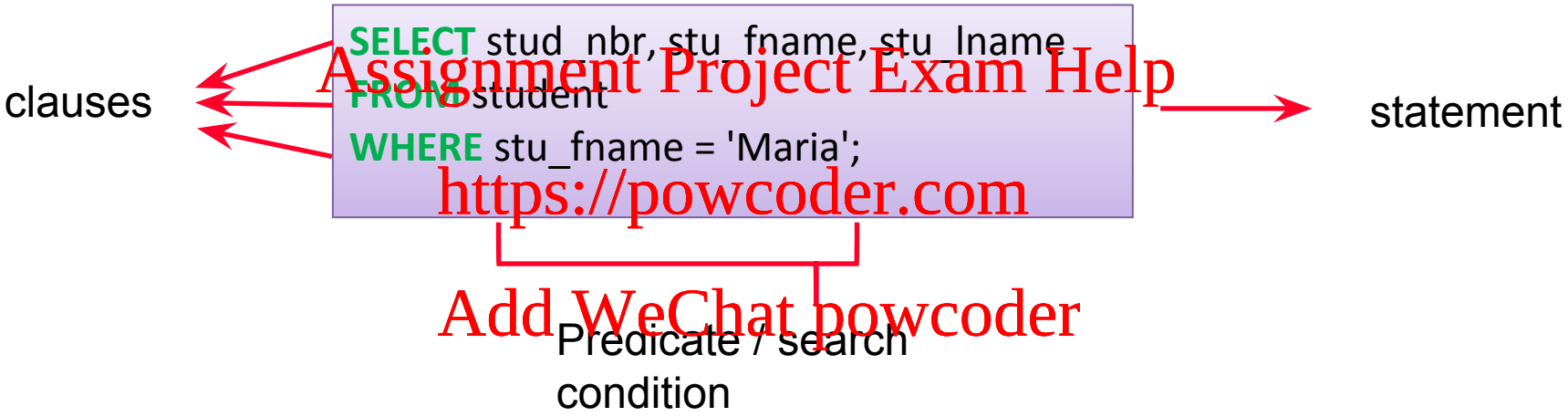
MONASH University

MONASH
INFORMATION
TECHNOLOGY

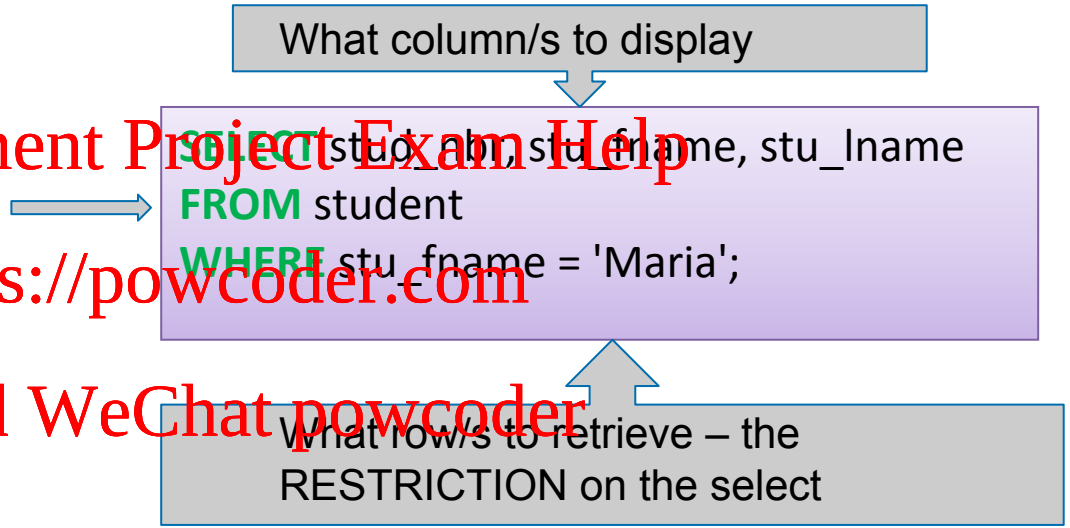
Structured Query Language (SQL) – Part 1

Anatomy of an SQL SELECT Statement



SQL SELECT Statement - Usage

What table(s) the data come from?



SQL Predicates or Search Conditions

- The search conditions are applied on each row, and the row is returned if the search conditions are evaluated to be TRUE for that row.
- **Comparison**
  - Compare the value of one expression to the value of another expression.
  - Operators: =, !=, <>, <, >, <=, >=
  - Example: salary > 5000
- **Range**
  - Test whether the value of an expression falls within a specified range of values.
  - Operator: BETWEEN
  - Example: salary BETWEEN 1000 AND 3000 (both are inclusive)

SQL Predicates or Search Conditions

- **Set Membership**
  - To test whether the value of expression equals one of a set of values.
  - Operator: IN
  - Example : city IN ('Melbourne', 'Sydney')
- **Pattern Match**
  - To test whether a string (text) matches a specified pattern.
  - Operator: LIKE
  - Patterns:
    - % character represents any sequence of zero or more character.
    - _ character represents any single character.
  - Example:
    - WHERE city LIKE 'M%'
    - WHERE unit_code LIKE 'FIT20__'

SQL Predicates or Search Conditions

- **NULL**
  - To test whether a column has a NULL (unknown) value.
  - Example: WHERE grade IS NULL.
- Use in subquery (to be discussed in the future)
  - ANY, ALL
  - EXISTS

What row will be retrieved?

- Predicate evaluation is done using three-valued logic.
  - **TRUE**, **FALSE** and **UNKNOWN**
- DBMS will evaluate the predicate against each row.
- Row that is evaluated to be TRUE will be retrieved.
- NULL is considered to be UNKNOWN.

Combining Predicates

- Logical operators
  - AND, OR, NOT
- Rules:
  - An expression is evaluated LEFT to RIGHT
  - Sub-expression in brackets are evaluated first
  - NOTs are evaluated before AND and OR
  - ANDs are evaluated before OR
  - Use of BRACKETS better alternative

Truth Table

- **AND** is evaluated to be TRUE if and only if **both** conditions are TRUE
- **OR** is evaluated to be TRUE if and only if at least one of the conditions is TRUE

AND

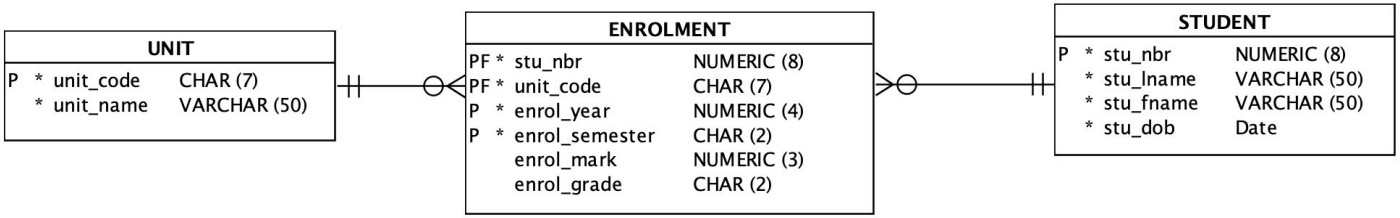
A \ B	T	U	F
T	T	U	F
U	U	U	F
F	F	F	F

OR

A \ B	T	U	F
T	T	T	T
U	T	U	U
F	T	U	F

T = TRUE
F = FALSE
U = Unknown

Unknown = NULL in relational database




```
SELECT * FROM UNIT;
```

Script Output | Query Result | SQL | All Rows Fetched: 3 in 0.015 seconds

	UNIT_CODE	UNIT_NAME
1	FIT1001	Computer Systems
2	FIT1002	Computer Programming
3	FIT1004	Database


```
SELECT * FROM STUDENT;
```

Script Output | Query Result | SQL | All Rows Fetched: 4 in 0.022 seconds

	STU_NBR	STU_LNAME	STU_FNAME	STU_DOB
1	11111111	Bloggs	Fred	01/JAN/90
2	11111112	Nice	Nick	10/OCT/94
3	11111113	Wheat	Wendy	05/MAY/90
4	11111114	Sheen	Cindy	25/DEC/96


```
SELECT * FROM ENROLMENT;
```

Script Output | Query Result | SQL | All Rows Fetched: 8 in 0.016 seconds

	STU_NBR	UNIT_CODE	ENROL_YEAR	ENROL_SEMESTER	ENROL_MARK	ENROL_GRADE
1	11111111	FIT1001	2012	1	78	D
2	11111111	FIT1002	2013	1	(null)	(null)
3	11111111	FIT1004	2013	1	(null)	(null)
4	11111112	FIT1001	2012	1	35	N
5	11111112	FIT1001	2013	1	(null)	(null)
6	11111113	FIT1001	2012	2	65	C
7	11111113	FIT1004	2013	1	(null)	(null)
8	11111114	FIT1004	2013	1	(null)	(null)

Arithmetic Operations

- Can be performed in SQL.
- For example:

```
SELECT stu_nbr, enrol_mark/10
FROM enrolment;
```


	STU_NBR	ENROL_MARK/10
1	11111111	7.8
2	11111111	(null)
3	11111111	(null)
4	11111112	3.5
5	11111112	(null)
6	11111113	6.5
7	11111113	(null)
8	11111114	(null)

Oracle NVL function

- It is used to replace a NULL with a value.

```
SELECT stu_nbr,
   NVL(enrol_mark,0),
   NVL(enrol_grade,'WH')
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

Renaming Column

- Note column headings on slide 16
- Use the word "AS"
  - New column name in " " to maintain case or spacing
- Example


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



```
SELECT stu_nbr, enrol_mark/10 AS "New Mark"
FROM enrolment;
```

Sorting Query Result

- "ORDER BY" clause – *tuples have no order*
  - Must be used if more than one row may be returned
- Order can be ASCending or DESCending. The default is ASCending.
  - NULL values can be explicitly placed first/last using "NULLS LAST" or "NULLS FIRST" command
- Sorting can be done for multiple columns.
  - order of the sorting is specified for each column.
- Example:


```
SELECT stu_nbr, enrol_mark
FROM enrolment
ORDER BY enrol_mark DESC
```

	STU_NBR	ENROL_MARK
1	11111111	(null)
2	11111111	(null)
3	11111114	(null)
4	11111112	(null)
5	11111113	(null)
6	11111111	78
7	11111113	65
8	11111112	35

Removing Duplicate Rows in the Query Result

- Use "DISTINCT" as part of SELECT clause
  - *use with care*


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

	STU_NBR
1	11111114
2	11111111
3	11111112
4	11111113

SQL NATURAL JOIN

STUDENT

sno	name
1	alex
2	maria
3	bob

QUALIFICATION

sno	degree	year
1	bachelor	1990
1	master	2000
2	PhD	2001


```
SELECT *
FROM student JOIN qualification
        ON student.sno = qualification.sno
ORDER BY student.sno
```

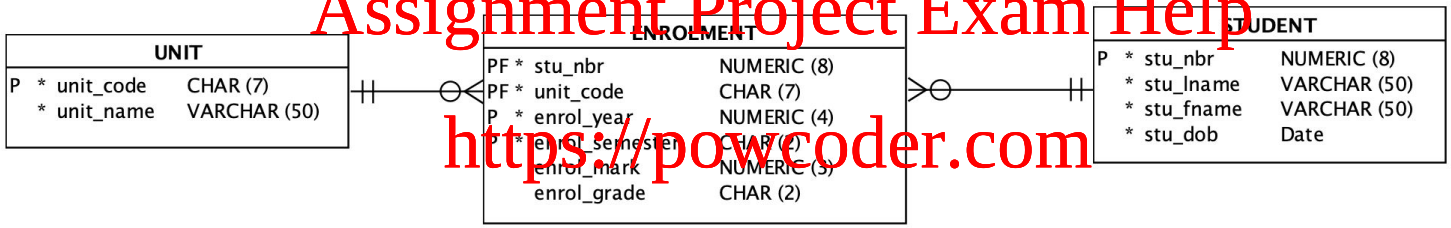

sno	name	degree	year
1	alex	bachelor	1990
1	alex	master	2000
2	maria	PhD	2001

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

JOIN-ing Multiple Tables

Pair the PK and FK in the JOIN condition
Note table aliasing e.g. unit u in FROM clause




```
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;
```

How many conditions will be used to join the two tables?


```
SELECT *
FROM table1 t1 JOIN table2 t2 ON
      (t1.T1_attribute1 = t2.T1_attribute1
       AND
       t1.T1_attribute2 = t2.T1_attribute2)
ORDER BY t1.T1_attribute1, t1.T1_attribute2;
```

Summary

- SQL statement, clause, predicate.
- Writing SQL predicates.
  - Comparison, range, set membership, pattern matching, is NULL
  - Combining predicates using logic operators (AND, OR, NOT)
- Arithmetic operation
  - NVL function
- Column alias.
- Ordering (Sorting) result.
- Removing duplicate rows.
- JOIN-ing tables

Oracle Date Data Type

Oracle Data Datatype

- Dates are stored differently from the SQL standard
  - standard uses two different types: date and time
  - Oracle uses one type: DATE
    - Stored in internal format contains date and time
      - Julian date as number (can use arithmetic)
    - Output is controlled by formatting
      - select **to_char**(sysdate,'dd-Mon-yyyy') from dual;
        - » 04-May-2020
      - select **to_char**(sysdate,'dd-Mon-yyyy hh:mi:ss PM') from dual;
        - » 04-May-2020 02:51:24 PM

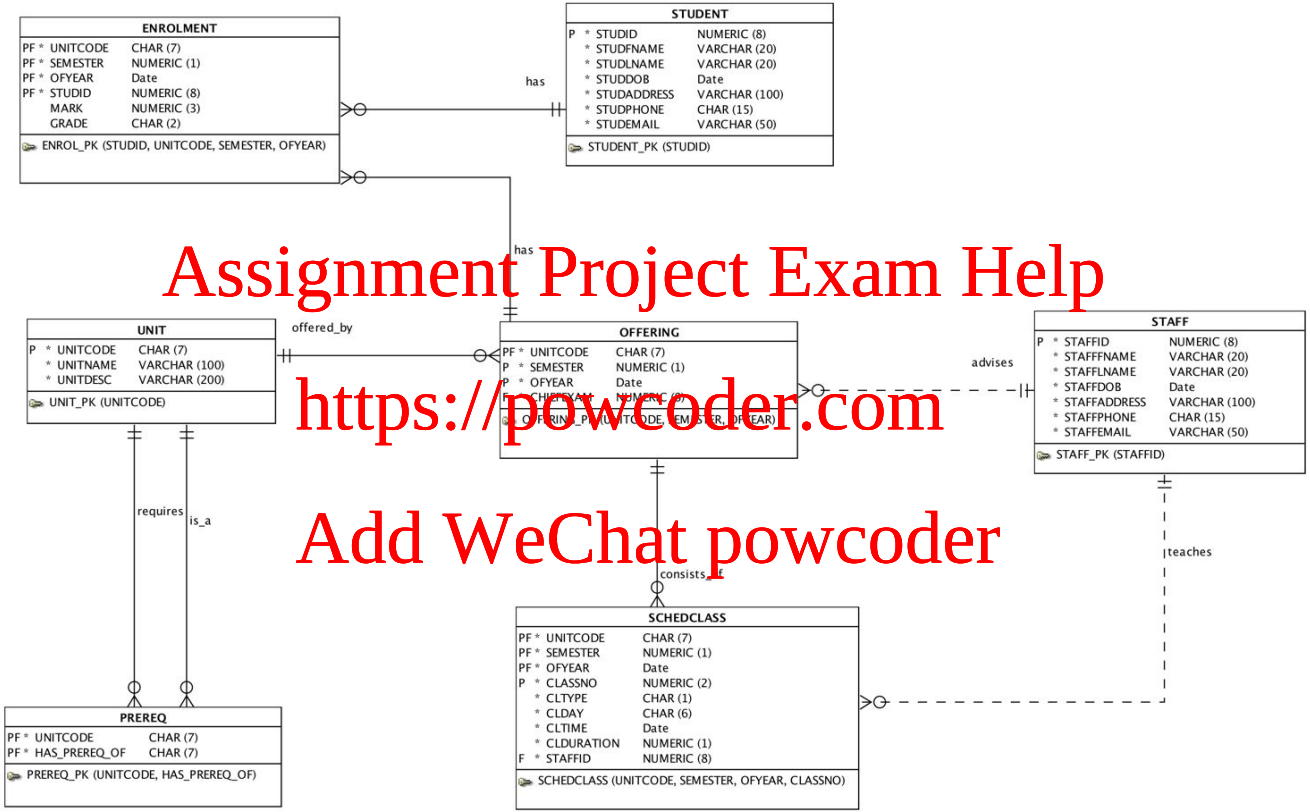
- DATE data type should be formatted with **TO_CHAR** when selecting for **display**.
- Text representing date **must be formatted** with **TO_DATE** when **comparing** or **inserting/updating.**
- Example:


```
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;
```

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

Uni Data Model



Putting this to Work

Q1. Show the ids, names of students as a single column called NAME and their DOBs. Order the output in date of birth order

Q2. Show the ids, names of students as a single column called NAME, unit code, and year and semester of enrolment where the mark is NULL. Order the output by student id, within unit code order