Assignment Project Exam Help

https://eduassistpro.github.i

Add WeChat edu_assist_pr

Query Optimisation

- In practice, query optimisers incorporate elements of the following three optimisation approaches:

- **Rule-based query optimisation**

  Use heuristic rules to transform a relational equivalent one with a possibly lower cost.

- **Cost-based query optimisation**

  Use a cost model to estimate the costs of plans, and then select the most cost-effective plan.

Semantic Query Optimisation

- Can we use **semantic information** stored in a database (such as integrity constraints) to optimise queries?
  -
  - R
  -
    - entity integrity constraints
    - referential integrity constraints
    - domain constraints
    - ...
    - user-defined integrity constraints

- **Key idea:** Integrity constraints may **not only be utilized to enforce consistency** of a database, but may **also optimise user queries**.

Semantic Query Optimisation

- **E**

sn].

Query: `SELECT DISTINCT ssn FROM Empl`

- We can avoid extra costs for duplicate elimi
  constraint tells us that tuples in the result will be unique.

Semantic Query Optimisation

- **Example 2**:


```
FROM Employee
WHERE salary > 300000
```


- We do not need to execute a query if the existing constraint tells us that the result will be empty.

Semantic Query Optimisation

- **Example 3**:

-

number]


```
FROM Works_on INNER JOIN Project
on Works_on.pno=Proje
```


- We can reduce the number of joins by execut
  since both queries always return the same r

```
SELECT DISTINCT ssn
FROM Works_on ;
```

Rule-based Query Optimisation

- A rule-based optimisation transforms the RA expression by using a set of heuristic rules that typically improve the execution performance.
- **K**
  w

  - Apply as early as possible to reduce the number of tuples;
  - **Push-down projection:**
    Apply as early as possible to reduce the num

  - **Re-ordering joins**:
    Apply restrictive joins first to reduce the size of the result.
- But we must ensure that the resulting query tree gives the same result as the original query tree, i.e., **the equivalence of RA expressions**.

Heuristic Rules

Staff(sno, fname, lname, salary, position, branchNo)
Branch(branchNo, name, street, suburb, city)

- T... by... s, utilized...

(1) $\sigma_{\varphi}(\sigma_{\psi}(R)) \equiv \sigma_{\varphi \wedge \psi}(R)$;

$\sigma_{branchNo='H'}(\sigma_{salary>60000}(Staff)) =$)

(2) $\pi_X(\pi_Y(R)) \equiv \pi_X(R)$ if $X \subseteq Y$

$\pi_{salary}(\pi_{branchNo,salary}(Staff)) = \pi_{salary}$

(3) $\sigma_{\varphi}(R_1 \times R_2) \equiv R_1 \bowtie_{\varphi} R_2$

$\sigma_{Staff.branchNo=Branch.branchNo}(Staff \times Branch) =$

$(Staff) \bowtie_{Staff.branchNo=Branch.branchNo} (Branch)$

Heuristic Rules


```
Staff(sno, fname, lname, salary, position, branchNo)
Branch(branchNo, name, street, suburb, city)
```


$(Staff) \bowtie_{Staff.branchNo=Branch.br}$

(5) $\sigma_{\varphi}(R_1 \bowtie R_2) = \sigma_{\varphi}(R_1) \bowtie R_2$, if R_1

$\sigma_{salary>60000}(Staff \bowtie Branch) = \sigma_{sal}$

(6) $\sigma_{\varphi_1 \wedge \varphi_2}(R_1 \bowtie R_2) \equiv \sigma_{\varphi_1}(R_1) \bowtie \sigma_{\varphi_2}$ 2 1

attributes in R_1 and φ_2 contains only attributes in R_2.

$\sigma_{salary>60000 \wedge city='Canberra'}(Staff \bowtie Branch) =$

$(\sigma_{salary>60000}(Staff)) \bowtie (\sigma_{city='Canberra'}(Branch))$

Heuristic Rules

Staff(sno, fname, lname, salary, position, branchNo)
Branch(branchNo, name, street, suburb, city)

$\pi_{branchNo,position,city}(Staff \bowtie Branch)$

$\pi_{branchNo,position}(Staff) \bowtie (\pi_{br}$

(8) If the join condition contains attributes no
$\pi_X(R_1 \bowtie R_2) \equiv \pi_X(\pi_{X_1}(R_1) \bowtie \pi_{X_2}($
both in R_1 and R_2, and ones in both R_i and X

$\pi_{position,city}(Staff \bowtie Branch) =$

$\pi_{position,city}(\pi_{branchNo,position}(Staff) \bowtie (\pi_{branchNo,city}(Branch)))$

- .....

Push-down Selection – Example

- Given the relation schemas:

  PERSON(id, first_name, last_name, year_born)

  DI

  M

- **Query: List the first and last names of the direct a movie that has won an 'Oscar' movie award**

  $\pi_{first\_name,last\_name}(\sigma_{award\_name='Oscar'}((\text{PERSO}$ MOVIE_AWARD))

- **Question**: Can we apply the following rule to optimise the query?

  $\sigma_{\varphi}(R_1 \bowtie R_2) \equiv \sigma_{\varphi}(R_1) \bowtie R_2$, if φ contains only attributes in R_1

Push-down Selection – Example

- Given the relation schemas:

  PERSON(id, first_name, last_name, year_born)

  DI

  M

- **Query: List the first and last names of the direct a movie that has won an 'Oscar' movie award**

  $\pi_{first\_name,last\_name}(\sigma_{award\_name='Oscar'}((\text{PERSO}$

  MOVIE_AWARD))

- We would have:

$$\pi_{first\_name,last\_name}((\text{PERSON} \bowtie \text{DIRECTOR}) \bowtie \sigma_{award\_name='Oscar'}(\text{MOVIE\_AWARD}))$$

Push-down Projection – Example

- Given the relation schemas:

  PERSON(id, first_name, last_name, year_born)

  DI

  M _ _ _ _ _ d)

- **Q**

  a movie that has won an 'Oscar' movie award

  $\pi_{first_name,last_name}($PERSON $\bowtie$ DIRECTOR _AWARD))

- **Question**: Can we apply the following rule to optim

$$\pi_X(R_1 \bowtie R_2) \equiv \pi_X(\pi_{X_1}(R_1) \bowtie \pi_{X_2}(R_2)),$$

where X_i contains attributes in both in R_1 and R_2, and ones in both R_i and X

Push-down Projection – Example

- Given the relation schemas:

  PERSON(id, first_name, last_name, year_born)

  DI

  M _ _ _ _ _ d)

- **Q**

  a movie that has won an 'Oscar' movie award

  $\pi_{first\_name,last\_name}$(PERSON ⋈ DIRECTOR _AWARD))

- we would have:

  $\pi_{first\_name,last\_name}(\pi_{first\_name,last\_name,title,production\_year}(\text{PERSON} \bowtie$

  $\text{DIRECTOR}) \bowtie \pi_{title,production\_year}(\sigma_{award\_name='Oscar'}(\text{MOVIE\_AWARD})))$

A Common Query Pattern (Be Careful)

- A common query pattern is **join-select-project** involving three steps:
  (1) **join** all the relevant relations,

- T

$$\pi_{A_1,\ldots,A_n}(\sigma_{\varphi}(R_1 \times \cdot$$

or as an equivalent SQL statement

```
SELECT DISTINCT A1, ..., An FROM R1, ..., Rk WHERE φ;
```

- Queries falling into this pattern can be **very inefficient**, which may yield huge intermediate result for the joined relations.

A Common Query Pattern (Be Careful)

push-down selection and **push-down projection**.

$$\pi_{A_1,\ldots,A_n}(\sigma_{\varphi}(R_1 \times \cdots \times R_k)),$$

Re-ordering Joins - Example

- Given the relation schemas:

  PERSON(id, first_name, last_name, year_born)

  Suppose that it has **10000 tuples**.

  DI

  Suppose that it has **100 tuples**.

  MOVIE_AWARD(title, production_year, awa

  Suppose that it has **1000 tuples**.

- **Example:** Consider the following two RA queries. Which one is better?
  - PERSON ⋈ MOVIE_AWARD ⋈ DIRECTOR
  - PERSON ⋈ DIRECTOR ⋈ MOVIE_AWARD

Cost-based Query Optimisation

- A q
  es

- It estimates and compares the costs of executing a query using different execution strategies and chooses one with .

- The query optimiser needs to **limit the nu** to be considered for improving efficiency.

Summary

- In general, there are **many ways of executing a query** in a database.
- T
  b
- B
  sh
- Nonetheless, SQL is not a suitable query languag
  optimised automatically.
- Instead, SQL queries are **transformed int
  queries** and optimised subsequently.
- A major advantage of relational algebra is to **make alternative forms of a query easy to explore**.