

Databases

Lecture 7

Relational Algebra

The Physical Structure of Databases

Relational Algebra

- *cross product*
 - binary operator
 - notation: $R_1 \times R_2$
 - 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`

- *union, set-difference, intersection*
 - binary operators
 - notation: $R_1 \cup R_2$, $R_1 - R_2$, $R_1 \cap R_2$
 - R_1 and R_2 must have compatible schemas
 - equivalent SELECT statements
 - `SELECT * FROM R1 UNION ALL SELECT * FROM R2`
 - `SELECT * FROM R1 EXCEPT SELECT * FROM R2`
 - `SELECT * FROM R1 INTERSECT SELECT * FROM R2`

- 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 meet a certain condition
 - definition $\Rightarrow R_1 \otimes_{\theta} R_2 = \sigma_{\theta}(R_1 \times R_2)$
 - equivalent SELECT statement
 - `SELECT * FROM R1 INNER JOIN R2 ON θ`

- join operators
 - *natural join*
 - notation: $R_1 * R_2$
 - resulting relation:
 - schema: the union of the attributes of the two relations
 - tuples: obtained from tuples $\langle r_1, r_2 \rangle$, 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:

$$R_1 * R_2 = \Pi_{\alpha \cup \beta} (R_1 \otimes_{R_1.A_1 = R_2.A_1 \text{ AND } \dots \text{ AND } R_1.A_m = R_2.A_m} R_2)$$
 - equivalent SELECT statement
 - SELECT * FROM R1 NATURAL JOIN R2

- join operators
 - *left outer join*
 - notation (in these notes): $R_1 \bowtie_C R_2$
 - resulting relation:
 - schema: the attributes of R_1 followed by the attributes of R_2
 - tuples: records from the condition join $R_1 \bowtie_C R_2$ + the records in R_1 that were not used in $R_1 \bowtie_C R_2$ 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: $R_1 \bowtie_C R_2$
 - resulting relation:
 - schema: the attributes of R_1 followed by the attributes of R_2
 - tuples: records from the condition join $R_1 \otimes_C R_2$ + the records 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`

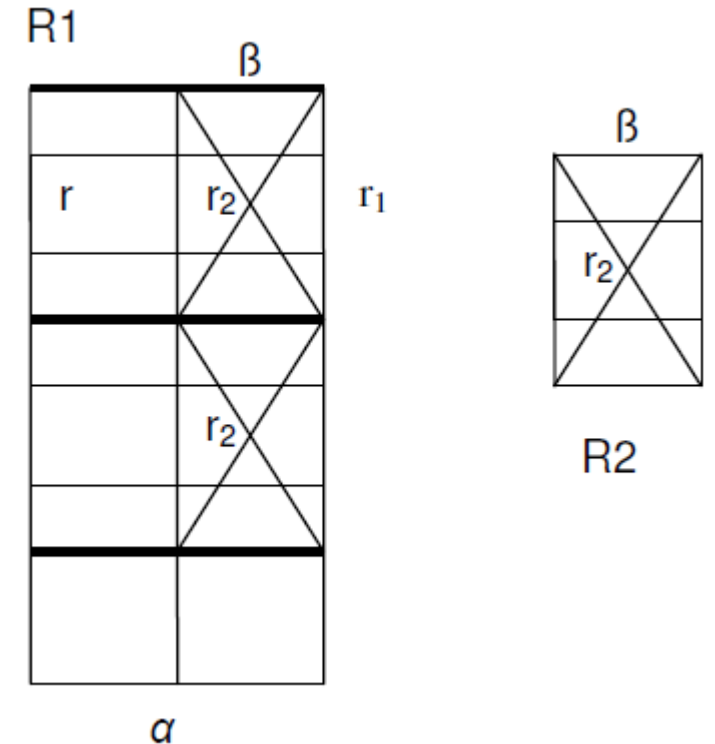
- join operators
 - *full outer join*
 - notation: $R_1 \bowtie_C R_2$
 - resulting relation:
 - schema: the attributes of R_1 followed by the attributes of R_2
 - tuples: the union of the left and right outer join results
 - equivalent SELECT statement
 - `SELECT * FROM R1 FULL OUTER JOIN R2 ON C`

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

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

- *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



- the *renaming* operator

$$\rho(R'(A_1 \rightarrow A_1', A_2 \rightarrow A_2', A_3 \rightarrow A_3'), R)$$

- the result, relation R' , has the same instance as R
- attributes A_1 , A_2 , and A_3 are renamed to A_1' , A_2' , and A_3' , respectively
- an algebra expression can be specified instead of R

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:
 $\{\sigma, \Pi, \times, \cup, \cap, -, *, \otimes, \ltimes, \rtimes, \bowtie, \triangleright, \triangleleft, \div\}$
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:

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

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

$$R_1 \bowtie_C R_2 = (R_1 \otimes_C R_2) \cup [R_1 - \Pi_{\alpha}(R_1 \otimes_C R_2)] \times R_3$$

$$R_1 \bowtie_C R_2 = (R_1 \otimes_C R_2) \cup R_4 \times [R_2 - \Pi_{\beta}(R_1 \otimes_C R_2)]$$

$$R_1 \bowtie_C R_2 = (R_1 \bowtie_C R_2) \cup (R_1 \bowtie_C R_2)$$

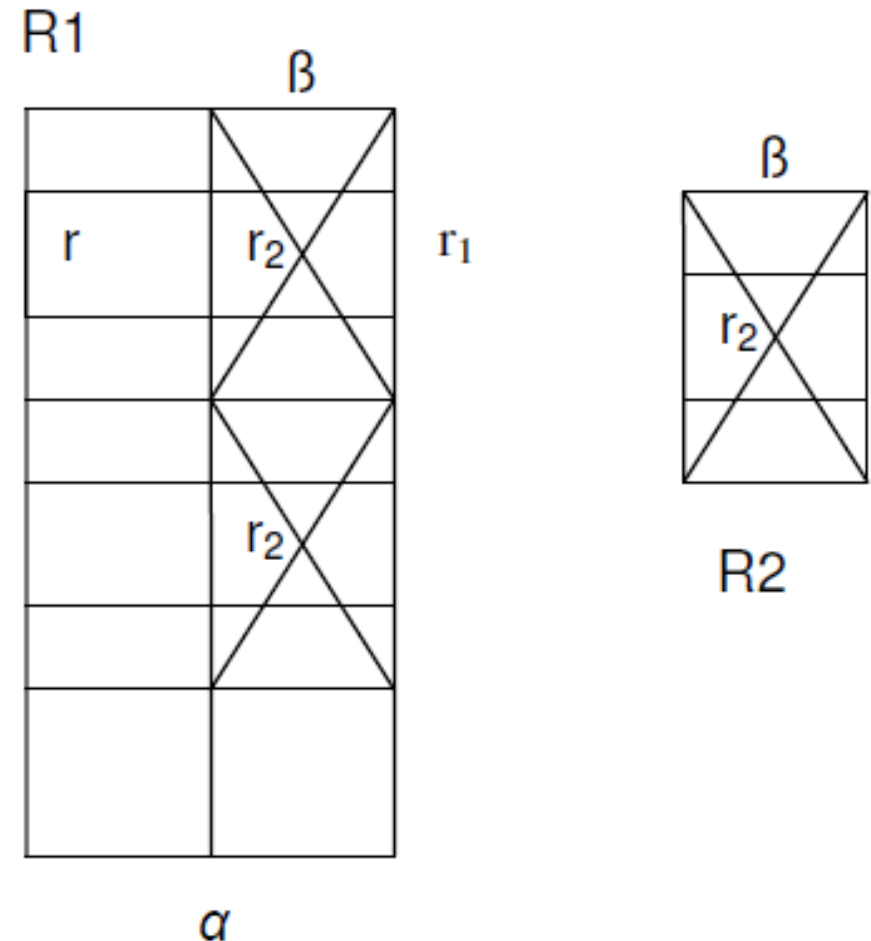
- $R_1[\alpha], R_2[\beta]$

$$R_1 \triangleright R_2 = \Pi_{\alpha}(R_1 * R_2)$$

$$R_1 \triangleleft R_2 = \Pi_{\beta}(R_1 * R_2)$$

- 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 $\Pi_{\beta}(r_1) = r_2$.
 - hence 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 eliminated from $(\Pi_{\alpha-\beta}(R_1)) \times R_2$, and the result is projected on $\alpha - \beta$
 - the final expression:

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



- see lecture examples (at the board) with algebra queries:
 - selection
 - projection
 - division
 - selection & projection
 - natural join with multiple tables, selection, projection
 - set-difference, natural join, selection, projection
 - different algebra expressions producing the same result (optimization - reducing the size of intermediate relations)

- the next examples use the statements below

- assignment:

$R[\text{list}] := \text{expression}$

- the expression's result, i.e., a relation, is assigned to a variable $R[\text{list}]$, specifying the name of the relation [and the names of its columns]

- eliminating duplicates from a relation

$\delta(R)$

- sorting records in a relation

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

- grouping - an extension for projection

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

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

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

groups [id, year, program]

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

faculty [id, name]

1. The names of students in a given group:

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

SELECT name

FROM students

WHERE sgroup='222'

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

$$G := \Pi_{\{id\}} \left(\sigma_{program='IG'}(groups) \right)$$
$$R := S_{\{sgroup, name\}} \left(\sigma_{sgroup \text{ is in } G}(students) \right)$$

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

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

$$ST := \sigma_{sgroup \text{ is in } \left(\Pi_{\{id\}} \left(\sigma_{program='IG'}(groups) \right) \right)}(students)$$

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

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

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

$$T := \sigma_{sgroup \text{ is in } \left(\Pi_{\{sgroup\}} \left(\sigma_{name='Ionescu'}(students) \right) \right)}(schedule)$$

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

$$F(no, sgroup) := \Pi_{\{endhour - starthour, sgroup\}}(schedule)$$
$$NoHours(sgroup, nohours) := \gamma_{\{sgroup, sum(no)\}} \text{ group by } \{sgroup\}(F)$$

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

$$A := (\sigma_{name='Ionescu'}(students)) \otimes_{students.sgroup=schedule.sgroup}(schedule)$$
$$B := \Pi_{\{faculty_id\}}(A)$$
$$C := faculty \otimes_{faculty.id=B.faculty_id}(B)$$
$$D := \Pi_{\{name\}}(C)$$

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

$$C := \Pi_{\{name\}}(faculty) - \Pi_{\{name\}}(schedule \otimes_{schedule.faculty_id=faculty.id} faculty)$$

* is there a problem if 2 different faculty members have the same name?

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

$$A := \delta \left(\Pi_{\{day\}}(schedule) \right)$$

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

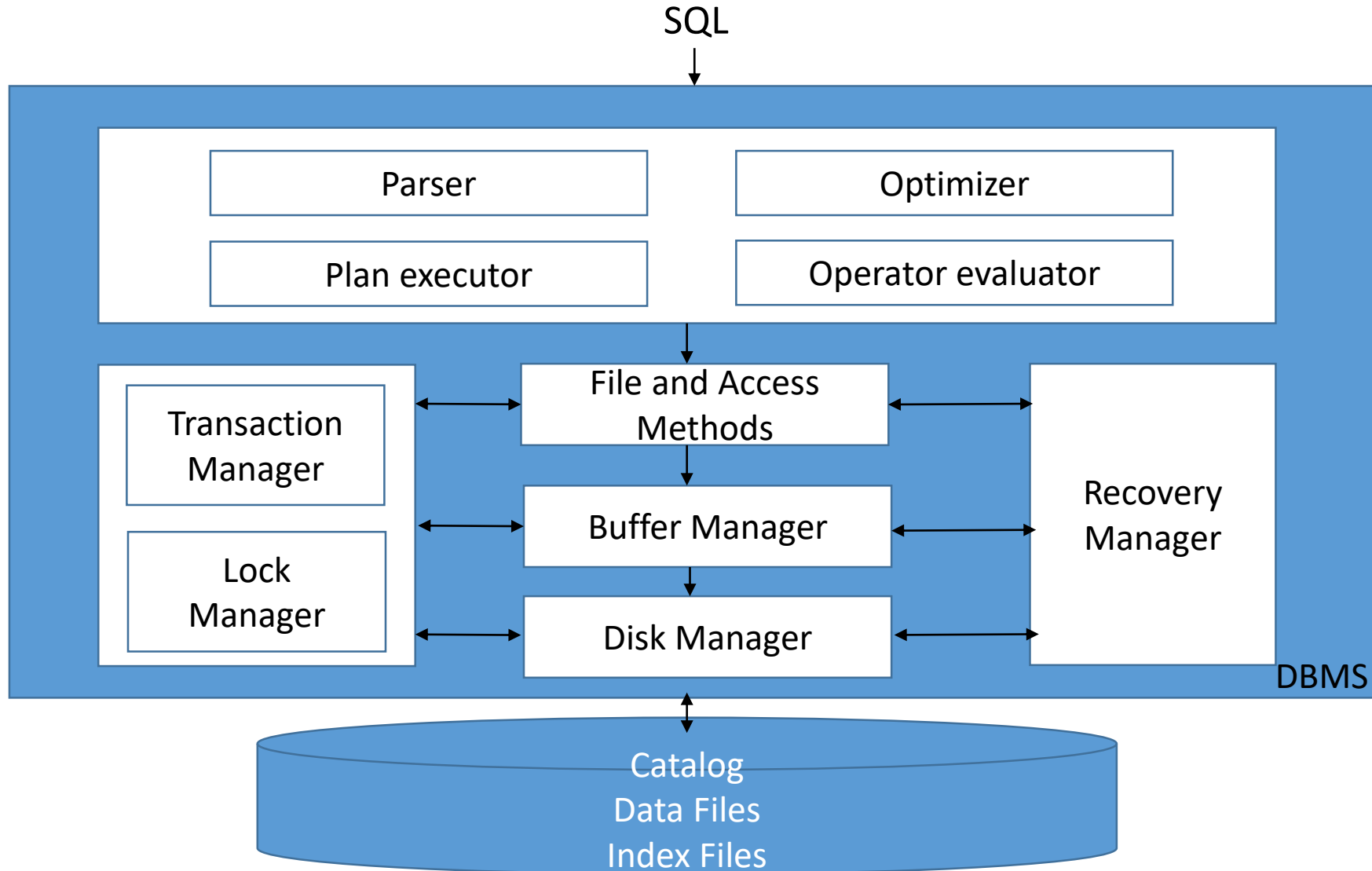
$$C := \Pi_{\{name, day\}}(B)$$

$$D := C \div A$$

* is there a problem if 2 different students have the same name?

The Physical Structure of Databases

DBMS Architecture



The Memory Hierarchy

- primary storage
 - cache, main memory
 - very fast access to data
 - volatile
 - currently used data
- secondary storage
 - slower devices, e.g., magnetic disks
 - nonvolatile
 - disks - sequential, direct access
 - main database

The Memory Hierarchy

- tertiary storage
 - slowest storage devices, e.g., optical disks, tapes
 - nonvolatile
 - tapes
 - only sequential access
 - good for archives, backups
 - unsuitable for data that is frequently accessed

The Memory Hierarchy

- secondary / tertiary storage
 - significantly cheaper than main memory
- large amounts of data that shouldn't be discarded when the system is restarted

=> the need for DBMSs that bring data from disks into main memory for processing

Secondary Storage – Magnetic Disks

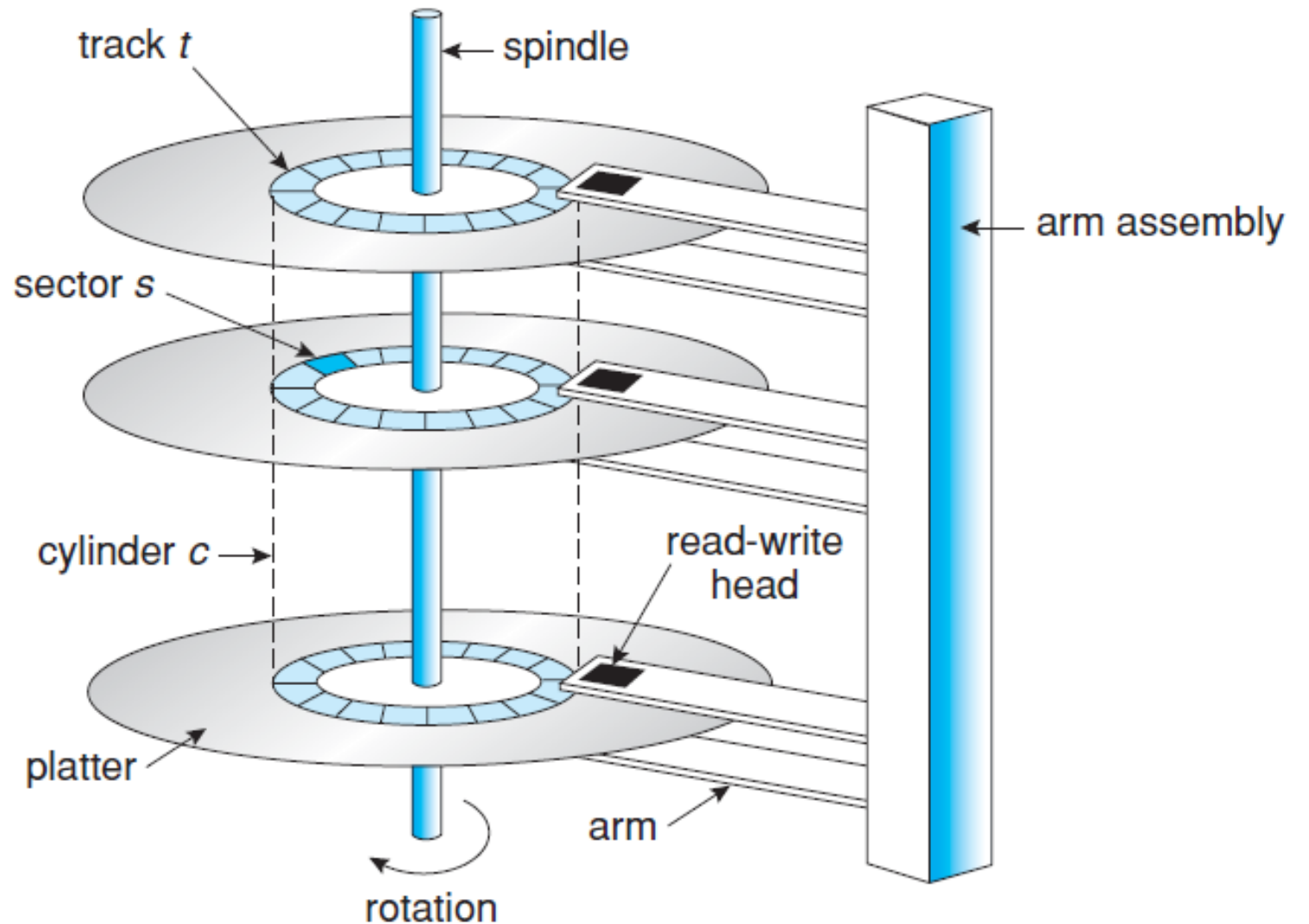
- direct access
- extremely used in database applications
- applications don't need to know whether the data is on disk or in main memory
- disk block
 - sequence of contiguous bytes
 - unit for data storage
 - unit for data transfer (reading / writing)
 - reading / writing a block - an input / output (I/O) operation
- tracks
 - concentric rings containing blocks, recorded on one or more platters

Secondary Storage – Magnetic Disks

- sectors
 - arcs on tracks
- platters
 - single-sided, double-sided (data recorded on one / both surfaces)
- cylinder
 - all tracks with the same diameter
- disk heads
 - one per recorded surface
 - to read / write a block, a head must be on top of the block
 - smaller platter size => decreased seek time
 - all the heads are moved as a unit
 - systems with one active head

Secondary Storage – Magnetic Disks

- sector size
 - characteristic of the disk, cannot be modified
- block size
 - multiple of the sector size



[Si08]

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>
- [Si08] SILBERSCHATZ, A., GALVIN, P.B., GAGNE, G., Operating System Concepts (8th Edition), Wiley Publishing, 2008
- [Ul11] ULLMAN, J., WIDOM, J., A First Course in Database Systems,
<http://infolab.stanford.edu/~ullman/fcdb.html>