Join and set-Operator

> Designer : ZHU Yueming in March 15th 2021

Experimental-Objective

- Understand cross join, left join and inner join.
- Learn Set Operator
- Exists and In

Part 1. Cross Join, Left Join, Inner Join

Before you start

To create following two tables, and insert data:

```
create table T1(
    A int,
    B varchar(10),
    C int
);

create table T2(
    A int,
    B varchar(10),
    C int
);

insert into T1 values(1,'hello',3);
insert into T1 values(2,'world',8);
insert into T1 values(2,'hi',3);

insert into T2 values(3,'database',4);
insert into T2 values(2,'hello',8);
insert into T2 values(4,'cs307',3);
```

1. Cross Join

Combine each rows in the left table with each rows in the right table. (Cartesian product)

```
select * from T1 cross join T2;
```

Two tables:

T1

a	b	c
1	hello	3
2	world	8
2	hi	3

T2

a	b	c
3	database	4
2	hello	8
4	cs307	3

Result:

	t1.a	t1.b	t1.c	t2.a	t2.b	t2.c
1	1	hello	3	3	database	4
2	1	hello	3	2	hello	8
3	1	hello	3	4	cs307	3
4	2	world	8	3	database	4
5	2	world	8	2	hello	8
6	2	world	8	4	cs307	3
7	2	hi	3	3	database	4
8	2	hi	3	2	hello	8
9	2	hi	3	4	cs307	3

2. Outer Join

- **Left outer join (left join)**: Keep all rows in the left table and combine matched rows in the right table.
- **Right outer join (right join)**: Keep all rows in the right table and combine matched rows in the left table. it can be always rewritten in left outer join.
- **Full outer join (full join)**: Keep all rows in both side.

Left Join example:

```
select * from T1 left join T2 on T1.A=T2.A;
```

Two tables:

T1

a	b	c
1	hello	3
2	world	8
2	hi	3

T2

a	b	c
3	database	4
2	hello	8
4	cs307	3

Result:

	t1.a	t1.b	t1.c	t2.a	t2.b	t2.c
1	1	hello	3	<null>	<null>	<null>
2	2	world	8	2	hello	8
3	2	hi	3	2	hello	8

Right Join example:

```
select * from T2 left join T1 on T1.A = T2.A;
or
select * from T1 right join T2 on T1.A = T2.A;
```

Two tables:

T1

a	b	c
1	hello	3
2	world	8
2	hi	3

T2

a	b	c
3	database	4
2	hello	8
4	cs307	3

Result:

	t2.a	t2.b	t2.c	t1.a	t1.b	t1.c
1	2	hello	8	2	world	8
2	2	hello	8	2	hi	3
3	3	database	4	<null>	<null>	<null>
4	4	cs307	3	<null>	<null>	<null>

Full Join example:

```
select * from T1 full outer join T2 on T1.A = T2.A;
```

Two tables:

T1

a	b	c
1	hello	3
2	world	8
2	hi	3

T2

a	b	c
3	database	4
2	hello	8
4	cs307	3

Result:

	t1.a	t1.b	t1.c	t2.a	t2.b	t2.c
1	1	hello	3	<null>	<null>	<null>
2	2	world	8	2	hello	8
3	2	hi	3	2	hello	8
4	<null>	<null>	<null>	3	database	4
5	<null>	<null>	<null>	4	cs307	3

3. Inner Join (join)

Combine matched rows under specific conditions from two or more tables.

```
select * from T1 join T2 on T1.A = T2.A;
```

Two tables:

T1

a	b	c
1	hello	3
2	world	8
2	hi	3

T2

a	b	c
3	database	4
2	hello	8
4	cs307	3

Result:

	t1.a	t1.b	t1.c	t2.a	t2.b	t2.c
1	2	world	8	2	hello	8
2	2	hi	3	2	hello	8

The rows in result sets for the join clause are the same for those two situations: `T1 join T2` and `T2 join T1`

```
select * from T2 join T1 on T2.A = T1.A;
```

Result:

	t2.a	t2.b	t2.c	t1.a	t1.b	t1.c
1	2	hello	8	2	world	8
2	2	hello	8	2	hi	3

4. Comparison of JOIN and LEFT JOIN: AND WHERE

The AND clause in the `left join` serves as the conditions of combination, so that the result set would remain all rows in the left table with the matched rows under the join conditions of the right table. While WHERE clause in the `left join` serves as the selection of the final result sets, which can filter rows dissatisfied of WHERE clause.

```
select * from T1 left join T2 on T1.A = T2.A and T1.C=3;
select * from T1 left join T2 on T1.A = T2.A where T1.C=3;
```

Two tables:

T1

a	b	c
1	hello	3
2	world	8
2	hi	3

T2

a	b	c
3	database	4
2	hello	8
4	cs307	3

Result of upper one:

	t1.a	t1.b	t1.c	t2.a	t2.b	t2.c
1	1	hello	3	<null>	<null>	<null>
2	2	world	8	<null>	<null>	<null>
3	2	hi	3	2	hello	8

Result of lower one:

	t1.a	t1.b	t1.c	t2.a	t2.b	t2.c
1	2	hi	3	2	hello	8
2	1	hello	3	<null>	<null>	<null>

The `AND` clause in `inner join` serves as the filter conditions, which can only remain the combine rows that satisfied of two (or more) tables, so that `AND` does similar effect as the `WHERE`. The result set of following two queries are the same.

```
select * from T1  join T2 on T1.A = T2.A and T1.C=3;
select * from T1 join T2 on T1.A = T2.A where T1.C=3;
```

Two tables:

T1

a	b	c
1	hello	3
2	world	8
2	hi	3

T2

a	b	c
3	database	4
2	hello	8
4	cs307	3

Result:

	t1.a	t1.b	t1.c	t2.a	t2.b	t2.c
1	2	hi	3	2	hello	8

Part 2. Set Operator

Before you start, you can create two tables as following queries:

Create two tables named `course` and `course_selected`

```
create table course
(
    cno    varchar(5) primary key,
    name   varchar not null,
    credit integer not null
);

create table course_selected
(
    id    serial primary key,
    sno   varchar(8) not null,
    cno   varchar(5) not null
        constraint cno_fk
            references course (cno),
    grade integer,
    date  date       not null
);
```

Insert data into those two tables:

```
insert into course(cno, name, credit)
VALUES ('CS307', 'database', 3);
insert into course(cno, name, credit)
VALUES ('CS102', 'Java', 3);
insert into course(cno, name, credit)
VALUES ('CS205', 'C++', 3);
insert into course(cno, name, credit)
VALUES ('CS203', 'data structure', 3);

insert into course_selected(sno, cno, grade, date)
values ('12000001', 'CS102', 59, '2020/5/1');
insert into course_selected(sno, cno, grade, date)
values ('12000001', 'CS102', 60, '2021/3/1');
insert into course_selected(sno, cno, grade, date)
values ('12000001', 'CS203', 75, '2021/3/1');
insert into course_selected(sno, cno, grade, date)
values ('12000001', 'CS307', 100, '2021/3/1');
insert into course_selected(sno, cno, grade, date)
values ('12000002', 'CS102', 60, '2021/5/1');
insert into course_selected(sno, cno, grade, date)
values ('12000002', 'CS203', 87, '2021/3/1');
```

```
insert into course_selected(sno, cno, grade, date)
values ('12000002', 'CS205', 96, '2021/3/1');
```

In this case, we can find that:

Student '12000001' selected following courses:

	cno	name	credit
1	CS307	database	3
2	CS102	Java	3
3	CS102	Java	3
4	CS203	data structure	3

Student '12000002' selected following courses:

	cno	name	credit
1	CS102	Java	3
2	CS205	C++	3
3	CS203	data structure	3

1. Union

Combine two result set into one, and remove the duplicate rows.

Compare to UNION ALL, UNION will see a duplicate.

Example:

```
select c.*
   from course_selected cs
           join course c on cs.cno = c.cno
   where cs.sno = '12000001'
union
select c.*
 from course_selected cs
          join course c on cs.cno = c.cno
 where cs.sno = '12000002';
```

Result:

	cno	name	credit
1	CS102	Java	3
2	CS203	data structure	3
3	CS205	C++	3
4	CS307	database	3

2. Union ALL

Combine two result set into one directly. When you know that, the two result sets cannot have any duplicates, then you don't need to go through the step of duplicate removal, which is costly, in this case you'd better using union all instead of union.

Example:

```
select c.*
   from course_selected cs
           join course c on cs.cno = c.cno
   where cs.sno = '12000001'
union all
select c.*
 from course_selected cs
          join course c on cs.cno = c.cno
 where cs.sno = '12000002';
```

Result:

	cno	name	credit
1	CS307	database	3
2	CS102	Java	3
3	CS102	Java	3
4	CS203	data structure	3
5	CS102	Java	3
6	CS205	C++	3
7	CS203	data structure	3

3. Except

Return the rows from the first result set, minus those that can also be found in the second result set.

Example:

```
select c.*
   from course_selected cs
           join course c on cs.cno = c.cno
   where cs.sno = '12000001'
except
select c.*
 from course_selected cs
          join course c on cs.cno = c.cno
 where cs.sno = '12000002';
```

Result:

	cno	name	credit
1	CS307	database	3

4. Intersect

Returns= the common rows in two result sets.

Example:

```
select c.*
   from course_selected cs
           join course c on cs.cno = c.cno
   where cs.sno = '12000001'
intersect
select c.*
 from course_selected cs
          join course c on cs.cno = c.cno
 where cs.sno = '12000002';
```

Result:

	cno	name	credit
1	CS203	data structure	3
2	CS102	Java	3

> Intersect -> inner join
>
> Except -> left join

Part 3. Exists and In

Example SQL 1. Exists:

```
select *
from course c
where exists(
        select null
        from course_selected cs
        where c.cno = cs.cno and  sno = '12000002'
    )
```

If the result set of `course_selected` is larger than `course`, using **Exists** is faster than **In**. The sample executing process is that for each row in outer-query, it scans the sub-query according to the matching condition, then only return the row is true or false.

The matching condition of sub-query and outer-query is `c.cno = cs.cno and  sno = '12000002'`

Logic pseudocode:

```
for each row in course:
    //usually create index in the column course_selected.cno
    if can find one (c.cno = cs.cno and  sno = '12000002')
            resultSet.add(row)
    else
            pass
```

Example SQL 2. In:

```
select *
from course
where cno in (
    select cno
    from course_selected
    where sno = '12000002');
```

If the result set of `course_selected` is smaller than following result set, , using **In** would be faster. The executing process will generate a temporary result set as follows, then for each row in course, scan each row in the temporary result set and find whether those are matched.

```
select cno from course_selected where sno = '12000002';
```

Logic pseudocode:

```
for each row1 in course:
    for each row2 in (select cno from course_selected where sno = '12000002')
as sub:
          if(course.cno == sub.cno )
                resultSet.add(row1)
```

Comparation：

> The part is referenced.

Exists	In
SQL Engine will stop the process as soon as it finds a single positive condition in EXISTS condition	SQL Engine compares all values in IN condition
The answer of EXISTS can be TRUE or FALSE	The answer of IN can be TRUE or FALSE or NULL
EXISTS cannot compare values between parent query and sub-query	IN compares values between parent query and sub-query
It can be used to determine if any values are returned or not	IN is used as multiple OR operator
EXISTS is faster than IN if sub-query result is large	IN is faster than EXISTS if sub-query result is less
NULL can be compared using EXISTS condition	NULL cannot be compared using IN condition
Direct values cannot be compared using EXISTS condition. It should have sub-query with SELECT clause	IN condition can have mul;ple direct values instead of sub-query