SQL Advanced

FIT9132

Outline

- **CASE**
- Subquery – nested, inline, correlated
- Views
- Joins - self join, outer join
- Set Operators
- Oracle Functions

SQL CASE statement

The CASE statement used in the select list enables the query to evaluate an attribute and output a particular value based on that evaluation

```
SELECT
    ...
    case cltype
        when 'L' then 'Lecture'
        when 'T' then 'Tutorial'
    end as Classtype,
    ...
    case
    when clduration < 2 then clduration || 'hr Short class'
    when clduration = 2 then clduration || 'hr Standard class'
    else clduration || 'hr Long class'
    end as classduration,
    ...
FROM uni.schedclass
ORDER BY ... ;
```

MONASH
University

Outline

- Case
- **Subquery – nested, inline, correlated**
- Views
- Joins - self join, outer join
- Set Operators
- Oracle Functions

MONASH
University

Query

For each unit, find the students(studid) who obtained the maximum mark in the unit

Subquery (NESTED)

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

> **select studid, unitcode, mark**
>
> **from uni.enrolment**
>
> **where (unitcode, mark) IN (select unitcode, max(mark)**
>
> > > **from uni.enrolment**
> > >
> > > **group by unitcode)**
>
> **order by unitcode, studid;**

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

Subquery (CORRELATED)

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

  select studid, unitcode, mark

  from uni.enrolment e1

  where mark = (select max(mark)

  from uni.enrolment e2

  where e1.unitcode = e2.unitcode)

  order by unitcode, studid;

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

Subquery (INLINE) – Derived table

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

select studid, e.unitcode, mark

from

  (select unitcode, max(mark) as max_mark

   from uni. enrolment

   group by unitcode) max_table

   join uni.enrolment e on e.unitcode = max_table.unitcode and
                    e.mark = max_table.max_mark

order by unitcode, studid;

Subquery (INLINE)

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

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

Outline

- Case
- Subquery – nested, inline, correlated
- **Views**
- Joins - self join, outer join
- Set Operators
- Oracle Functions

Views

- A virtual table derived from one or more base tables.
- Sometimes used as "Access Control" to the database

  CREATE OR REPLACE VIEW [view_name] AS

  SELECT … ;

  create *or replace* view max_view as

  select unitcode, max(mark) as max_mark

  from uni.enrolment

  group by unitcode;

  select * from max_view
  order by unitcode;

- What objects do I own?

  select * from user_objects;

Using Views

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

create or replace view max_view

  as select unitcode, max(mark) as max_mark

  from uni.enrolment group by unitcode;

select e.studid, e.unitcode, e.mark

from max_view v join uni.enrolment e on e.unitcode = v.unitcode

    and e.mark = v.max_mark

order by e.unitcode;

Please note VIEWS <u>MUST NOT</u> be used for Assignment 3

Outline

- Case
- Subquery – nested, inline, correlated
- Views
- **Joins - self join, outer join**
- Set Operators
- Oracle Functions

Self Join

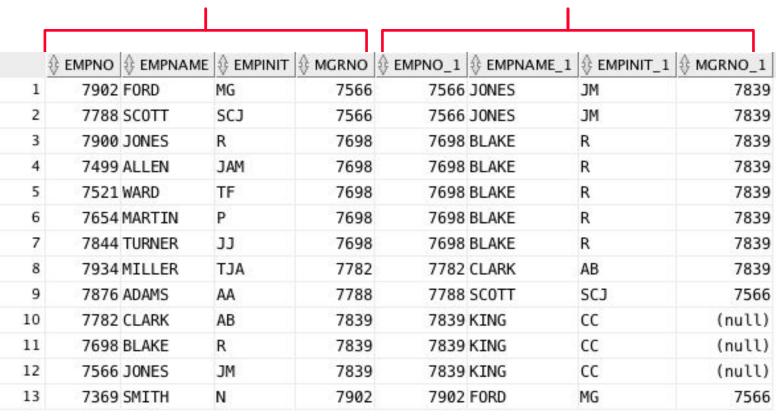- Show the name of the manager for each employee.

SELECT
    empno,
    empname,
    empinit,
    mgrno
FROM
    emp.employee;

	EMPNO	EMPNAME	EMPINIT	MGRNO
1	7839	KING	CC	(null)
2	7566	JONES	JM	7839
3	7902	FORD	MG	7566
4	7369	SMITH	N	7902
5	7698	BLAKE	R	7839
6	7499	ALLEN	JAM	7698
7	7521	WARD	TF	7698
8	7654	MARTIN	P	7698
9	7782	CLARK	AB	7839
10	7788	SCOTT	SCJ	7566
11	7844	TURNER	JJ	7698
12	7876	ADAMS	AA	7788
13	7900	JONES	R	7698
14	7934	MILLER	TJA	7782

SELECT *
FROM emp.employee e1 JOIN emp.employee e2
ON e1.mgrno = e2.empno;

e1   e2

	EMPNO	EMPNAME	EMPINIT	MGRNO	EMPNO_1	EMPNAME_1	EMPINIT_1	MGRNO_1
1	7902	FORD	MG	7566	7566	JONES	JM	7839
2	7788	SCOTT	SCJ	7566	7566	JONES	JM	7839
3	7900	JONES	R	7698	7698	BLAKE	R	7839
4	7499	ALLEN	JAM	7698	7698	BLAKE	R	7839
5	7521	WARD	TF	7698	7698	BLAKE	R	7839
6	7654	MARTIN	P	7698	7698	BLAKE	R	7839
7	7844	TURNER	JJ	7698	7698	BLAKE	R	7839
8	7934	MILLER	TJA	7782	7782	CLARK	AB	7839
9	7876	ADAMS	AA	7788	7788	SCOTT	SCJ	7566
10	7782	CLARK	AB	7839	7839	KING	CC	(null)
11	7698	BLAKE	R	7839	7839	KING	CC	(null)
12	7566	JONES	JM	7839	7839	KING	CC	(null)
13	7369	SMITH	N	7902	7902	FORD	MG	7566

Joined rows

1,12

2,12

3,11

Note some columns have been hidden   **Why now only 13 rows?**

SELECT e1.empno, e1.empname, e1.empinit, e1.mgrno,
e2.empname AS MANAGER
FROM emp.employee e1 JOIN emp.employee e2
ON e1.mgrno = e2.empno
ORDER BY e1.empname;

	EMPNO	EMPNAME	EMPINIT	MGRNO	MANAGER
1	7876	ADAMS	AA	7788	SCOTT
2	7499	ALLEN	JAM	7698	BLAKE
3	7698	BLAKE	R	7839	KING
4	7782	CLARK	AB	7839	KING
5	7902	FORD	MG	7566	JONES
6	7900	JONES	R	7698	BLAKE
7	7566	JONES	JM	7839	KING
8	7654	MARTIN	P	7698	BLAKE
9	7934	MILLER	TJA	7782	CLARK
10	7788	SCOTT	SCJ	7566	JONES
11	7369	SMITH	N	7902	FORD
12	7844	TURNER	JJ	7698	BLAKE
13	7521	WARD	TF	7698	BLAKE

NATURAL JOIN

Student

ID	NAME
1	Alice
2	Bob
3	Chris

Mark

ID	SUBJECT	MARK
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_1	SUBJECT	MARK
1	Alice	1	1004	95
2	Bob	2	1045	55
1	Alice	1	1045	90

Select * from student s join mark m on s.id = m.id;
Note that this is an INNER JOIN

MONASH University

FULL OUTER JOIN

Student

ID	NAME
1	Alice
2	Bob
3	Chris

Mark

ID	SUBJECT	MARK
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_1	SUBJECT	MARK
1	Alice	1	1004	95
2	Bob	2	1045	55
1	Alice	1	1045	90
(null)	(null)	4	1004	100
3	Chris	(null)	(null)	(null)

select * from
student s full outer join mark m on s.id = m.id;

LEFT OUTER JOIN

Student

ID	NAME
1	Alice
2	Bob
3	Chris

Mark

ID	SUBJECT	MARK
1	1004	95
2	1045	55
1	1045	90
4	1004	100

Get (incomplete) information of only Chris

ID	NAME	ID_1	SUBJECT	MARK
1	Alice	1	1004	95
2	Bob	2	1045	55
1	Alice	1	1045	90
3	Chris	(null)	(null)	(null)

**select * from
student s left outer join mark m
on s.id = m.id;**

RIGHT OUTER JOIN

Student

ID	NAME
1	Alice
2	Bob
3	Chris

Mark

ID	SUBJECT	MARK
1	1004	95
2	1045	55
1	1045	90
4	1004	100

Get (incomplete) information of the student with ID 4

ID	NAME	ID_1	SUBJECT	MARK
1	Alice	1	1045	90
1	Alice	1	1004	95
2	Bob	2	1045	55
(null)	(null)	4	1004	100

select * from
student s right outer join mark m
on s.id = m.id;

ID	Name	Manager
1	Alice	2
2	Bob	3
3	Chris	

Employee

Q1. What is the output from the following SQL

**select e1.NAME as name, e2.name as manager
from employee e1 right outer join employee e2
    on e1.manager = e2.id;**

(A)

```
NAME          MANAGER
---------     --------

Alice         Bob
Bob           Chris
              Alice
```

(B)

```
NAME          MANAGER
----------    ----------

Alice         Bob
Bob           Chris
Chris
```

Employee

ID	Name	Salary
1	Alice	100,000
2	Bob	150,000
3	Chris	200,000

Project

Project	Cost	EmpID
Alpha	4000	1
Beta	3000	2
Gamma	5000	2

Q2. Which of the following shows, for *each* employee, the total amount of projects they are assigned to? (E.g., Alice is assigned to Alpha with total cost 4000, Bob is assigned to Beta and Gamma with total cost 8000)

A. select e.name, sum(cost) as total from employee e left outer join project p on e.id = p.empid group by e.name;

B. select e.name, sum(cost) as total from employee e right outer join project p on e.id = p.empid group by e.name;

C. select e.name, NVL(sum(cost),0) as total from employee e left outer join project p on e.id = p.empid group by e.name;

D. None of the above

Outline

- Case
- Subquery – nested, inline, correlated
- Views
- Joins - self join, outer join
- **Set Operators**
- Oracle Functions

Q3. Two or more queries that are connected using a set operator have to be union compatible. When would two relations be union compatible? It is when the two relations have:

A.   the same degree and similar domain for the attributes

B.   the same degree and attributes' name

C.   the same degree and cardinality.

D.   the same cardinality.

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 staffid, stafflname, stafffname
from uni.staff
where staffid IN
    (**select staffid from uni.staff**
    minus
    select chiefexam from uni.offering);

STAFFID
1
2
3
4
5
6
7

CHIEFEXAM
1
1
1
1
3
4
5
5
7
7
7
7
7
7

STAFFID	STAFFLNAME	STAFFFNAME
2	Burbage	Charity
6	Umbridge	Dolores

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

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

```sql
SELECT unitcode, AVG(mark) AS Average, 'Below Distinction' AS Average_Status
FROM
 uni.enrolment
GROUP BY
 unitcode
HAVING
 AVG(mark) < 70
UNION
SELECT unitcode, AVG(mark) AS Average, 'Distinction and Above' AS
Average_Status
FROM
 uni.enrolment
GROUP BY
 unitcode
HAVING
 AVG(mark) >= 70
ORDER BY
 Average DESC ;
```

INTERSECTION

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

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

```
SELECT studid, studfname, studlname
FROM
  uni.student
WHERE
  studlname IN
  ( SELECT DISTINCT studlname
    FROM
      uni.student
    INTERSECT
    SELECT DISTINCT stafflname
    FROM
      uni.staff)
ORDER BY studid;
```

Outline

- Case
- Subquery – nested, inline, correlated
- Views
- Joins - self join, outer join
- Set Operators
- **Oracle Functions**

MONASH University

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,'$0999.99') FROM employee;
Group	Sets of Values	avg(), count(), etc

See document on Moodle

Q4. **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.   SELECT avg(round(mark,2))

B.   SELECT round(avg(mark,2))

C.   SELECT round(avg(mark),2)

D.   SELECT avg(mark(round(2)))