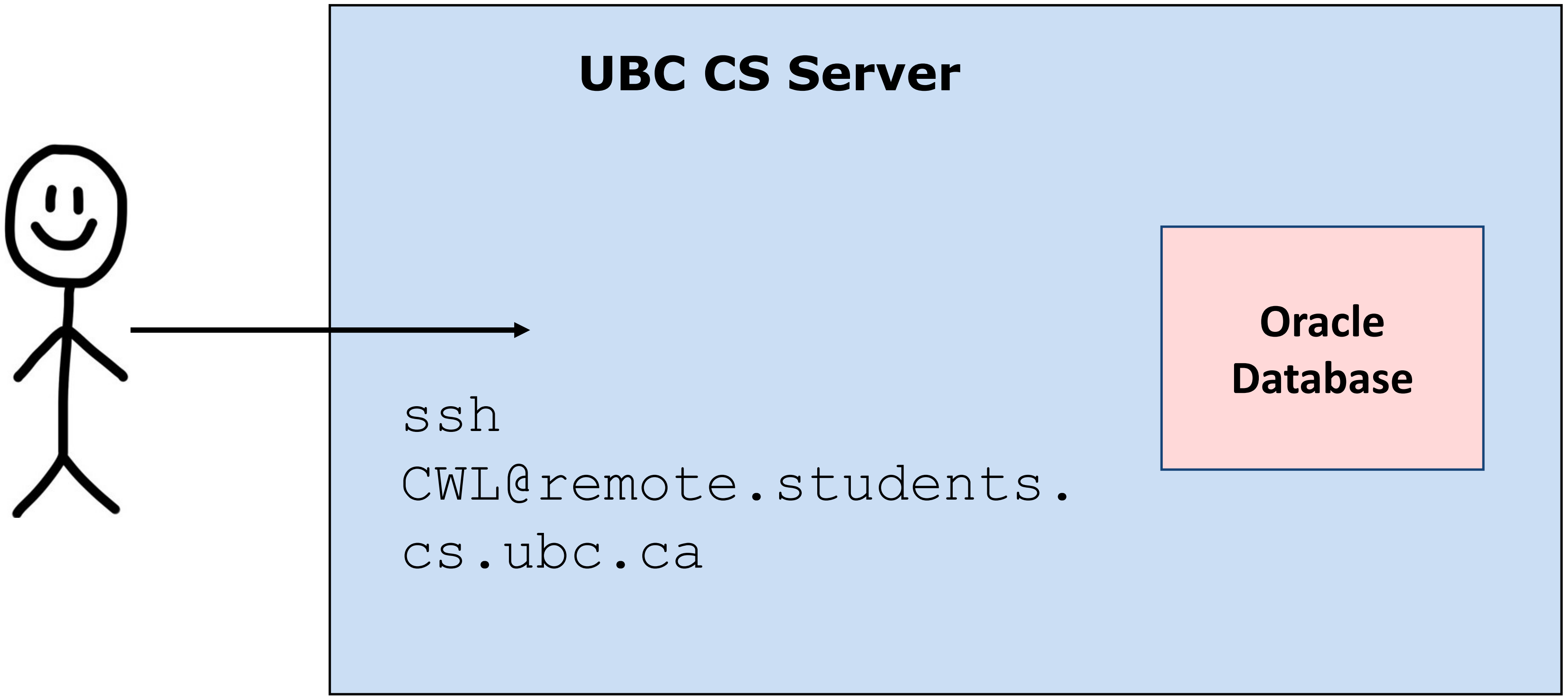
## Administrative Notes – February 7, 2023

- Feb 17: Assignment 3 (pairs if you want) due
  - Make sure you sign up in pairs beforehand as we need time to configure Canvas to allow both partners to view the submission and subsequent feedback
    - Everyone who has signed up as of last night has been put into a group on Canvas

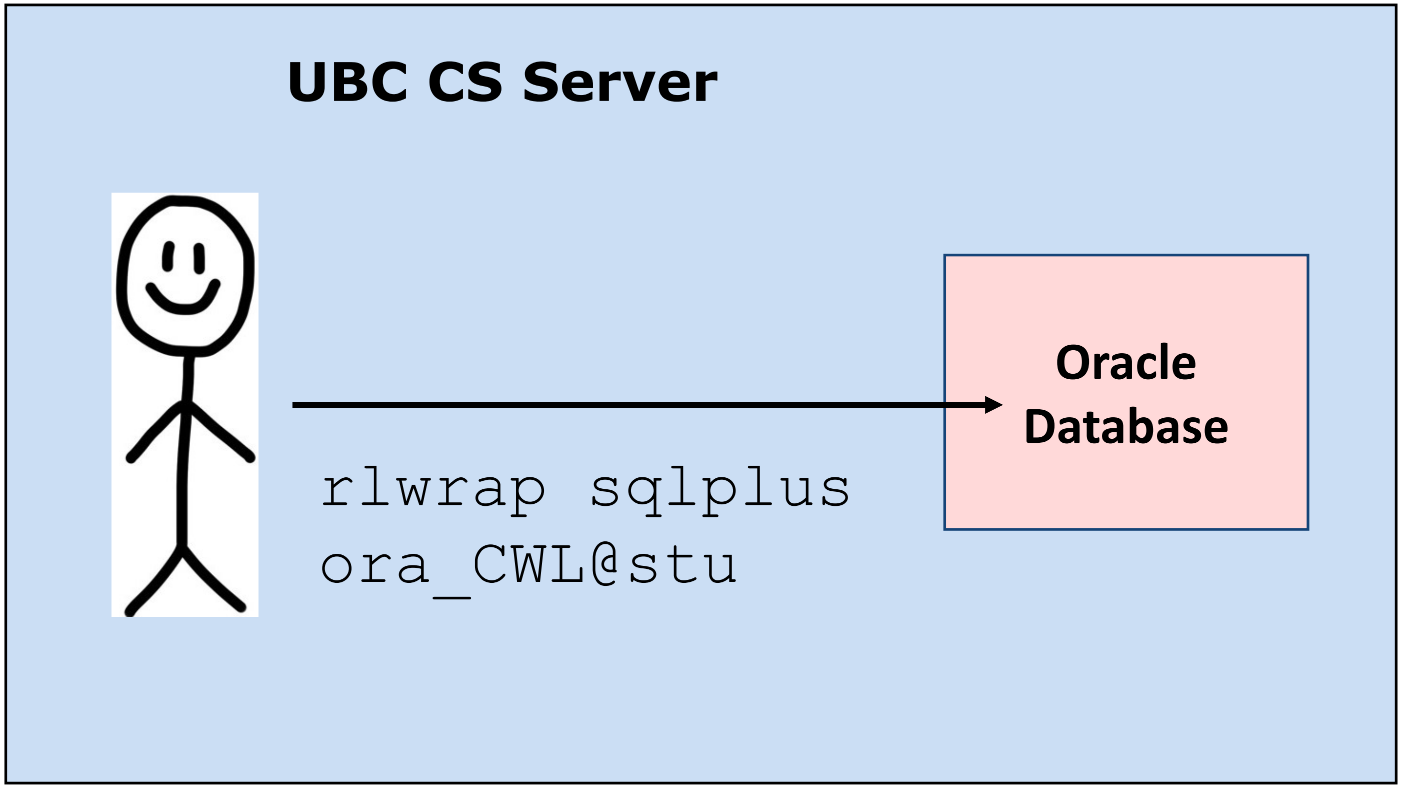
- Feb 20 – 24: Reading Break! (yay!)
  - No lectures, tutorials, or office hours during this week

# Logging into SQL Plus



```
ssh
CWL@remote.students.
cs.ubc.ca
```

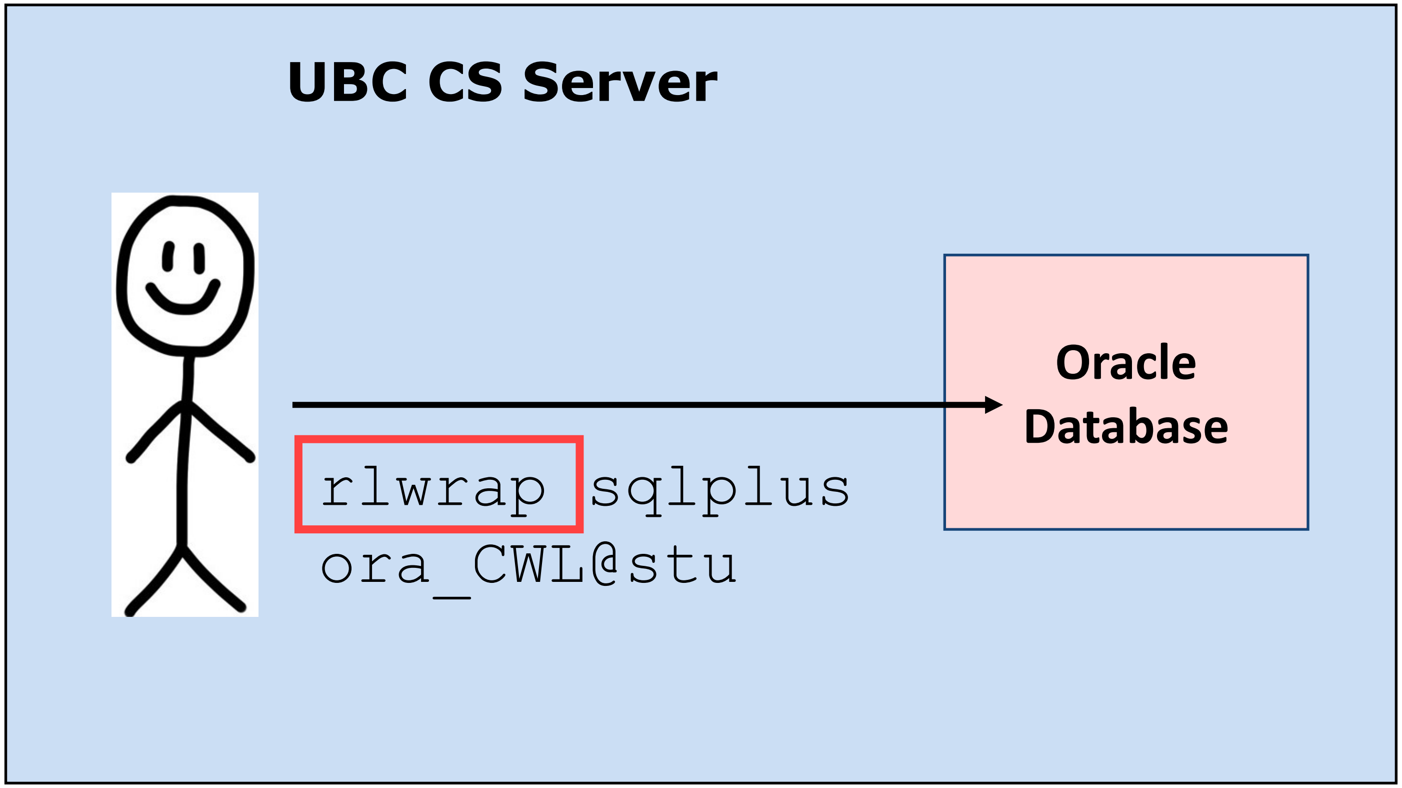
## Logging into SQL Plus



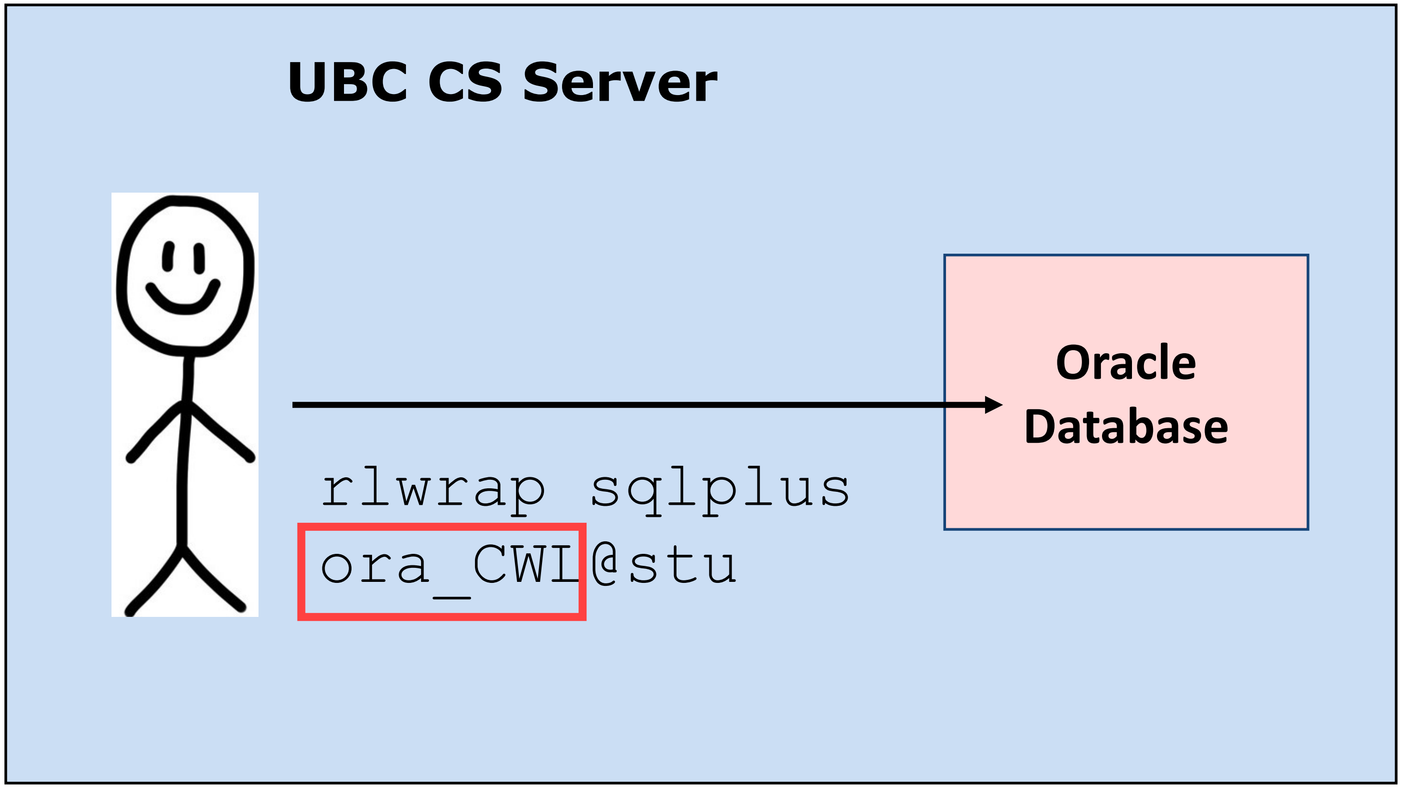
```
rlwrap sqlplus
ora_CWL@stu
```

```
Password is a<student number>
like a12345678.
```

## Logging into SQL Plus

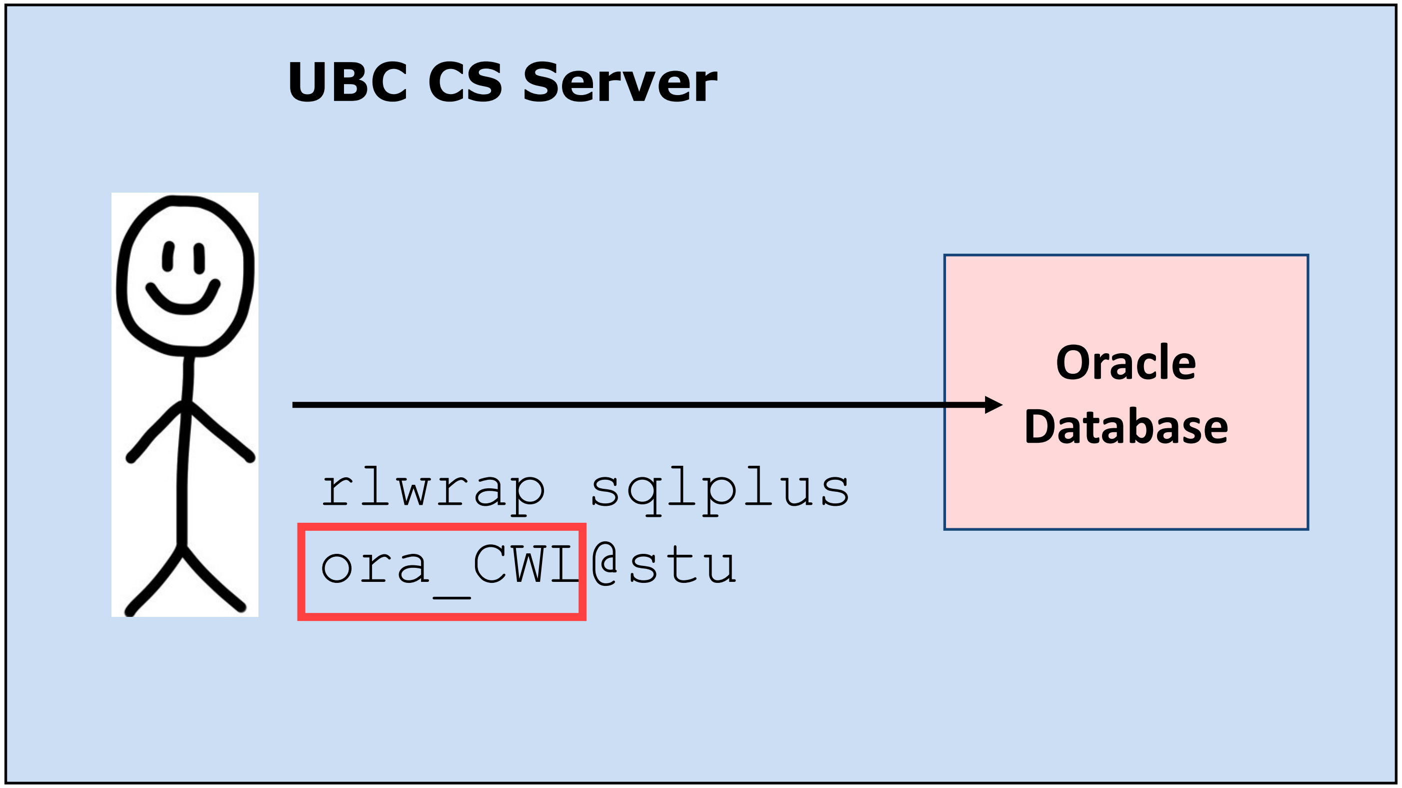


## Logging into SQL Plus



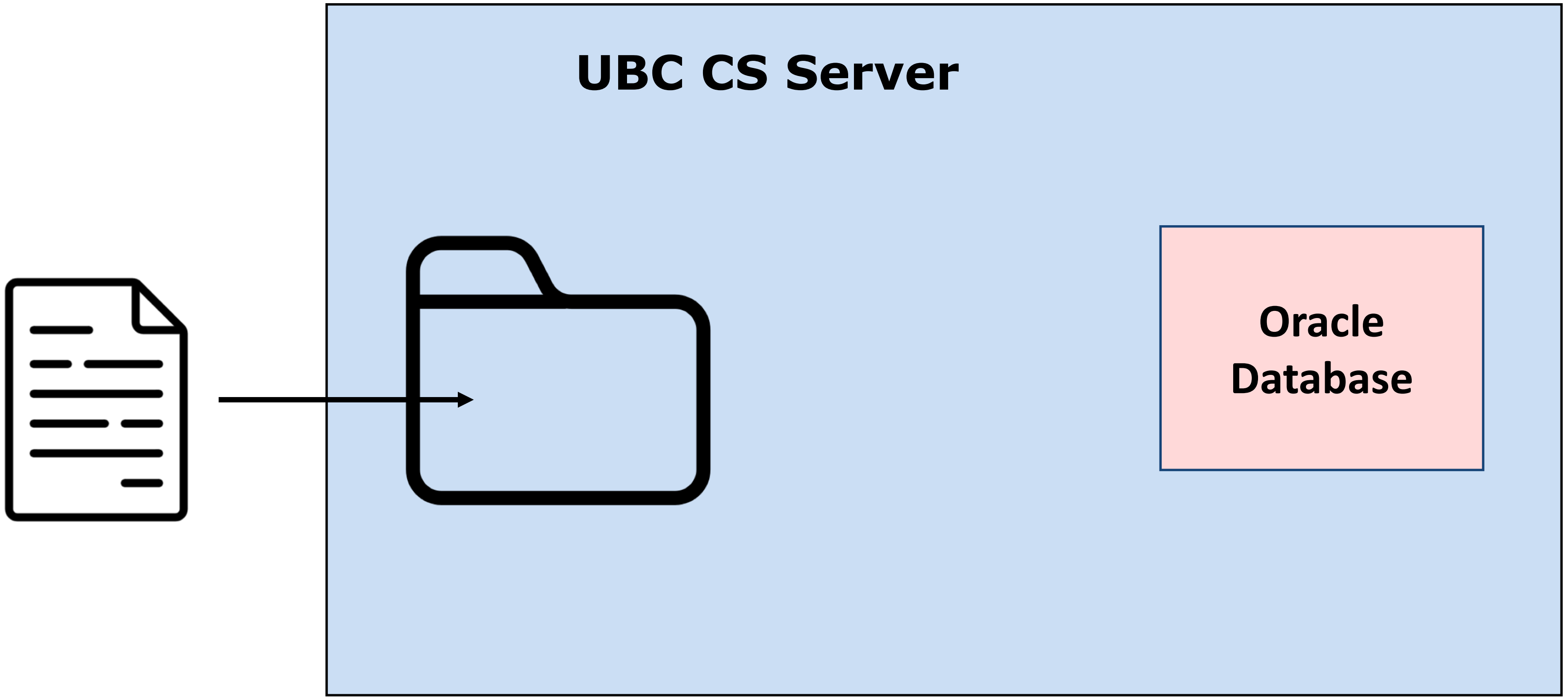
```
rlwrap sqlplus
ora_CWL@stu
```

## If you want to run a file of SQL statements in Oracle…

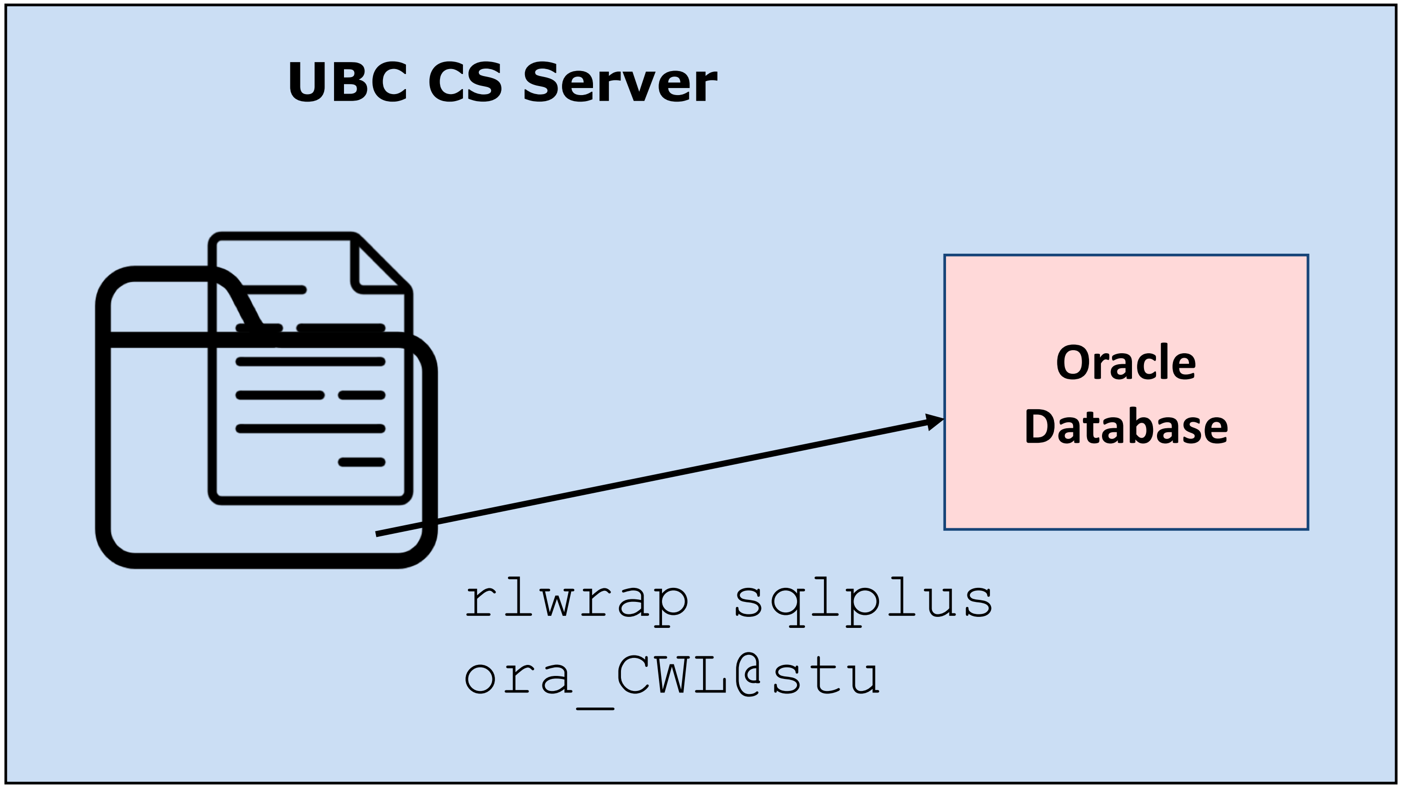


```
rlwrap sqlplus
ora_CWL@stu
```

## If you want to run a file of SQL statements in Oracle...



## If you want to run a file of SQL statements in Oracle…



```
rlwrap sqlplus
ora_CWL@stu
```

## Once you have logged into SQLPlus…

**Oracle Database**

```
start <filename>.sql;
```

Tutorial will be focused on SQLPlus for the next two weeks. Go to tutorial to get help if you are still stuck!

## Now where were we…

- SELECT, FROM, WHERE

```
SELECT    target-list
FROM      relation-list
WHERE     qualification
```

- We know how to work across multiple relations
- We know how to alias a relation (super important when relations have the same attribute name)
- We know how to rename any attributes in our result

## Conceptual Procedural Evaluation Strategy

1. Compute the cross-product of *relation-list*.
2. Discard resulting tuples if they fail *qualifications*.
3. Delete attributes that are not in *target-list*.
4. If DISTINCT is specified, eliminate duplicate rows.

```
SELECT    target-list
FROM      relation-list
WHERE     qualification
```

## Example of Conceptual Procedural Evaluation

```
SELECT Name
FROM MovieStar M, StarsIn S
WHERE S.StarID = M.StarID AND
      MovieID = 276
```

| StarID | Name | Gender | MovieID | StarID | Character |
|---|---|---|---|---|---|
| 1273 | Nathalie Portman | Female | 272 | 1269 | Leigh Anne Touhy |
| 1273 | Nathalie Portman | Female | 273 | 1270 | Mary |
| 1273 | Nathalie Portman | Female | 274 | 1271 | King George VI |
| 1273 | Nathalie Portman | Female | 276 | 1273 | Nina Sayers |
| … | … | … | … | … | … |

## New Students Example

- Class(name,meets_at,room,fid)
- Student(snum,sname,major,standing,age)
- Enrolled(snum,cname)
- Faculty(fid,fname,deptid)

## Class Table

| Name | Meets_at | Room | FID |
|---|---|---|---|
| Data Structures | MWF 10 | R128 | 489456522 |
| Database Systems | MWF 12:30-1:45 | 1320 DCL | 142519864 |
| Operating System Design | TuTh 12-1:20 | 20 AVW | 489456522 |
| Archaeology of the Incas | MWF 3-4:15 | R128 | 248965255 |
| Aviation Accident Investigation | TuTh 1-2:50 | Q3 | 011564812 |
| Air Quality Engineering | TuTh 10:30-11:45 | R15 | 011564812 |
| Introductory Latin | MWF 3-4:15 | R12 | 248965255 |
| American Political Parties | TuTh 2-3:15 | 20 AVW | 619023588 |
| Social Cognition | Tu 6:30-8:40 | R15 | 159542516 |
| Perception | MTuWTh 3 | Q3 | 489221823 |
| Multivariate Analysis | TuTh 2-3:15 | R15 | 090873519 |
| Patent Law | F 1-2:50 | R128 | 090873519 |
| Urban Economics | MWF 11 | 20 AVW | 489221823 |
| Organic Chemistry | TuTh 12:30-1:45 | R12 | 489221823 |
| Marketing Research | MW 10-11:15 | 1320 DCL | 489221823 |
| Seminar in American Art | M 4 | R15 | 489221823 |
| Orbital Mechanics | MWF 8 1320 | DCL | 011564812 |
| Dairy Herd Management | TuTh 12:30-1:45 | R128 | 356187925 |
| Communication Networks | MW 9:30-10:45 | 20 AVW | 141582651 |
| Optical Electronics | TuTh 12:30-1:45 | R15 | 254099823 |
| Introduction to Math | TuTh 8-9:30 | R128 | 489221823 |

# Student Table

```
      SNUM SNAME                MAJOR                         ST AGE
---------- ------------------------------------------------ ---
  51135593 Maria White          English                       SR  21
  60839453 Charles Harris       Architecture                  SR  22
  99354543 Susan Martin         Law                           JR  20
 112348546 Joseph Thompson      Computer Science              SO  19
 115987938 Christopher Garcia   Computer Science              JR  20
 132977562 Angela Martinez      History                       SR  20
 269734834 Thomas Robinson      Psychology                    SO  18
 280158572 Margaret Clark       Animal Science                FR  18
 301221823 Juan Rodriguez       Psychology                    JR  20
 318548912 Dorthy Lewis         Finance                       FR  18
 320874981 Daniel Lee           Electrical Engineering        FR  17
 322654189 Lisa Walker          Computer Science              SO  17
 348121549 Paul Hall            Computer Science              JR  18
 351565322 Nancy Allen          Accounting                    JR  19
 451519864 Mark Young           Finance                       FR  18
 455798411 Luis Hernandez       Electrical Engineering        FR  17
 462156489 Donald King          Mechanical Engineering        SO  19
 550156548 George Wright        Education                     SR  21
 552455318 Ana Lopez            Computer Engineering          SR  19
 556784565 Kenneth Hill         Civil Engineering             SR  21
 567354612 Karen Scott          Computer Engineering          FR  18
 573284895 Steven Green         Kinesiology                   SO  19
 574489456 Betty Adams          Economics                     JR  20
 578875478 Edward Baker         Veterinary Medicine           SR  21
```

## Enrolled Table

```
 SNUM      CNAME
---------- ----------------------------------------
 112348546 Database Systems
 115987938 Database Systems
 348121549 Database Systems
 322654189 Database Systems
 552455318 Database Systems
 455798411 Operating System Design
 552455318 Operating System Design
 567354612 Operating System Design
 112348546 Operating System Design
 115987938 Operating System Design
 322654189 Operating System Design
 567354612 Data Structures
 552455318 Communication Networks
 455798411 Optical Electronics
 455798411 Organic Chemistry
 301221823 Perception
 301221823 Social Cognition
 301221823 American Political Parties
 556784565 Air Quality Engineering
  99354543 Patent Law
 574489456 Urban Economics
```

## Faculty Table

```
       FID FNAME                DEPTID
---------- ----------------------
 142519864 I. Teach             20
 242518965 James Smith          68
 141582651 Mary Johnson         20
 011564812 John Williams        68
 254099823 Patricia Jones       68
 356187925 Robert Brown         12
 489456522 Linda Davis          20
 287321212 Michael Miller       12
 248965255 Barbara Wilson       12
 159542516 William Moore        33
 090873519 Elizabeth Taylor     11
 486512566 David Anderson       20
 619023588 Jennifer Thomas      11
 489221823 Richard Jackson      33
 548977562 Ulysses Teach        20
```

## Running Examples

Movie(MovieID, Title, Year)

StarsIn(MovieID, StarID, Character)

MovieStar(StarID, Name, Gender)

Student(snum,sname,major,standing,age)

Class(name,meets_at,room,fid)

Enrolled(snum,cname)

Faculty(fid,fname,deptid)

## What kinds of queries can you answer so far?

Find the student ids of those who have taken a course named “Database Systems”.

Student(snum,sname,major,standing,age)

Class(name,meets_at,room,fid)

Enrolled(snum,cname)

Faculty(fid,fname,deptid)

## What kinds of queries can you answer so far?

Find the student ids of those who have taken a course named “Database Systems”.

```
SELECT snum
FROM enrolled e
WHERE cname = 'Database Systems'
```

Do we need distinct? A. Yes. [✓ B. No]

Student(snum,sname,major,standing,age)

Class(name,meets_at,room,fid)

Enrolled(snum,cname)

Faculty(fid,fname,deptid)

## What kinds of queries can you answer so far?

Find the names of all classes taught by Elizabeth Taylor.

Student(snum,sname,major,standing,age)

Class(name,meets_at,room,fid)

Enrolled(snum,cname)

Faculty(fid,fname,deptid)

## What kinds of queries can you answer so far?

Find the names of all classes taught by Elizabeth Taylor.

```
SELECT name
FROM Faculty f, class c
WHERE f.fid = c.fid and
      fname = 'Elizabeth Taylor'
```

Do we need distinct? A. Yes. [✓ B. No]

Student(snum,sname,major,standing,age)

Class(name,meets_at,room,fid)

Enrolled(snum,cname)

Faculty(fid,fname,deptid)

## What kinds of queries can you answer so far?

Find the departments that have at least one faculty member

Student(<u>snum</u>,sname,major,standing,age)

Class(<u>name</u>,meets_at,room,fid)

Enrolled(<u>snum,cname</u>)

Faculty(<u>fid</u>,fname,deptid)

## What kinds of queries can you answer so far?

Find the departments that have at least one faculty member

```
SELECT DISTINCT deptid
FROM faculty
```

Do we need distinct? [✓ A. Yes.] B. No

Student(<u>snum</u>,sname,major,standing,age)

Class(<u>name</u>,meets_at,room,fid)

Enrolled(<u>snum,cname</u>)

Faculty(<u>fid</u>,fname,deptid)

## What kinds of queries can you answer so far?

Find the departments that have more than one faculty member (express not equal by "<>")

```
SELECT DISTINCT f1.deptid
FROM faculty f1, faculty f2
WHERE f1.fid <>f2.fid AND
      f1.deptid = f2.deptid
```

That is why renaming is important.

f1

| fid | fname | Deptid |
|---|---|---|
| 90873519 | Elizabeth Taylor | 11 |
| 619023588 | Jennifer Thomas | 11 |
| … | … | … |

f2

| fid | fname | Deptid |
|---|---|---|
| 90873519 | Elizabeth Taylor | 11 |
| 619023588 | Jennifer Thomas | 11 |
| … | … | … |

Do you need DISTINCT?

## String Comparisons

What are the student ids of those who have taken a course with “System” in the name?

## A string walks into a bar…

```
SELECT DISTINCT snum
FROM   enrolled
WHERE cname LIKE '%System%'
```

- LIKE is used for string matching:
  - '_' stands for any one character and
  - '%' stands for 0 or more arbitrary characters.
- SQL supports string operations such as
  - concatenation (using "||")
  - converting from upper to lower case (and vice versa)
  - finding string length, extracting substrings, etc.

## A string walks into a bar…

```
SELECT DISTINCT snum
FROM   enrolled
WHERE cname LIKE '%System%'
```

Do we need DISTINCT?

[✓ A. Yes]

B. No

## Ordering of Tuples

List in alphabetic order the names of actors who were in a movie in 1939.

```
SELECT DISTINCT name
FROM Movie, StarsIn, MovieStar
WHERE Movie.MovieID = StarsIn.MovieID AND
      StarsIn.StarID = MovieStar.StarID AND
      year = 1939
```

## Ordering of Tuples

List in alphabetic order the names of actors who were in a movie in 1939.

```
SELECT DISTINCT name
FROM Movie, StarsIn, MovieStar
WHERE Movie.MovieID = StarsIn.MovieID AND
      StarsIn.StarID = MovieStar.StarID AND
      year = 1939
ORDER BY name
```

## Ordering of Tuples

Order is specified by:

- **desc** for descending order
- **asc** for ascending order (default)
- E.g. **order by** *Name* **desc**
- You can order within order: for example, … “ORDER BY Year, Name” would first order by Year, then Name within years

## Clicker Question: Sorting

Relation R has schema R(a,b,c). In the result of the query

```
SELECT a, b, c
FROM R
ORDER BY c DESC, b ASC;
```

What condition must a tuple *t* satisfy so that *t* **necessarily precedes** the tuple (5,5,5)? Identify one such tuple from the list below.

A. (3,6,3)

B. (1,5,5)

C. (5,5,6)

D. All of the above

E. None of the above

## Clicker Question: Sorting

Relation R has schema R(a,b,c). In the result of the query

```
SELECT a, b, c
FROM R
ORDER BY c DESC, b ASC;
```

What condition must a tuple *t* satisfy so that *t* **necessarily precedes** the tuple (5,5,5)? Identify one such tuple from the list below.

A. (3,6,3) 3 < 5
B. (1,5,5) Not specified
C. (5,5,6) Right
D. All of the above
E. None of the above

clickerorder.sql and clickerorder2.sql produce different ordering for 7,5,5 vs. 1,5,5

## Set Operations

- **union, intersect,** and **except** correspond to the relational algebra operations $\cup$, $\cap$, $-$.
- Each automatically eliminates duplicates
- To retain all duplicates use the corresponding multiset versions:

  **union all, intersect all** and **except all**
- **A Union B** (OR) – combine the results from A and B
- **A Intersect B** (AND) – only keep results that appear in both A and B
- **A Except B** – only keep results found in A and not B

## Set Operations

- Suppose a tuple occurs *m* times in *r* and *n* times in *s*, then, it occurs:
  - *m* + *n* times in *r* **union all** *s*
  - min(*m, n)* times in *r* **intersect all** *s*
  - max(0, *m* – *n)* times in *r* **except all** *s*

## Find IDs of MovieStars who've been in a movie in 1944 *or* 1974

Movie(MovieID, Title, Year)
StarsIn(MovieID, StarID, Character)
MovieStar(StarID, Name, Gender)

- **UNION:** Can union any two *union-compatible* sets of tuples (i.e., the result of SQL queries).

## Find IDs of MovieStars who've been in a movie in 1944 *or* 1974

Movie(<u>MovieID</u>, Title, Year)
StarsIn(<u>MovieID, StarID</u>, Character)
MovieStar(<u>StarID</u>, Name, Gender)

- **UNION:** Can union any two *union-compatible* sets of tuples (i.e., the result of SQL queries).

```
SELECT  StarID
FROM  Movie M, StarsIn S
WHERE  M.MovieID=S.MovieID AND
( year = 1944 OR year = 1974)
```

- The two queries though quite similar return different results, why?
  - Use UNION ALL to get the same answer

```
SELECT  StarID
FROM     Movie M, StarsIn S
WHERE   M.MovieID = S.MovieID AND
year = 1944
UNION
SELECT  StarID
FROM     Movie M, StarsIn S
WHERE   M.MovieID = S.MovieID AND
year = 1974
```

## Set Operations: Intersect

Movie(<u>MovieID</u>, Title, Year)
StarsIn(<u>MovieID, StarID</u>, Character)
MovieStar(<u>StarID</u>, Name, Gender)

Example: Find IDs of stars who have been in a movie in 1944 *and* 1974.

- **INTERSECT:** Can be used to compute the intersection of any two *union-compatible* sets of tuples.
- In SQL/92, but some systems don't support it.

# Set Operations: Intersect

Movie(<u>MovieID</u>, Title, Year)
StarsIn(<u>MovieID, StarID</u>, Character)
MovieStar(<u>StarID</u>, Name, Gender)

Example: Find IDs of stars who have been in a movie in 1944 *<u>and</u>* 1974.

```
SELECT  StarID
FROM    Movie M, StarsIn S
WHERE   M.MovieID = S.MovieID AND
year = 1944
INTERSECT
SELECT  StarID
FROM    Movie M, StarsIn S
WHERE   M.MovieID = S.MovieID AND
year = 1974
```

- **INTERSECT:** Can be used to compute the intersection of any two *union-compatible* sets of tuples.
- In SQL/92, but some systems don't support it.

Oracle does
MYSQL doesn't

## Rewriting INTERSECT with Joins

Example: Find IDs of stars who have been in a movie in 1944 *and* 1974 without using **INTERSECT.**

Movie(MovieID, Title, Year)
StarsIn(MovieID, StarID, Character)
MovieStar(StarID, Name, Gender)

## Rewriting INTERSECT with Joins

Example: Find IDs of stars who have been in a movie in 1944 *and* 1974 without using **INTERSECT.**

```
SELECT  distinct S1.StarID
FROM     Movie M1, StarsIn S1,
         Movie M2, StarsIn S2
WHERE
         M1.MovieID = S1.MovieID AND M1.year = 1944 AND
         M2.MovieID = S2.MovieID AND M2.year = 1974 AND
         S2.StarID = S1.StarID
```

## Set Operations: EXCEPT

Find the sids of all students who took Operating System Design but did not take Database Systems

Student(<u>snum</u>,sname,major,standing,age)

Class(<u>name</u>,meets_at,room,fid)

Enrolled(<u>snum,cname</u>)

Faculty(<u>fid</u>,fname,deptid)

## Set Operations: EXCEPT

Find the sids of all students who took Operating System Design but did not take Database Systems

```
SELECT snum
FROM enrolled e
WHERE cname = 'Operating System Design'
EXCEPT ← Oracle uses MINUS rather than EXCEPT
SELECT snum
FROM enrolled e
WHERE cname = 'Database Systems'
```

Can we do it in a different way?
(We'll come back to this)

## But what about…

Select the IDs of all students who have not taken "Operating System Design"

- One way to do is to find all students that have taken a course.
- `MINUS` those who have taken "Operating System Design"

```
SELECT snum
FROM enrolled e
EXCEPT    ← Oracle uses MINUS rather than EXCEPT
SELECT snum
FROM enrolled e
WHERE cname = 'Operating System
        Design'
```

## Motivating Example for Nested Queries

Find ids and names of female stars who have been in movie with ID 28:

## Motivating Example for Nested Queries

Find ids and names of female stars who have been in movie with ID 28:

```
SELECT M.StarID, name
FROM MovieStar M, StarsIn S
WHERE M.StarID = S.starID AND S.MovieID = 28
      AND gender = ‘female’;
```

## Motivating Example for Nested Queries

Find ids and names of female stars who have been in movie with ID 28:

```
SELECT M.StarID, name
FROM MovieStar M, StarsIn S
WHERE M.StarID = S.starID AND S.MovieID = 28
      AND gender = ‘female’;
```

Find ids and names of female stars who have not been in movie w/ ID 28 w/o using EXCEPT/MINUS:

- Would the following be correct?

```
~~SELECT M.StarID, name~~
~~FROM MovieStar M, StarsIn S~~
~~WHERE M.StarID = S.starID AND S.MovieID <> 28~~
~~      AND gender = ‘female’;~~
```

# Motivating Example for Nested Queries

MovieStar

| <u>StarID</u> | Name | Gender |
|---|---|---|
| 1 | Jessica Wong | Female |
| 2 | Jia Lu | Male |
| 3 | Carol Huang | Female |

StarsIn

| <u>MovieID</u> | StarID | Character |
|---|---|---|
| 28 | 1 | A tree |
| 28 | 3 | Background |
| 29 | 3 | A tired grad student |

Does "WHERE StarsIn.MovieID <> 28" correctly remove Carol from the results?

## Nested Queries

- A very powerful feature of SQL:

```
SELECT A₁,A₂,…,Aₙ
FROM   R₁,R₂, …,Rₘ
WHERE  condition
```

- A nested query is a query that has another query embedded with it.
  - A **SELECT, FROM, WHERE, or HAVING** clause can itself contain an SQL query!
  - Being part of the `WHERE` clause is the most common

## Nested Queries (IN/Not IN)

Find ids and names of stars who have been in movie with ID 28:

## Nested Queries (IN/Not IN)

Find ids and names of stars who have been in movie with ID 28:

```
SELECT M.StarID, M.Name
FROM MovieStar M
WHERE M.Gender = 'female' AND
      M.StarID IN (SELECT S.StarID
                   FROM StarsIn S
                   WHERE MovieID = 28)
```

There's also NOT IN

- To find stars who have *not* been in movie 28, use **NOT IN**.
- To understand nested query semantics, think of a ***nested loops*** evaluation:
    - *For each MovieStar tuple, check the qualification by computing the subquery.*