SQL Part 3 (Advanced)

Outline

1. Subquery – nested, inline, correlated
2. Views
3. Joins - self join, outer join
4. Set Operators
5. Oracle Functions

Subquery (NESTED)

- For each unit, find the students who obtained maximum mark in the unit

```
Select studid, unitcode, mark
from enrolment
where (unitcode, mark) IN (select unitcode, max(mark)
                           from enrolment group by unitcode);
```

- the subquery is independent of the outer query and is executed only once.

Subquery (CORRELATED)

- For each unit, find the students who obtained maximum mark in the unit

```
Select studid, unitcode, mark
from enrolment e1
where mark  =  (select max(mark)
                from enrolment e2
                where e2.unitcode = e1.unitcode)
```

- the subquery is related to the outer query and is considered to be evaluated once **for each row** of the outer query

Subquery (INLINE)

- For each unit, find the students who obtained maximum mark in the unit

```
select studid, e.unitcode, mark
from (select unitcode, max(mark) as max_mark
 from enrolment group by unitcode)
temp  join enrolment e on temp.unitcode =
 e.unitcode and e.mark = temp.max_mark;
```

Subquery (INLINE)

- For each grade, compute the percentage of the students who got that grade

```
SELECT
   grade,
   count(grade) as grade_count,
   (SELECT count(*) from enrolment) as total_rows,
   100*count(grade)/(SELECT count(*) FROM enrolment)
   as percentage
FROM enrolment
where grade is NOT NULL
GROUP BY grade;
```

Outline

1. Subquery – nested, inline, correlated
2. Views
3. Joins - self join, outer join
4. Set Operators
5. Oracle Functions

Views

- A virtual table derived from one or more base tables.

CREATE OR REPLACE VIEW [view_name] AS

SELECT … ;

Create or replace view max_view as

Select unitcode, max(mark) as max_mark from enrolment group by unitcode;

Select * from max_view;

Using Views

- For each unit, find the students who obtained maximum mark in the unit

```
create or replace view max_view
as select unitcode, max(mark) as max_mark
 from enrolment group by unitcode;

select e.studid, e.unitcode, e.mark
from max_view v join enrolment e
on e.unitcode = v.unitcode
and e.mark = v.max_mark
order by e.unitcode;
```

Outline

1. Subquery – nested, inline, correlated
2. Views
3. Joins - self join, outer join
4. Set Operators
5. Oracle Functions

Self Join

Employee

ID	Name	Manager
1	Alice	2
2	Bob	3
3	Chris	3

Print the name of the manager for each employee.

Self Join

Select * from employee e1, employee e2;

ID	Name	Manager
1	Alice	2
2	Bob	3
3	Chris	3

Employee e1

ID	Name	Manager
1	Alice	2
1	Alice	2
1	Alice	2
2	Bob	3
2	Bob	3
2	Bob	3
3	Chris	3
3	Chris	3
3	Chris	3

Employee e2

ID	Name	Manager
1	Alice	2
2	Bob	3
3	Chris	3
1	Alice	2
2	Bob	3
3	Chris	3
1	Alice	2
2	Bob	3
3	Chris	3

Self Join

Select * from employee e1 join employee e2 on e1.manager = e2.id;

Employee e1

ID	Name	Manager
1	Alice	2
1	Alice	2
1	Alice	2
2	Bob	3
2	Bob	3
2	Bob	3
3	Chris	3
3	Chris	3
3	Chris	3

Employee e2

ID	Name	Manager
1	Alice	2
2	Bob	3
3	Chris	3
1	Alice	2
2	Bob	3
3	Chris	3
1	Alice	2
2	Bob	3
3	Chris	3

Self Join

Select * from employee e1 join employee e2

on e1.manager = e2.id;

Employee e1

ID	Name	Manager
1	Alice	2
2	Bob	3
3	Chris	3

Employee e2

ID	Name	Manager
2	Bob	3
3	Chris	3
3	Chris	3

Self Join

Select e1.name, e2.name as manager

from employee e1 join employee e2 on e1.manager = e2.id

Employee e1

ID	Name	Manager
1	Alice	2
2	Bob	3
3	Chris	3

Employee e2

ID	Name	Manager
2	Bob	3
3	Chris	3
3	Chris	3

name	manager
Alice	Bob
Bob	Chris
Chris	Chris

OUTER JOIN

Students

ID	Name
1	Alice
2	Bob
3	Chris

Marks

ID	Subj	Marks
1	1004	95
2	1045	55
1	1045	90
4	1004	100

Natural Join gives no information for Chris and the student with ID 4

ID	Name	ID	Subj	Marks
1	Alice	1	1004	95
1	Alice	1	1045	90
2	Bob	2	1045	55

Select * from students s join marks m
on s.id = m.id;

FULL OUTER JOIN

Students

ID	Name
1	Alice
2	Bob
3	Chris

Marks

ID	Subj	Marks
1	1004	95
2	1045	55
1	1045	90
4	1004	100

Get (incomplete) information of both Chris and student with ID 4

ID	Name	ID	Subj	Marks
1	Alice	1	1004	95
1	Alice	1	1045	90
2	Bob	2	1045	55
3	Chris	Null	Null	Null
Null	Null	4	1004	100

```
select * from
students s full outer join marks m on s.id = m.id;
```

LEFT OUTER JOIN

Students

ID	Name
1	Alice
2	Bob
3	Chris

Marks

ID	Subj	Marks
1	1004	95
2	1045	55
1	1045	90
4	1004	100

Get (incomplete) information of only Chris

ID	Name	ID	Subj	Marks
1	Alice	1	1004	95
1	Alice	1	1045	90
2	Bob	2	1045	55
3	Chris	Null	Null	Null

```
select * from
students s left outer join marks m
on s.id = m.id;
```

RIGHT OUTER JOIN

Students

ID	Name
1	Alice
2	Bob
3	Chris

Marks

ID	Subj	Marks
1	1004	95
2	1045	55
1	1045	90
4	1004	100

Get (incomplete) information of the student with ID 4

ID	Name	ID	Subj	Marks
1	Alice	1	1004	95
1	Alice	1	1045	90
2	Bob	2	1045	55
Null	Null	4	1004	100

```
select * from
students s right outer join marks m
on s.id = m.id;
```

Outline

1. Subquery – nested, inline, correlated
2. Views
3. Joins - self join, outer join
4. Set Operators
5. Oracle Functions

Relational Set Operators

- Using the set operators you can combine two or more sets to create new sets (relations)
- Union All
  - All rows selected by **either** query, including all duplicates
- Union
  - All rows selected by **either** query, removing duplicates (e.g,, DISTINCT on Union All)
- Intersect
  - All distinct rows selected by **both** queries
- Minus
  - All distinct rows selected by the first query but not by the second
- All set operators have equal precedence. If a SQL statement contains multiple set operators, Oracle evaluates them from the left to right if no parentheses explicitly specify another order.
- The two sets must be UNION COMPATIBLE (i.e., same number of attributes and similar data types)

MINUS

- List the name of staff who are not a chief examiner in an offering.

```
select staffIname, stafffname
from staff
where staffid IN
(select staffid from staff
Minus
select chiefexam from offering);
```

UNION

Create a list of units with its average mark. Give the label “Below distinction” to all units with the average less than 70 and “Distinction and Above” for those units with average greater or equal to 70.

UNITCODE	Average	Average status
FIT2077	64.5	Below Distinction
FIT1040	70	Distinction and Above
FIT1004	71.7	Distinction and Above
FIT5132	74	Distinction and Above
FIT5136	75.5	Distinction and Above
FIT5131	77.5	Distinction and Above

1. Select units with average marks less than 70 and set status
2. Select units with average marks greater or equal to 70 and set status
3. Take a union of 1 and 2

```
SELECT unitcode, AVG(mark) AS "Average”, 'Below Distinction' AS "Average status"
FROM enrolment
GROUP BY unitcode
HAVING AVG(mark) < 70

UNION

SELECT unitcode, AVG(mark) AS “Average”, 'Distinction and Above' AS "Average status"
FROM enrolment
GROUP BY unitcode
HAVING AVG(mark) >= 70;
```

INTERSECTION

Find students who have the same surname as a staff member's surname.

1. Find the common surnames in staff and student table.
2. Find students with the surname present in 1

```
SELECT studid, studfname, studlname
FROM student
WHERE studlname IN
  ( SELECT  studlname
    FROM student
    INTERSECT
    SELECT stafflname
    FROM staff);
```

Outline

1. Subquery – nested, inline, correlated
2. Views
3. Joins - self join, outer join
4. Set Operators
5. Oracle Functions

Function Type	Applicable to	Example
Arithmetic	Numerical data	SELECT ucode, round(avg(mark)) FROM enrolment GROUP BY ucode;
Text	Alpha numeric data	SELECT studsurname FROM enrolment WHERE upper(studsurname) LIKE ‘B%’;
Date	Date/Time-related data	
General	Any data type	NVL function
Conversion	Data Type conversion	SELECT to_char(empmsal,'$999.99') FROM employee;
Group	Sets of Values	avg(), count(), etc

Given the following oracle syntax for round function;

ROUND(n [,integer]) where n is a number and integer determines the decimal point;

what would be the right SELECT clause for rounding the average mark of all marks in the enrolment (not including the NULL values) to the nearest 2 decimal point?

a. avg(round(mark,2))
b. round(avg(mark,2))
c. round(avg(mark),2)
d. avg(mark(round(2)))

HANDLING DATE IN ORACLE

- Oracle uses two different types: date and timestamp
    - Output is controlled by formatting
        - select to_char(sysdate,'dd-Mon-yyyy') from dual;
            - 01-Aug-2012
        - select to_char(sysdate,'dd-Mon-yyyy hh:mi:ss PM') from dual;
            - 01-Aug-2012 10:56:50 AM

Current Date

- Current date can be queried from the DUAL table using the SYSDATE attribute.
  - SELECT sysdate FROM dual;
- Oracle internal attributes include:
  - sysdate: current date/time
  - systimestamp: current date/time as a timestamp
  - user: current logged in user

Operations on Date Data Type

- Substraction

```
select sysdate - studdob
from uni.student;
```

  – Return days as a floating point, i.e will have fraction.

- Addition

```
SELECT SYSDATE + 1 FROM DUAL;
```

2. The following SQL statement aims to display student who are born after 01-April-1991. However it returns rows which include those show below. Why this statement causes this error?

STUDID	STUDENTNAME	STUDENTDOB
11111123	Tay Lee	01-Mar-1988
11111126	John Tse	03-Dec-1988
11111117	Jake Ryan	01-Jan-1990
11111127	Jake Brown	01-Jan-1990

```
select studid,
  studfname || ' ' || studlname as StudentName,
  to_char(studdob,'dd-Mon-yyyy') as StudentDOB
from uni.student
where to_char(studdob,'dd-Mon-yyyy')  >  '01-Apr-1991‘
order by studdob;
```

a. The comparison operator should be “<“.
b. The to_char function causes the comparison in the condition to be a character-based comparison rather than date comparison.
c. The date ‘01-Apr-1991’ is not in a correct date format.
d. More than one.

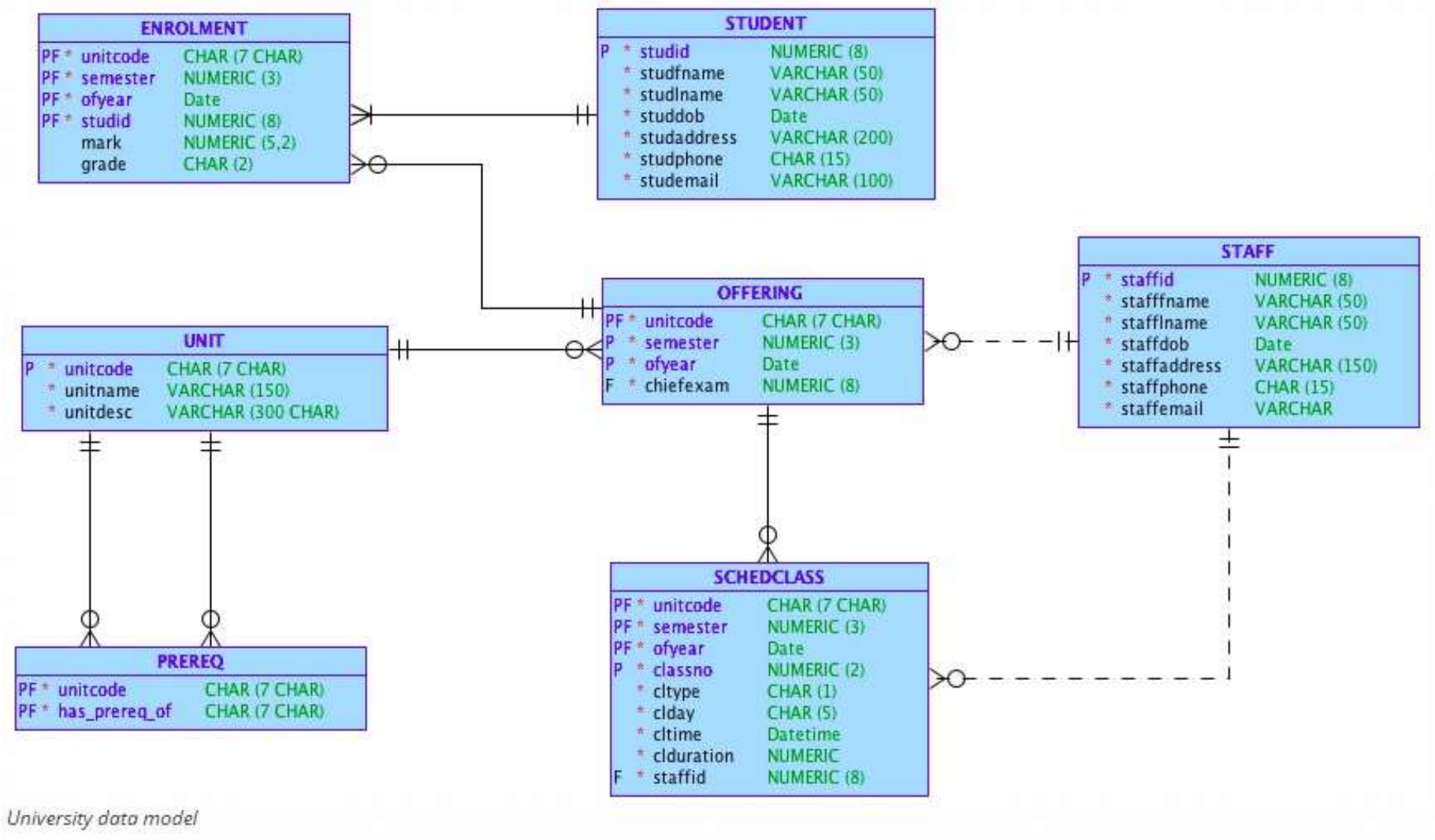
Integer sorting: 1,3,11,15,16,20

String based sorting: 1,11,15,16,20,3

- DATE data type should be formatted with TO_CHAR when selecting for display
- String literal representing date must be formatted with TO_DATE when comparing or inserting/updating
- Example:

```
select studid,
   studfname || ' ' || studlname as StudentName,
   to_char(studdob,'dd-Mon-yyyy') as StudentDOB
from uni.student
where studdob >  to_date('01-Apr-1991','dd-Mon-yyyy')
order by studdob;
```

PRACTICE



University data model