

# 上周回顾

重要程度: ★★★★  
难易程度: ★★★★★

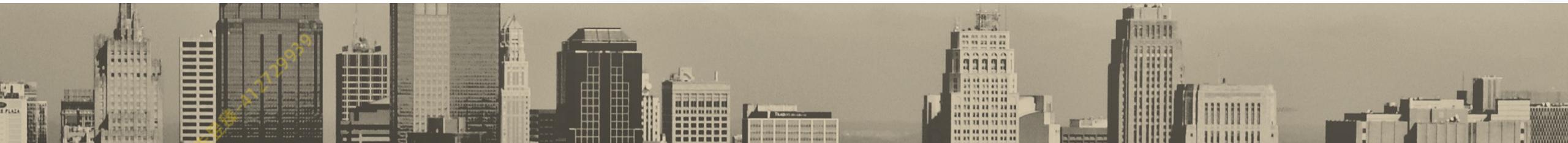
## 1、Transaction Management

- Atomicity
  - 要么全发生，要么全不发生
- Consistency
  - 要从一个完整状态转到另一个完整状态
- Isolation
  - 多个transaction之间的状态互不干扰
- Durability
  - Transaction结束后，结果应一段时间停留，直到下一次transaction
- Shared lock
  - 所有transaction都只能读，不能写
- Exclusive lock
  - 某一个transaction上此锁后只能它写和读，其他transaction不能进行任何读写操作

HD@朱昱臻-412729939

# 本章节知识点

知识点1 SQL QUERY



# 知识点讲解

# SQL – DML (insert data)

重要程度:   
难易程度: 

- ❖ 要和create table里attribute的顺序一一对应

```
INSERT INTO table [(column [, column...])]  
VALUES (value [, value...]);
```

```
INSERT INTO unit VALUES ('FIT9132', 'Databases');  
INSERT INTO student VALUES (112233, 'Wild', 'Wilbur',  
                            '01-Jan-1995')
```

*Role of: to\_date and to\_char*

# SQL – DML (Update)

重要程度: ★★★★★  
难易程度: ★★★★★

UPDATE table

SET column1 = xxx, column1 = yyy WHERE condition;

```
UPDATE enrolment  
SET mark = 80,  
    grade ='HD'  
WHERE sno = 112233  
and .....
```

```
UPDATE enrolment  
SET mark = 85  
WHERE unit_code = (SELECT unit_code FROM unit WHERE  
                    unit_name='Introduction to databases')  
      AND mark = 80;
```

HD@朱昱臻-4127000

# SQL – DML (Delete)

重要程度: ★★★★★  
难易程度: ★★★★★

DELETE FROM table WHERE condition;

```
DELETE FROM enrolment
WHERE sno='112233'
    AND
        unit_code= (SELECT unit_code FROM unit
                    WHERE unit_name='Introduction to Database' )
    AND
        semester='1'
    AND
        year='2012';
```

HD@朱昱臻-412729939

# SQL – DML (COMMIT & ROLLBACK)

重要程度:   
难易程度: 

- ❖ COMMIT: 保存
- ❖ ROLLBACK: 撤回

```
INSERT INTO enrolment VALUES (112233,  
                               'FIT9132',1,2018,45,'N');  
INSERT INTO enrolment VALUES (112233,  
                               'FIT9001',1,2018,80,'HD');  
COMMIT;
```

**COMMIT** makes the changes to the database permanent.

**ROLLBACK** will undo the changes.

# SQL – DML (sequence)

重要程度:   
难易程度: 

❖ 为什么PK最好选择numeric data?

❖ auto increment for PK

- Steps

1. Create sequence

2. Access sequence(currval & nextval)

– Create sequence

```
CREATE SEQUENCE sno_seq  
INCREMENT BY 1;
```

– Access the sequence using two built-in variables (pseudocolumns):

- NEXTVAL and CURRVAL

- ```
- INSERT INTO student  
VALUES(sno_seq.nextval, 'Bond', 'James', '01-Jan-1994');
```

- ```
- INSERT INTO enrolment  
VALUES(sno_seq.currval, 'FIT9132', ...);
```

# SQL - Query

重要程度: ★★★★★  
难易程度: ★★★★★

## Search Condition

### Comparison

<, >, <=, >=, =, !=

Example: display employee\_id which the employee salary is higher than 80,000

```
SELECT emp_id  
FROM employee  
WHERE emp_salary > 80000;
```

### Range:

#### **BETWEEN xxx AND xxx**

Example: display employee id which the employee salary is between 10000 and 80000

```
SELECT emp_id  
FROM employee  
WHERE emp_salary BETWEEN 10000 AND 80000;
```

HD@朱昱臻-4173939

# SQL - Query

重要程度: ★★★★★  
难易程度: ★★★★★

## Search Condition (continue)

### Set Membership: IN

To test whether the value of expression equals one of a set of values  
city IN ('Melbourne','Sydney') 查找city这一列带有'Melbourne' 或 'Sydney'的行

### Pattern Match: LIKE

To test whether a string matches a specified pattern  
%: represents any sequence of zero or more characters \_: represents any single character

#### example:

```
SELECT emp_id  
FROM employees  
WHERE emp_lname LIKE 'C%'
```

找Employee last name 以c开头的

#### other example:

%m: 以m结尾的

%abc%: 在任何位置带有abc的

\_ C% : C在第三位的

# SQL – Query (NULL)

重要程度:   
难易程度: 

**NULL:**

IS NULL / IS NOT NULL: to test whether a column has a NULL value

**Example:**

```
SELECT *  
FROM student  
WHERE grade IS NULL;
```

HD@朱昱臻-412729939

# SQL – Query (NVL, AS)

重要程度: ★★★★★  
难易程度: ★★★★★

- ❖ NVL(colname, value replace NULL) - in SELECT statement

NVL(enrol\_mark, 0) 把enrol\_mark 为NULL的替换成0 NVL(enrol\_grade,'WH')

- ❖ Rename: AS

```
SELECT stud_id, enrol_mark/10 AS new_mark FROM enrolment;
```

```
SELECT stud_id AS "student id" FROM ENROLMENT;
```

	STU_NBR	NVL(ENROL_MARK,0)	NVL(ENROL_GRADE,'WH')
1	11111111	78	D
2	11111111	0	WH
3	11111111	0	WH
4	11111112	35	N
5	11111112	0	WH
6	11111113	65	C
7	11111113	0	WH
8	11111114	0	WH

HD@朱昱臻-412729939

# SQL – Query (ORDER BY, NULLS LAST/FIRST)

重要程度:   
难易程度: 

## ❖ ORDER BY

Ascending & descending: 默认asc  
可以有多个column参与sorting

```
SELECT stud_id  
FROM enrolment  
ORDER BY enrol_mark DESC;
```

## ❖ NULLS LAST & NULLS FIRST

NULLS LAST: 如果排序遇到null行, null行放在最后

NULLS FIRST: 如果排序遇到null行, null行放在最前

```
SELECT stud_id  
FROM enrolment  
ORDER BY enrol_mark DESC NULLS LAST;
```

HD@吴昊12729939

# SQL – Query (DISTINCT)

重要程度:   
难易程度: 

## ❖ DISTINCT

如果出现重复的，只选择一行

```
SELECT DISTINCT stud_id  
FROM enrolment  
WHERE enrol_mark IS NULL;
```

HD@朱昱臻-412729939

# SQL – Query (Join)

重要程度: ★★★★★  
难易程度: ★★★★★

SELECT \*

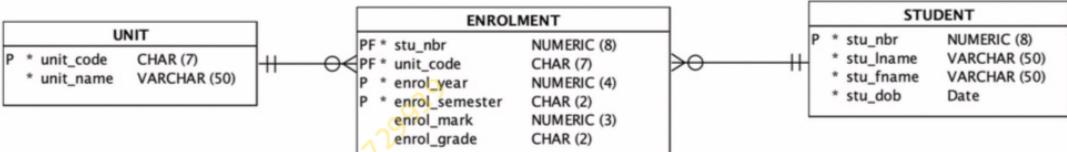
FROM table 1 JOIN table 2 ON join condition

ORDER BY attribute;

STUDENT		QUALIFICATION		
sno	name	sno	degree	year
1	alex	1	bachelor	1990
2	maria	1	master	2000
3	bob	2	PhD	2001

<b>SELECT *</b>																
<b>FROM student JOIN qualification</b>																
<b>ON student.sno = qualification.sno</b>																
<b>ORDER BY student.sno</b>																
<table border="1"> <thead> <tr> <th>sno</th> <th>name</th> <th>degree</th> <th>year</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>alex</td> <td>bachelor</td> <td>1990</td> </tr> <tr> <td>1</td> <td>alex</td> <td>master</td> <td>2000</td> </tr> <tr> <td>2</td> <td>maria</td> <td>PhD</td> <td>2001</td> </tr> </tbody> </table>	sno	name	degree	year	1	alex	bachelor	1990	1	alex	master	2000	2	maria	PhD	2001
sno	name	degree	year													
1	alex	bachelor	1990													
1	alex	master	2000													
2	maria	PhD	2001													



```

SELECT s.stu_nbr, s.stu_lname, u.unit_name
FROM ((unit u JOIN enrolment e ON u.unit_code=e.unit_code)
      JOIN student s ON e.stu_nbr=s.stu_nbr)
ORDER BY s.stu_nbr, u.unit_name;
    
```

## SQL JOIN

- For database students are **required to use ANSI JOINS**
  - placing the join in the where clause is **not acceptable** and will be *marked as incorrect for all assessment purposes*
    - such a join is sometimes known as "implicit join notation" - effectively a cross product and then restricted by the where clause
- **ANSI JOINS**
  - **ON**
    - the general form which always works, hence the syntax we tend to use
    - FROM student JOIN qualification ON student.sno = qualification.sno
  - **USING**
    - requires matching attribute names for the PK and FK
    - FROM student JOIN qualification USING (sno)
  - **NATURAL**
    - requires matching attribute names for the PK and FK
    - FROM student NATURAL JOIN qualification

# SQL – Query (CASE...WHEN...THEN...ELSE...END)

重要程度:   
难易程度: 

```
SELECT CASE
    WHEN STUD_FNAME IS NULL THEN STUD_LNAME
    ELSE STUD_LNAME || ' ' || STUD_FNAME
END AS "STUDENT NAME";
```

Artist Code	Artist Name	Artist State
20	Adrianna Dugget	Victoria
10	Dorette	Victoria
17	Duddy	Victoria
4	Rosalinda Zavattiero	Victoria

HD@朱昱臻-412729939

# SQL – Query (Aggregate Function)

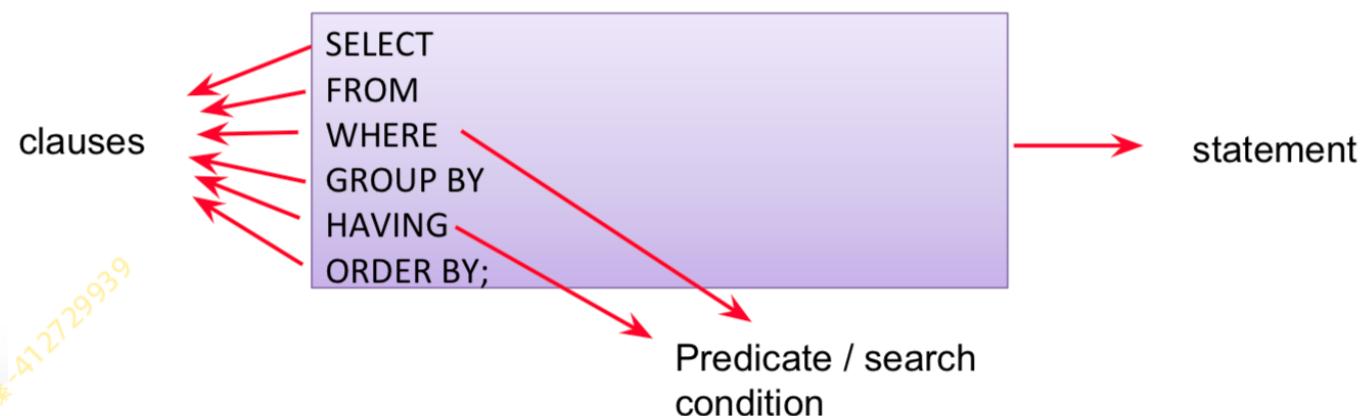
重要程度: ★★★★★  
难易程度: ★★★★★

```
SELECT max(mark)  
FROM enrolment;
```

```
SELECT avg(mark)  
FROM enrolment;
```

```
SELECT min(mark)  
FROM enrolment;
```

```
SELECT count(stu_nbr)  
FROM enrolment  
WHERE mark >= 50;
```



HD@朱昱臻-412729939

# SQL – Query (Aggregate Function)

重要程度: ★★★★★  
难易程度: ★★★★★

```
SELECT avg(mark)  
FROM enrolment;
```

```
SELECT unit_code, avg(mark)  
FROM enrolment  
GROUP BY unit_code  
ORDER BY unit_code;
```

Unit_code	Mark	Studid	Year
FIT2094	80	111	2016
FIT2094	20	111	2015
FIT2004	100	111	2016
FIT2004	40	222	2015
FIT2004	40	333	2015

```
SELECT unit_code, count(*)  
FROM enrolment  
GROUP BY unit_code  
HAVING count(*) > 2;
```

# SQL – Query (Aggregate Function)

重要程度: ★★★★★  
难易程度: ★★★★★

## WHERE

- applied to ALL rows

## HAVING

- applied to groups

## Order:

SELECT

FROM

WHERE

GROUP BY

HAVING

Unit_code	Mark	Studid	Year
FIT2094	80	111	2016
FIT2094	20	111	2015
FIT2004	100	111	2016
FIT2004	40	222	2015
FIT2004	40	333	2015

```
SELECT unit_code, avg(mark), count(*)  
FROM enrolmentA  
WHERE year = 2015  
GROUP BY unit_code  
HAVING avg(mark) > 30  
ORDER BY avg(mark) DESC;
```

HD@朱昱臻-412729939

# SQL – Query (Subquery)

重要程度: ★★★★★  
难易程度: ★★★★★

当需要的条件不在select里面的时候，需要subquery

avg(mark) 不在最后select的output里，但需要它来对data进行比较

"Find all students whose mark is higher than the average mark of all enrolled students"

```
SELECT *
FROM enrolment
WHERE mark > (SELECT avg (mark)
                FROM enrolment );
```

```
UPDATE enrolment
SET mark = 85
WHERE unit_code = (SELECT unit_code FROM unit WHERE
                    unit_name='Introduction to databases')
                    AND mark = 80;
```

# SQL – Query (Subquery)

重要程度: ★★★★★  
难易程度: ★★★★★

三种类型:

## 1. Nested

Subquery 独立于外面的query，并且只会执行一次

## 2. Correlated

每经过一行subquery，就会执行一次，但是里面的query和外部的query有联系，无法独立运行

## 3. Inline

要通过计算得到的，output经常作为一个‘table’跟在from后面

```
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;
```

check whether result  
execute once.

11111111	FIT1004
11111112	FIT1040
11111112	FIT1004
11111113	FIT1040
11111113	FIT1004

UNITCODE	MA
FIT1040	
FIT1004	
FIT5132	
FIT5136	
FIT2077	
.....	

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

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

UNITCODE	MAX_MARK
FIT1040	80
FIT1004	90
FIT5132	78
FIT5136	80
FIT2077	74
FIT5131	88

# SQL – Query (View)

重要程度: ★★★★★  
难易程度: ★★★★★

虚拟table，一个或多个表依照某个条件组合而成的结果集, 对视图只能进行select操作, 没有实际的物理记录

```
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;
```

# SQL – Query (Relational Set Operator)

重要程度:   
难易程度: 

## UnionAll

– All rows selected by either query, including all duplicates • Union (重复也合并)

## Union

– All rows selected by either query, removing duplicates (e.g., DISTINCT on Union All) (去重合并)

## Intersect

– All distinct rows selected by both queries (找交集)

## Minus

– All distinct rows selected by the first query but not by the second (前表有后表没有的)

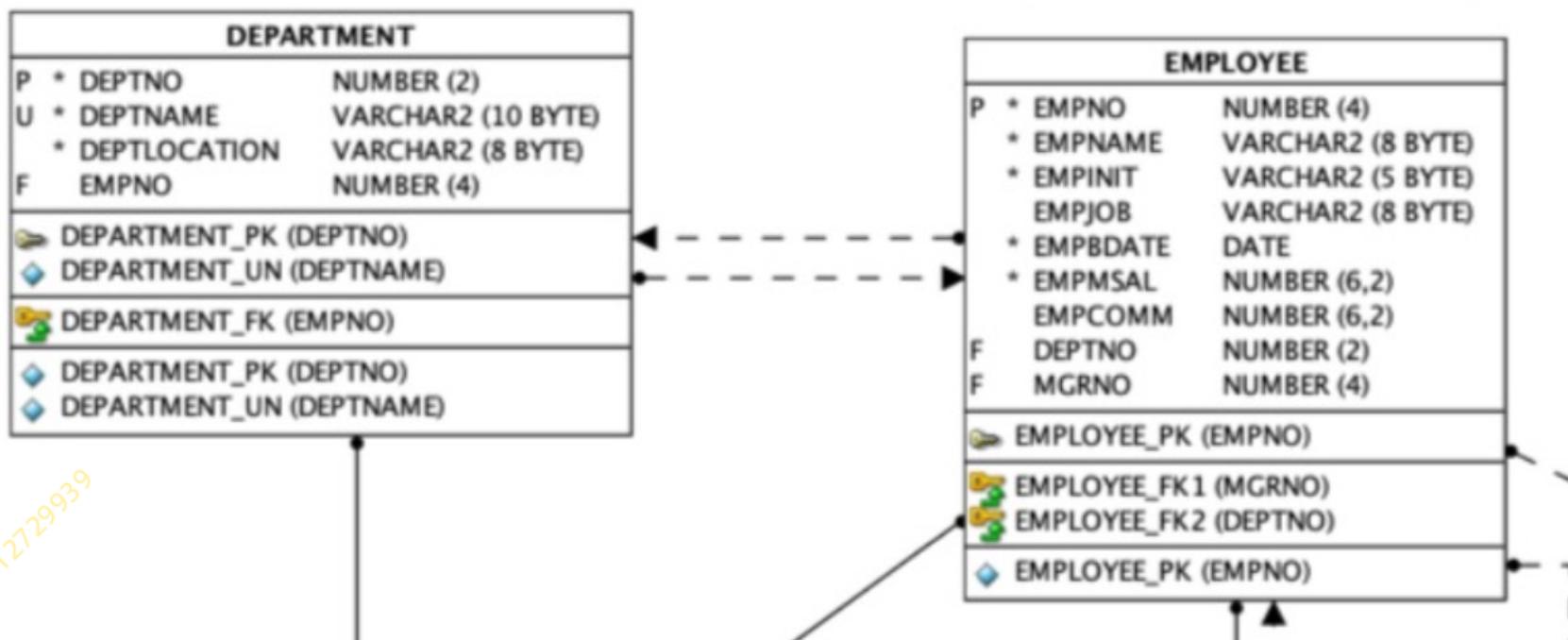
HD@朱昱臻-41272955

## 例题练习

重要程度: ★★★★★  
难易程度: ★★★★★

- v. The employee named KING who has a job as the only company DIRECTOR has been assigned to manage the new EXAM department. Record this in the database.

[4 marks]



## 例题练习

重要程度: ★★★★★  
难易程度: ★★★★★

```
UPDATE department
SET
    empno = (
        SELECT
            empno
        FROM
            employee
        WHERE
            empname = 'KING'
            AND empjob = 'DIRECTOR'
    )
WHERE
    deptname = 'EXAM';
COMMIT;
```

HD@朱昱臻-412729939

# 重难点总结

# 重难点总结

## SQL QUERY

- where 用于搜索条件， having 用于搜索 group by 之后的条件
- Query 顺序为：  
Select attribute1,attribute2,...  
from tablename  
where condition (optional)  
group by (optional)  
having (optional)  
order by xxx asc/desc (optional)
- 如果所需要的条件不在目前提取的表里时，需要 subquery 来实现

- 凡是出现了 aggregate function，后面都要带 group by
- Aggregate function:  
Max(), min(), avg(), sum(), count() ...
- Join 有 left join, right join, full outer join 以及 join，要根据不同场景选择合适的 join

HD@朱昱臻-412