteBand
f	c1	b1
f	c2	b2

=>

Name	FavoriteCake	FavoriteBand
f	c1	b1
f	c2	b2
f	c1	b2
f	c2	b1

Property. Let $R[A]$ be a relation, $A = \alpha \cup \beta \cup \gamma$. If $\alpha \rightrightarrows \beta$, then $\alpha \rightrightarrows \gamma$.

Justification.

- Let u be a value of α in R.
- Let $\beta(u) = \Pi_\beta\big(\sigma_{\alpha=u}(R)\big)$, $\gamma(u) = \Pi_\gamma\big(\sigma_{\alpha=u}(R)\big)$ (the β and γ values in the tuples where $\alpha = u$).

Since $\alpha \rightrightarrows \beta \Rightarrow$

$\forall w \in \gamma(u), \forall v \in \beta(u), \exists r = (u, v, w)$, or

$\forall v \in \beta(u), \forall w \in \gamma(u), \exists r = (u, v, w)$,

therefore $\alpha \rightrightarrows \gamma$.

- for relation DFM' (in the previous example):

{Department} $\rightrightarrows$ *{FacultyMember}*, *{Department}* $\rightrightarrows$ *{MeetingDate}*

Definition. A relation R is in 4NF if, for every multi-valued dependency $\alpha \rightrightarrows \beta$ that holds over R, one of the statements below is true:

- $\beta \subseteq \alpha$ or $\alpha \cup \beta = R$, or ← dependency is trivial
- α is a superkey.

- trivial multi-valued dependency $\alpha \rightrightarrows \beta$ in relation R: $\beta \subseteq \alpha$ or $\alpha \cup \beta = R$

- if $R[\alpha, \beta, \gamma]$ and $\alpha \rightrightarrows \beta$ (non-trivial, α not a superkey), R is decomposed into the following relations:

$$R_1[\alpha, \beta] = \Pi_{\alpha \cup \beta}(R)$$

$$R_2[\alpha, \gamma] = \Pi_{\alpha \cup \gamma}(R)$$

Name	FavoriteCake
Tinker Bell	Diplomat cake
Tinker Bell	Tiramisu
Tinker Bell	Porumbita
Galadriel	Frog cake

Name	FavoriteBand
Tinker Bell	Phoenix
Tinker Bell	Inna
Galadriel	Iron Maiden
Galadriel	Metallica

- relation DFM' is decomposed into:

  DF [Department, FacultyMember]

  DM [Department, MeetingDate]

- a dependency (simple, multi-valued) in a relation can be eliminated via decompositions (the original relation is decomposed into a collection of new relations)
- nevertheless, there are relations without such dependencies that can still contain redundant information, which can be a source of errors in the database

->

Example 12. Consider relation FaPrCo[FacultyMember, Program, Course], storing the programs and courses for different faculty members; this relation has no functional dependencies; its key is {FacultyMember, Program, Course}

- consider the following data in the relation:

Fa	Pr	Co
F1	P1	C2
F1	P2	C1
F2	P1	C1
F1	P1	C1

- some associations appear in multiple records (redundant data):
    - faculty member F1 is teaching in program P1
    - faculty member F1 is teaching course C1
    - course C1 is taught in program P1

Fa	Pr	Co
F1	P1	C2
F1	P2	C1
F2	P1	C1
F1	P1	C1

- if some values in the relation are changed, e.g., "F1 will teach course C3 instead of course C1", several updates should be carried out (without knowing in how many records)
- the same is true for the following changes: "in P1, course C3 should replace course C1", "F1 is switching from program P1 to P3"

- the previous relation cannot be decomposed into 2 relations (via projection), because new data would be introduced through the join
- this claim can be justified by considering the three possible projections on two attributes:

FaPr	Fa	Pr
	F1	P1
	F1	P2
	F2	P1

FaCo	Fa	Co
	F1	C2
	F1	C1
	F2	C1

PrCo	Pr	Co
	P1	C2
	P2	C1
	P1	C1

- when evaluating $\mathrm{FaPr} * \mathrm{PrCo}$, the following data is obtained:

$\mathbf{R}' = \mathbf{FaPr} * \mathbf{PrCo}$	Fa	Pr	Co
	F1	P1	C2
	F1	P1	C1
	F1	P2	C1
	F2	P1	C2
	F2	P1	C1

- this result set contains an extra tuple, which didn't exist in the original relation
- the same is true for the other join combinations: $\mathrm{FaPr} * \mathrm{FaCo}$ and $\mathrm{PrCo} * \mathrm{FaCo}$

- when evaluating $\mathrm{R}' * \mathrm{FaCo}$ (i.e., $\mathrm{FaPr} * \mathrm{PrCo} * \mathrm{FaCo}$), the original relation FaPrCo is obtained
- conclusion: FaPrCo cannot be decomposed into 2 projections, but it can be decomposed into 3 projections, i.e., FaPrCo is *3-decomposable*:

$$\mathrm{FaPrCo} = \mathrm{FaPr} * \mathrm{PrCo} * \mathrm{FaCo}, \text{ or } \mathrm{FaPrCo} = *\,(\mathrm{FaPr}, \mathrm{PrCo}, \mathrm{FaCo})$$

- this conclusion (FaPrCo is 3-decomposable) is true for the data in the relation
- 3-decomposability can be specified as a constraint:

* *if (F1, P1) ∈ FaPr and (F1, C1) ∈ FaCo and (P1, C1) ∈ PrCo then (F1, P1, C1) ∈ FaPrCo*

- this restriction can be expressed on FaPrCo (all legal instances must satisfy the constraint):

* *if (F1, P1, C2) ∈ $FaPrCo$ and (F1, P2, C1) ∈ $FaPrCo$ and (F2, P1, C1) ∈ $FaPrCo$ then (F1, P1, C1) ∈ $FaPrCo$*

- consider the following relation instance:

Fa	Pr	Co
F1	P1	C2
F1	P2	C1

R[Fa, Pr, Co]

JD * {FaPrCo, FaPr}

↳ joining these gets you the og.

- if the previous restriction holds, then, if (F2, P1, C1) is added to the relation, (F1, P1, C1) must be also added:

Fa	Pr	Co
F1	P1	C2
F1	P2	C1
F2	**P1**	**C1**
F1	**P1**	**C1**

* what if (F1, P1, C1) is removed from the instance?

Definition. Let $R[A]$ be a relation and $R_i[\alpha_i]$, i=1,2, ...,m, the projections of R on α_i. R satisfies the join dependency $*\{\alpha_1, \alpha_2, \ldots, \alpha_\mathrm{m}\}$ if $R = R_1 * R_2 * \cdots * R_m$.

- FaPrCo has a join dependency because FaPrCo = FaPr $*$ PrCo $*$ FaCo

Definition. Relation R is in 5NF if every non-trivial JD is implied by the candidate keys in R.

- JD $*\{\alpha_1, \alpha_2, \ldots, \alpha_\mathrm{m}\}$ on R is trivial if at least one α_i is the set of all attributes of R.
- JD $*\{\alpha_1, \alpha_2, \ldots, \alpha_\mathrm{m}\}$ on R is implied by the candidate keys of R if each α_i is a superkey in R.

=> FaPrCo not in 5NF

- decomposition: projections on FaPr, PrCo, FaCo

Relational Algebra

- query languages in the relational model
  - relational algebra and calculus - formal query languages with a significant influence on SQL
    - relational algebra
      - queries are specified in an operational manner
    - relational calculus
      - queries describe the desired answer, without specifying how it will be computed (declarative)
  - not expected to be Turing complete
  - not intended for complex calculations
  - provide efficient access to large datasets
  - allow optimizations

- relational algebra
  - used by DBMSs to represent query execution plans
  - a relational algebra query:
    - is built using a collection of operators
    - describes a step-by-step procedure for computing the result set
    - is evaluated on the input relations' instances
    - produces an instance of the output relation
  - every operation returns a relation, so operators can be composed; the algebra is closed
  - the result of an algebra expression is a relation, and a relation is a set of tuples
- relational algebra on bags (multisets) - duplicates are not eliminated

Conditions

- conditions that can be used in several algebraic operators
- similar to the SELECT filter conditions

1. *attribute_name relational_operator value*

- *value* - attribute name, expression

2. *attribute_name IS [NOT] IN single_column_relation*

- a relation with one column can be considered a set
- the condition tests whether a value belongs to a set

3. *relation {IS [NOT] IN | = | <>} relation*

- the relations in the condition must be union-compatible

Conditions

4. *(condition)*
NOT condition
$condition_1$ *AND* $condition_2$
$condition_1$ *OR* $condition_2$,

where *condition*, $condition_1$, $condition_2$ are conditions of type 1-4.

Operators in the Algebra

- equivalent SELECT statements can be specified for the relational algebra expressions
- *selection*
  - notation: $\boxed{\sigma_C(R)}$
  - resulting relation:
    - schema: R's schema
    - tuples: records in R that satisfy condition C
  - equivalent SELECT statement

```
SELECT *
FROM R
WHERE C
```

PID	Name	Ruthlessness	Ship
p1	Jack Sparrow	1	BP
p2	Elizabeth Swann	2	BP
p3	Hector Barbossa	3	BP
p4	Will Turner	1	FD
p5	Davy Jones	5	FD
p6	Blackbeard	5	QAR

Pirates

$\sigma_{Ruthlessness < 4}(Pirates)$

- *projection*
  - notation: $\boxed{\pi_\alpha(R)}$
  - resulting relation:
    - schema: attributes in α
    - tuples: every record in R is projected on α
  - α can be extended to a set of expressions, specifying the columns of the relation being computed
  - equivalent SELECT statement

PID	Name	Ruthlessness	Ship
p1	Jack Sparrow	1	BP
p2	Elizabeth Swann	2	BP
p3	Hector Barbossa	3	BP
p4	Will Turner	1	FD
p5	Davy Jones	5	FD
p6	Blackbeard	5	QAR

Pirates

SELECT DISTINCT α
FROM R

$\pi_{PID, Name}(Pirates)$

$\pi_{PID, Name}(\sigma_{Ruthlessness<4}(Pirates))$

$\boxed{\text{SELECT } \alpha}$
FROM R   -- algebra on bags

* they can be composed since σ produces a relation

- *cross-product*
  - notation: $\boxed{R_1 \times R_2}$
  - resulting relation:
    - schema: the attributes of R_1 followed by the attributes of R_2
    - tuples: every tuple r_1 in R_1 is concatenated with every tuple r_2 in R_2
  - equivalent SELECT statement

```
SELECT *
FROM R1 CROSS JOIN R2
```

PID	Name	Ruthlessness	Ship
p1	Jack Sparrow	1	BP
p2	Elizabeth Swann	2	BP
p3	Hector Barbossa	3	BP
p4	Will Turner	1	FD

Pirates

PID	TID	Value
p1	t1	100
p2	t9	5000
p3	t1	200

Shares

12 entries

- *union, set-difference, intersection*
  - notation: $R_1 \cup R_2$, $R_1 - R_2$, $R_1 \cap R_2$
  - R_1 and R_2 must be union-compatible:
    - same number of columns
    - corresponding columns, taken in order from left to right, have the same domains
  - equivalent SELECT statements

Pirates ~~∩ Shares~~
not compatible

```
SELECT *
FROM R1
UNION
SELECT *
FROM R2
```

```
SELECT *
FROM R1
EXCEPT
SELECT *
FROM R2
```

```
SELECT *
FROM R1
INTERSECT
SELECT *
FROM R2
```

```
-- algebra on bags: SELECT statements that don't eliminate
duplicates (e.g., UNION ALL)
```

- join operators
  - *condition join* (or *theta join*)
    - notation: $R_1 \otimes_\Theta R_2$
    - result: the records in the cross-product of R_1 and R_2 that satisfy a certain condition
  - definition $\Rightarrow$ $\boxed{R_1 \otimes_\Theta R_2 = \sigma_\Theta(R_1 \times R_2)}$
  - equivalent SELECT statement

```
SELECT *
FROM R1 INNER JOIN R2 ON Θ
```

$Pirates \otimes_{Pirates.PID > Shares.PID} Sh$

PID	Name	Ruthlessness	Ship
p1	Jack Sparrow	1	BP
p2	Elizabeth Swann	2	BP
p3	Hector Barbossa	3	BP
p4	Will Turner	1	FD

Pirates

PID	TID	Value
p1	t1	100
p2	t9	5000
p3	t1	200

Shares

- join operators
  - *natural join*
    - notation: $R_1 * R_2$
    - resulting relation:
      - schema: the union of the attributes of the two relations (attributes with the same name in R_1 and R_2 appear once in the result)
      - tuples: obtained from tuples $<r_1, r_2>$, where r_1 in R_1, r_2 in R_2, and r_1 and r_2 agree on the common attributes of R_1 and R_2
    - let $R_1[\alpha], R_2[\beta], \alpha \cap \beta = \{A_1, A_2, \dots, A_m\}$; then:

$$R_1 * R_2 = \pi_{\alpha \cup \beta}(R_1 \otimes_{R_1.A_1 = R_2.A_1 \ AND \ \dots \ AND \ R_1.A_m = R_2.A_m} R_2)$$

    - equivalent SELECT statement

```
SELECT *
FROM R1 NATURAL JOIN R2
```

PID	Name	Ruthlessness	Ship
p1	Jack Sparrow	1	BP
p2	Elizabeth Swann	2	BP
p3	Hector Barbossa	3	BP
p4	Will Turner	1	FD

Pirates

PID	TID	Value
p1	t1	100
p2	t9	5000
p3	t1	200

Shares

Pirates * Shares

$\pi_{Name}(\sigma_{TID=t1}(Shares \bowtie Pirates))$ ← get names of pirates on ship 3

Name
Jack Sparrow
Hector Barbossa

PID	Name	Ruthlessness	Ship
p1	Jack Sparrow	1	BP
p2	Elizabeth Swann	2	BP
p3	Hector Barbossa	3	BP
p4	Will Turner	1	FD

Pirates

PID	TID	Value
p1	t1	100
p2	t9	5000
p3	t1	200

Shares

$\pi_{Name}(\sigma_{TID=t1}(Shares) \bowtie Pirates)$ ← does the same

Name
Jack Sparrow
Hector Barbossa

* more efficient

Because $\sigma_{TID=t1}$ is a smaller subset that is then joined with Pirates

PID	Name	Ruthlessness	Ship
p1	Jack Sparrow	1	BP
p2	Elizabeth Swann	2	BP
p3	Hector Barbossa	3	BP
p4	Will Turner	1	FD

Pirates

PID	TID	Value
p1	t1	100
p2	t9	5000
p3	t1	200

Shares

- Compute cruel pirates with no shares from serbia

Ruthlessness > 4

PID	Name	Ruthlessness	Ship
p1	Jack Sparrow	1	BP
p2	Elizabeth Swann	2	BP
p3	Hector Barbossa	3	BP
p4	Will Turner	1	FD

Pirates

PID	TID	Value
p1	t1	100
p2	t9	5000
p3	t1	200

Shares

$$\left(\tilde{\pi}_{PID}\left(\sigma_{Ruthlessness > 4}(Pirates)\right) - \tilde{\pi}_{PID}\left(Shares * \left(\sigma_{country = \text{"serbia"}}(Treasures)\right)\right)\right)$$

- join operators
  - *left outer join*
    - notation (in these notes): $R_1 \ltimes_C R_2$
    - resulting relation:
      - schema: the attributes of R_1 followed by the attributes of R_2
      - tuples: tuples from the condition join $R_1 \otimes_c R_2$ + the tuples in R_1 that were not used in $R_1 \otimes_c R_2$ combined with the *null* value for the attributes of R_2
    - equivalent SELECT statement

```
SELECT *
FROM R1 LEFT OUTER JOIN R2 ON C
```

- join operators
  - *right outer join*
    - notation: $R_1 \rtimes_C R_2$
    - resulting relation:
      - schema: the attributes of R_1 followed by the attributes of R_2
      - tuples: tuples from the condition join $R_1 \otimes_c R_2$ + the tuples in R_2 that were not used in $R_1 \otimes_c R_2$ combined with the *null* value for the attributes of R_1
    - equivalent SELECT statement

```
SELECT *
FROM R1 RIGHT OUTER JOIN R2 ON C
```

PID	Name	Ruthlessness	Ship
p1	Jack Sparrow	1	BP
p2	Elizabeth Swann	2	BP
p3	Hector Barbossa	3	BP ..
p4	Will Turner	1	FD ...

Pirates

PID	TID	Value
p1	t1	100
p2	t9	5000
p5	t1	200

null

null

Shares

Pirates $\bowtie_{P.PID=T.PID}$ Shares

- join operators
  - *full outer join*
    - notation: $R_1 \bowtie_C R_2$
    - resulting relation:
      - schema: the attributes of R_1 followed by the attributes of R_2
      - tuples:
        - tuples from the condition join $R_1 \otimes_c R_2$ +
        - the tuples in R_1 that were not used in $R_1 \otimes_c R_2$ combined with the *null* value for the attributes of R_2 +
        - the tuples in R_2 that were not used in $R_1 \otimes_c R_2$ combined with the *null* value for the attributes of R_1
    - equivalent SELECT statement

```
SELECT *
FROM R1 FULL OUTER JOIN R2 ON C
```

- join operators
  - *left semi join*
    - notation: $R_1 \rhd R_2$
    - resulting relation:
      - schema: R_1's schema
      - tuples: the tuples in R_1 that are used in the natural join $R_1 * R_2$

Pirates ⊳ Shares

PID	Name	Ruthlessness	Ship
p1	Jack Sparrow	1	BP
p2	Elizabeth Swann	2	BP
p3	Hector Barbossa	3	BP
p4	Will Turner	1	FD

Pirates

PID	TID	Value
p1	t1	100
p2	t9	5000
p5	t1	200

Shares

- join operators
  - *right semi join*
    - notation: $R_1 \lhd R_2$
    - resulting relation:
      - schema: R_2's schema
      - tuples: the tuples in R_2 that are used in the natural join $R_1 * R_2$

Pirates ⊳ Shares

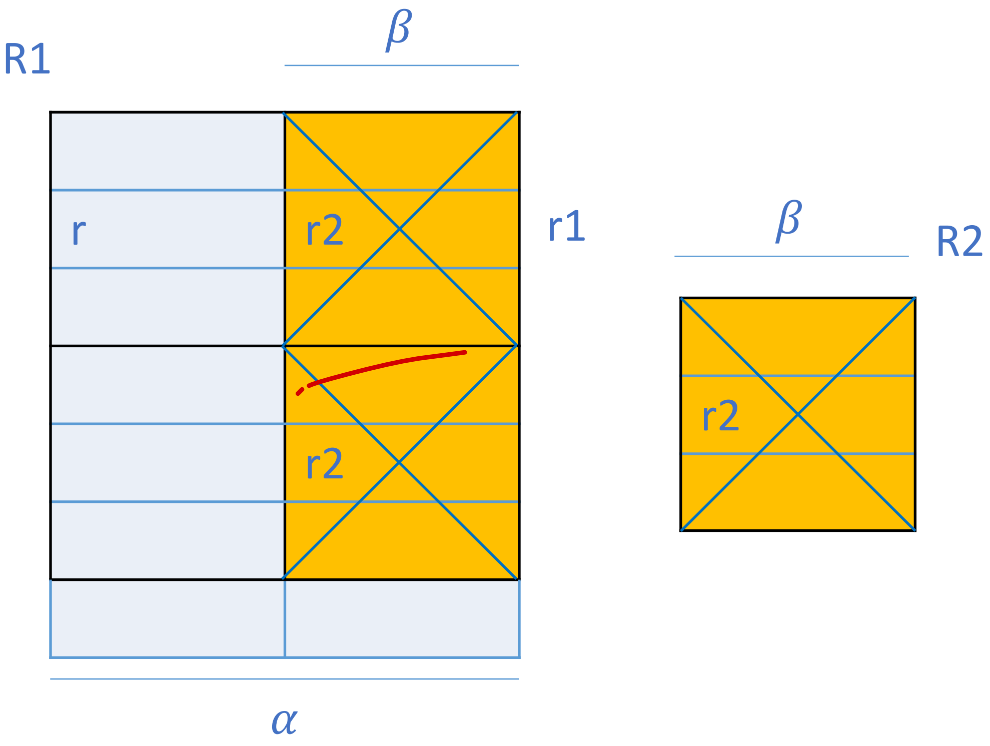
PID	Name	Ruthlessness	Ship
p1	Jack Sparrow	1	BP
p2	Elizabeth Swann	2	BP
p3	Hector Barbossa	3	BP
p4	Will Turner	1	FD

Pirates

PID	TID	Value
p1	t1	100
p2	t9	5000
p5	t1	200

Shares

- *division*
  - notation: $R_1 \div R_2$
  - $R_1[\alpha]$, $R_2[\beta]$, $\beta \subset \alpha$!
  - resulting relation:
    - schema: $\alpha - \beta$
    - tuples: a record $r \in R_1 \div R_2$ if $\forall r_2 \in R_2$, $\exists r_1 \in R_1$ such that:
      - $\pi_{\alpha-\beta}(r_1) = r$
      - $\pi_{\beta}(r_1) = r_2$
      - i.e., a record r belongs to the result if in R_1 r is concatenated with every record in R_2



PID	TID
p1	t1
p1	t2
p1	t3
p2	t1
p2	t2
p3	t1

S

T_1

TID
t1

T_2

TID
t1
t2

T_3

TID
t1
t2
t3

PID	Name	Ruthlessness	Ship
p1	Jack Sparrow	1	BP
p2	Elizabeth Swann	2	BP
p3	Hector Barbossa	3	BP
p4	Will Turner	1	FD

Pirates

PID	TID	Value
p1	t1	100
p2	t9	5000
p5	t1	200

Shares

$S \div T_1 =$

PID
p1
p2
p3

$S \div T_2 =$

PID
p1
p2

$S \div T_3 =$

PID
p1

- see lecture examples (at the board) with algebra queries:

- selection

- projection

- division

- selection, projection

- natural join, selection, projection

- set-difference, natural join, selection, projection

- different algebra expressions producing the same result (optimization - reducing the size of intermediate relations)

References

- [Ta13] ȚÂMBULEA, L., Curs Baze de date, Facultatea de Matematică și Informatică, UBB, 2013-2014
- [Ra00] RAMAKRISHNAN, R., GEHRKE, J., Database Management Systems (2nd Edition), McGraw-Hill, 2000
- [Da03] DATE, C.J., An Introduction to Database Systems (8th Edition), Addison-Wesley, 2003
- [Ga08] GARCIA-MOLINA, H., ULLMAN, J., WIDOM, J., Database Systems: The Complete Book, Prentice Hall Press, 2008
- [Ha96] HANSEN, G., HANSEN, J., Database Management And Design (2nd Edition), Prentice Hall, 1996
- [Ra07] RAMAKRISHNAN, R., GEHRKE, J., Database Management Systems, McGraw-Hill, 2007, http://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/slides/slides3ed.html
- [Ul11] ULLMAN, J., WIDOM, J., A First Course in Database Systems, http://infolab.stanford.edu/~ullman/fcdb.html