CS143: Database Systems

SQL Notes—Part 1

Select statement

- Query statements in SQL start with the keyword

`select`

and return a result in table form

```
select   Attribute … Attribute
from     Table … Table
[where   Condition]
```

- The three parts are usually called
  - target list
  - from clause
  - where clause

MotherChild

mother	child
Lisa	Mary
Lisa	Greg
Anne	Kim
Anne	Phil
Mary	Andy
Mary	Rob

FatherChild

father	child
Steve	Frank
Greg	Kim
Greg	Phil
Frank	Andy
Frank	Rob

Person

name	age	income
Andy	27	21
Rob	25	15
Mary	55	42
Anne	50	35
Phil	26	30
Greg	50	40
Frank	60	20
Kim	30	41
Mike	85	35
Lisa	75	87

```
select person.name, person.income
from   person
where  person.age < 30
```

```
select name, income
from   person
where  age < 30
```

Two Kinds of Projection

employee

empNo	surname	branch	salary
7309	Black	York	55
5998	Black	Glasgow	64
9553	Brown	London	44
5698	Brown	London	64

```
select
       surname, branch
from employee
```

```
select distinct
       surname, branch
from employee
```

surname	branch
Black	York
Black	Glasgow
Brown	London
Brown	London

surname	ranch
Black	York
Black	Glasgow
Brown	London

Naming and aliases

```
select name, income
from   person
where  age < 30
```

is an abbreviation for:

```
select person.name, person.income
from   person
where  person.age < 30
```

and also for:

```
select p.name as name,  p.income as income
from   person p           /* Same as "person as p" */
where  p.age < 30
```

Expressions in the Target List

```
select income/4 as quarterlyIncome
from   person
where  name = 'Greg'
```

Complex Conditions in the "where" Clause

```
select *                    /* the star means all the columns */
from   person
where  income > 25
       and (age < 30 or age > 60)
```

SQL and Relational Algebra

Given the relations: R1(A1,A2) and R2(A3,A4)

the semantics of the query

```
select R1.A1, R2.A4
from   R1, R2
where  R1.A2 = R2.A3
```

can be described in terms of

- cartesian product (`from`)
- selection (`where`)
- projection (`select`)

Note: This does not mean that the system really calculates the cartesian product!

MotherChild

mother	child
Lisa	Mary
Lisa	Greg
Anne	Kim
Anne	Phil
Mary	Andy
Mary	Rob

FatherChild

father	child
Steve	Frank
Greg	Kim
Greg	Phil
Frank	Andy
Frank	Rob

Person

name	age	income
Andy	27	21
Rob	25	15
Mary	55	42
Anne	50	35
Phil	26	30
Greg	50	40
Frank	60	20
Kim	30	41
Mike	85	35
Lisa	75	87

"The fathers of persons who earn more than 20K"

$$\pi_{father}(\text{fatherChild} \bowtie_{child=name} \sigma_{income>20}(\text{person}))$$

“The fathers of persons who earn more than 20K”

```
select distinct fc.father
from   person p, fatherChild fc
where  fc.child = p.name
       and p.income > 20
```

MotherChild

mother	child
Lisa	Mary
Lisa	Greg
Anne	Kim
Anne	Phil
Mary	Andy
Mary	Rob

FatherChild

father	child
Steve	Frank
Greg	Kim
Greg	Phil
Frank	Andy
Frank	Rob

Person

name	age	income
Andy	27	21
Rob	25	15
Mary	55	42
Anne	50	35
Phil	26	30
Greg	50	40
Frank	60	20
Kim	30	41
Mike	85	35
Lisa	75	87

$$\pi_{father}(\text{fatherChild} \bowtie_{child=name} \sigma_{income>20}(\text{person}))$$

For each child show the father and mother

R
A

$$\text{fatherChild} \bowtie \text{motherChild}$$

S
Q
L

```
select fc.child, fc.father, mc.mother
from   motherChild mc, fatherChild fc
where  fc.child = mc.child
```

MotherChild

mother	child
Lisa	Mary
Lisa	Greg
Anne	Kim
Anne	Phil
Mary	Andy
Mary	Rob

FatherChild

father	child
Steve	Frank
Greg	Kim
Greg	Phil
Frank	Andy
Frank	Rob

Person

name	age	income
Andy	27	21
Rob	25	15
Mary	55	42
Anne	50	35
Phil	26	30
Greg	50	40
Frank	60	20
Kim	30	41
Mike	85	35
Lisa	75	87

"Persons that earn more than their father, showing name, income, and income of the father"

```
select f.name, f.income, c.income
from   person f, fatherChild fc, person c
where  f.name = fc.father and
       fc.child = c.name and
       c.income > f.income
```

MotherChild

mother	child
Lisa	Mary
Lisa	Greg
Anne	Kim
Anne	Phil
Mary	Andy
Mary	Rob

FatherChild

father	child
Steve	Frank
Greg	Kim
Greg	Phil
Frank	Andy
Frank	Rob

Person

name	age	income
Andy	27	21
Rob	25	15
Mary	55	42
Anne	50	35
Phil	26	30
Greg	50	40
Frank	60	20
Kim	30	41
Mike	85	35
Lisa	75	87

Aliases: **you should use `as’ and you must not forget commas**!

```
select   P1.name, P2.name
from MotherChild as  M1, MotherChild as M2,
          Person P1,  Person P2
where   M1.Child =P1.Name and
            M2.Child =P2.Name   and
             M1.mother=M2.mother  and
             P1.name < P2.name
```

MotherChild

mother	child
Lisa	Mary
Lisa	Greg
Anne	Kim
Anne	Phil
Mary	Andy
Mary	Rob

FatherChild

father	child
Steve	Frank
Greg	Kim
Greg	Phil
Frank	Andy
Frank	Rob

Person

name	age	income
Andy	27	21
Rob	25	15
Mary	55	42
Anne	50	35
Phil	26	30
Greg	50	40
Frank	60	20
Kim	30	41
Mike	85	35
Lisa	75	87

Ordering the Result: `order by`

```
select name, income
from   person
where  age < 30
```

name	income
Andy	21
Rob	15
Mary	42

```
select name, income
from   person
where  age < 30
order by name
```

name	income
Andy	21
Mary	42
Rob	15

Aggregate Operators

Among the expressions in the target list, we can also have expressions that calculate values based on multisets of tuples:

- count, minimum, maximum, average, sum

Basic Syntax (simplified):

Function ([`distinct`] *ExpressionOnAttributes*)

Aggregate Operator `count`: Example

Example: How many children has Frank?

```
select count(*) as NumFranksChildren
from   fatherChild
where  father = 'Frank'
```

Semantics: The aggregate operator (`count`), which counts the tuples, is applied to the result of the query:

```
select *
from   fatherChild
where  father = 'Frank'
```

fatherChild

father	child
Steve	Frank
Greg	Kim
Greg	Phil
Frank	Andy
Frank	Rob

Other Aggregate Operators

`sum`, `avg`, `max`, `min`

- argument can be an attribute or an expression (but not "`*`")
- `sum` and `avg`: numerical and temporal arguments
- `max` and `min`: arguments on which an ordering is defined

Example: Average income of Frank's children

```
select  avg(p.income)
from    person p join fatherChild fc on
        p.name = fc.child
where   fc.father = 'Frank'
```

Aggregate Operators and the Target List

An incorrect query (whose name should be returned?):

```
select name, max(income)
from   person
```

The target list has to be homogeneous, for example:

```
select min(age), avg(income)
from   person
```

“For each group of adult persons who have the same age, return the maximum income for that group and show the age”

Write the query in SQL!

person	name	age	income

```
select age, max(income)
from   person
where  age > 17
group by age
```

Aggregate Operators and Grouping

- Aggregation functions can be applied to partitions of the tuples of a relations
- To specify the partition of tuples, on uses the`group by` clause:

`group by` *attributeList*

Aggregate Operators and Grouping

The number of children of every father.

```
select father, count(*) as NumChildren
from   fatherChild
group by father
```

fatherChild

father	child
Steve	Frank
Greg	Kim
Greg	Phil
Frank	Andy
Frank	Rob

father	NumChildren
Steve	1
Greg	2
Frank	2

“For each group of adult persons who have the same age, return the maximum income for that group and show the age”

```
select age, max(income)
from   person
where  age > 17
group by age
```

person	name	age	income
	Andy	27	21
	Rob	25	NULL
	Mary	55	21
	Anne	50	35

Grouping and Target List

In a query that has a `group by` clause, only such attributes can appear in the target list (except for aggregation functions) the appear in the `group by` clause.

Example: Incorrect: income of persons, grouped according to age

```
select age, income
from   person
group by age
```

There could exist several values for the same group.

Correct: average income of persons, grouped by age.

```
select age, avg(income)
from   person
group by age
```

The syntactic restriction on the attributes in the select clause holds also for queries that would be semantically correct (i.e., for which there is only a single value of the attribute for every group).

Conditions on Groups

It is also possible to filter the groups using selection conditions. Clearly, the selection of groups differs from the selection of the tuples in the `where` clause: the tuples form the groups.

To filter the groups, the “having clause” is used.

The having clause must appear after the “`group by`”

Example: Fathers whose children have an average income greater 25.

```
select fc.father, avg(c.income)
from   person c join fatherChild fc
       on c.name = fc.child
group by fc.father
having avg(c.income) > 25
```

Having or Where?

Find fathers whose children under 20 have an average income >25.

Syntax of SQL `select` (Summary)

SQLSelect ::=

```
select      ListOfAttributesOrExpressions
from        ListOfTables
[ where     ConditionsOnTuples ]
[ group by  ListOfGroupingAttributes ]
[ having    ConditionsOnAggregates ]
[ order by  ListOfOrderingAttributes ]
```