Overview

- Structured query language (SQL) is a user-friendly language for specifying relational algebra queries. It is supported by all the major database systems.

SQL provides:

- Data Manipulation Language (DML)
  - retrieve, insert and modify database contents
- Data Definition Language (DDL)
  - add and delete database objects
- Data Control Language (DCL)
  - configure security access

- In this lecture, we will learn how to rewrite algebra operators in SQL (DML).

Syntax of an SQL Statement


```
select distinct A1, A2, ..., An
from T1, ..., Tm
where P
```


where $T_1, \ldots, T_m$ are tables, $A_1, \ldots, A_n$ are attributes, and P is a predicate. The statement returns a table, and corresponds to the following relational algebra query:

$$\Pi_{A_1,\ldots,A_n}(\sigma_P(T_1 \times \cdots \times T_m))$$

Selection σ


```
select *
from T
where P
```

corresponds to

$$\sigma_P(T)$$

PROF

pid	name	dept	rank	sal
$p1$	Adam	CS	asst	6000
$p2$	Bob	EE	asso	8000
$p3$	Calvin	CS	full	10000
$p4$	Dorothy	EE	asst	5000
$p5$	Emily	EE	asso	8500
$p6$	Frank	CS	full	9000


```
select *
from PROF
where rank = 'asst'
```

$\sigma_{\text{rank} = \text{“asst”}}(\text{PROF})$

PROF

pid	name	dept	rank	sal
$p1$	Adam	CS	asst	6000
$p2$	Bob	EE	asso	8000
$p3$	Calvin	CS	full	10000
$p4$	Dorothy	EE	asst	5000
$p5$	Emily	EE	asso	8500
$p6$	Frank	CS	full	9000


```
select *
from PROF
where not(rank = 'asst' and dept = 'EE')
```

$\sigma_{\neg(\text{rank} = \text{“asst”} \wedge \text{dept}=\text{“EE”})}(\text{PROF})$

Selection Predicate


```
select *
from T
where P
```


In P, you can specify the standard comparisons and logic operators:

- $=, <>, <, <=, >, >=$
- Connect multiple comparisons with: AND, OR, NOT.

Projection Π

select distinct $A_1, ..., A_n$
from T

corresponds to

$$\Pi_{A_1,...,A_n}(T)$$

PROF

pid	name	dept	rank	sal
$p1$	Adam	CS	asst	6000
$p2$	Bob	EE	asso	8000
$p3$	Calvin	CS	full	10000
$p4$	Dorothy	EE	asst	5000
$p5$	Emily	EE	asso	8500
$p6$	Frank	CS	full	9000

select distinct dept, rank
from PROF

$\Pi_{\text{dept,rank}}(\text{PROF})$

Note

The keyword distinct removes all duplicate rows in the output. Omitting the keyword keeps all duplicates. See the next slide.

PROF

pid	name	dept	rank	sal
$p1$	Adam	CS	asst	6000
$p2$	Bob	EE	asso	8000
$p3$	Calvin	CS	full	10000
$p4$	Dorothy	EE	asst	6000
$p5$	Emily	EE	asso	8500
$p6$	Frank	CS	full	9000

"select dept, rank from PROF" returns:

dept	rank
CS	asst
EE	asso
CS	full
EE	asst
EE	asso
CS	full

This duplicate-retaining feature is useful for aggregate queries as we will discuss later in the course.

Cartesian Product $\times$

select *
from T_1, T_2

corresponds to $T_1 \times T_2$

select *
from $T_1, ..., T_m$

corresponds to $T_1 \times ... \times T_m$

PROF

pid	name	dept	rank	sal
$p1$	Adam	CS	asst	6000
$p2$	Bob	EE	asso	8000
$p3$	Calvin	CS	full	10000
$p4$	Dorothy	EE	asst	5000
$p5$	Emily	EE	asso	8500

TEACH

pid	cid	year
$p1$	$c1$	2011
$p2$	$c2$	2012
$p1$	$c2$	2012


```
select *
from PROF, TEACH
```


PROF $\times$ TEACH

Putting Multiple Operators Together

PROF

pid	name	dept	rank	sal
$p1$	Adam	CS	asst	6000
$p2$	Bob	EE	asso	8000
$p3$	Calvin	CS	full	10000
$p4$	Dorothy	EE	asst	5000
$p5$	Emily	EE	asso	8500

TEACH

pid	cid	year
$p1$	$c1$	2011
$p2$	$c2$	2012
$p1$	$c2$	2012


```
select distinct dept
from PROF, TEACH
where PROF.pid = TEACH.pid
```

$$\Pi_{\text{dept}}(\sigma_{\text{PROF.pid} = \text{TEACH.pid}}(\text{PROF} \times \text{TEACH}))$$

Rename ρ


```
select ...
from T as S
where ...
```

corresponds to

$$...\rho_S(T)...$$

PROF

pid	name	dept	rank	sal
$p1$	Adam	CS	asst	6000
$p2$	Bob	EE	asso	8000
$p3$	Calvin	CS	full	10000
$p4$	Dorothy	EE	asst	5000
$p5$	Emily	EE	asso	8500

TEACH

pid	cid	year
$p1$	$c1$	2011
$p2$	$c2$	2012
$p1$	$c2$	2012


```
select distinct dept
from PROF as A, TEACH as B
where A.pid = B.pid
```


$$\Pi_{\text{dept}}(\sigma_{\text{A.pid = B.pid}}(\rho_{\text{A}}(\text{PROF}) \times \rho_{\text{B}}(\text{TEACH})))$$

Set Difference –

([SQL statement 1])
minus
([SQL statement 2])

corresponds to

$$T_1 - T_2$$

where T_1 (T_2) is the table returned by SQL statement 1 (2).

Note

- T_1 and T_2 need to have the same schema.
- Duplicates in T_1 and T_2 will first be removed before performing the set difference.
- In some systems (e.g., SQL server from Microsoft), the set difference operator is named "except", instead of "minus".

PROF

pid	name	dept	rank	sal
$p1$	Adam	CS	asst	6000
$p2$	Bob	EE	asso	8000
$p3$	Calvin	CS	full	10000
$p4$	Dorothy	EE	asst	5000
$p5$	Emily	EE	asso	8500
$p6$	Frank	CS	full	9000

(select rank from PROF)
minus
(select rank from PROF where dept = 'CS')

$\Pi_{\text{rank}}(\text{PROF}) - \Pi_{\text{rank}}(\sigma_{\text{dept} = \text{“CS”}}(\text{PROF}))$

Set Union ∪

([SQL statement 1])
union
([SQL statement 2])

corresponds to

$$T_1 \cup T_2$$

where T_1 (T_2) is the table returned by SQL statement 1 (2).

Note

- T_1 and T_2 need to have the same schema.
- Duplicates in T_1 and T_2 will first be removed before performing the set union.

PROF

pid	name	dept	rank	sal
$p1$	Adam	CS	asst	6000
$p2$	Bob	EE	asso	8000
$p3$	Calvin	CS	full	10000
$p4$	Dorothy	EE	asst	5000
$p5$	Emily	EE	asso	8500
$p6$	Frank	CS	full	9000


```
(select * from PROF where sal <= 6000)
union
(select * from PROF where sal >= 9000)
```

$\sigma_{\text{sal} \leq 6000}(\text{PROF}) \cup \sigma_{\text{sal} \geq 9000}(\text{PROF})$

We have shown how to rewrite the 6 fundamental algebra operators in SQL. How about the extended operators $\leftarrow, \cap, \bowtie$ and $\div$? As we will see next, there is an explicit statement only for $\cap$. Nevertheless, as $\cap$ and $\bowtie$ can be implemented using the 6 fundamental operators, they can also be written in SQL using the statements introduced earlier. We will, however, ignore $\leftarrow$ from our discussion (this operator is the least useful one, anyway).

Set Intersection ∩

```
([SQL statement 1])
intersect
([SQL statement 2])
```


corresponds to

$$T_1 \cap T_2$$

where T_1 (T_2) is the table returned by SQL statement 1 (2).

Note

- T_1 and T_2 need to have the same schema.
- Duplicates in T_1 and T_2 will first be removed before performing the set union.

PROF

pid	name	dept	rank	sal
$p1$	Adam	CS	asst	6000
$p2$	Bob	EE	asso	8000
$p3$	Calvin	CS	full	10000
$p4$	Dorothy	EE	asst	5000
$p5$	Emily	EE	asso	8500
$p6$	Frank	CS	full	9000

(select * from PROF where sal $>=$ 6000)
intersect
(select * from PROF where dept = 'CS')

$\sigma_{\text{sal} \geq 6000}(\text{PROF}) \cap \sigma_{\text{dept} = \text{“CS”}}(\text{PROF})$

Natural Join

PROF

pid	name	dept	rank	sal
$p1$	Alan	CS	asst	6000
$p2$	Bob	EE	asso	8000
$p3$	Calvin	CS	full	10000
$p4$	Dorothy	EE	asst	5000
$p5$	Emily	EE	asso	8500

TEACH

pid	cid	year
$p1$	$c1$	2011
$p2$	$c2$	2012
$p1$	$c2$	2012


```
select distinct PROF.pid, name, dept, rank, sal, cid, year
from PROF, TEACH
where PROF.pid = TEACH.pid
```


$\Pi_{\text{PROF.pid, name, dept, rank, sal, cid, year}}(\sigma_{\text{PROF.pid=TEACH.pid}}(\text{PROF} \times \text{TEACH}))$

$=$

$\text{PROF} \bowtie \text{TEACH}$

Division

T_1

pid	cid
$p1$	$c1$
$p1$	$c2$
$p1$	$c3$
$p2$	$c2$
$p2$	$c3$
$p3$	$c1$
$p4$	$c1$
$p4$	$c2$
$p4$	$c3$

T_2

cid
$c1$
$c2$
$c3$

(select pid from T_1)
minus
select pid from (
(select * from (select pid from T_1), T_2)
minus
(select * from T_1))

Note

Notice how an SQL statement can be nested in a from clause.

$$\Pi_{S_1-S_2}(T_1) - \Pi_{S_1-S_2}\left(\Pi_{S_1-S_2}(T_1) \times T_2 - T_1\right) = T_1 \div T_2$$

```
SELECT * FROM politicians WHERE clue > 0;
```



```
SQL> select intelligence_level from developer;
select intelligence_level from developer
       *
ERROR at line 1:
ORA-00904: "INTELLIGENCE_LEVEL": invalid identifier
```


http://www.orafaq.com/wiki/Fun_stuff