Software Installation

- Downloading and Installing Oracle Express Edition (Slides 2-14)
- Downloading SQL Developer (Slides 15-18)
- Connecting SQL Developer to Oracle Express Edition (Slides 19-21)

Downloading and Installing Oracle Express Edition

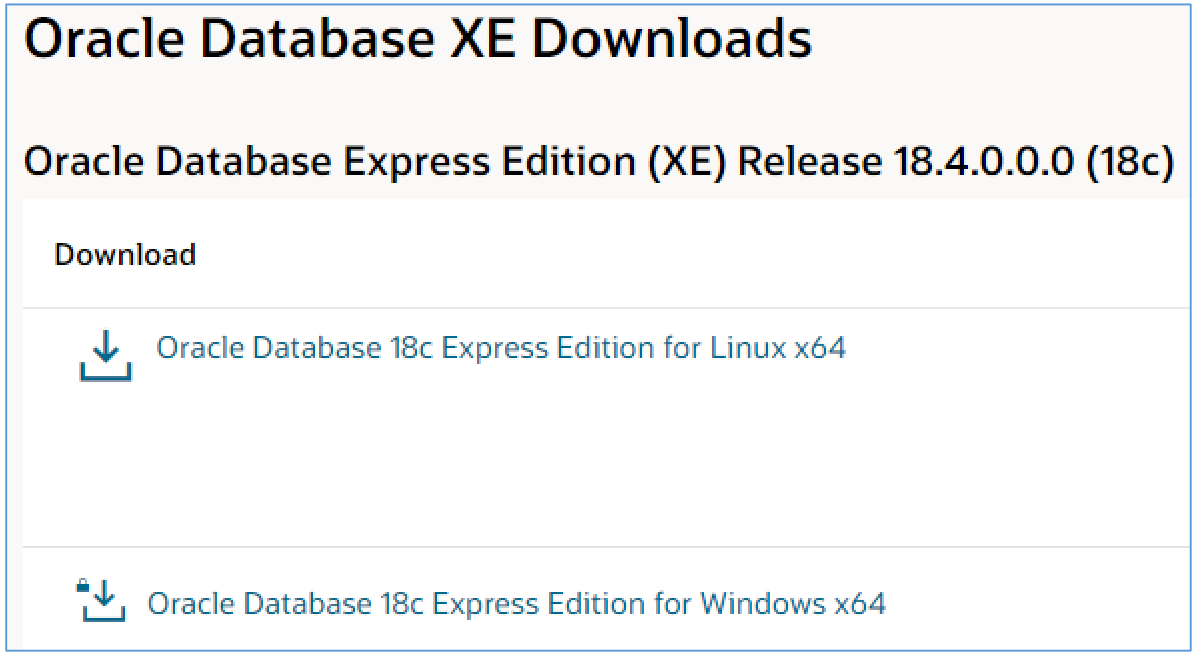
Step 1

- Visit Oracle Express Edition download page
  - https://www.oracle.com/database/technologies/xe-downloads.html

- Notice that the above link might change

Step 2

Click on the download appropriate to your operating system



Oracle Database XE Downloads

Oracle Database Express Edition (XE) Release 18.4.0.0.0 (18c)

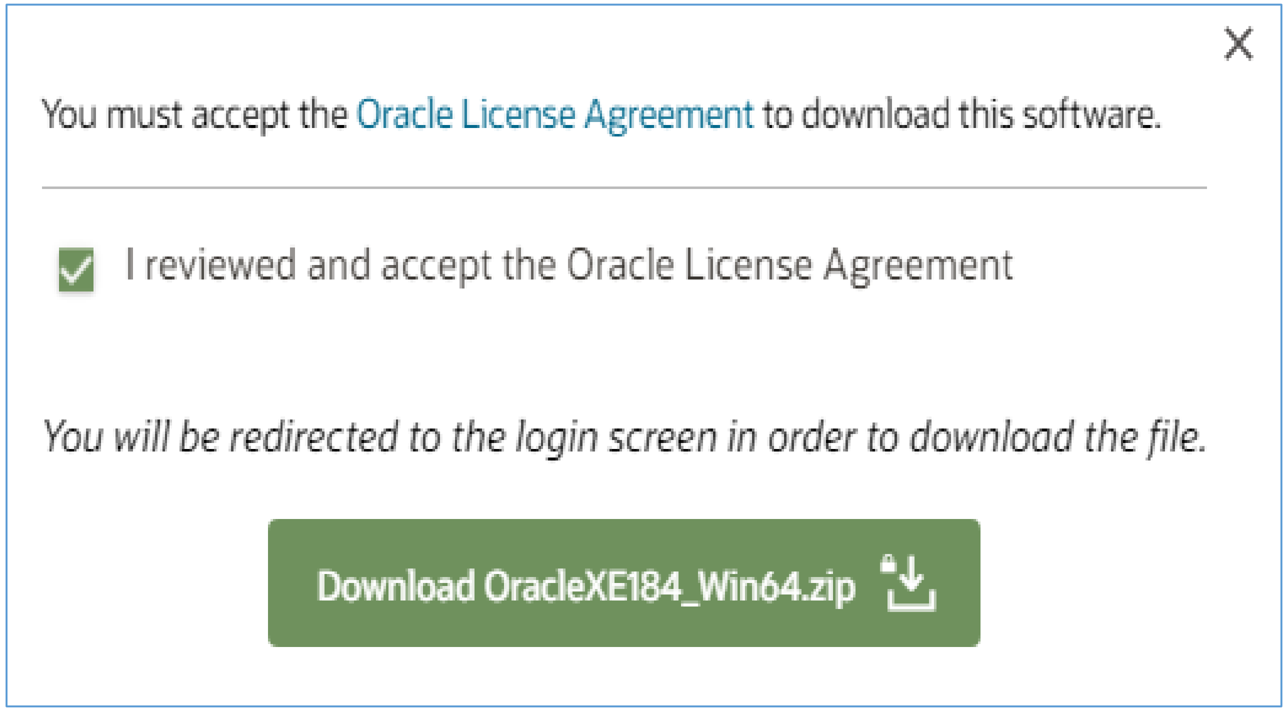
Download

Oracle Database 18c Express Edition for Linux x64

Oracle Database 18c Express Edition for Windows x64

Step 3

Check the box for reviewing and accepting license, then click on download button



You must accept the Oracle License Agreement to download this software.

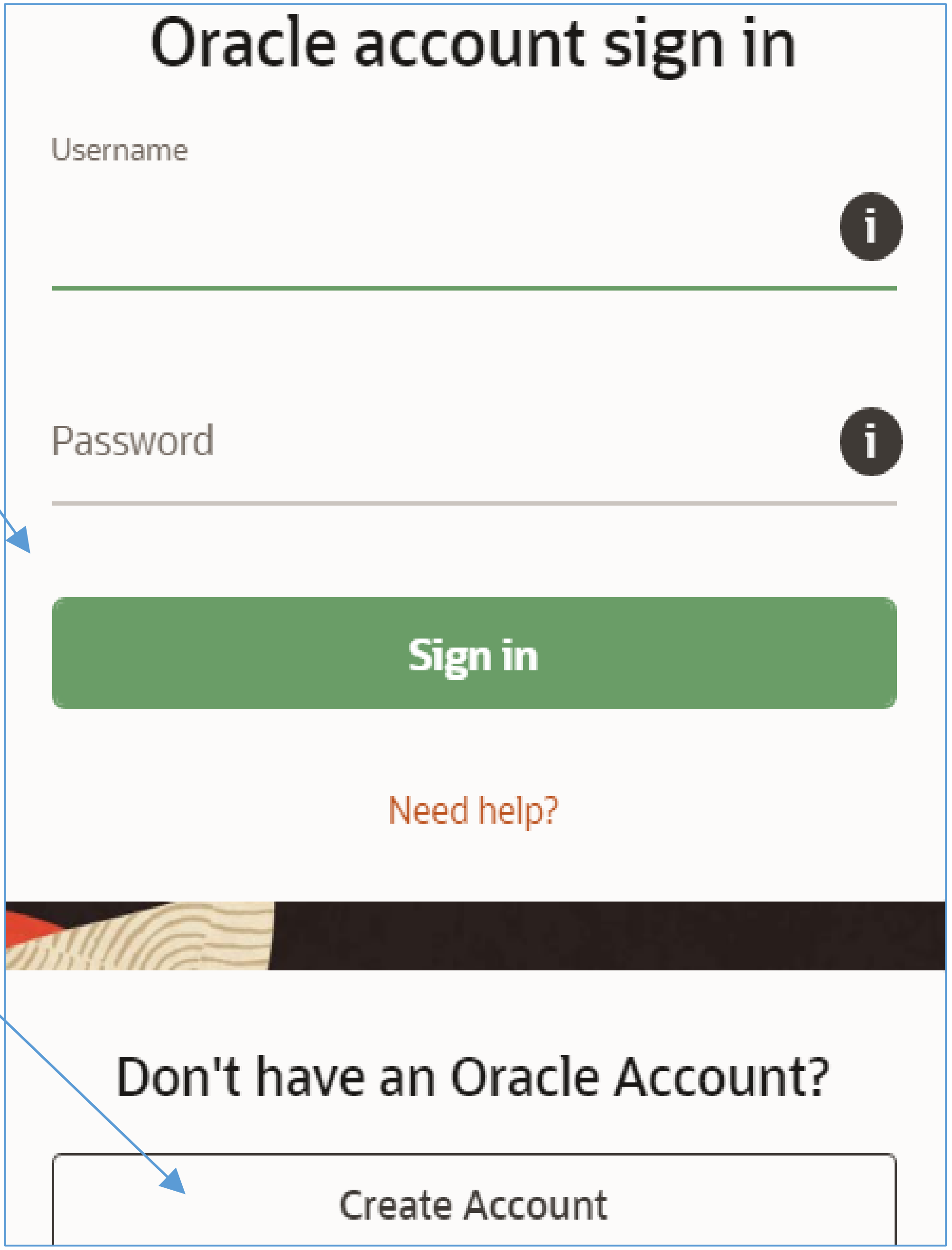
✓ I reviewed and accept the Oracle License Agreement

You will be redirected to the login screen in order to download the file.

Step 4

You should login using your oracle account.

If you don't have an account create one


Oracle account sign in

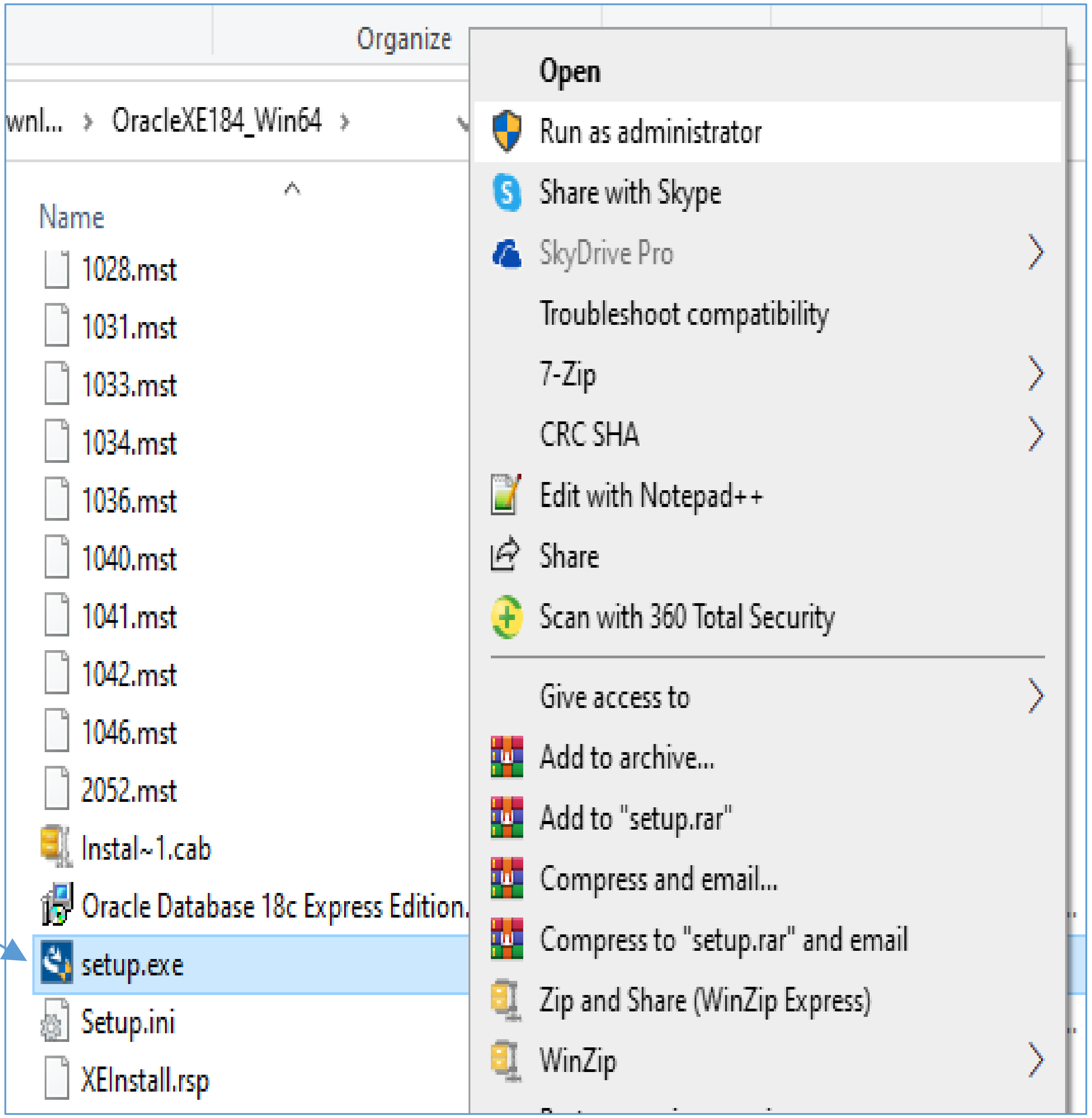
Username

Password

Don't have an Oracle Account?

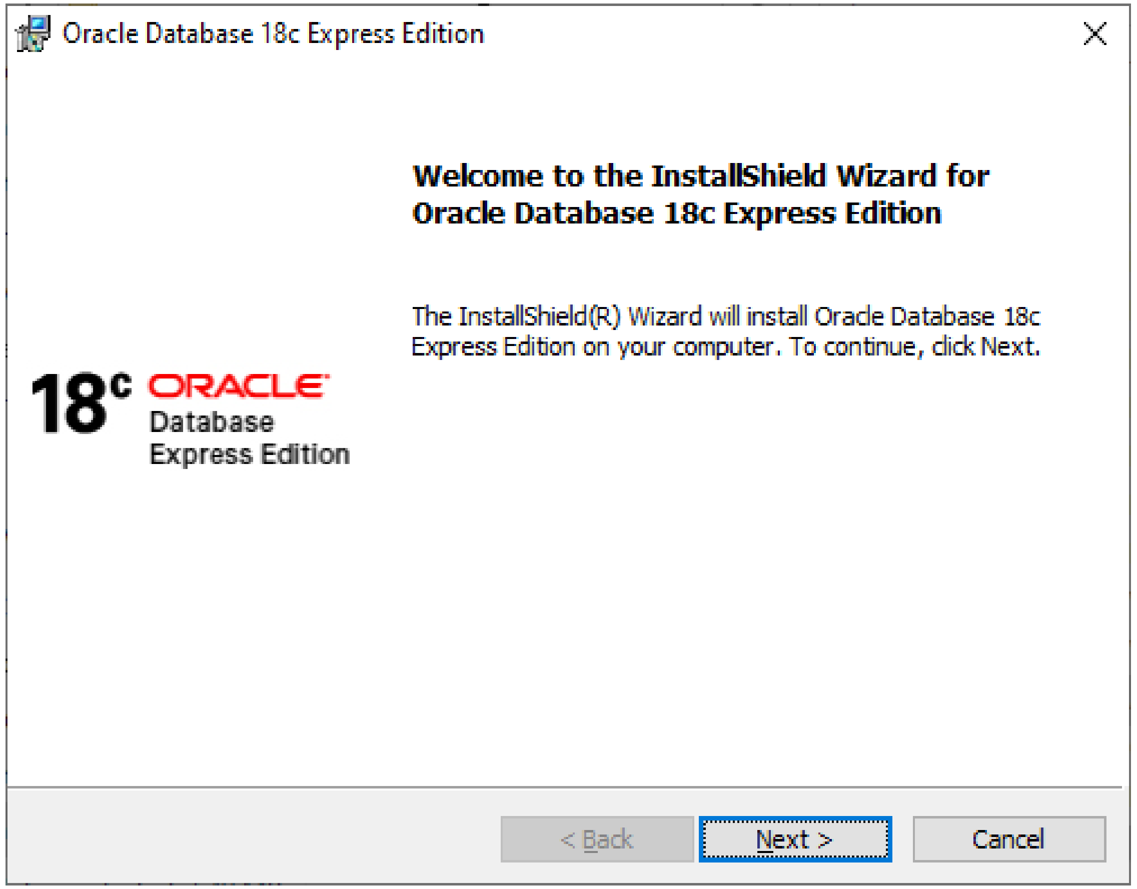
Step 5

After your download is complete:
- unzip the download file
- get into the extracted folder
- Right click on “setup.exe”
- Click “Run as Administrator”



Step 6

Click “Next”



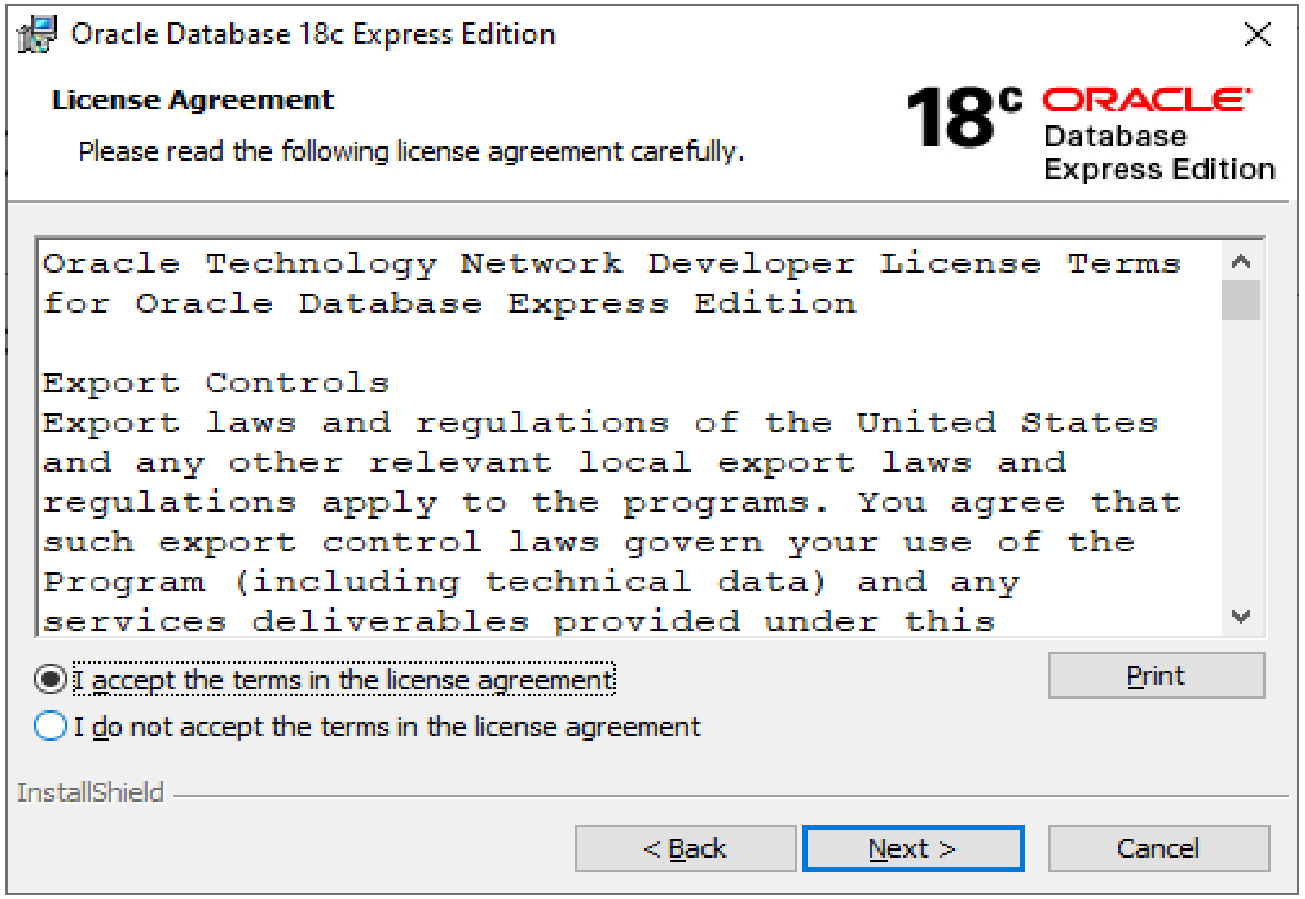
Oracle Database 18c Express Edition

Welcome to the InstallShield Wizard for Oracle Database 18c Express Edition

The InstallShield(R) Wizard will install Oracle Database 18c Express Edition on your computer. To continue, click Next.

Step 7

Accept the license and click "Next"



Oracle Database 18c Express Edition

License Agreement

Please read the following license agreement carefully.

18c ORACLE Database Express Edition

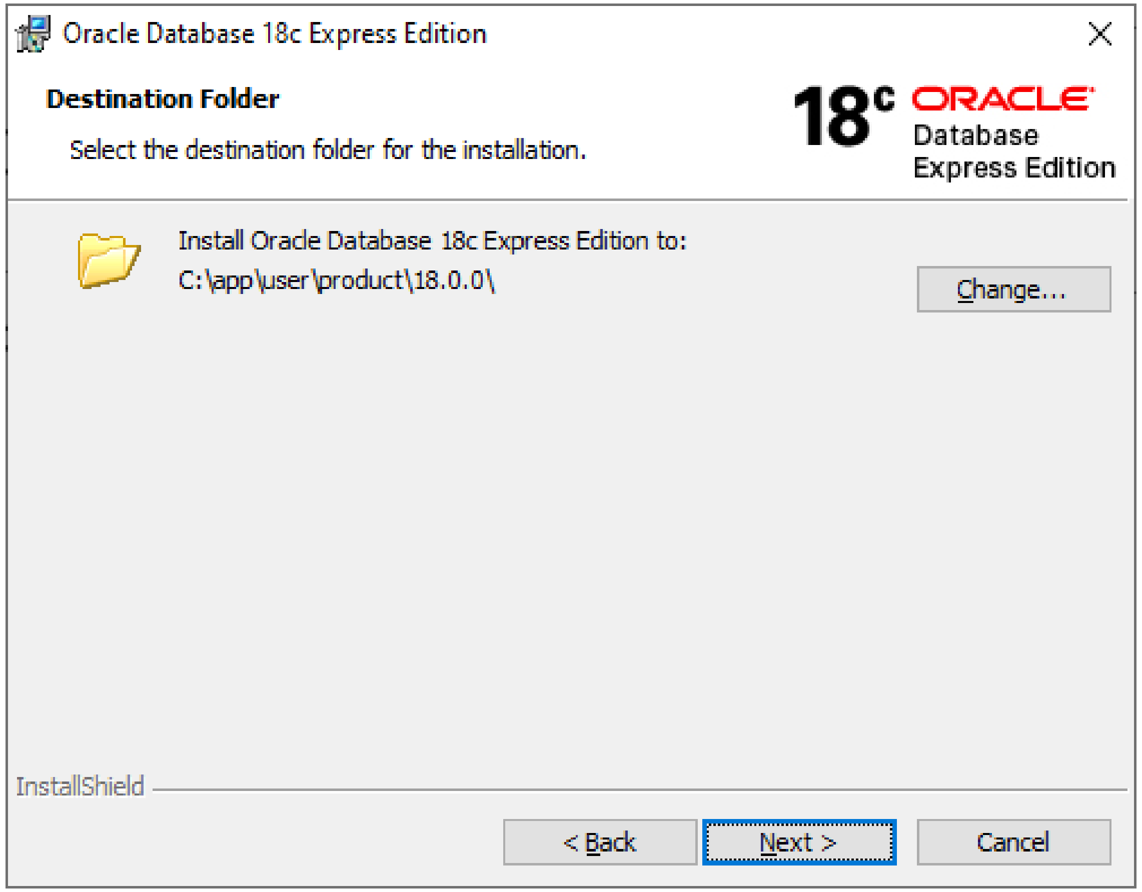
```
Oracle Technology Network Developer License Terms
for Oracle Database Express Edition

Export Controls
Export laws and regulations of the United States
and any other relevant local export laws and
regulations apply to the programs. You agree that
such export control laws govern your use of the
Program (including technical data) and any
services deliverables provided under this
```

[✓ I accept the terms in the license agreement]

I do not accept the terms in the license agreement

Step 8

click “Next”


Oracle Database 18c Express Edition

Destination Folder

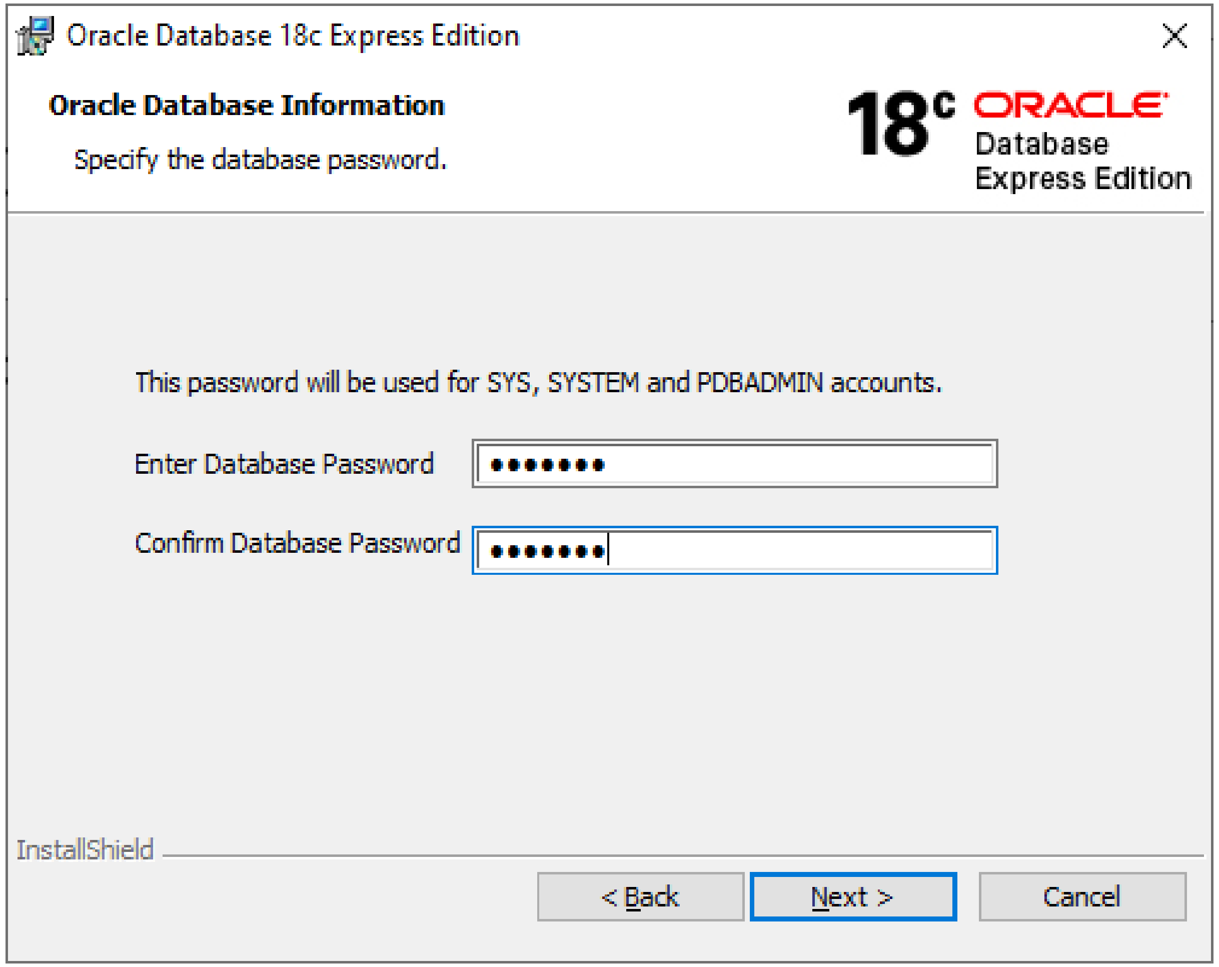
Select the destination folder for the installation.

Install Oracle Database 18c Express Edition to:
C:\app\user\product\18.0.0\

Step 9

Choose a password and confirm it, then click next

Make sure you remember the password. It is very important for connecting to SQL developer



Oracle Database 18c Express Edition

Oracle Database Information

Specify the database password.

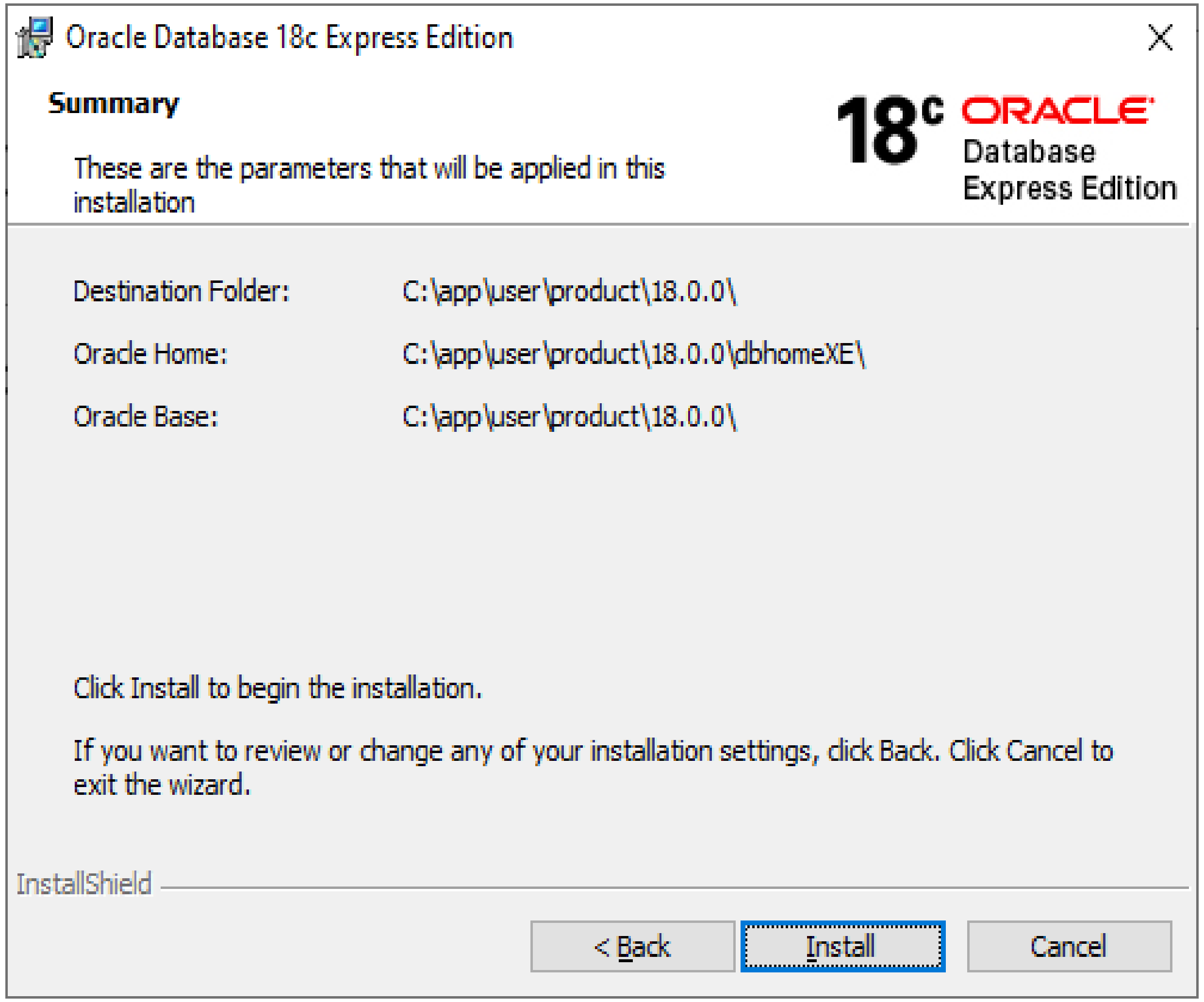
This password will be used for SYS, SYSTEM and PDBADMIN accounts.

Enter Database Password ••••••••

Confirm Database Password ••••••••

Step 10

Click "Install"


Oracle Database 18c Express Edition
18c ORACLE Database Express Edition
InstallShield
< Back
Install
Cancel

Summary

These are the parameters that will be applied in this installation

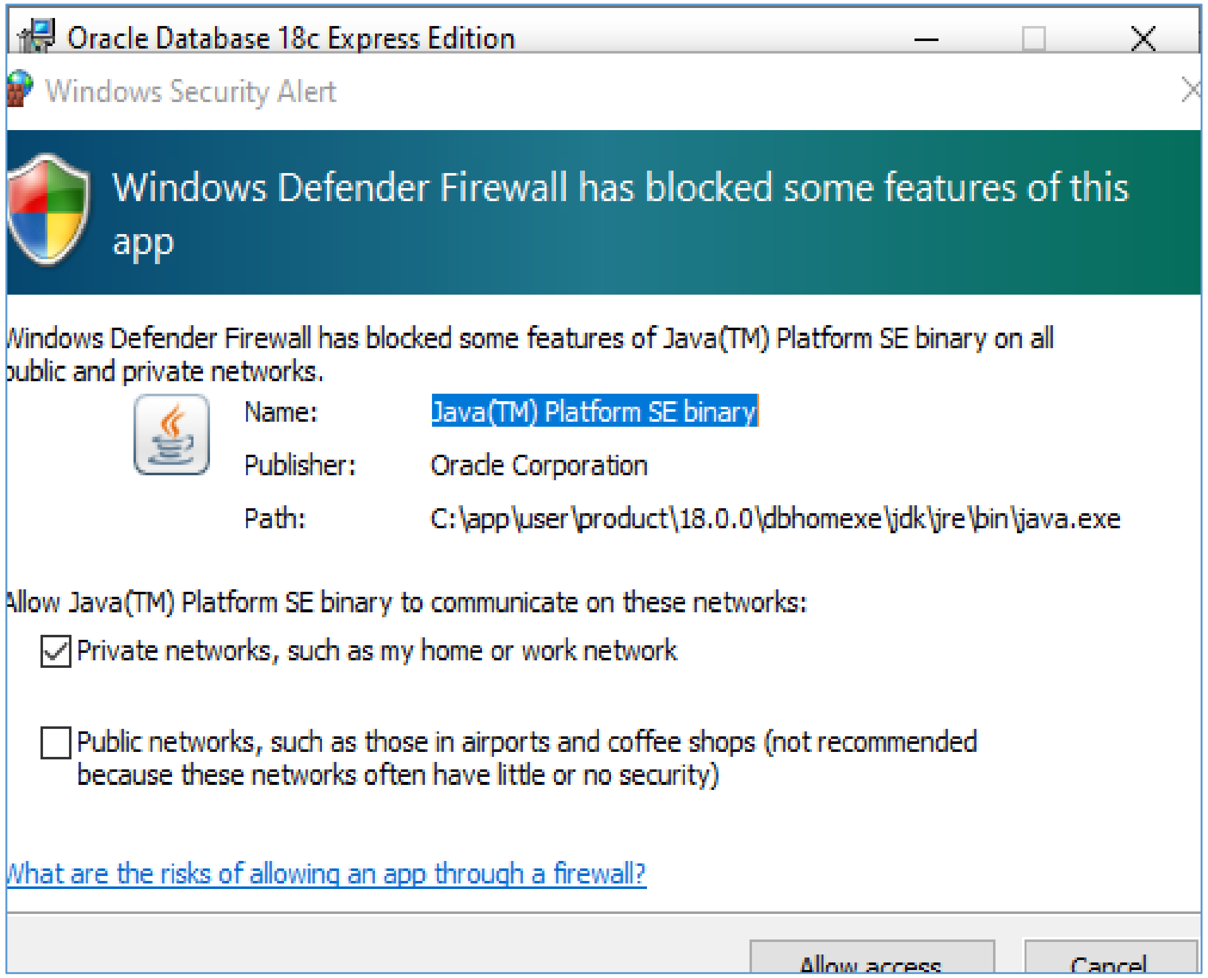
Destination Folder:	C:\app\user\product\18.0.0\
Oracle Home:	C:\app\user\product\18.0.0\dbhomeXE\
Oracle Base:	C:\app\user\product\18.0.0\

Click Install to begin the installation.

If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.

Step 11

If a message like this one pops, then click “Allow Access”


Oracle Database 18c Express Edition

Windows Security Alert

Windows Defender Firewall has blocked some features of this app

Windows Defender Firewall has blocked some features of Java(TM) Platform SE binary on all public and private networks.

Name: Java(TM) Platform SE binary

Publisher: Oracle Corporation

Path: C:\app\user\product\18.0.0\dbhomexe\jdk\jre\bin\java.exe

Allow Java(TM) Platform SE binary to communicate on these networks:

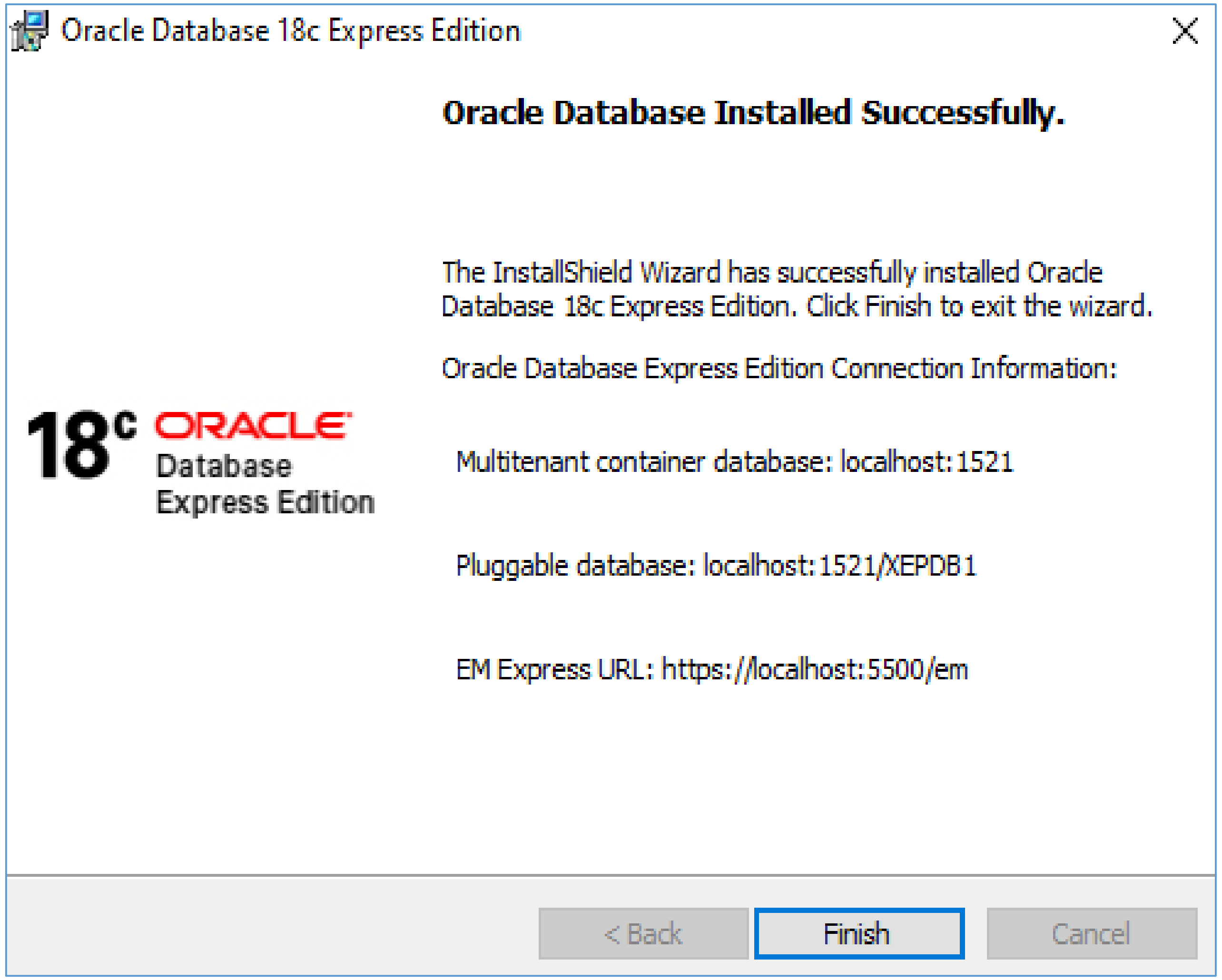
☑ Private networks, such as my home or work network

☐ Public networks, such as those in airports and coffee shops (not recommended because these networks often have little or no security)

What are the risks of allowing an app through a firewall?

Step 12

Click “Finish”


Oracle Database 18c Express Edition

Oracle Database Installed Successfully.

The InstallShield Wizard has successfully installed Oracle Database 18c Express Edition. Click Finish to exit the wizard.

Oracle Database Express Edition Connection Information:

Multitenant container database: localhost:1521

Pluggable database: localhost:1521/XEPDB1

EM Express URL: https://localhost:5500/em

Downloading SQL Developer

Step 1

- Go to SQL Developer download page
  - https://www.oracle.com/tools/downloads/sqldev-v192-downloads.html

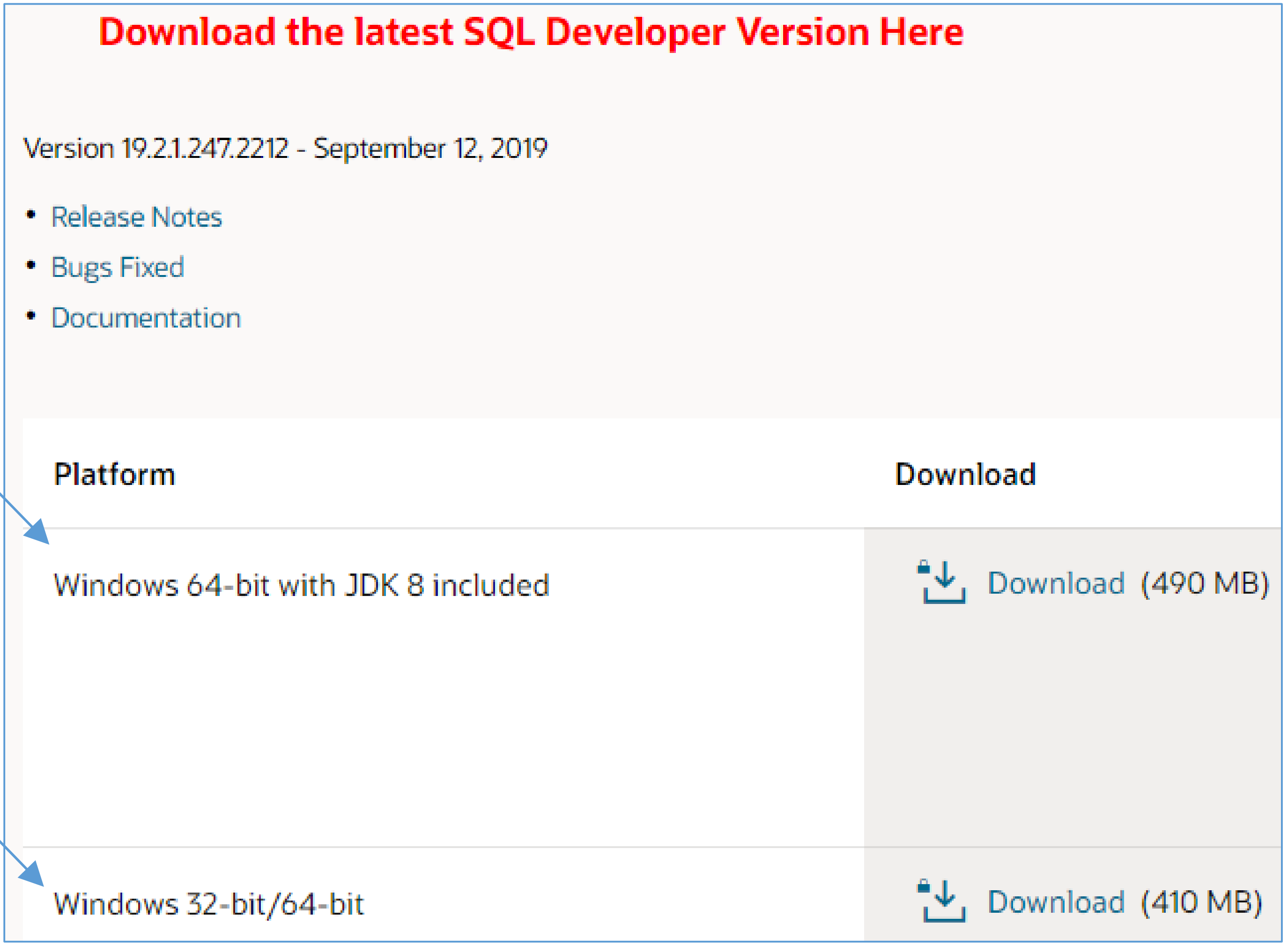
- Notice that the above link might change

Step 2

Install the version appropriate to your operating system.

Notice that the windows46 version already comes with jdk 8. But your windows must be 46bit

If you choose this one, then you should have jdk 8 or higher installed on your system before using sql developer



Download the latest SQL Developer Version Here

Version 19.2.1.247.2212 - September 12, 2019

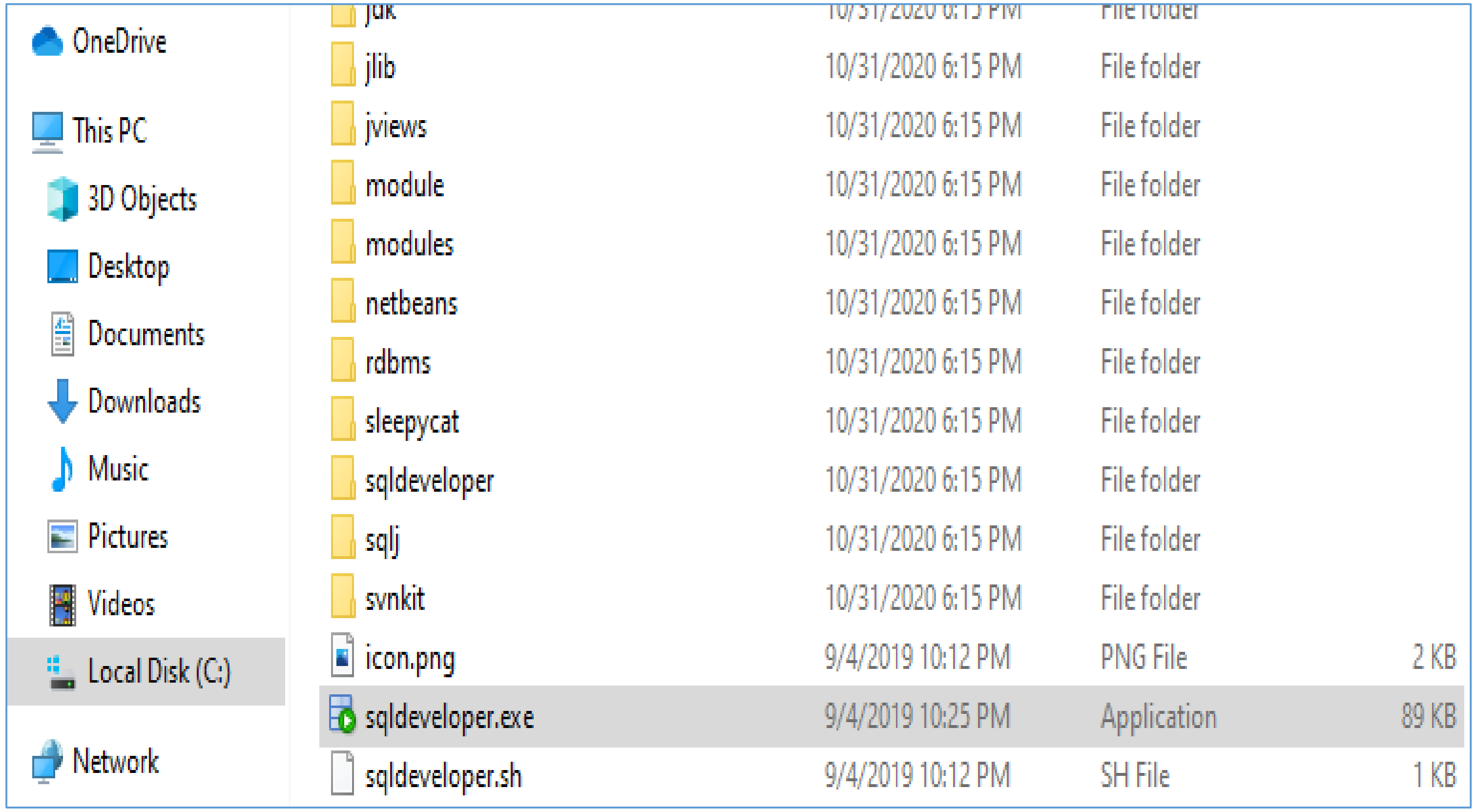
- Release Notes
- Bugs Fixed
- Documentation

Platform	Download
Windows 64-bit with JDK 8 included	Download (490 MB)
Windows 32-bit/64-bit	Download (410 MB)

Step 3

After downloading sql developer:

- Extract the downloaded file
- Get into the extracted folder
- Double click on "sqldeveloper.exe" to run it

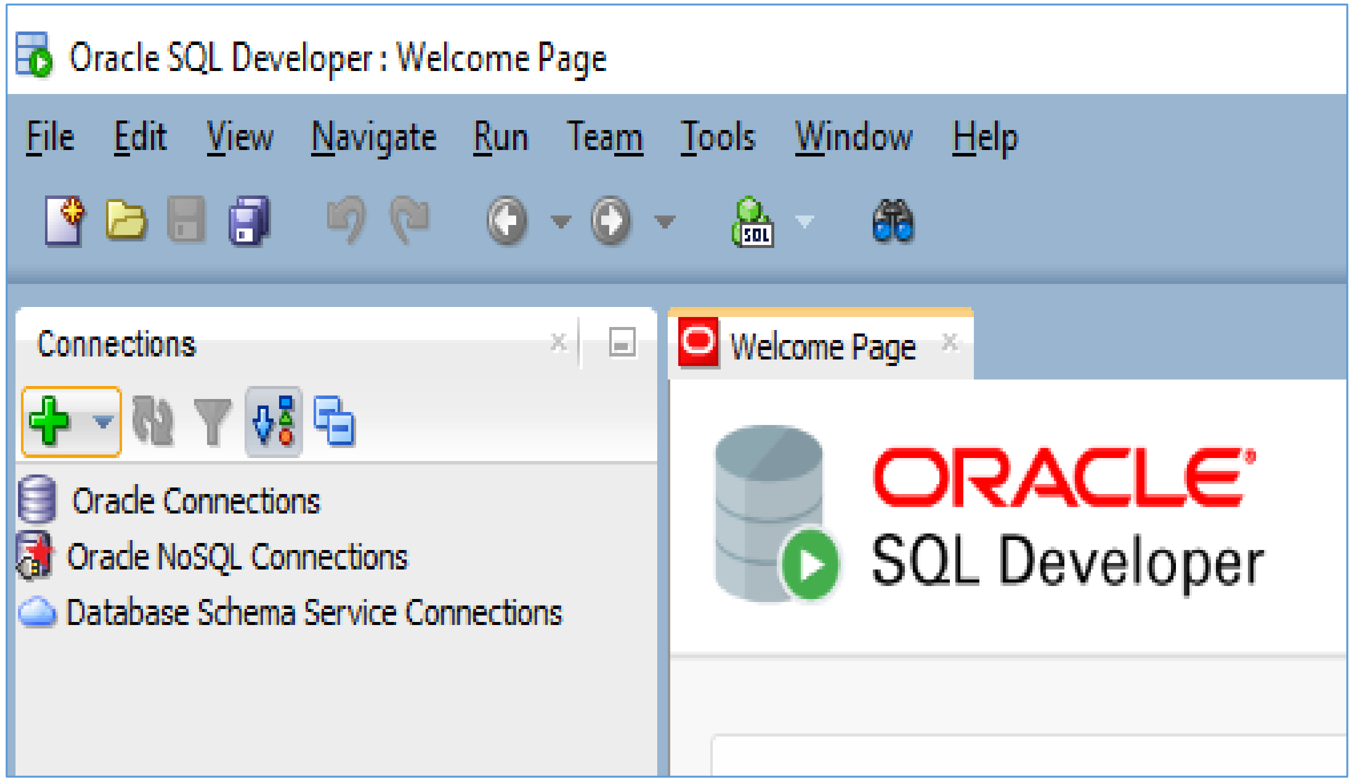


jlib	10/31/2020 6:15 PM	File folder	
jviews	10/31/2020 6:15 PM	File folder	
module	10/31/2020 6:15 PM	File folder	
modules	10/31/2020 6:15 PM	File folder	
netbeans	10/31/2020 6:15 PM	File folder	
rdbms	10/31/2020 6:15 PM	File folder	
sleepycat	10/31/2020 6:15 PM	File folder	
sqldeveloper	10/31/2020 6:15 PM	File folder	
sqlj	10/31/2020 6:15 PM	File folder	
svnkit	10/31/2020 6:15 PM	File folder	
icon.png	9/4/2019 10:12 PM	PNG File	2 KB
sqldeveloper.exe	9/4/2019 10:25 PM	Application	89 KB
sqldeveloper.sh	9/4/2019 10:12 PM	SH File	1 KB

Connecting SQL Developer to Oracle Express Edition

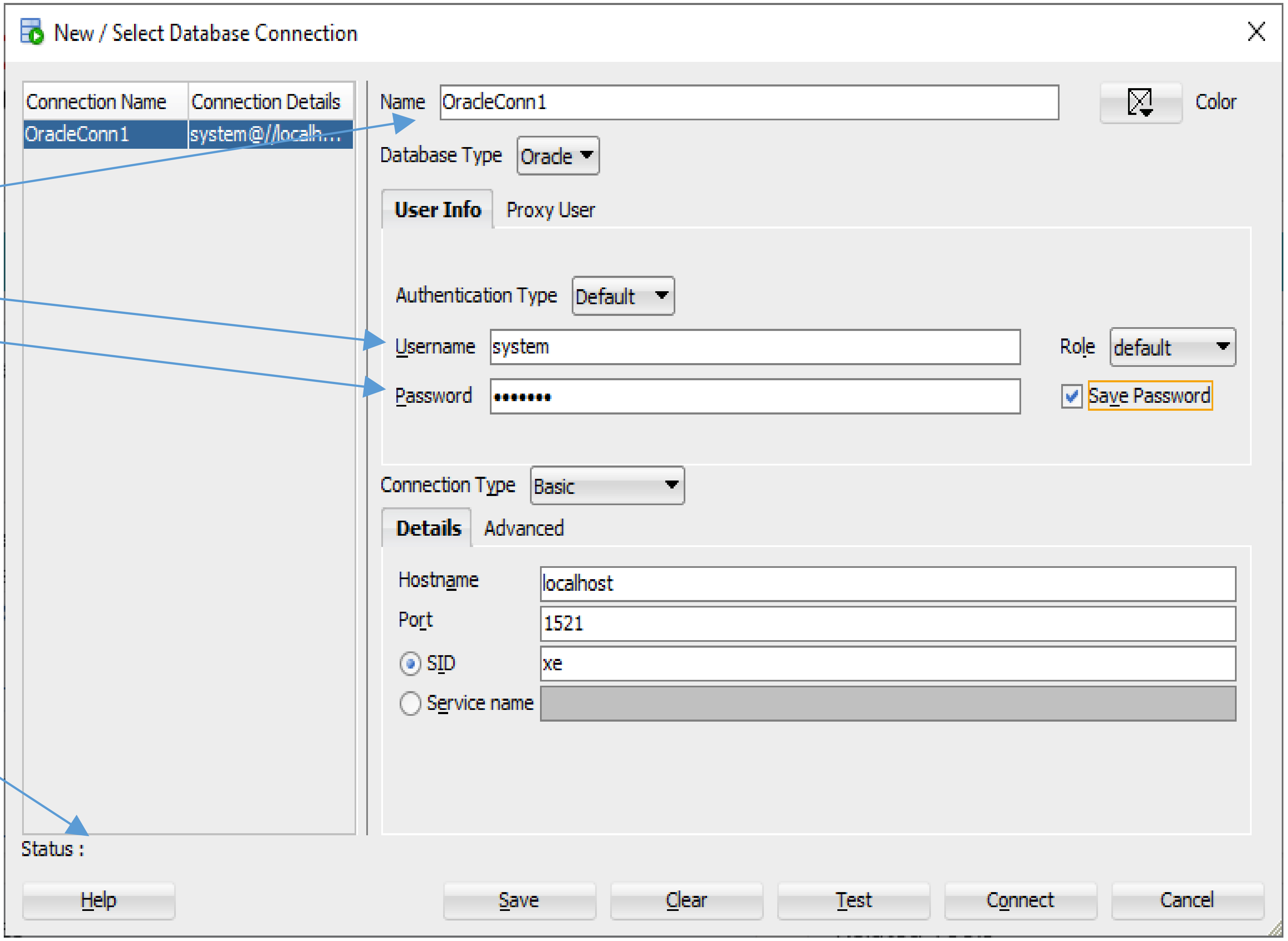
Step 1

- Open SQL Developer
- Click the plus button to create a new connection


Oracle SQL Developer : Welcome Page
File Edit View Navigate Run Team Tools Window Help
Connections
Oracle Connections
Oracle NoSQL Connections
Database Schema Service Connections
Welcome Page
ORACLE
SQL Developer

Step 2

- Enter any connection name
- Username: system
- Password: The one you chose when installing Oracle express edition
- Click “Test”
- Check the “save password” option
- If everything is OK, you will see a success message
- Click “Connect”



New / Select Database Connection

Connection Name	Connection Details
OracleConn1	system@//localh...

Name: OracleConn1
Database Type: Oracle
User Info
Authentication Type: Default
Username: system
Role: default
Password: •••••••
[✓] Save Password
Connection Type: Basic
Details
Hostname: localhost
Port: 1521
(●) SID: xe
( ) Service name
Status :

Lab1

- Database Languages
- Command Format
- Data Types

Database Languages

- Data Definition Language (DDL)
  - Defines DB structure or schema.
  - Example: Create, drop, and alter a table.
- Data Control Language (DCL)
  - Controls actions on DB objects.
  - Example: grant and revoke privileges.
- Data Manipulation Language (DML)
  - Deals with data in DB objects.
  - Example: retrieve, add, and delete data from a table.
- Transaction Control Language (TCL)
  - Groups statement together into one logical unit.

Data Definition Language (DDL)

- Statements which are used to define database schema or structure.

- Most famous statements:
  - Create: creates objects in DB.
  - Alter: changes structure of objects in DB.
  - Drop: Deletes objects from the DB.

- Example of Objects: tables, views, and indexes.

Command Format

- In all Lab material, SQL commands contain keywords and variables
  - Keywords will be written in **<u>lower case letters.</u>**
  - Variables will be **<u>preceded with</u>** **$** sign.

- Example:
  - Select $attribute From $table
    - 'Select' and 'from' are **keywords**.
    - 'attribute' and 'table' are **variables** that should be replaced with some value.

- We use this terminology so that you can distinguish keywords from variables.

Data Types

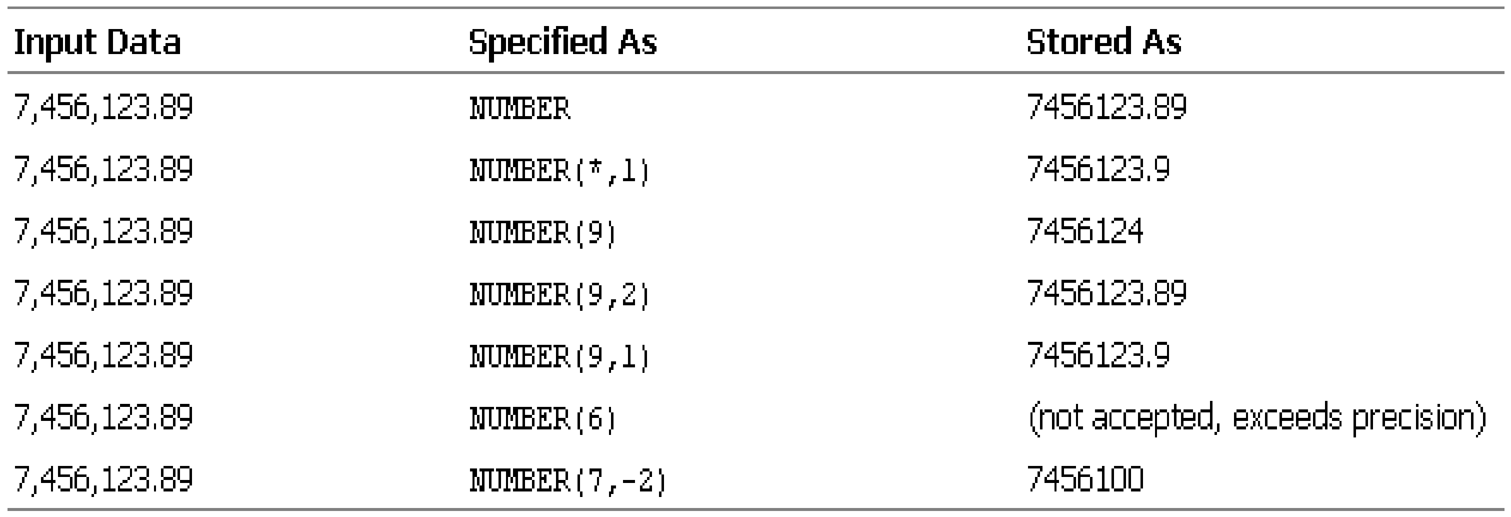
- char
  - Fixed length character strings.
  - Uses a number between 1-2000 (bytes).
  - Format: **char($size)**
  - Example: **char(20)**
  - If content is shorter, the remaining is filled with spaces.
  - If content is larger, error.

Data Types

- varchar2
  - Variable length character strings.
  - Uses a number between 1-4000 (bytes).
  - Format: **varchar2($size)**
  - Example: **varchar2(40)**
  - If content is shorter, the remaining of the total size is truncated to save space.
  - If content is larger, error.
  - Always use varchar2 instead of char unless you know your data has a fixed length.

Data Types

- number
  - Numeric data
  - Format: **number($precision, $scale)**
  - Precision: is the number of digits to the **<u>left</u>** of decimal point
  - Scale: is the number of digits to the **<u>right</u>** of decimal point



Input Data	Specified As	Stored As
7,456,123.89	`NUMBER`	7456123.89
7,456,123.89	`NUMBER(*,1)`	7456123.9
7,456,123.89	`NUMBER(9)`	7456124
7,456,123.89	`NUMBER(9,2)`	7456123.89
7,456,123.89	`NUMBER(9,1)`	7456123.9
7,456,123.89	`NUMBER(6)`	(not accepted, exceeds precision)
7,456,123.89	`NUMBER(7,-2)`	7456100

Note: This figure is taken from oracle.com

Data Types

- Date:
  - Stores Century, year, month, day, hour, minute, and second.
  - Each of the previous parts has 7 bytes reserved for it.
  - Each Oracle version has a default standard date format, this format can be **DD-MON-YY**, for example, ‘**21-Apr-80**’.
  - You can use function **to_date** to enter dates in a way different than standard format. For example: **to_date(‘April 21, 1980’, ‘MONTH DD, YYYY’).**

Default Date Format

- Default date format differs according to installed Oracle version

- This command shows the default date format

```
SELECT value FROM v$nls_parameters WHERE parameter ='NLS_DATE_FORMAT';
```

Lab 2

- Oracle Software
- User name and password
- Data Definition Language (DDL)
- Creating Tables
- Viewing description of a table
- Deleting Tables
- Exercise

Dr. Osama Al-Haj Hassan

Oracle Software

- During semester, we will use “Oracle 9i”.
- In “Oracle 9i”, we will only use “SQL Plus”.
- If “Oracle 9i” is installed on your windows system, you can run SQL Plus as follows (Path might differ based on installation):
  - Start Menu → All Programs → Oracle for Windows NT → SQL Plus 8.0
- SQL Plus is a command line window using which you can write commands to build and manipulate databases.

User Name and Password

- When you run SQL Plus, you will be prompted for a user name and a password, you can use one of the following:
  - UserName: system   Password: manager
  - UserName: Scott   Password: tiger

Setting Line Size

- Sometimes, the default line size is short.
  - So, output would be broken to multiple lines.

- To increase line size, you use this command:
  - set line $numOfChars;

- Example:
  - set line 300;

Spool Command

- If you do some work on SQL Plus and close it, then anything that appears on SQL Plus window will be lost.
- To save all what appears on SQL Plus window, you need to spool it to an external file using the following command.
- Spool $FilePath;
- Example:
  - Spool “c:/myfile.txt”;
- Now, everything that appears on SQL Plus window will be saved in “c:/myfile.txt”.

Data Definition Language (DDL)

- Used to change the structure of the database.

- Part of DDL is:
  - Creating tables.
  - Deleting tables.
  - Changing table definition.

Create Table

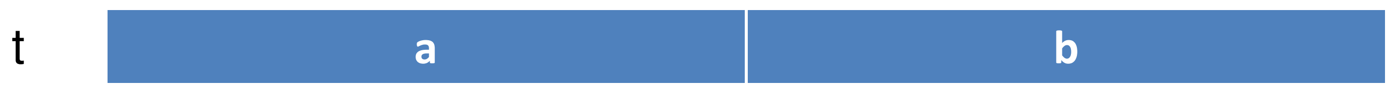
- Syntax:
  - create table $table_name

```
(
    $attribute_name1     $data_type,
    $attribute_name2     $data_type
 );
```

- You can have **more attributes** such that you provide a **comma** after each attribute definition.

Examples

-
  ```
  create table t (
      a   number,
      b   varchar2(10)
  );
  ```



Examples

-
  ```
  create table orders (
     order_id    number,
     order_date   date,
     customer_id      number,
     customer_name    varchar2(40)
  );
  ```

orders	**order_id**	**order_date**	**customer_id**	**customer_name**

View description of a table

- The ‘**desc**’ command outputs the description of the table which includes attribute names and their data types.

- Syntax:
  - desc $table_name;

- Example:
  - desc t;
  - desc orders;

Deleting Tables

- ‘drop table’ command deletes a table.

- Syntax:
  - drop table $table_name;

- Examples:
  - drop table t;
  - drop table orders;

Excersice

- Create the registration database illustrated in the following slides.

- Data type is given below each attribute name.

The “ed” Command

- Unfortunately, SQL Plus is not an easy interface to work with.
- It is not possible to move cursor forward, backward, top, or down. But, you can use backspace in one line.
- As a result, if you made a mistake while writing a command, it is not easy to fix it.
- A way to go around this is using the “ed” command.

Using “ed” Command

- Write the following command which contains a mistake.

```
create tab
(
```

- You realized that you made a mistake “tab” instead of “table”.
- End your command with a semicolon “;”.
- Now write the “ed” command.
  - ed;

Using "ed" Command

- Notepad window appears containing the command you wrote last.
- Also, there is a backslash "/" at the end of the command.

```
Create tab
(

/
```

- Correct your command and make sure to do the following:
  - Do **<u>not write</u>** a ";" at the end of your command.
  - Do **<u>not remove</u>** the backslash "/" character.
- Now close the Notepad window and click yes to save your changes.
- The corrected command will show up in the SQL Plus window.
- Now, write a semicolon ";" and hit enter to execute your command.

Registration Database

student

std_id	std_name	class	dep_id
number	varchar2(30)	number	number

course

crs_Id	crs_name	credit	dep_id
number	varchar2(30)	number	number

prerequisite

crs_id	prereq_id
number	number

department

dep_id	dep_name
number	varchar2(30)

Registration Database

instructor

ins_id	ins_name	dep_id
number	varchar2(30)	number

section

sec_id	crs_id	semester	year	ins_id
number	number	number	number	number

grade_report

std_id	sec_id	crs_id	semester	year	grade
number	number	number	number	number	number

Lab 3

- Alter Table
- Rename Table
- Data Control Language
    - Primary Key
    - Composite Key
    - Constraints

Dr. Osama Al-Haj Hassan

Alter Table

- Sometimes you create a table and you remember you need to change its definition because of
  - Mistake
  - Change in design
- To do that, you use command “alter table”.

Alter Table

- Things that can be done using alter table:
  - Adding a column or multiple columns.
  - Modifying an existing column or multiple columns.
  - Renaming an existing column or multiple columns.
  - Deleting a column or multiple columns.
  - Renaming a table.

Adding a column or multiple columns

- Adding one column:
  - Syntax:
    - alter table $table_name
      add $column_name $column-definition;

- Adding multiple columns:
  - Syntax:
    - alter table $table_name
      add ($column_name1 $column1-definition,
             $column_name2 $column2-definition
            );

    - You can add as many columns as you want.

Examples

- This is a table to work with.

- create table employee

```
(
        emp_id number,
        emp_name varchar2(30)
);
```

Examples

-
  ```
  alter table employee
  add salary number;
  ```

-
  ```
  alter table employee
  add ( address varchar2(30),
         gender number,
         birth_date date
        );
  ```

Modify an existing column or multiple columns

- Modifying one column:
  - Syntax:
    - alter table $table_name
      modify $column_name $description;

- Modifying multiple columns:
  - Syntax:
    - alter table $table_name
      ```
      modify($column_name1   $column1-definition,
             $column_name2   $column2-definition);
      ```

    - You can modify as many columns as you want.

Examples

- alter table employee
  modify emp_name varchar2(40);

- alter table employee
  modify(gender varchar2(1),
               emp_name varchar2(45)
              );

Deleting a column or multiple columns

- Deleting one column:
  - Syntax:
    - alter table $table_name
      drop column $column_name;

- Deleting multiple columns:
  - Syntax:
    - alter table $table_name
      drop ($column_name1,
      $column_name2);

    - You can delete as many columns as you want

Examples

- alter table employee
  drop column gender;

- alter table employee
  drop (address, salary);

Renaming an existing column

- Renaming an existing column is performed by:
  - First: Deleting the existing column.
  - Second: Adding it again with the new name.

Example

- In this example, I want to rename the “salary” column to “emp_salary”
  - alter table employee
    drop column salary;
  - alter table employee
    add emp_salary number;

Renaming a table

- Syntax:
  - alter table $table_name
    rename to $new_table_name;

- Example:
  - alter table employee
    rename to company_employee;

Data Control Language (DCL)

- It is used to:
  - Control actions on DB objects.
  - Example: grant and revoke privileges.
  - Creating keys is also considered part of DCL.
  - Creating constraints is also considered part of DCL.

Primary Key

- It is a key attribute(s) in an entity.
- It uniquely identifies an entity.
- It is called primary because one entity can have multiple candidate keys.
  - One of them is called “**Primary Key**”
  - The others are called “**Alternative Keys**” or “**Secondary Keys**”
- A primary key value are not allowed to be null.

Defining a primary key

1) Using create table statement (Definition 1)
   - Syntax:
     - Create table $table_name
       ```
       (
           $attribute1  $data_type  primary key,
           $attribute2  $data_type
       );
       ```
     - Here **attribute1** is the primary key of the table.
     - Example:
       ```
       create table employee
       (
           emp_id number primary key,
           emp_name varchar2(30)
       );
       ```

Defining a primary key

2) Using create table statement (Definition 2)

– Syntax:

–
```
Create table $table_name
(
    $attribute1  $data_type,
    $attribute2  $data_type,
    primary key($attribute1)
);
```

– Example:

```
create table employee
(
    emp_id number,
    emp_name varchar2(30),
     primary key (emp_id)
);
```

Defining a primary key

3) By adding a constraint on create table (Definition 3)

- Syntax:

```
Create table $table_name
(
    $attribute1  $data_type,
    $attribute2  $data_type,
    constraint $const_name primary key($attribute1);
);
```

- Example:

```
create table employee
(
    emp_id number,
    emp_name varchar2(30),
    constraint pk_const primary key (emp_id)
);
```

Composite Keys

- Sometimes primary keys might consist of more than one attribute.

- For example, in the case of “sections” table, the primary key of the table cannot be a single attribute. The key in this case is a combination of:
  - sec_id, crs_id, semester, year

- That is why it is called composite key.

Defining Composite Key

- You can use either definition 2 or 3 which you saw in the previous slides.
- Example:
  - create table sections

```
(
   sec_id number,
   crs_id number,
   semester number,
   year number,
   primary key(sec_id,crs_id,semester,year)
);
```

- How many keys table “sections” have? Only “1” key that is composed of 4 attributes.

Defining Composite Key

- You can use either definition 2 or 3 which you saw in the previous slides.

- Example:
  - create table sections

```
(
    sec_id number,
    crs_id number,
    semester number,
    year number,
    constraint c1 primary key(sec_id,crs_id,semester,year)
);
```

Note

- Look at the table description after you defined a primary key.
- Use the command desc $table_name
- Example: desc sections;
- You can see that a primary key attribute has “Not Null” written beside it.

Defining a primary key

- What if we forgot to define a primary key at the time of using “create table”?

- In this case, we can use “alter table” to define a primary key.

Defining a primary key

4) By adding a constraint using alter table command:

- Syntax:
  - alter table $table_name
    add constraint $const_name primary key($attribute1, $attribute2);
  - You can have as many attributes as you want.

Example

- create table employee

```
(
    emp_id number,
    emp_name varchar2(30)
);
```

- alter table employee

```
add constraint c2 primary key (emp_id);
```

Lab 4

- Data Control Language
    - Primary Key
    - Composite Key
    - Constraints

- DML
    - insert statement
    - update statement
    - delete statement
    - predicates

Prepared By: Dr.Osama Al-Haj Hassan

Removing a primary key?

- Removing a primary key is done by removing the constraint that is used to define it in the first place.

- How can you remember the name of the constraint that you used to define your primary key?

- You can look it up from view “user_constraints”.

View “user_constraints”

- It is a view that contains information about all constraints on all tables.

- Important attributes of the view:
  - constraint_name
  - table_name

Example

-
```
create table employee
(
        emp_id number primary key,
        emp_name varchar2(30)
);
```

- In table employee, what is the name of the constraint that represents the primary key????

Example (Continue)

- `select constraint_name, table_name from user_constraints where table_name= 'EMPLOYEE' ;`

Upper Case

- Output returned by this statement:

```
  CONSTRAINT_NAME              TABLE_NAME
  ---------------------------- ----------------------------
        SYS_C00595               EMPLOYEE
```

- If you want to use object name between single quotations, it has to be written in “upper case” letters.

Example (Continue)

- Now, we know the primary key constraint for employee table is “SYS_C00595” because it is the only constraint defined on the table.
- In case many constraints exist on the same table, you should remember the name of the constraint that represents the primary key.
- Now, we can remove the constraint and therefore remove the primary key.
- `alter table employee drop constraint SYS_C00595;`

Null Values

- You can have a situation where you have an attribute that is not a key, but you do not permit NULL values for it.

- How can you specify that an attribute should not have NULL values?

Null Values

- You can specify that while creating a table:
  - Example:

  - create table employee

```
(
        emp_id number primary key,
        emp_name varchar2(30) not null
);
```

  - This is logical because each employee must have a name.

Null Values

- If you forget to specify “Not Null” for an attribute while creating a table, you can use alter table to change that.

- Example:

```
-alter table employee
 add address varchar2(30);

 - alter table employee
   modify address varchar2(30) not null;
```

Constraints

- We saw a few examples of constraints so far:
  - Creating a primary key constraint.
  - Deleting a primary key constraint.

- There is another type of constraints called “Check Constraints”.
  - It is used to check on the value of an attribute.

- You can add a constraint in
  - Create table command or
  - Alter table command

Defining Check Constraints

- Syntax1:
  - create table $table_name
    ```
    (   $attribute1    $data_type,
        $attribute2   $data_type,
        constraint   $constraint_name   check($condition)
    );
    ```
  - Example:
    ```
    create table student
    (
         std_id number,
         std_gpa number,
         constraint    ccc1   check(std_gpa>0)
    );
    ```

Defining Check Constraints

- Syntax2

  -
    ```
    alter table $table_name
     add constraint $constraint_name check($condition);
    ```
  - Example:

    -
      ```
      alter table student
      add constraint ccc1 check(gpa>0);
      ```

Check Constraint Examples

- If grade is defined as varchar2(1):

  -
    ```
    alter table grade_report
    add constraint grade_cc
    check(grade in ( ‘A’ ,’ B’ ,’ C’ ,’ D’ ,’ F’ ));
    ```

- If grade is defined as number:

  -
    ```
    alter table grade_report
    add constraint grade_cc
    check(grade>=35 and grade<=100);
    ```

Check Constraint Examples

- If grade is defined as number:

  -
    ```
    alter table grade_report
    add constraint grade_cc
    check(grade between 35 and 100);
    ```
  - Note: the previous example is equivalent to "grade>=35 and grade<=100"

-
  ```
  alter table employee
  add constraint cons2
  check(salary+commision<3000);
  ```

Deleting a constraint

- You can delete a check constraint in the same way you delete a primary key constraint:
  - Example:

    -
      ```
      alter table student
      drop   constraint   ccc1;
      ```

Data Manipulation Language (DML)

- It is used to manipulate the content of a DB.

- Examples:
  - Insert a record.
  - Update a record.
  - Delete a record.
  - Retrieve records from a table.

A Table to work with

-
  ```
  create table employee
  (
          id number primary key,
          name varchar2(30),
          address varchar2(30),
          birth_date date
  );
  ```

Retrieving “all” records in a table (Basic Select Statement)

- Syntax:
  - `Select * from $table-name;`
- Syntax:
  - * means “all attributes” in the table.
- Example:
  - `Select * from employee;`
  - `Select * from student;`

Inserting records into a table

- Syntax 1:
  - `Insert into $table-name values($val1,$val2,…);`
  - Here, you have to have values for all attributes in the same order they appear in table definition.

- Example:
  - `Insert into employee values(1,’kamal’,’Jbaiha’,’21-Jan-80’);`

Inserting records into a table

- Syntax 2:
  - `Insert into $table-name ($col1,$col2,…) values($val1,$val2,…);`
  - Here, you can choose what attributes(columns) to enter.
  - Use values (val) in the same order that corresponds to columns (col).
- Example:
  - `Insert into employee (id, birth_date) values(1,’21-Jan-80’);`

Inserting records into a table

- If you define a primary key for a table, then you must provide a primary key value when adding a new entity to the table.

- Example:
  - `Insert into employee (name, birth_date) values( ‘kamal’ ,’ 21-Jan-80’ );`
  - This insert statement is invalid because it treats “id” as Null. “id” is the primary key of table employee. It cannot be Null.

Inserting records into a table

- If the primary key is a sequential number automatically assigned by DBMS (Auto_Increment), then you do not provide a value for it in the insert statement.

Updating Records in a Table

- Syntax:

  -
    ```
    Update $table-name
    set $col1=$val1, $col2=$val2
    where $predicate;
    ```
  - You can have as many columns as you want.
  - Predicate is a condition on which records are updated.

Updating Records in a Table

- Examples:

  -
    ```
    Update employee
    set name= ‘kamal’
    where id=1;
    ```
  - In this example, for the employee with ID =1, we change his name to ‘kamal’.

Updating Records in a Table

- Examples:

  -
    ```
    Update employee
    set name= ‘Salem’ , address= ‘Jbaiha’
    where id=3;
    ```
  - In this example, for the employee with ID =3, we change his name to ‘Salem’ and his address to ‘Jbaiha’.

Updating Records in a Table

- Examples:
  -
    ```
    Update employee
    set address= ‘Jbaiha’ ;
    ```
  - In this example, for all records (employees), we change their address to ‘Jbaiha’.

Deleting Records in a Table

- Syntax:
  - `Delete from $table-name`
    `where $predicate;`
  - Predicate is a condition on which records are deleted.
- Example:
  - `Delete from employee where id =1;`
  - This example deletes the record that represents employee who has id = 1

Deleting Records in a Table

- Examples:
  - `Delete from employees where id>7 or name= ‘Ahmad’ ;`
  - This example delete records that represent employees with id>7 or their name is ‘Ahmad’
- Deleting all records from a table:
  - `Delete from employees;`

“And” “OR” in predicates

- Predicates can be complex, we can add a mix of predicates using “and/or”.

- Here, you should keep attention to operator precedence.

- Example:
  - name=‘Ahmad’ and id=1;
  - Address=‘jbaiha’ or name=‘ahmad’
  - Id=1 and address=‘jbaiha’ or address=‘amman’

Operator Precedence

- Priorities are from highest to lowest are treated as follows:
  - Parentheses

    ()
  - Arithmetic - multiplication related

    * , / , %
  - Arithmetic - addition related

    + , -
  - Logical – NOT
  - Logical – AND
  - Logical - OR

Operator precedence in Predicates

- Note that the order of operators execution may make the result of the query different.

- Example:
  - Suppose you have the following table

Emp_work

empno	payrate	Dept_code
1	18	Sales
2	20	Sales
3	18	Adver
4	20	QA

Operator precedence in Predicates

- Example (Continue):
  - The result of:
    - Select * from emp_work where dept_code='sales' and payrate=18 or payrate=20;

empno	payrate	Dept_code
1	18	Sales
2	20	Sales
4	20	QA

  - The result of:
    - Select * from emp_work where dept_code='sales' and (payrate=18 or payrate=20);

empno	payrate	Dept_code
1	18	Sales
2	20	Sales

Lab 5

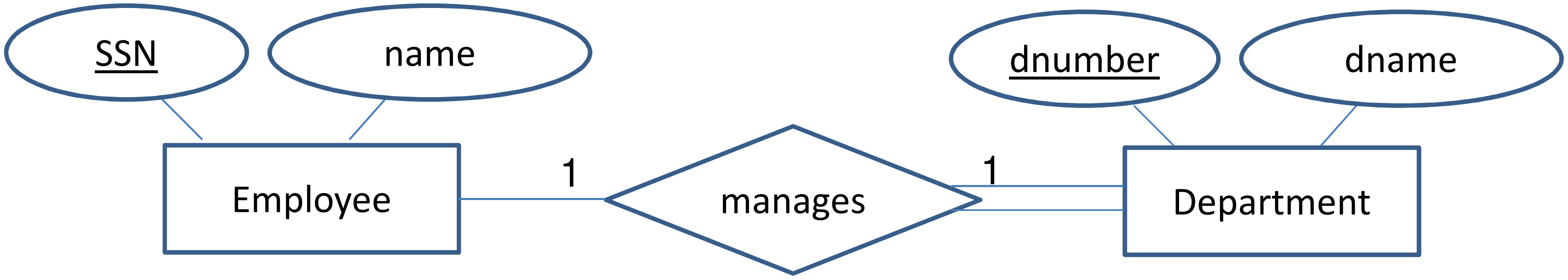
- Foreign keys
- Using “unique” to define secondary keys
- Referential Integrity
- Inserting records in tables involved in PK/FK relationship
- Deleting tables that are involved in PK/FK relationship
- Select statement

Foreign Key

- A foreign key is an attribute in table “X” that refers to a primary key in table “Y”.

- It is used to represent relationships in tables.
  - Entity 1 is represented as a table
  - Entity 2 is represented as a table
  - How do you connect the two tables (relationship)
    - Using Foreign Keys

Example 1 (1:1 Relationship)



- How Can you represent relationship "manages" in the DB?
- Put the Primary Key of "one" of the entities into the "other one".

Employee

<u>SSN</u>	**name**
1	Ahmad
2	Kamal
3	Sara

Primary Key

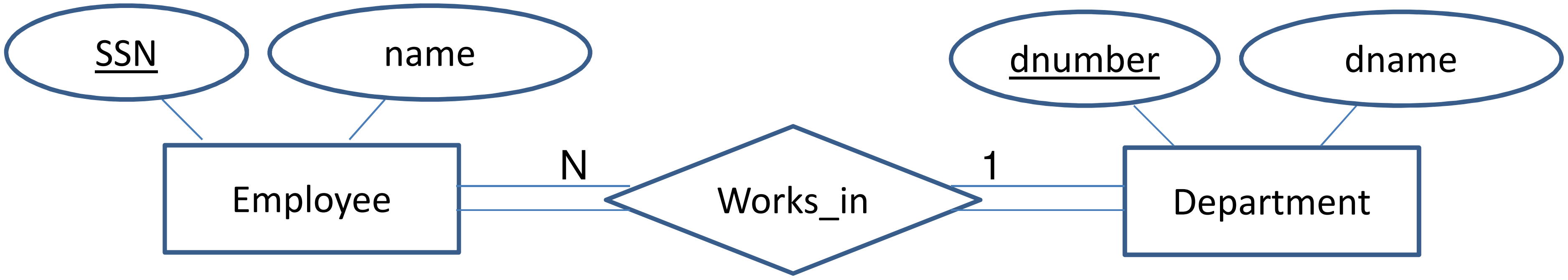
Department

<u>dnumber</u>	**dname**	**mgrSSN**
1	CS	3
2	QA	2

Primary Key

Foreign Key

Example 2 (1:N Relationship)



- How Can you represent relationship "works_in" in the DB?
- Put the Primary Key of the "1" side in the "many" side.

Employee

SSN	name	dno
1	Ahmad	2
2	Kamal	1
3	Sara	1

Department

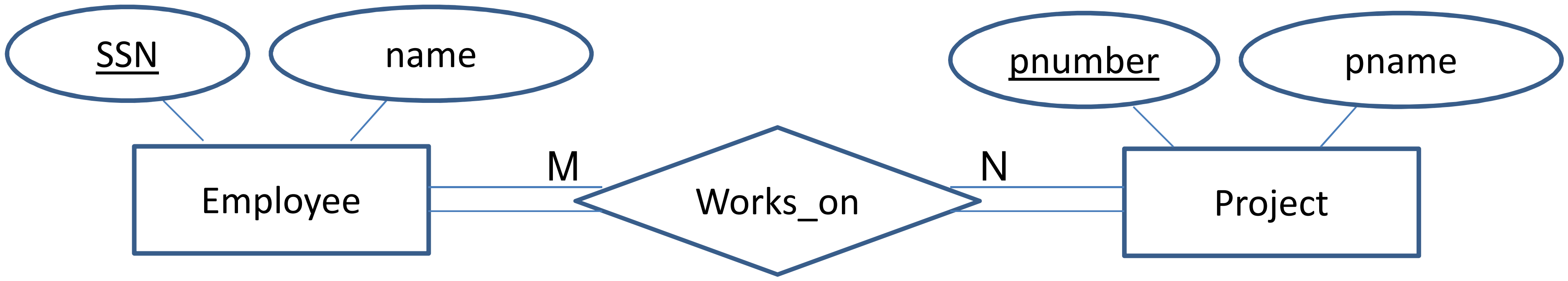
dnumber	dname
1	CS
2	QA

Primary Key

Foreign Key

Primary Key

Example 3 (M:N Relationship)



- How Can you represent relationship “works_on” in the DB?
- Put the Primary Key of the 2 entities in a separate table.

Employee

<u>SSN</u>	name
1	Ahmad
2	Kamal
3	Sara

Primary Key

Works_on

SSN	pno
1	10
1	30
2	20
3	30

Foreign Key　　Foreign Key

Project

<u>pnumber</u>	pname
10	Abdali
20	Jaleel
30	Shaheq

Primary Key

Note

- There are more rules for mapping ER to relations (tables).
- We will study them in the theoretical lectures.

Important Note

- The **value** of a foreign key **must be** a value that **exist** among the **values** of the primary key it is related to.

Defining Foreign Keys

- Syntax 1:
  - Create table $table_name

```
(
    $attr_name  $datatype,
    $attr_name  $datatype,
    $attr_name  $datatype references $PK_table ( $PK_Attr )
);
```

Foreign Key Table (Referencing Table)

Foreign Key

Primary Key

Primary Key Table (Referenced Table)

Example

- Suppose you have table department:

  -
    ```
    create table department
    (
            dnumber number primary key,
            dname varchar2(30)
    );
    ```
- Create a table employee which relates employee to the department he is working on.

Example (Continue)

```
create table employee
(
    ssn number primary key,
    name varchar2(30),
    dno number references department(dnumber)
 );
```

Defining Foreign Keys

- Syntax 2:

```
Create table $table_name
  (
 $attr_name  $datatype,
 $attr_name  $datatype,
 $attr_name  $datatype,
 foreign key ($attr-name) references $PK_table ($PK_Attr)
 );
```

Foreign Key Table (Referencing Table)

Foreign Key

Primary Key

Primary Key Table (Referenced Table)

Example

- Using syntax 2 in the previous example :

```
create table employee
(
      ssn number primary key,
      name varchar2(30),
      dno number,
      foreign key(dno) references department(dnumber)
);
```

Defining Foreign Keys

- Syntax 3:

```
Create table $table_name
(
    $attr_name  $datatype,
    $attr_name  $datatype,
    $attr_name  $datatype,
    constraint $const_name foreign key ($attr-name) reference $PK_table($PK_Attr)
);
```

Example

- Using syntax 3 in the previous example :

```
create table employee
(
     ssn number primary key,
     name varchar2(30),
     dno number,
     constraint   fk1  foreign key(dno) references department(dnumber)
 );
```

Defining foreign key

- If you did not define foreign key in create table statement, you can do that in alter table statement.

- Syntax:

```
alter table $table_name
add constraint $const_name
foreign key ($attr-name) references $PK_table($PK_Attr);
```

Example

- Suppose we did not define “dno” as a foreign key in employee.

```
alter table employee
add constraint pck2
foreign key (dno) references department(dnumber);
```

Recommendation

- The best way to create tables with foreign keys is by:
  - First, create tables with foreign key **attributes**.
  - Second, **connect** foreign keys to primary keys using **alter table**.

Defining Foreign Keys

```
Create table department
(
 dnumber number primary key,
 dname varchar2(30)
);
```

Command 1

```
Create table project
(
  pnumber number,
  pname varchar2(30),
  dnum number,
  foreign key(dnum) references department(dnumber)
);
```

Command 2

Question: Can we execute command 2 before command 1?
Answer : No, because we are referencing "dnumber" in table department before it actually exists.

This is why I recommend connecting foreign keys to primary keys in a separate "Alter Table" step.

Unique Attributes

- Suppose you want to define more than one key for a table.
- One of them is “**primary key**”.
- The rest are “**Secondary**” or “**Alternative**” keys.
- You define a secondary key using “**unique**” keyword.
- “unique” means: Its value cannot be repeated in the table (different for each record).

Example 1

- Create table project
```
(
        pnumber  number primary key,
        pname varchar2(30) unique
);
```

Example 2

- The previous example can be done this way:

-
  ```
  Create table project
   (
            pnumber number primary key,
            pname varchar2(30),
            constraint con49 unique(pname)
  );
  ```

- If you forgot to define secondary keys in create table statement, you can do that using alter table statement by adding a constraint.

- You can remove a foreign key by removing its constraint.

Defining Foreign Key

- A foreign key can be formed of “several attributes” (composite) and they refer to “several attributes” in primary key table.

Example

```
- Create table X1
  (
     a1 number primary key,
     a2 varchar2(10) unique,
     constraint con36 unique(a1,a2)
  );
- Create table X2
  (
     a3 number,
     a4 varchar2(10),
     foreign key (a3,a4) references X1(a1,a2)
  );
```

Unique(a1,a2)
Means that the
Combination of
(a1,a1) is unique,
So it is not repeated
In the table

Referential Integrity

- Referential Integrity is making sure the value of a foreign key attribute in one table:
  - exists in the primary key attribute of the other table. Or
  - It equals “Null”.
- Suppose dno is a foreign key that references department dnumber.

✓

Employee

ssn	name	dno
1	Ahmad	1
2	Kamal	1
3	Salma	2

Department

dnumber	Dname
1	CS
2	CIS

✗

Employee

ssn	name	dno
1	Ahmad	1
2	Kamal	3

There is no department with Dnumber =3

Inserting records in tables that are involved in a PK/FK relationship

- Because of referential integrity, the value of a FK must exist among values of PK it is related to. (Unless Fk = Null).
- Example:

```
Create table department
(
  dnumber number primary key,
  dname varchar2(30)
);
```

```
Create table project
(
 pnumber number,
 pname varchar2(30)
 dnum number,
 foreign key(dnum) references department(dnumber)
);
```

- Suppose that the department table is empty. Can we execute the following command:
  - `Insert into project values(10,’ Shaheq’ , 1);`

Inserting records in tables that are involved in a PK/FK relationship

- **No we cannot**, because the project belongs to department “1”, but department “1” does not exist yet (Referential Integrity)
- So, we can do this in any of the following ways:
  1) First, insert department “1” information into table department, then insert project “10” information into project table.

```
insert into department values(1,’ QA’ );
insert into project values(10,’ Shaheq’ , 1);
```

2) Insert project “10” information while having dno value as null. When department “1” information is added, you update project table by setting the dno value to “1”.

```
insert into project values(10,’ Shaheq’ , null);
insert into department values(1,’ QA’ );
Update project set dno=1 where pnumber=10;
```

Note

```
insert into project values(10,’ Shaheq’ , null);
```

These two are equivalent

```
insert into project (pnumber,pname) values(10,’ Shaheq’ );
```

Deleting tables that is involved in a PK/FK relationship

- Important Rules:
  - You **can delete** the table that has the **FK** (Referencing Table).
  - You **cannot delete** the table that has the **PK** which is **connected to the FK** (Referenced Table).
    - You can do this only after you delete the PK/FK connection.

Example 1

```
Create table department
(
    dnumber number primary key,
    dname varchar2(30)
);
```

```
Create table project
(
    pnumber number,
    pname varchar2(30),
    dnum number
);
```

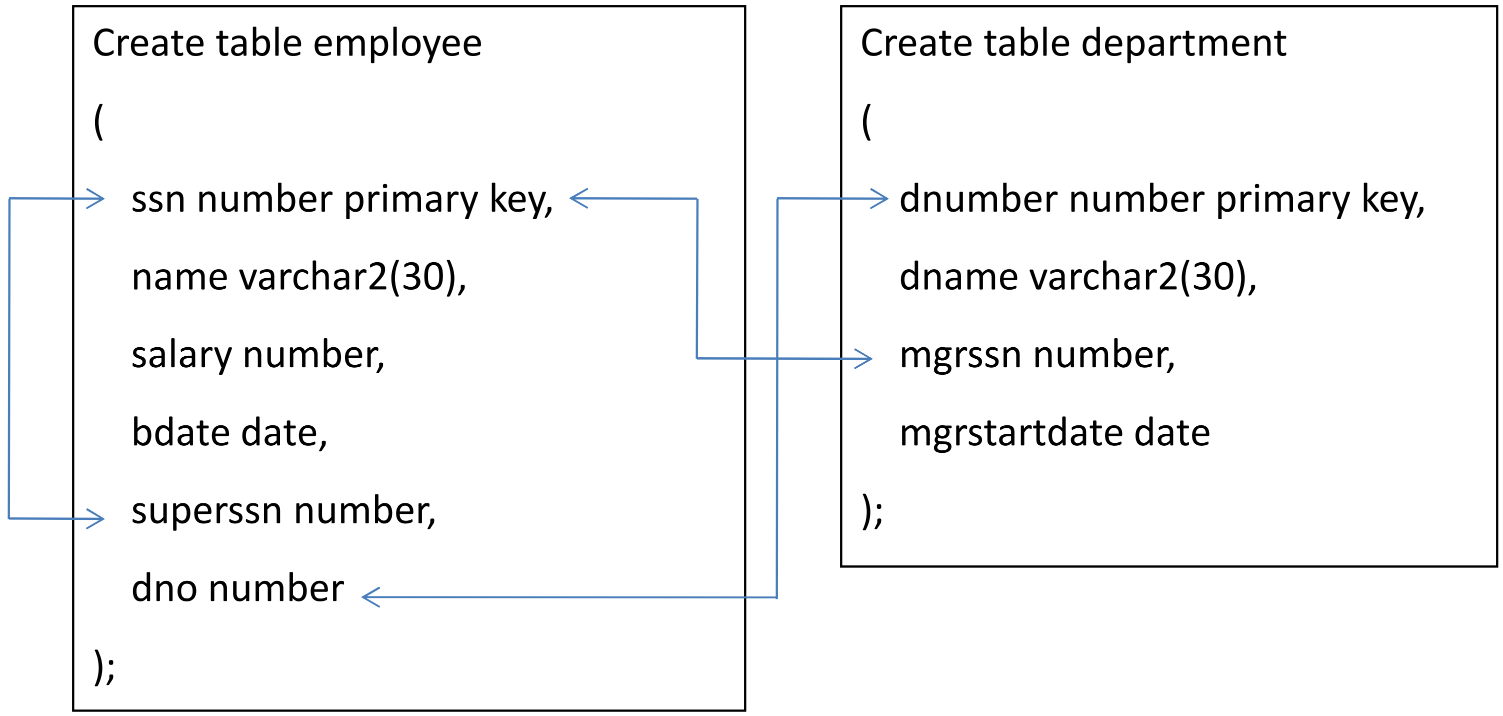
```
alter table project
 add constraint fk1 foreign key(dnum) references department(dnumber);
```

Question: Can we **directly** delete table department?
Answer : No, because it has a PK (dnumber) which is **connected to** foreign key in table project (dnum)

Question: How can we delete table department?
Answer : remove the PK/FK relationship, then delete table department

```
alter table project drop constraint fk1;
drop table department;
```

Example 2



```
Create table employee
(
    ssn number primary key,
    name varchar2(30),
    salary number,
    bdate date,
    superssn number,
    dno number
);
```

```
Create table department
(
    dnumber number primary key,
    dname varchar2(30),
    mgrssn number,
    mgrstartdate date
);
```

```
alter table employee
add constraint fk1 foreign key(superssn) references employee(ssn);
add constraint fk2 foreign key(dno) references department(dnumber);
```

```
alter table department
add constraint fk3 foreign key(mgrssn) references employee(ssn);
```

Example 2 (Continue)

Question: Can we **directly** delete table department?
Answer : No, because it has a PK (dnumber) which is connected to foreign key in table employee.

Question: Can we **directly** delete table employee?
Answer : No, because it has a PK (ssn) which is connected to foreign key in table department (mgrssn).
Also, no, because it has a PK (ssn) which is connected to foreign key in itself (superssn).

Question: How can we delete table department, then delete table employee?
Answer : remove the PK/FK relationship, then delete tables

```
alter table employee drop constraint fk2;
drop table department;
alter table employee drop constraint fk1;
drop table employee;
```

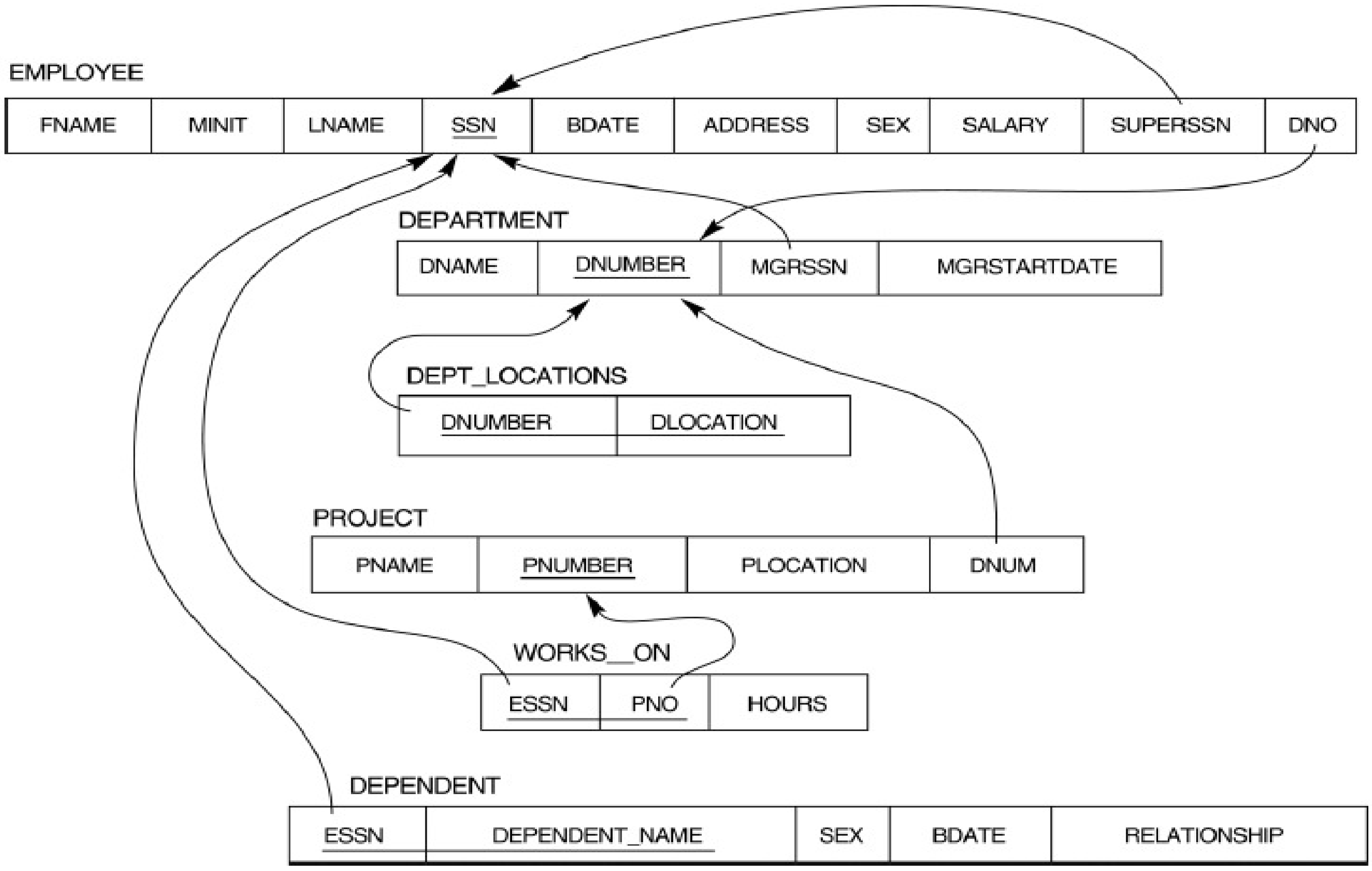
Query

- In English: Query means to ask for something.
- In Databases:
  - Querying (Asking) the database for records in tables.
- A “**select**” statement is considered a query.

Select Statement

- Used to retrieve data from tables.
- It does not change the content of tables.
- It consists of many parts:
  - “Select” & “From” clauses , they are mandatory.
  - “Where” clause.
  - “order by” clause.
  - “group by” clause.
  - “having” clause.

Company DB represented in Relational Model



“Select” & “From” clauses

- It is a mandatory part in select statement.

- Syntax 1:
  - Select * From $table_name1, $table_name2, …;
    - * means: “all attributes”.
- Syntax 2:
  - Select attribute1, attribute2, …
    from $table_name1, $table_name2, …;

- Examples:
  - select * from employee;
  - select ssn, name from employee;

“where” clause

- It is used to indicate what “records” we want to retrieve.

- Syntax:
  - `where $predicate`

- Predicate is a condition on which we specify what records we need.

Examples

- `select name, salary from employee where ssn=1;`
  - Here, we get the record for employeeswho’s ssn=1

-
  ```
  Select ssn, name from employee
  where salary >= 500 and salary <= 1000;
  ```

-
  ```
  Select ssn, name from employee
  where salary between 500 and 1000;
  ```

“order by” clause

- Used to order records retrieved in select statement.
- The default ordering is “**<u>ascending</u>**”.

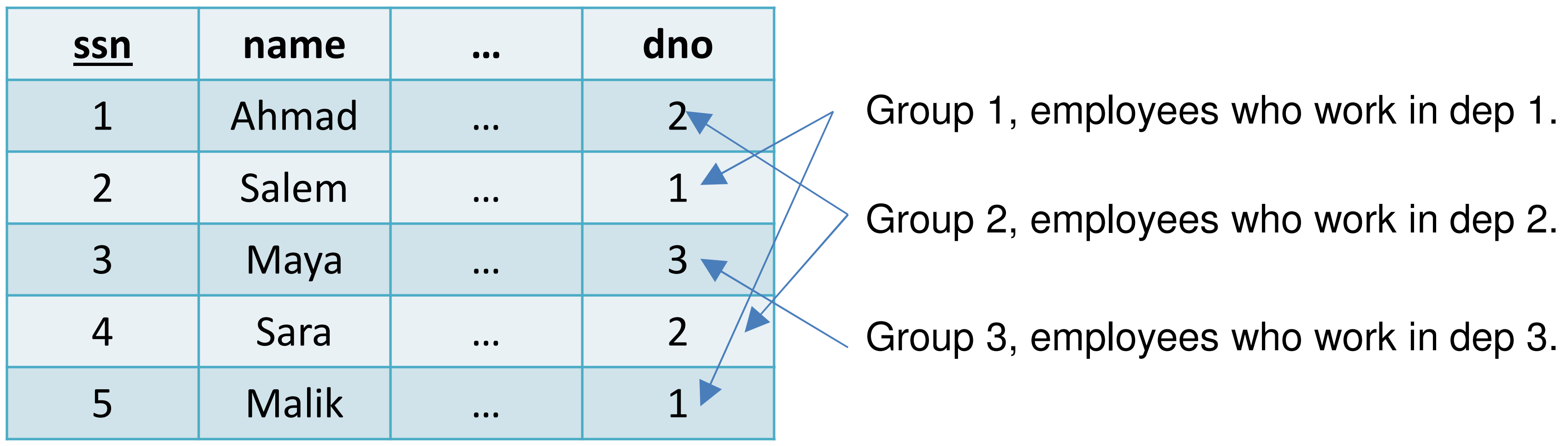
- Syntax:
  - `Order by $attr1, $attr2, …`
  - `Order by $attr1 desc`
  - `Order by $attr1 asc`

- Examples:
  - `select * from employee order by salary;`
  - `select * from employee order by dno, salary;`
  - `select * from employee order by salary desc;`
  - `select * from employee order by dno desc, salary asc;`

“group by” Clause

- “group by” is used to group records based on a common attribute.

- Example: Based on dno.



<u>ssn</u>	name	…	dno
1	Ahmad	…	2
2	Salem	…	1
3	Maya	…	3
4	Sara	…	2
5	Malik	…	1

Group 1, employees who work in dep 1.

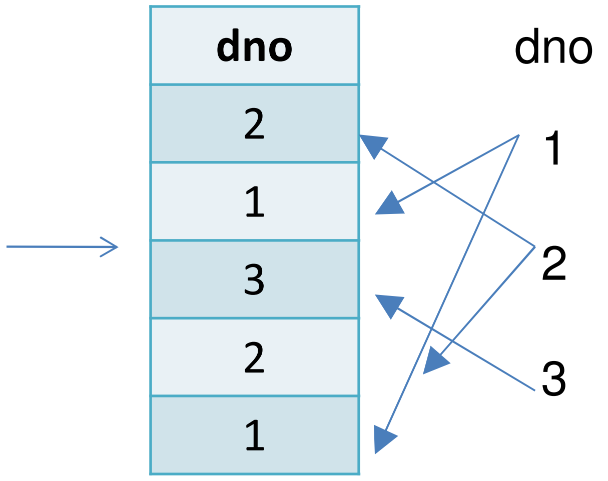
Group 2, employees who work in dep 2.

Group 3, employees who work in dep 3.

Example

- List department numbers in which employees work. Each department number should appear 1 time.
  - select dno from employee group by dno;

ssn	name	…	dno
1	Ahmad	…	2
2	Salem	…	1
3	Maya	…	3
4	Sara	…	2
5	Malik	…	1



“group by” clause

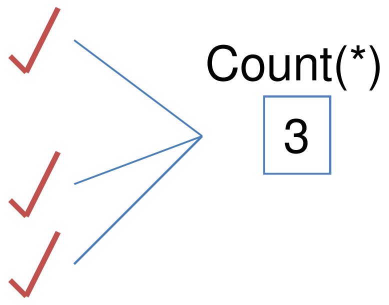
- “Group by” is also used with “aggregate” functions.
- Aggregate functions take many items as input and produce a single value as output.
- Important aggregate functions:
  - Sum, max, min, average, STddev.
- Aggregate functions work on groups. This is why using “group by” with aggregate functions is beneficial.

Example (without group by)

- Find number of employees with salary > 400
  - `select count(*) from employee where salary > 400;`

- Here, the table employee is one group, the previous statement counts how many records satisfy the condition "salary > 400".

<u>ssn</u>	name	salary	…	dno
1	Ahmad	356	…	2
2	Salem	500	…	1
3	Maya	333	…	3
4	Sara	405	…	2
5	Malik	734	…	1

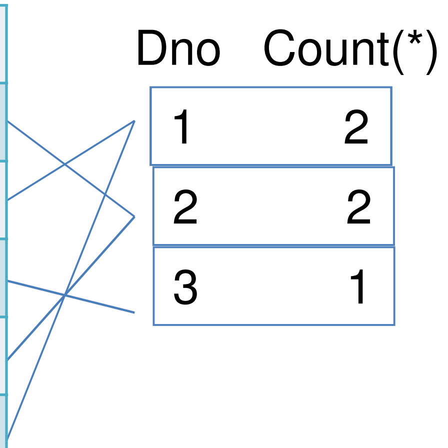


Example

- Find department number and number of employees in that department
  - `select dno, count(*) from employee group by dno;`

- The previous statement works as follows:
  - Classify records into groups based on dno.
  - Count number of records in each group.

<u>ssn</u>	name	salary	…	dno
1	Ahmad	356	…	2
2	Salem	500	…	1
3	Maya	333	…	3
4	Sara	405	…	2
5	Malik	734	…	1



Dno	Count(*)
1	2
2	2
3	1

Example

- Find employees with salary > 400 and count how many they are in each department.

- `select dno, count(*) from employee where salary > 400 group by dno;`

- The previous statement works as follows:
  - Get employees with salary > 400.
  - Group records based on dno and count number of records in each group.

<u>**ssn**</u>	**name**	**salary**	**…**	**dno**	
1	Ahmad	356	…	2	
2	Salem	500	…	1	✓
3	Maya	333	…	3	
4	Sara	405	…	2	✓
5	Malik	734	…	1	✓

Dno	Count(*)
1	2
2	1

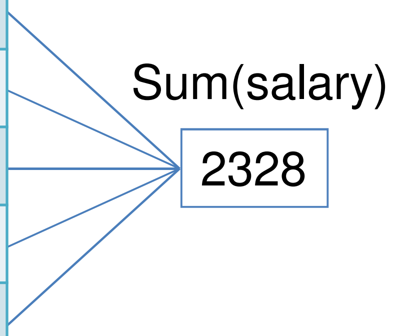
Example

- Find sum of salaries for all employees.

- `select sum(salary) from employee;`

- The previous statement works as follows:
  - Go over all employees and accumulate salary.

<u>**ssn**</u>	**name**	**salary**	**…**	**dno**
1	Ahmad	356	…	2
2	Salem	500	…	1
3	Maya	333	…	3
4	Sara	405	…	2
5	Malik	734	…	1

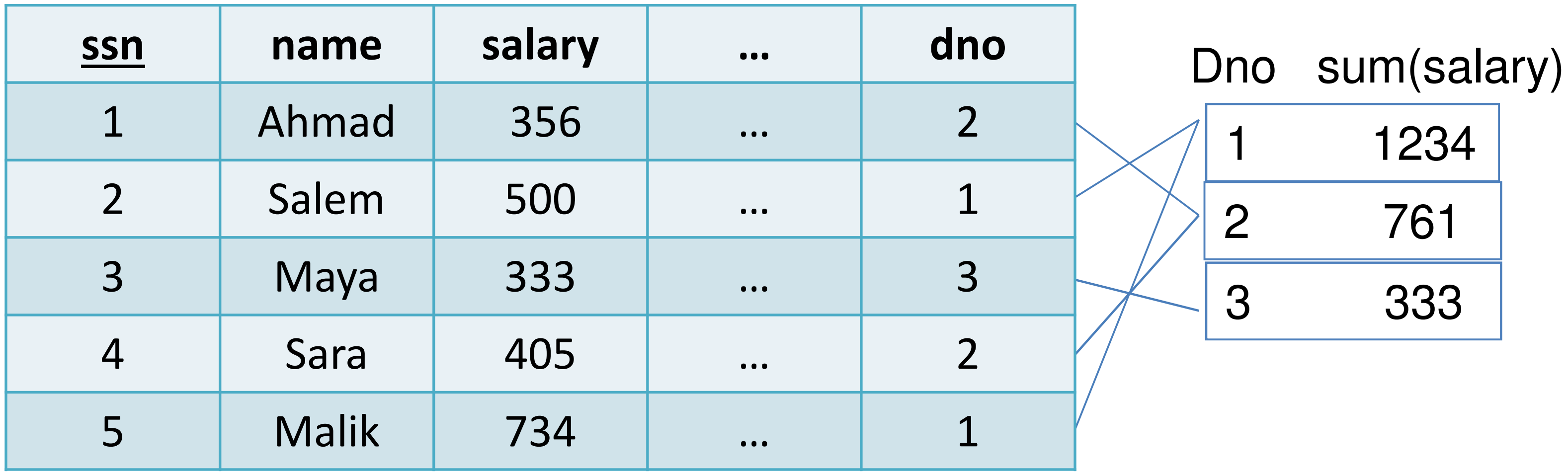


Example

- Find department number and sum of salaries for all employees in that department.

- `select dno, sum(salary) from employee group by dno;`

- The previous statement works as follows:
  - Group employees based on dno.
  - Find sum of salary for employees in each group.



ssn	name	salary	…	dno
1	Ahmad	356	…	2
2	Salem	500	…	1
3	Maya	333	…	3
4	Sara	405	…	2
5	Malik	734	…	1

Dno	sum(salary)
1	1234
2	761
3	333

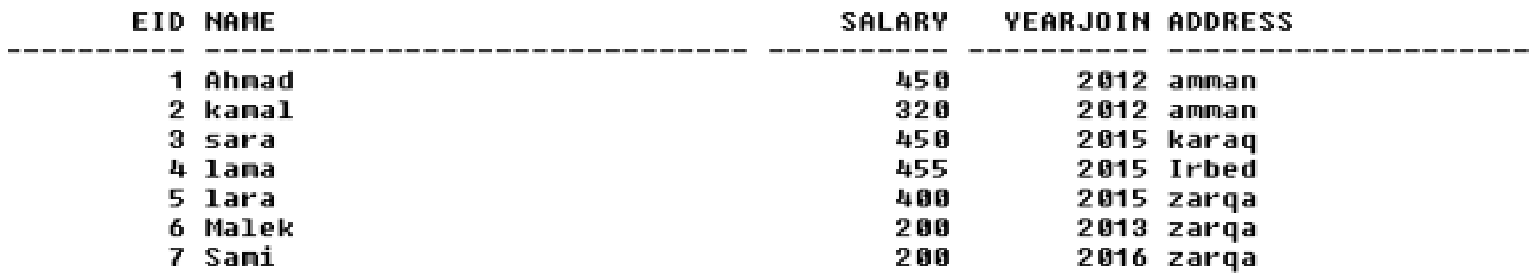
Important rule about “group by”

- The attributes that are allowed to appear in select statement with “group by” are:
  - The attributes on which grouping is done.

- Example 1:
  - Select dno, count(*) from employee group by dno;
  - This is correct.

- Example 2:
  - Select dno,ssn, count(*) from employee group by dno;
  - This is wrong because “ssn” is not an attribute on which “group by” is performed.

Example about group by rule



```
       EID NAME                           SALARY   YEARJOIN ADDRESS
---------- ------------------------------ ---------- ---------- --------------------
         1 Ahmad                                450       2012 amman
         2 kamal                                320       2012 amman
         3 sara                                 450       2015 karaq
         4 lama                                 455       2015 Irbed
         5 lara                                 400       2015 zarqa
         6 Malek                                200       2013 zarqa
         7 Sami                                 200       2016 zarqa
```

select address,name, sum(salary) from employee group by address;

The previous command gives error? Why?

Address	Name	Sum(salary)
Amman	???	770
Karaq	???	450
Irbed	???	455
Zarqa	???	800

What can you put here? Multiple values. It does not work.

Example

- Find the highest salary in each department.
- `select dno, max(salary) from employee group by dno;`
- The previous statement works as follows:
  - Group employees based on dno.
  - Find highest salary for employees in each group.

ssn	name	salary	…	dno
1	Ahmad	356	…	2
2	Salem	500	…	1
3	Maya	333	…	3
4	Sara	405	…	2
5	Malik	734	…	1

Dno	max(salary)
1	734
2	405
3	333

Example

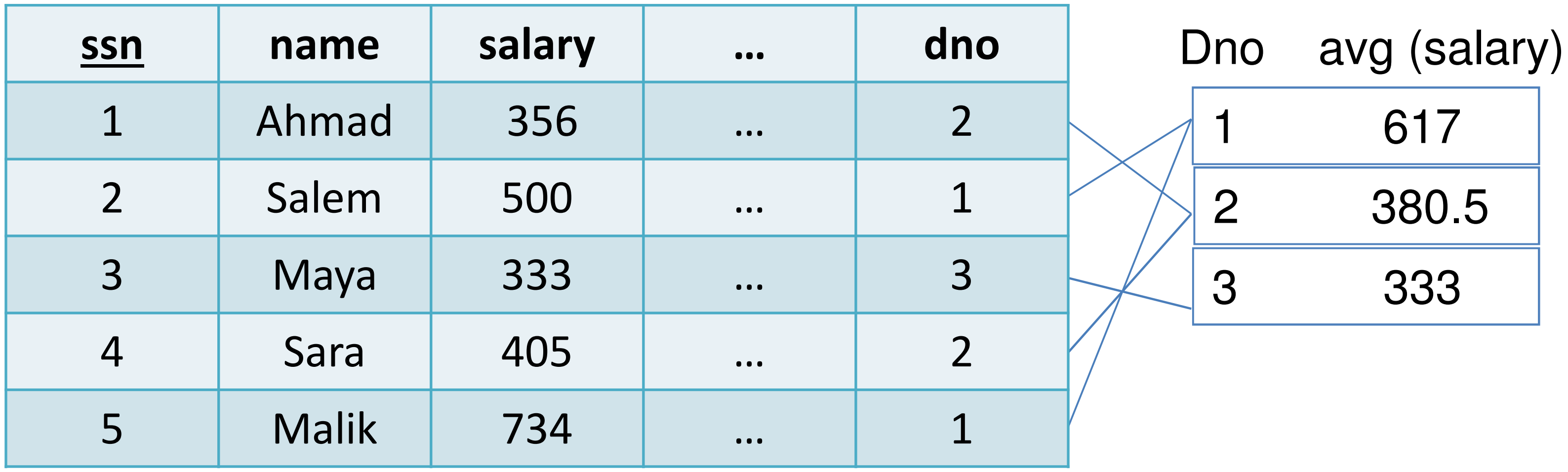
- Find the lowest salary in each department.
- `select dno, min(salary) from employee group by dno;`
- The previous statement works as follows:
  - Group employees based on dno.
  - Find lowest salary for employees in each group.

<u>ssn</u>	name	salary	…	dno
1	Ahmad	356	…	2
2	Salem	500	…	1
3	Maya	333	…	3
4	Sara	405	…	2
5	Malik	734	…	1

Dno	min(salary)
1	500
2	356
3	333

Example

- Find the average salary in each department.
- `select dno, avg(salary) from employee group by dno;`
- The previous statement works as follows:
  - Group employees based on dno.
  - Find average of salaries for employees in each group.



<u>**ssn**</u>	**name**	**salary**	**…**	**dno**
1	Ahmad	356	…	2
2	Salem	500	…	1
3	Maya	333	…	3
4	Sara	405	…	2
5	Malik	734	…	1

Dno	avg (salary)
1	617
2	380.5
3	333

Example

- Find the standard deviation of salaries in the company.

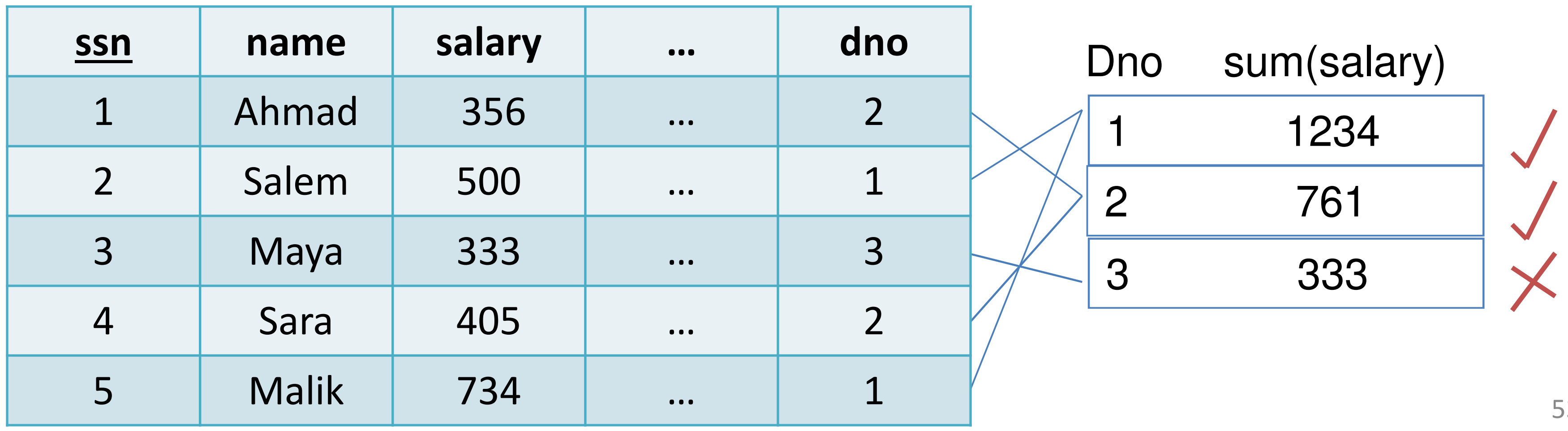
- `select stddev(salary) from employee;`

ssn	name	salary	…	dno
1	Ahmad	356	…	2
2	Salem	500	…	1
3	Maya	333	…	3
4	Sara	405	…	2
5	Malik	734	…	1

stddev(salary)
163.16342

“Having” Clause

- It is used to filter records resulting from “group by”.
- Find departments and the sum of salaries in the department such that sum of salaries is > 400.
- `select dno, sum(salary) from employee group by dno having sum(salary)>400;`



ssn	name	salary	…	dno
1	Ahmad	356	…	2
2	Salem	500	…	1
3	Maya	333	…	3
4	Sara	405	…	2
5	Malik	734	…	1

Dno	sum(salary)	
1	1234	✓
2	761	✓
3	333	✗

Using “distinct”

- Distinct keyword allows you to remove duplicate values from result of select statement.
- Example: Suppose that table “employee” has an address attribute.
- We need to know different (unique) addresses of employees

ssn	Name	salary	…	address	dno
1	Ahmad	356	…	Amman	2
2	Salem	500	…	Salt	1
3	Maya	333	…	Amman	3
4	Sara	405	…	Karak	2
5	Malik	734	…	Salt	1

Example

- If we use:
  - `Select address from employee;`
  - We will get: →

```
Amman
Salt
Amman
Karak
Salt
```

- But, if we use:
  - `Select distinct address from employee;`
  - We will get →

```
Amman
Salt
Karak
```

Example

- Retrieve unique records of this table “mytab”.

A1	A2
1	Nissan
1	Nissan
2	Honda

- `Select distinct * from mytab;`

- The result is

```
1    Nissan
2    Honda
```

Attribute Names

- You can change the name of attributes that appear in the result.
- Example

```
Select id, name from employee;
```

```
       ID                      NAME
--------------------    -------------------------
        1                      Ahmad
        3                      Maya
        4                      Sara
        5                      Malik
        2                      Salem
```

```
Select id as employee_id, name
as employee_name from employee;
```

```
EMPLOYEE_ID      EMPLOYEE_NAME
-------------------    -------------------------
        1                      Ahmad
        3                      Maya
        4                      Sara
        5                      Malik
        2                      Salem
```

Qualifying Names

- If you have two tables
  - Tab1(a,b,c)
  - Tab2(a,x,y)
- How can you distinguish between attribute “a” in “Tab1” and attribute “a” in “Tab2”?
- We can use:
  - Tab1.a
  - Tab2.a
- This is useful when both attributes appear in the same query.

Lab 6

-Select statement

- Like condition
- subqueries
- Cartesian Product
- Join
- Calculated Fields
- Views
- Using “All”, “Any”
- Intersect
- Union

Dr. Osama Al-Haj Hassan

“Like” Condition

- “Like condition” enables you to match attribute values against some pattern.
- “Like Condition” uses “wild cards” for matching
- “Wild Cards” include:
  - Percent (%): indicates any sequence of characters of length zero or more
  - Underscore (_): indicates one character

Examples

- Find employees who’s name starts with ‘Ma’
  - Select * from employee where name like ‘Ma%’;
- Find employees who’s name contains the sequence ‘al’
  - Select * from employee where name like ‘%al%’;
- Find employees who’s salary starts with 33 plus one more digit (total of 3 digits only)
  - Select * from employee where salary like ’33_’;

Subqueries

- A subquery is a query within a query.

- A subquery can appear in:
  – “where” clause
  – “from” clause
  – “select” clause

Subqueries in “where” Clause “exists” subquery

- The “where exists” clause has the value:
  - “true” if the statement following “where exists” produces at least 1 record.
  - Otherwise, it’s values is “false”

- Example:Select employee ssn and name for all employees who have dependents.
  - Select ssn,name from employee where exists (select * from dependent where essn=ssn);

How is the previous query executed?

Query	Result
Select ssn,name from employee	
Where exists (select * from dependent where essn=1)	Result: true ✓
Where exists (select * from dependent where essn=2)	Result: false
Where exists (select * from dependent where essn=3)	Result: true ✓
Where exists (select * from dependent where essn=4)	Result: false
Where exists (select * from dependent where essn=5)	Result: false

Employee

Ssn	name	salary	Bdate	superssn	dno
1	Ahmad	356	12-JAN-80	2	2
2	Salem	500	30-APR-85	3	1
3	Maya	333	10-FEB-70	4	3
4	Sara	405	06-JAN-90	5	2
5	Malik	734	15-MAR-86	1	1

dependent

essn	dep_name
1	Faten
3	Sami

Final Result

Ssn	name
1	Ahmad
3	Maya

Subqueries in “where” Clause “not exists” subquery

- The “where not exists” is the opposite of “where exists”
- The “where not exists” clause has the value:
  - “true” if the statement following “where not exists” produces zero record.
  - Otherwise, it’s values is “false”
- Example:Select employee ssn and name for all employees who do **not** have dependents.
  - Select ssn,name from employee where not exists (select * from dependent where essn=ssn);

Subqueries in “where” Clause “in” subquery

- “in” option is used to check if a “given value” exists among “other values”.

- Example: find employee ssn and name for employees who have dependents.
  – Select ssn,name from employee where ssn in (select essn from dependent);

How is the previous query executed?

Select ssn,name from employee

Where 1 in (1,3)	Result: true ✓
Where 2 in (1,3)	Result: false
Where 3 in (1,3)	Result: true ✓
Where 4 in (1,3)	Result: false
Where 5 in (1,3)	Result: false

Employee

Ssn	name	salary	Bdate	superssn	dno
1	Ahmad	356	12-JAN-80	2	2
2	Salem	500	30-APR-85	3	1
3	Maya	333	10-FEB-70	4	3
4	Sara	405	06-JAN-90	5	2
5	Malik	734	15-MAR-86	1	1

dependent

essn	dep_name
1	Faten
3	Sami

Final Result

Ssn	name
1	Ahmad
3	Maya

Note

- The “given value” and “other values” should have the same dimension and type.

- Example:
  - Select ssn,name from employee where ssn in (select essn,dep_name from dependents;
    - This is wrong because “ssn” is one attribute and “essn,dep_name” are two attributes

  - Select ssn,name from employee where ssn in (select dep_name from dependent);
    - This is wrong because “ssn” is number and “dep_name” is a string.

Subqueries in “where” Clause
“not in” subquery

- “not in” is the opposite of “in”.
- “not in” option is used to check if a “given value” does **not** exist among “other values”.
- Example: find employee ssn and name for employees who do not have dependents.
  - Select ssn,name from employee where ssn not in (select essn from dependent);

Subqueries in “from” clause

- Find employee ssn and number of dependents for him only if the number of dependents > 1

- Straight forward solution (Without Subqueries):
  - Select essn,count(*) from dependent group by essn having count(*)>1;

- More complex solution (With Subqueries):
  - Select subq.essn ,subq.dep_count from (select essn,count(*) as dep_count from dependent group by essn) subq where subq.dep_count>1;

How is the previous subquery executed?

```
Select subq.essn ,subq.dep_count from
(select essn,count(*) as dep_count from dependent group by essn) subq
 where subq.dep_count>1;
```

Step 1

subq

```
(select essn,count(*) as dep_count
from dependent group by essn) subq
```

subq

essn	dep_count
1	2
3	1

Step 2

```
Select subq.essn ,subq.dep_count
from subq where subq.dep_count>1;
```

essn	dep_count	
1	2	✓
3	1	✗

Cartesian Product

- Cartesian Product: The operation of pairing records of two tables.
- The symbol for Cartesian Product is "X"
- (Table1 X Table2) is performed as follows
  - Number of attributes in the result of Cartesian Product is the sum of number of attributes in "Table 1" and "Table2"
  - Number of records in the result of Cartesian Product is the multiplication of number of records in "Table 1" and "Table2"
  - Take each record from "Table1" and pair it with each record in "Table2"

A	B	C
A1	B1	C1
A2	B2	C2

2 records

X

X	Y	Z
X1	Y1	Z1
X2	Y2	Z2
X3	Y3	Z3

3 records

=

A	B	C	X	Y	Z
A1	B1	C1	X1	Y1	Z1
A1	B1	C1	X2	Y2	Z2
A1	B1	C1	X3	Y3	Z3
A2	B2	C2	X1	Y1	Z1
A2	B2	C2	X2	Y2	Z2
A2	B2	C2	X3	Y3	Z3

6 records

Example

- What is the Cartesian Product of employee and dependent?
  - Select * from employee,dependent;

Cartesian Product

- Cartesian product produces **logically incorrect** records for tables which have relationships.
- Example: Employee X Dependent

Employee

Ssn	name	salary	Bdate	superssn	dno
1	Ahmad	356	12-JAN-80	2	2
2	Salem	500	30-APR-85	3	1
3	Maya	333	10-FEB-70	4	3
4	Sara	405	06-JAN-90	5	2
5	Malik	734	15-MAR-86	1	1

Dependent

essn	dep_name	relationship
1	Sofi	Daughter
1	Basem	Son
3	Bana	Daughter

Employee X Dependent

Ssn	name	salary	Bdate	superssn	dno	essn	dep_name	relationship	
1	Ahmad	356	12-JAN-80	2	2	1	Sofi	Daughter	✓
1	Ahmad	356	12-JAN-80	2	2	1	Basem	Son	✓
1	Ahmad	356	12-JAN-80	2	2	3	Bana	Daughter	✗
2	Salem	500	30-APR-85	3	1	1	Sofi	Daughter	✗
2	Salem	500	30-APR-85	3	1	1	Basem	Son	✗
2	Salem	500	30-APR-85	3	1	3	Bana	Daughter	✗
3	Maya	333	10-FEB-70	4	3	1	Sofi	Daughter	✗
3	Maya	333	10-FEB-70	4	3	1	Basem	Son	✗
3	Maya	333	10-FEB-70	4	3	3	Bana	Daughter	✓
4	Sara	405	06-JAN-90	5	2	1	Sofi	Daughter	✗
4	Sara	405	06-JAN-90	5	2	1	Basem	Son	✗
4	Sara	405	06-JAN-90	5	2	3	Bana	Daughter	✗
5	Malik	734	15-MAR-86	1	1	1	Sofi	Daughter	✗
5	Malik	734	15-MAR-86	1	1	1	Basem	Son	✗
5	Malik	734	15-MAR-86	1	1	3	Bana	Daughter	✗

Some tuples are **not logically correct**.

Fixing Cartesian Product Result (Join and Join Predicate)

- To fix the previous problem, we make Cartesian Product based on a predicate that relates primary key to foreign key
- Example: Employee X Dependent (Employee.ssn = Dependent.essn)
- In this case, the operation is called “Join” and the predicate called “join predicate” or “join condition”
- Because of that, the incorrect records will be deleted. They will be deleted because (ssn is not equal to essn)

Employee X Dependent

Ssn	name	salary	Bdate	superssn	dno	essn	dep_name	relationship
1	Ahmad	356	12-JAN-80	2	2	1	Sofi	Daughter
1	Ahmad	356	12-JAN-80	2	2	1	Basem	Son
3	Maya	333	10-FEB-70	4	3	3	Bana	Daughter

- Now, the result is **<u>logically correct</u>**

Example

- This is an example of a “join”.
- Find all information related to employees and their dependents.
  - Select * from employee,dependent where employee.ssn=dependent.essn;

Important Note

- The number of join conditions we use to fix Cartesian Product result = number of tables involved in join – 1

- Example:

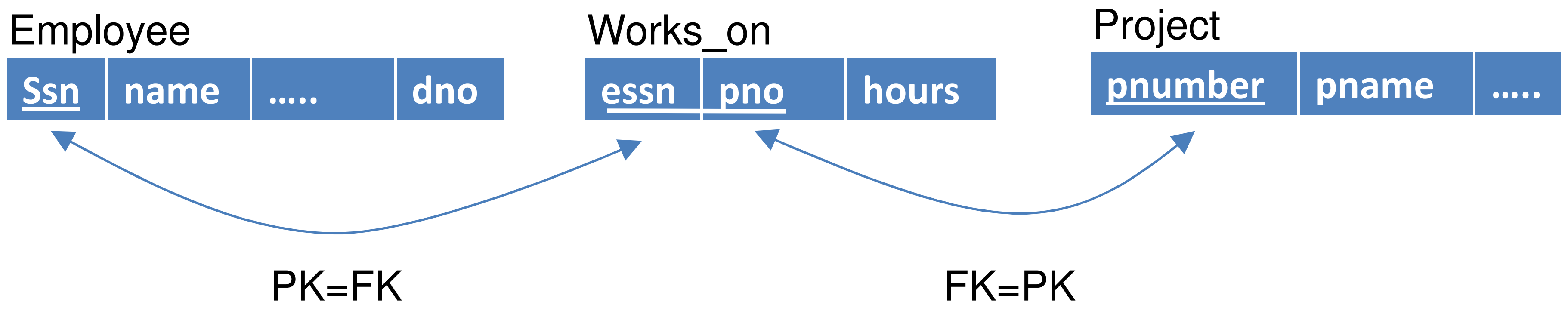
```
select * from employee,works_on,project where employee.ssn = works_on.essn
    and works_on.pno=project.pnumber;
```

- Number of tables in join = 3

- Number of join conditions to fix Cartisian product result = 3-1 = 2

Important Note

- The number of join conditions we use to fix Cartesian Product result = number of tables involved in join – 1
- Joining tables employee,works_on,project



Note

- When solving any query, do the following:
  1) What are the attributes you need?
  2) In which tables do those attributes exist?
  3) What is the correct query design?

Calculated Fields

- You can use mathematical equations in your select statements.

- Example1: Find employee salary multiplied by 2.
  - Select (salary*2) from employee;

- Example 2: Suppose you have a table of products that a company sells. You have sales of a product and cost of product. We need to know the product which gives us more profit.
  - Select max(prod_sales/prod_cost) from product;

Views

- Tables are physically stored in disk.
- Views are not physically stored in disk.
- Views are generated from existing tables or other existing views.
- Views are used to display part of the DB that a user wants to see.

Views

- Syntax:
  - Create view $view_name as $selectstatement;

- Create a view of all employees who have salary >200;
  - Create view emp200 as

    select * from employee where salary > 200;

Views

- You can query views the same way you do with tables.

- Example: what are the records in emp200 view?
  - Select * from emp200;

More Examples

- Find employee ssn, employee name, and the dependent name for all employees with salary > 300.

- Tables we need:
  - Ssn, name (Employee)
  - Dep_name (Dependent)

- Select employee.ssn,employee.name, dependent.dep_name from employee,dependent where employee.ssn=dependent.essn and salary>300;

More Examples

- Find department name and the name of its manager.

- Tables we need:
  - Name (Employee)
  - Dname (Department)

- Select name, dName
  from employee,department
  where employee.ssn = department.mgrssn;

More Examples (all)

- Find employee ssn,name in ‘CS’ department who’s salary is greater than each salary for employees in ‘CIS’ department;
  - Select ssn,name from employee,department where employee.dno = department.dnumber and

    dname = 'CS' and

    salary>all (select salary from employee,department where

    employee.dno=department.dnumber and

    dname= 'CIS');

How is the previous query executed?

A join between Employee and Department such that dname=CS

ssn	name	dno	dnumber	dname	salary
2	Salem	1	1	CS	500
5	Malik	1	1	CS	370

> all

A join between Employee and Department such that dname=CIS

ssn	name	dno	dnumber	dname	salary
1	Ahmad	2	2	CIS	356
4	Sara	2	2	CIS	405

Only Salem SSN and Name will be in the final result

More Examples (any)

- Find employees ssn,name in ‘CS’ department who’s salary is greater than any salary for employees in ‘CIS’ department;
  - Select ssn,name from employee,department where employee.dno = department.dnumber and

    dname = 'CS' and

    salary>any (select salary from employee,department where employee.dno=department.dnumber and

    dname= 'CIS');

How is the previous query executed?

A join between Employee and Department such that dname=CS

ssn	name	dno	dnumber	dname	salary
2	Salem	1	1	CS	500
5	Malik	1	1	CS	370

> any

A join between Employee and Department such that dname=CIS

ssn	name	dno	dnumber	dname	salary
1	Ahmad	2	2	CIS	356
4	Sara	2	2	CIS	405

Salem and Malek SSN and Name will be in the final result

More Examples

- Find ssn,name for employees who work in projects that belong to department 1 but they do not work in projects that belong to department 2.
  - Select ssn from employee,works_on,project where employee.ssn = works_on.essn and

    works_on.pno=project.pnumber and project.dnum=1 and employee.ssn not in

    (Select ssn from employee,works_on,project where employee.ssn = works_on.essn and

    works_on.pno=project.pnumber and project.dnum=2);

What if we want to use department name in the query?
In this case, we need to use one more table and one more join condition

More Examples

- Find employees who work on projects that belong to department 1 and department 2.
  - Select distinct ssn from employee,works_on,project where employee.ssn = works_on.essn and

    works_on.pno=project.pnumber and project.dnum=1 and employee.ssn in

    (Select ssn from employee,works_on,project where employee.ssn = works_on.essn and

    works_on.pno=project.pnumber and project.dnum=2);

More Examples (intersect)

- The previous example can be done as:
  - (Select ssn,name from employee,works_on,project where employee.ssn = works_on.essn and

    works_on.pno=project.pnumber and project.dnum=1)

    intersect

    (Select ssn,name from employee,works_on,project where employee.ssn = works_on.essn and

    works_on.pno=project.pnumber and project.dnum=2);

Can we say “project.dnum=1 and project.dnum=2” instead of using intersect?

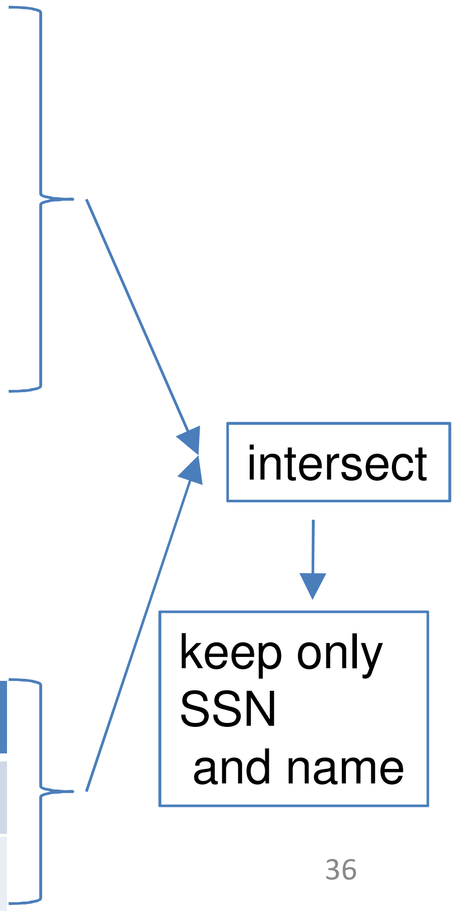
How is the previous query executed?

Select ssn,name from employee,works_on,project where employee.ssn = works_on.essn and works_on.pno=project.pnumber and project.dnum=1

Ssn	name	…	Essn	pno	…	pnumber	pname	…	dnum
1	Ahmad		1	1		1	Abraj		1
1	Ahmad		1	2		2	Univ		1
2	Salem		2	1		1	Abraj		1
3	Maya		3	2		2	Univ		1

Select ssn,name from employee,works_on,project where employee.ssn = works_on.essn and works_on.pno=project.pnumber and project.dnum=2

Ssn	name	…	Essn	pno	…	pnumber	pname	…	dnum
1	Ahmad		1	3		3	Sabek		2
3	Maya		3	3		3	Sabek		2



More Examples

- Find ssn for employees who work on projects that belong to department 1 “or” projects that belong to department 2.
  - Select distinct ssn,name from employee,works_on,project where employee.ssn = works_on.essn and

    works_on.pno=project.pnumber and

    (project.dnum=1 or project.dnum=2);

More Examples (Union)

- Solve the previous query using “Union”
  - (Select ssn,name from employee,works_on,project where employee.ssn = works_on.essn and

    works_on.pno=project.pnumber and project.dnum=1)

    union

    (Select ssn,name from employee,works_on,project where employee.ssn = works_on.essn and

    works_on.pno=project.pnumber and project.dnum=2);

How is the previous query executed?

Select ssn,name from employee,works_on,project where
 employee.ssn = works_on.essn and
works_on.pno=project.pnumber and project.dnum=1

Ssn	name	…	Essn	pno	…	pnumber	pname	…	dnum
1	Ahmad		1	1		1	Abraj		1
1	Ahmad		1	2		2	Univ		1
2	Salem		2	1		1	Abraj		1
3	Maya		3	2		2	Univ		1

Select ssn,name from employee,works_on,project where
 employee.ssn = works_on.essn and
works_on.pno=project.pnumber and project.dnum=2

Ssn	name	…	Essn	pno	…	pnumber	pname	…	dnum
1	Ahmad		1	3		3	Sabek		2
3	Maya		3	3		3	Sabek		2

Union

Keep only SSN
 and name

Lab 7

- Inner join
- Left Outer Join
- Right Outer Join
- Full Outer Join
- To_Char function
- To_Number function
- To_Date function
- Creating indexes
- Commit and Rollback
- Creating Users
- Granting and Revoking Privileges
- Changing user password
- Switching between users
- Dropping Users

Dr. Osama Al-Haj Hassan

Inner Join

- Inner join is the same as “join”.

- Syntax:

```
select $col1, $col2, …
from $table_name1
inner join $table_name2
on $table_name1.column_name=$table_name2.column_name;
```

Inner Join

- Example:
  - Perform a join between tables employee and department.
  - Select * from employee inner join department on employee.dno = department.dnumber;

Inner Join

```
Select * from employee inner join department on employee.dno = department.dnumber;
```

These 2 statements are equivalent

```
Select * from employee, department where employee.dno = department.dnumber;
```

Left Outer Join

- Left Outer Join returns records from the table on the left even when they do not have a match in the table on the right.
- Syntax:

```
select $col1, $col2, …
from $table_name1
left outer join $table_name2
on $table_name1.column_name=$table_name2.column_name;
```

Example of Left Outer Join

Emp1

SSN	name	Lives-in
1	Ahmad	Amman
2	Kamal	Karak
3	Sara	Salt
4	Majed	Ma3an

Emp2

SSN	name	Salary	works-in
1	Ahmad	370	Amman
2	Kamal	460	Salt
4	Majed	600	Irbid

$\text{Emp1} ⟕_{\text{Emp1.ssn=Emp2.ssn}} \text{Emp2}$

SSN	name	Lives-in	SSN	name	Salary	works-in
1	Ahmad	Amman	1	Ahmad	370	Amman
2	Kamal	Karak	2	Kamal	460	Salt
3	Sara	Salt	null	null	null	null
4	Majed	Ma3an	4	Majed	600	Irbid

Left Outer Join

- Example: Perform a left outer join between emp1 and emp2;
- Select * from emp1 left outer join emp2 on emp1.ssn = emp2.ssn;

Right Outer Join

- Right Outer Join returns records from the table on the right even when they do not have a match in the table on the left.

- Syntax:

```
select $col1, $col2, …
from $table_name1
right outer join $table_name2
on $table_name1.column_name=$table_name2.column_name;
```

Example of Right Outer Join

Emp1

SSN	name	Lives-in
1	Ahmad	Amman
2	Kamal	Karak
3	Sara	Salt

Emp2

SSN	name	Salary	works-in
1	Ahmad	370	Amman
2	Kamal	460	Salt
3	Sara	250	Salt
4	Majed	600	Irbid

$\text{Emp1} ⋊_{\text{Emp1.ssn=Emp2.ssn}} \text{Emp2}$

SSN	name	Lives-in	SSN	name	Salary	works-in
1	Ahmad	Amman	1	Ahmad	370	Amman
2	Kamal	Karak	2	Kamal	460	Salt
3	Sara	Salt	3	Sara	250	Salt
null	null	null	4	Majed	600	Irbid

Right Outer Join

- Example: Perform a right outer join between emp1 and emp2.
- Select * from emp1 right outer join emp2 on emp1.ssn = emp2.ssn;

Full Outer Join

- Full Outer Join returns records from the tables on the left and on the right even when they do not have a match in each other.

- Syntax:

```
select $col1, $col2, …
from $table_name1
full outer join $table_name2
on $table_name1.column_name=$table_name2.column_name;
```

Example of Full Outer Join

Emp1

SSN	name	Lives-in
2	Kamal	Karak
3	Sara	Salt
4	Majed	Ma3an

Emp2

SSN	name	Salary	works-in
1	Ahmad	370	Amman
2	Kamal	460	Salt
4	Majed	600	Irbid

$$Emp1 \mathbin{⟗}_{Emp1.ssn=Emp2.ssn} Emp2$$

SSN	name	Lives-in	SSN	name	Salary	works-in
null	null	null	1	Ahmad	370	Amman
2	Kamal	Karak	2	Kamal	460	Salt
3	Sara	Salt	null	null	null	null
4	Majed	Ma3an	4	Majed	600	Irbid

Full Outer Join

- Example: Perform a full outer join between emp1 and emp2.
- Select * from emp1 full outer join emp2 on emp1.ssn = emp2.ssn;

To_Char Function

- “To_Char” function converts a number or a date to string.
- Syntax: To_Char($value,$format)
- Examples:
  - **to_char(1210.76, '9999.9')** would return '1210.8‘
  - **to_char(1210.76, '9,999.99')** would return '1,210.76‘
  - **to_char(1210.76, '$9,999.00')** would return '$1,210.76‘
  - **to_char(21, '000099')** would return '000021‘
  - **to_char(1210.76, '9999.999')** would return '1210.760‘
  - **to_char(1210.76, ‘99999.99')** would return ‘01210.76‘
  - **to_char(1210.76, ‘999.99')** would return a badly formatted string because number of 9s is more than number of digits in the input number.

To_Char Function

- “Sysdate” is a keyword that returns the current date.
- Assume today’s date is “07/09/2012”
- Examples:
  - **to_char(sysdate, 'yyyy/mm/dd');** would return '2012/07/09‘
  - **to_char(sysdate, 'Month DD, YYYY');** would return 'July 09, 2012‘
  - **to_char(sysdate, 'MON DDth, YYYY');** would return 'JUL 09TH, 2012‘

To_Number Function

- “To_Number“ function converts a string to number.
- Syntax: To_Number($value,$format)
- Examples:
  - **to_number('1210.73', '9999.99')** would return the number 1210.73
  - **to_number('1210.73', '99999.99')** would return the number 01210.73
  - **to_number('1210.73', '9999.999')** would return the number 1210.730
  - **to_number('1210.73', '999.99')** would return an error
  - **to_number('1210.73', '9999.9')** would return an error
  - **to_number('546', '999')** would return the number 546

To_Date Function

- “To_Date” function converts a string to date.
- Syntax: To_Date($value,$format)
- The ‘format’ <u>should be the same as</u> the format of the supplied date value.
- Examples:
  - **to_date('2012/07/09', 'yyyy/mm/dd')** would return a date value of July 9, 2012.
  - **to_date('2012/07/09', 'yyyy/mm/dddd')** would return an error.
  - **to_date('070912', 'MMDDYY')** would return a date value of July 9, 2012.
  - **to_date('20120315', 'yyyymmdd')** would return a date value of Mar 15, 2012.

to_date Examples

- to_date('10-12-06','MM-DD-YY')
- to_date('jan 2007','MON YYYY')
- to_date('2007/05/31','YYYY/MM/DD')
- to_date('12-31-2007 12:15','MM-DD-YYYY HH:MI')
- to_date('2006,091,00:00:00' , 'YYYY,DDD,HH24:MI:SS')
- to_date('15-may-2006 06:00:01','dd-mon-yyyy hh24:mi:ss')

Examples of using the previous functions

- insert into employee(ssn,bdate) values(8, to_date('20020315', 'yyyymmdd'));

- In this example, we needed to use “to_date” function because we are using a date value different than the default date format in Oracle.

Examples of using the previous functions

- Insert into employee(ssn,salary) values (10,to_number('234.76','999.99'));

- Update employee set bdate = to_date('10-12-06','MM-DD-YY') where ssn=10;

Creating Indexes

- Index: is a data structure that speeds up access to table records.

Example (Without Index)

Employee

<u>SSN</u>	name	salary	dno
1	Ahmad	350	5
2	Kamal	450	3
.	.	.	.
.	.	.	.
1000	Lama	444	5

1000 employee exist in table

Question: How many steps do you need to access employee 1000?
“select * from employee where ssn=1000;”

Answer: 1000 steps
- you start from employee 1
- then employee 2
- and so on, until you reach employee 1000

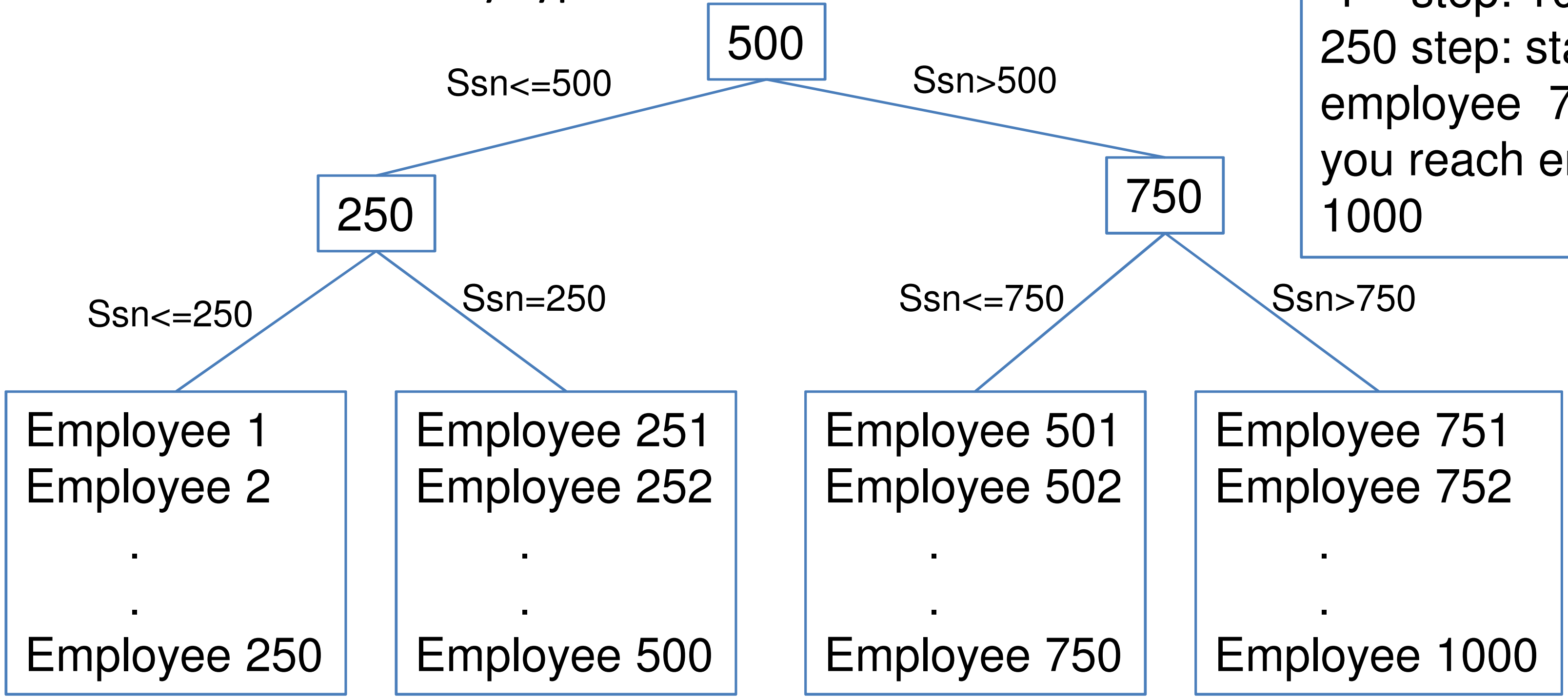
<u>**This is not efficient**</u>

Example (With Index)

- SSN is the primary key of table employee.
- An index on attribute “ssn” speeds up access.
- Let us have a simple “B+ tree” index on SSN
- There are many types of indexes

Now, you only need 252 steps to reach Employee 1000. How?

1 step: 1000 > 500
1 step: 1000 > 750
250 step: start from employee 751 until you reach employee 1000



<u>This is more efficient</u>

Creating Indexes

- Syntax:
  - create index $index_name on $table_name($attribute_name);
  - Example:
    - create index emp_dno on employee(dno);
    - This way, queries which contain tests on the attribute dno will execute faster.

Creating Indexes

- An index is created “by default” on the primary key of tables.
- For example, you cannot create an index on employee ssn because an index on ssn is already created because ssn is the primary key of the table.

Commit/rollback

- In a session, usually the effect of your actions on DB are not "<u>immediately</u>" stored "<u>permanently</u>" on disk.

- This means that other users using DB might not see the effect of your actions on DB immediately.

- In this case, you can use the "commit" command to "immediately" reflect changes on DB. Other users will be able to see the effect of your actions on DB.

Commit/rollback

- In a session, if you want to cancel the effect of your actions on DB, you use command “rollback”

Type of Users

- Local: Needs a password.
- External: Does not need a password, it will be authenticated by OS.
- Global: Does not need a password, it will be authenticated by directory service.

Creating Users

- Two steps are needed to create a user.
  - Step 1: Create user name and password.
  - Step 2: Give user roles.

Creating Users (Step 1:Create user and Password)

- Creating user name and password:
  - create user $user_name identified by $password;
  - Example: create user kamal identified by kamal123;
- User cannot log in SQL Plus unless system administrator gives him privilege to do that.

Creating Users (Step 2: Assign roles)

- There are many “roles” which include:
  - DBA: includes most administrative functions.
  - Connect: allows user to log in locally to database.
  - Resource: does not have all options available to DBA.

- Syntax:
  - Grant $role to $user;

Creating Users (Step 2: Assign roles)

- Examples:
  - grant connect to kamal;
    - Now kamal can log in to SQL Plus.
  - grant resource to kamal;
    - Now kamal can create tables.

System Privileges

- If a certain role does not have a certain privilege, then you can grant it using:
  - grant $privilege_name to $user_name;

- Example: role “resource” does not allow users to create a view. We can grant “create view” privilege to user “kamal” by:
  - grant create any view to kamal;

Object Privileges

- Object Privilege: The right to perform a particular action on a specific object.

- What do you do with privileges?
  - Grant: give privilege to users.
  - Revoke: take privilege away from users.

Object Privileges

- Syntax of Granting Object Privilege:
  - grant $privilege_name ON $object_name to $schema_name;

- Syntax of Revoking Object Privileges:
  - revoke $privilege_name on $object_name from $schema_name;

Examples

- Assume you are user “Ahmad”. You have a table “employee” and you want to give user “kamal” some privileges on table “employee”.

- grant select on employee to kamal;

  - Allows user kamal to perform select statement on object employee.

- grant select,update,insert,delete on employee to kamal;

- grant all on employee to kamal;

  - Allows kamal to perform any action on object employee.

Changing Password of a User

- Syntax:
  - alter user $user_name identified by $password;
- Example:
  - alter user kamal identified by xyz99;

Switching between users

- If you are connected as user1 and you want to disconnect and reconnect as user2, you use the command “conn”.

- Format:
  - conn $user_name/password or
  - conn $user_name then SQL Plus will ask for password

- Example:
  - conn kamal/abc123
  - conn kamal

Dropping Users

- To delete a user you use, first, you need to make sure the users do not have objects(tables, views, indexes,…etc). After that, you can use this command:
  – drop user $user_name;
  – Example: drop user kamal;

- If you want to delete a user and all objects related to him, you can use:
  – drop user $user_name cascade;
  – Example: drop user kamal cascade;