Databases

Lecture 7

Relational Algebra (II)

Sabina S. CS

- the *renaming* operator

$\rho(\mathrm{R}'(\mathrm{A}_1 \to \mathrm{A}_1{}', \mathrm{A}_2 \to \mathrm{A}_2{}', \mathrm{A}_3 \to \mathrm{A}_3{}'), \mathrm{E})$

  - E - relational algebra expression
  - the result, relation R', has the same tuples as the result of E
  - attributes A_1, A_2, and A_3 are renamed to $\mathrm{A}_1{}'$, $\mathrm{A}_2{}'$, and $\mathrm{A}_3{}'$, respectively

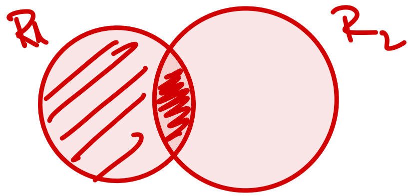
An Independent Subset of Operators

- independent set of operators M:
  - eliminating any operator *op* from M: there will be a relation that can be obtained using M's operators, but cannot be obtained with the operators in M-{*op*}
- for the previously described query language, with operators:

  $\{\sigma, \pi, \times, \cup, -, \cap, \otimes, *, \ltimes, \rtimes, \bowtie, \rhd, \lhd, \div\}$

an independent set of operators is $\{\sigma, \pi, \times, \cup, -\}$

- the other operators are obtained as follows (some expressions have already been introduced):
  - $R_1 \cap R_2 = R_1 - (R_1 - R_2)$
  - $R_1 \otimes_C R_2 = \sigma_C(R_1 \times R_2)$



- the other operators are obtained as follows (some expressions have already been introduced):
  - $R_1[\alpha], R_2[\beta], \alpha \cap \beta = \{A_1, A_2, \dots, A_m\}$, then:

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

  - $R_1[\alpha], R_2[\beta], R_3[\beta] = \{(null, \dots, null)\}, R_4[\alpha] = \{(null, \dots, null)\}$

    all R_1 with no match in R_2

    $$R_1 \ltimes_C R_2 = (R_1 \otimes_c R_2) \cup (R_1 - \pi_\alpha(R_1 \otimes_c R_2)) \times R_3$$
    $$R_1 \rtimes_C R_2 = (R_1 \otimes_c R_2) \cup R_4 \times (R_2 - \pi_\beta(R_1 \otimes_c R_2))$$
    $$R_1 \bowtie_C R_2 = (R_1 \ltimes_C R_2) \cup (R_1 \rtimes_C R_2)$$

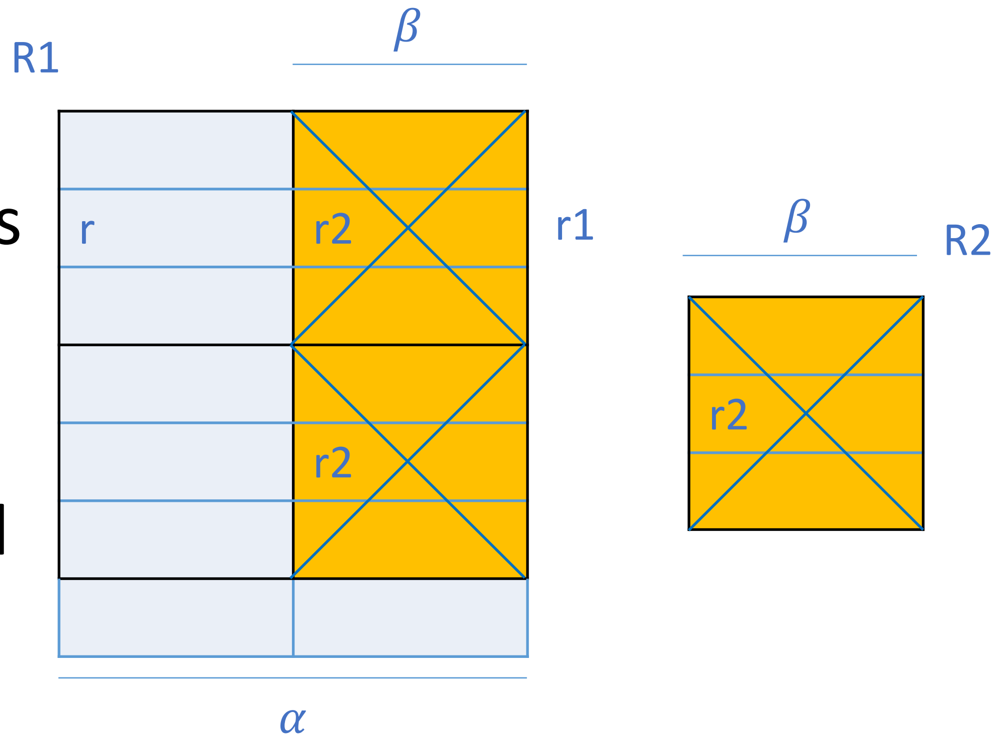
    all R_2 with no match in R_1

  - $R_1[\alpha], R_2[\beta]$

    $$R_1 \rhd R_2 = \pi_\alpha(R_1 * R_2)$$
    $$R_1 \lhd R_2 = \pi_\beta(R_1 * R_2)$$

- the other operators are obtained as follows (some expressions have already been introduced):
  - if $R_1[\alpha], R_2[\beta], \beta \subset \alpha$, then $r \in R_1 \div R_2$ if $\forall r_2 \in R_2, \exists r_1 \in R_1$ such that: $\pi_{\alpha-\beta}(r_1) = r$ and $\pi_{\beta}(r_1) = r_2$

    => r is in $\pi_{\alpha-\beta}(R_1)$, but not all the elements in $\pi_{\alpha-\beta}(R_1)$ are in the result
  - $(\pi_{\alpha-\beta}(R_1)) \times R_2$ contains all the elements with one part in $\pi_{\alpha-\beta}(R_1)$ and the second part in R_2
  - to obtain values that are disqualified, R_1 is subtracted from the obtained relation, and the result is projected on $\alpha - \beta$
  - the final expression:



$$R_1 \div R_2 = \pi_{\alpha-\beta}(R_1) - \pi_{\alpha-\beta}((\pi_{\alpha-\beta}(R_1)) \times R_2 - R_1)$$

$(\pi_{\alpha-\beta}(R_1)) \times R_2$ elems with one part in $\pi_{\alpha-\beta}(R_1)$ and 2nd in R_2

α β

Fairy	Castle
Tinker Bell	Hogwarts School of Witchcraft and Wizardry
Tinker Bell	Far Far Away Palace
Craiasa Zanelor	Hogwarts School of Witchcraft and Wizardry
Tinker Bell	Rivendell
Craiasa Zanelor	Far Far Away Palace
Galadriel	Hogwarts School of Witchcraft and Wizardry
Galadriel	Rivendell
Galadriel	Far Far Away Palace

Castle
Hogwarts School of Witchcraft and Wizardry
Far Far Away Palace
Rivendell

* all fairies teaching at all castles

$\Rightarrow$

Fairy	Castle
Tinker Bell	Hogwarts School of Witchcraft and Wizardry
Tinker Bell	Far Far Away Palace
Tinker Bell	Rivendell
Craiasa Zanelor	Hogwarts School of Witchcraft and Wizardry
Craiasa Zanelor	Far Far Away Palace
Craiasa Zanelor	Rivendell
Galadriel	Hogwarts School of Witchcraft and Wizardry
Galadriel	Far Far Away Palace
Galadriel	Rivendell

$-$

$\Rightarrow$ Craiasa Zanelor – Rivendell

$\pi_{\alpha-\beta}$(Cră...) = Crăiasa Zanelor

=

Fairy
Tinker Bell
Galadriel

* the next examples use the statements below:

- assignment

$R[list] := expression$

  - the expression's result (a relation) is assigned to a variable (R[list]), specifying the name of the relation [and the names of its columns]

- eliminating duplicates from a relation

$\delta(R)$

- sorting records in a relation

$S_{\{list\}}(R)$

- grouping

$\gamma_{\{list1\}\ group\ by\ \{list2\}}(R)$

  - R's records are grouped by the columns in *list2*
  - *list1* (that can contain aggregate functions) is evaluated for each group of records

FK

```
students [id, name, sgroup, gpa, dob]
groups [id, year, program]
```

FK

```
schedule [day, starthour, endhour, activtype, room, sgroup,
facultym_id]
```

FK

```
faculty_members [id, name]
```

1. The names of students in a given group:

$$R := \boxed{\pi_{\{name\}}}\left(\boxed{\sigma_{sgroup='222'}}(students)\right)$$

```
SELECT name
FROM students
WHERE sgroup='222'
```

2. The students in a given program (alphabetical list, by groups):

$$\boxed{G} := \pi_{\{id\}}\left(\sigma_{program='IG'}(groups)\right)$$

$$R := \underbrace{S_{\{sgroup,name\}}}_{\text{sort}}\left(\boxed{\sigma_{sgroup\ is\ in\ G}}(students)\right)$$

```
SELECT *
FROM students
WHERE sgroup IN
   (SELECT id
   FROM groups
   WHERE program='IG')
ORDER BY sgroup, name
```

```
students [id, name, sgroup, gpa, dob]
groups [id, year, program]
schedule [day, starthour, endhour, activtype, room,
  sgroup, facultym_id]
faculty_members [id, name]
```

3. The number of students in every group of a given program:

$$ST := \sigma_{sgroup\ is\ in\ \left(\underbrace{\pi_{\{id\}}\left(\sigma_{program='IG'}(groups)\right)}_{\text{IDs of groupd doing IG}}\right)}(students)$$

studs from those groups

$$NR := \gamma_{\{sgroup,\ count(*)\}\ group\ by\ \{sgroup\}}(ST)$$

```
SELECT sgroup, COUNT(*)
FROM (SELECT *
      FROM students
      WHERE sgroup IN
          (SELECT id
           FROM groups
           WHERE program='IG')
     ) t
GROUP BY sgroup
```

```
students [id, name, sgroup, gpa, dob]
groups [id, year, program]
schedule [day, starthour, endhour, activtype, room,
  sgroup, facultym_id]
faculty_members [id, name]
```

4. A student's schedule (the student is given by name):

$$T \coloneqq \sigma_{sgroup\ is\ in\left(\pi_{\{sgroup\}}\left(\sigma_{name='Ionescu\ M.\ Razvan'}(students)\right)\right)}(schedule)$$

5. The number of hours per week for every group:

$$F(\overset{\text{name of column}}{no}, sgroup) \coloneqq \pi_{\{\underbrace{endhour-starthour}_{\#hours},sgroup\}}(schedule)$$

$$NoHours(sgroup, nohours) \coloneqq \gamma_{\{sgroup,sum(no)\}\ group\ by\ \{sgroup\}}(F)$$

```
students [id, name, sgroup, gpa, dob]
groups [id, year, program]
schedule [day, starthour, endhour, activtype, room, sgroup, facultym_id]
faculty_members [id, name]
```

6. The faculty members (their names) who teach a given student:

$$A := (\sigma_{name='Ionescu\ M.\ Razvan'}(students)) \otimes_{students.sgroup=schedule.sgroup} schedule$$

$$B := \pi_{\{facultym\_id\}}(A)$$ ← gets all FM of A

$$C := faculty\_members \otimes_{faculty\_members.id=B.facultym\_id} B$$

$$D := \pi_{\{name\}}(C)$$

```
students [id, name, sgroup, gpa, dob]

groups [id, year, program]

schedule [day, starthour, endhour, activtype, room, sgroup, facultym_id]

faculty_members [id, name]
```

7. The faculty members with no teaching assignments (i.e., not on the schedule):

$$C := \pi_{\{name\}}(faculty\_members) - \pi_{\{name\}}(schedule \otimes_{schedule.facultym\_id=faculty\_members.id} faculty\_members)$$

* Is there a problem if two different faculty members have the same name?

yes, better take PIDs

```
students [id, name, sgroup, gpa, dob]

groups [id, year, program]

schedule [day, starthour, endhour, activtype, room, sgroup, facultym_id]

faculty_members [id, name]
```

8. Students with school activities on every day of the week (all days with school activities considered):

no duplicates

$$A \coloneqq \boxed{\delta}\left(\pi_{\{day\}}(schedule)\right)$$

$$B \coloneqq students \otimes_{students.sgroup=schedule.sgroup} schedule$$

$$C \coloneqq \delta\left(\pi_{\{name,day\}}(B)\right)$$

$$D \coloneqq C \div A$$

* Is there a problem if two different students have the same name?

yes, take SID

```
students [id, name, sgroup, gpa, dob]
groups [id, year, program]
schedule [day, starthour, endhour, activtype, room, sgroup, facultym_id]
faculty_members [id, name]
```

Milestone - review

- Databases Fundamentals
- The Relational Model
- SQL
- Functional Dependencies. Normal Forms
- Relational Algebra

See lecture problem (solved at the board)

- Create a database for a system that manages several funding portals, which bring together investors and entrepreneurs seeking funding for their startups. The entities of interest to the problem domain are: Funding Portals, Investors, Entrepreneurs, Startups, and Investments. A funding portal has a name and a website URL. An investor can offer funding through several portals, has a first name, last name, and date of birth. An entrepreneur has a first name, last name, and a startup success probability score; (s)he can own several startups. A startup has a name and description; it belongs to an entrepreneur. An investment is made by an investor for a startup through one of the funding portals the investor is registered on; it has a value (the invested amount of money) and a date. An investor can finance the same startup multiple times (through the same portal or through different portals).

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
- [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