

Tema 4 - Llenguatge de Definició de Dades (DDL)

4.1. Introducció

En aquest tema estudiarem el llenguatge de definició de dades (DDL) emprat en les bases de dades Oracle, en concret l'utilitzat en Oracle Database 18c (però es podria utilitzar quasi qualsevol altra versió de Oracle Database). Aquest llenguatge ens permet crear i definir l'estructura d'una base de dades .

Començarem tractant alguns conceptes bàsics en les bases de dades Oracle, així com els primers passos que haurem de seguir per a treballar amb un servidor Oracle Database.

Si preferixes emprar un altre gestor de bases de dades, has de tindre en compte que, encara que un percentatge molt gran de SQL és comú, Oracle té unes certes peculiaritats i la seua sintaxi a vegades se separa de l'estàndard. Per tant, en cas d'emprar un altre gestor, és probable que hages de fer canvis en algunes de les instruccions que se't presentaran.

4.2. Oracle

Oracle Corporation és una multinacional americana amb seu en Redwood Shores, Califòrnia. Està especialitzada principalment en el desenvolupament i comercialització d'aplicacions de bases de dades, sistemes en el núvol i programari per a grans empreses. L'any 2018 va anar la tercera major companyia de programari per ingressos.

4.2.1. Bases de dades de Oracle.

Oracle Corporation ha adquirit i/o desenvolupat les següents tecnologies de bases de dades :

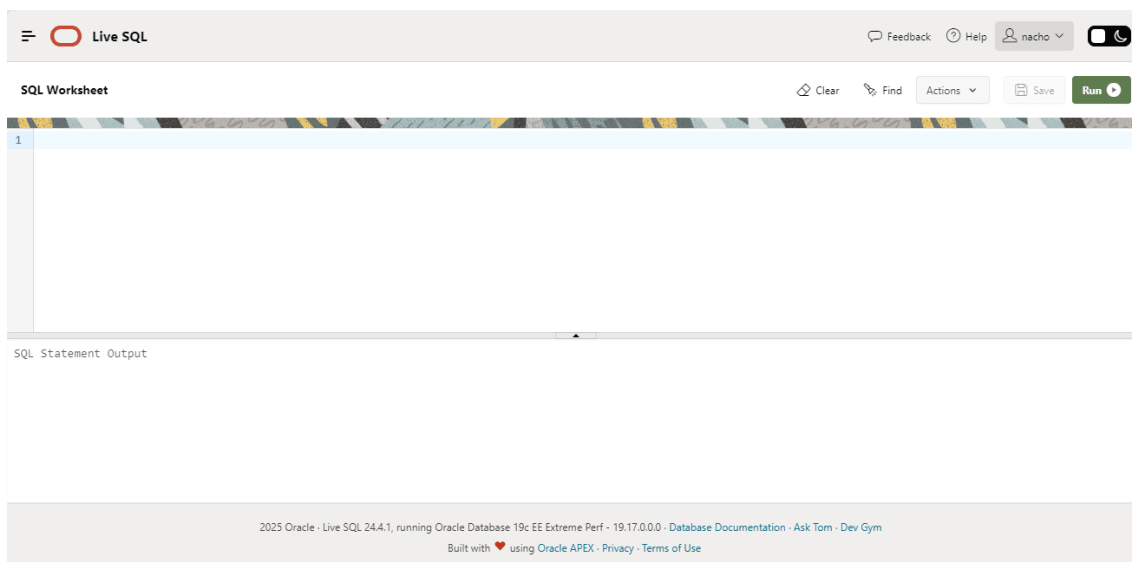
- Berkeley DB, que ofereix maneig de bases de dades embegudes ((SGBD) que està integrat amb l'aplicació programari que requereix accés a les dades emmagatzemades, de tal forma que la base de dades està oculta de l'aplicació d'usuari final i requereix poc o cap manteniment.
- Oracle Rdb, un sistema de bases de dades relacionals que es pot executar en plataformes OpenVMS.
- TimesTen, que optimitza les operacions de les bases de dades en memòria
- Oracle Essbase, que continua amb la tradició de Hyperion Essbase en el maneig de bases de dades multidimensionals
- MySQL, un sistema de bases de dades relacionals, sota llicència GNU
- Oracle NoSQL Database

4.2.2. Ús a través de Oracle Live SQL

La forma més senzilla de practicar SQL i PL/SQL emprant Oracle és utilitzar una interfície web, creat per la pròpia Oracle, anomenat **Oracle Live SQL**, al qual es pot accedir en la URL:

<https://livesql.oracle.com/>

Serà necessari crear-se un compte en Oracle, però, a canvi, el maneig és molt més senzill que si necessitàrem instal·lar tot el gestor de bases de dades. L'aparença de la interfície és esta:



La pantalla està dividida en dos panells: en el superior introduïrem les nostres instruccions SQL i en l'inferior se'ns mostraran els resultats.

4.2.3. Alternativa: Instal·lació local

Utilitzarem Oracle Database 18c Express Edition (XE), o posterior, juntament amb l'eina SQL Developer. La instal·lació i posada en marxa de totes dues aplicacions s'ha tractat en un document a part, però també es pot descarregar una màquina virtual basada en Linux, que conté Oracle 19.3 y SQL Developer 19.1, llestes per a usar, en la URL:

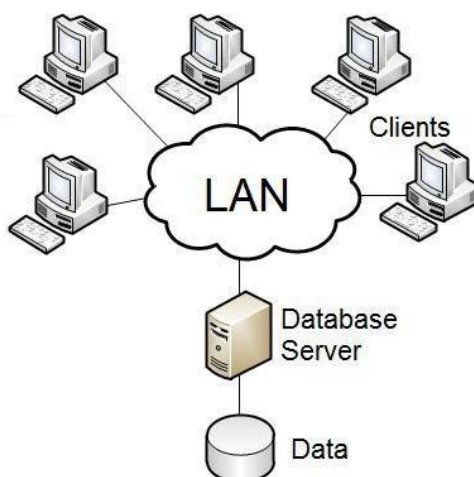
<https://www.oracle.com/database/technologies/databaseappdev-vm.html>

4.2.4. Arquitectura Client/Servidor

En l'arquitectura client/servidor de Oracle ens trobem amb dues parts clarament diferenciades: el front-end o part client i el back-end o part servidor. El client executa una aplicació que accedeix a la informació de la base de dades, a través de peticions que són

processades en el servidor, qui retornarà una resposta al client amb els resultats de la seua petició.

Encara que l'aplicació client i servidor es poden executar en la mateixa màquina, en el món real és més habitual treballar amb diferents màquines connectades en xarxa.



En el nostre cas concret, per a aquest curs, el que anomenem part servidor estaria format per Oracle Database 19c, mentre que el client seria l'aplicació SQL Developer.

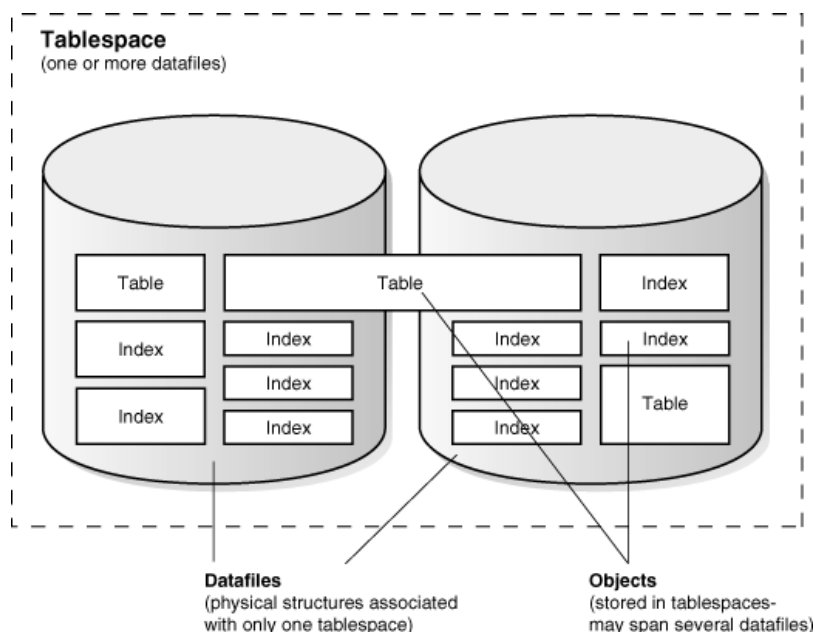
4.2.5. Instància de base de dades

Una instància de Oracle està formada pel conjunt de processos i estructures de dades en memòria que el servidor de base de dades necessita quan està en funcionament. Aquests processos proporcionen el mecanisme necessari per a accedir a un conjunt d'arxius físics on s'emmagatzema la informació de la base de dades.

És possible tindre múltiples instàncies de Oracle 18c executant-se en un mateix servidor, amb espai de memòria diferent per a donar suport a cadascuna d'elles. Una instància de Oracle s'assigna sempre a una única base de dades.

4.2.6. Tablespaces

Oracle emmagatzema les dades de manera lògica en "tablespaces" i físicament en fitxers associats amb el seu corresponent "tablespace". La següent imatge mostra aquesta relació



Bases de dades , tablespaces i fitxers de dades estan estretament relacionats, però presenten importants diferències:

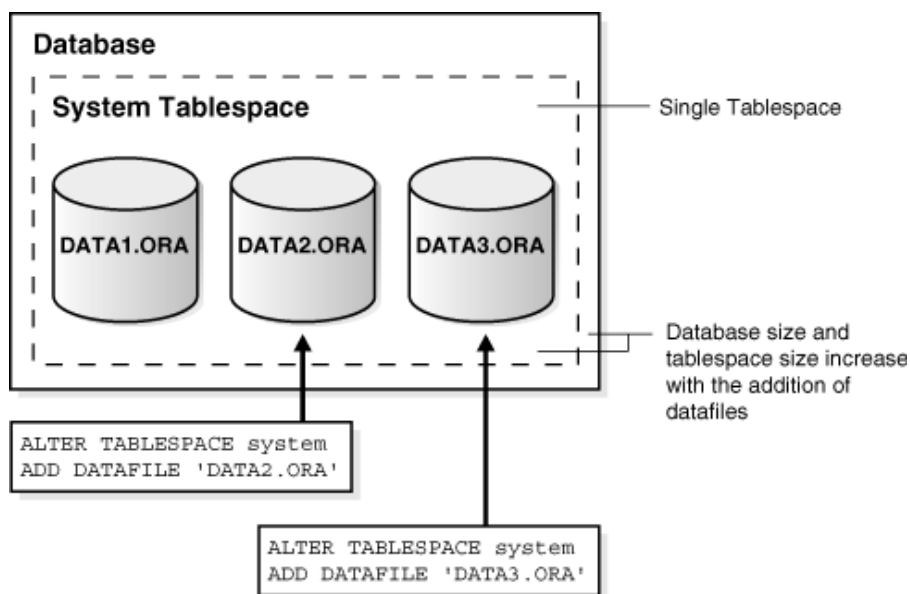
- Una base de dades de Oracle està formada d'una o més unitats lògiques d'emmagatzematge, anomenades tablespaces que en el seu conjunt emmagatzemen tota la informació de la base de dades
- Cada tablespace d'una base de dades conté un o més arxius de dades, amb una estructura física conforme al sistema operatiu on Oracle s'està executant
- La informació de la base de dades s'emmagatzema en els arxius de dades que formen cada tablespace. Per exemple , la base de dades més simple de Oracle, tindria un únic tablespace amb un únic arxiu de dades. Una altra base de dades podria tindre quatre tablespaces amb dos arxius de dades cadascun, amb un total de huit arxius

Taules, índexs i altres objectes es col·loquen dins de tablespaces, per la qual cosa el primer pas per a introduir informació en una base de dades nova i independent de la resta, hauria de ser crear un primer tablespace i associar-ho a un o diversos arxius físics.

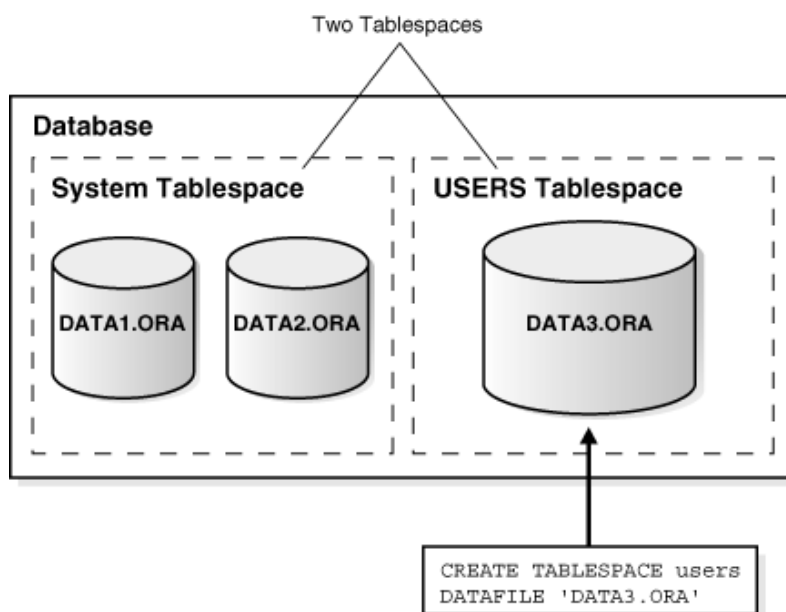
La grandària d'un tablespace és la grandària dels arxius de dades que el formen. La grandària d'una base de dades és la suma de les grandàries de tots els tablespaces que la formen.

Podem ampliar la grandària reservada per a una base de dades de Oracle de tres maneres:

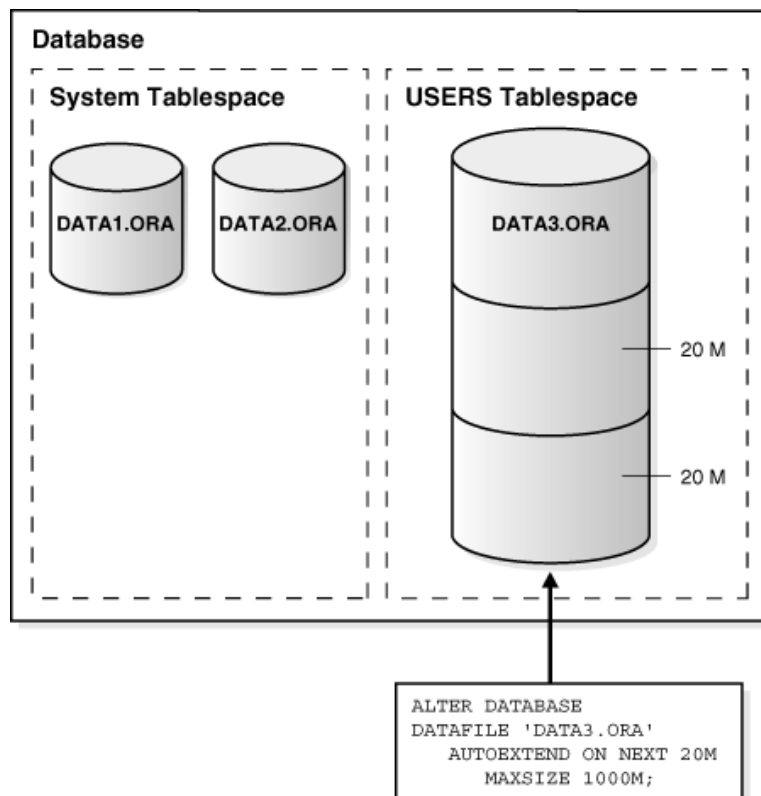
- Afegint un arxiu de dades a un tablespace



- Afegint un nou tablespace



- Augmentant la grandària d'un arxiu de dades.



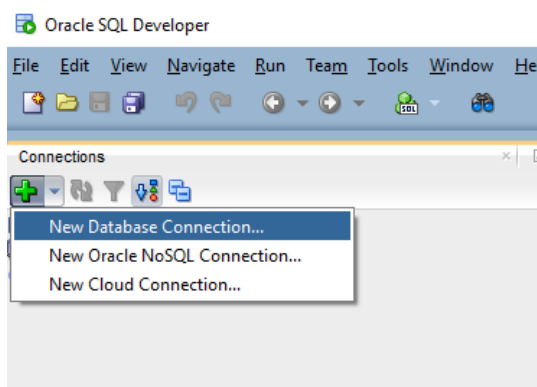
Per defecte, Oracle proporciona els següents tablespaces:

- SYSTEM: magatzem d'objectes del sistema com el diccionari de dades.
- SYSAUX: dedicat a components addicionals de la base de dades , com per exemple el repositori del Enterprise Manager
- USERS: magatzem per defecte per als diferents usuaris de la base de dades emmagatzemen els seus objectes
- TEMP: utilitzat per a emmagatzemar resultats parcials en ordenacions, operacions complexes, etc.

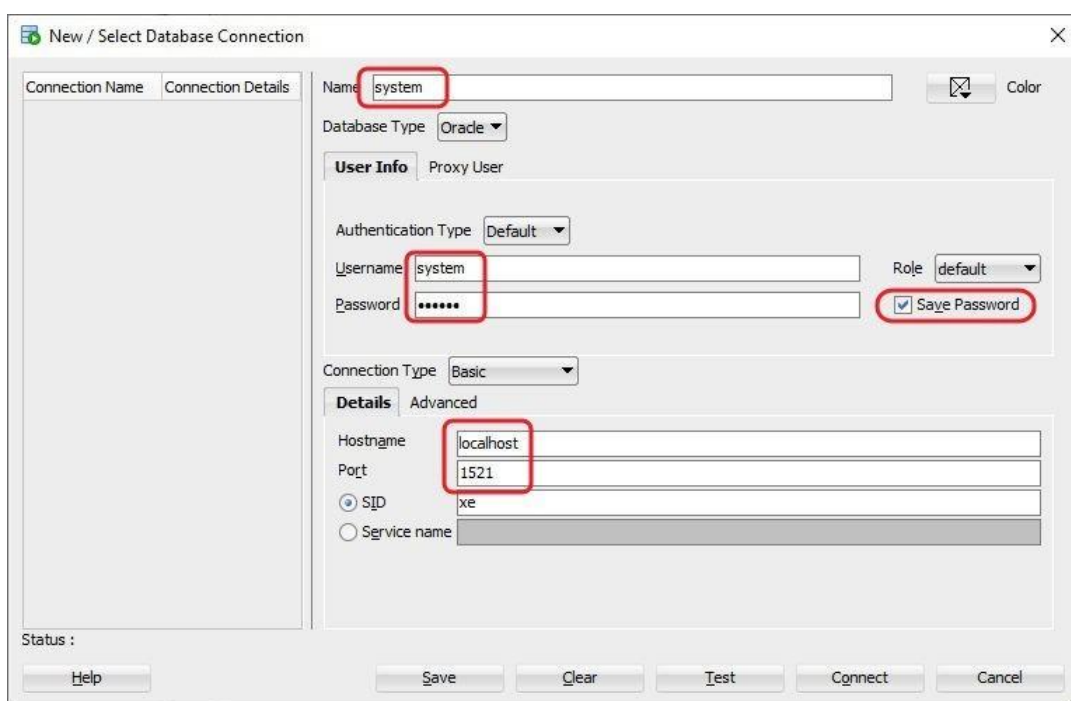
4.2.7. Connexió a la base de dades com a DBA.

En un document a part s'ha tractat la instal·lació i posada en marxa del servidor de base de dades Oracle Database Express Edition, així com l'eina gràfica SQL Developer, que ens permet interactuar de manera còmoda contra la base de dades .

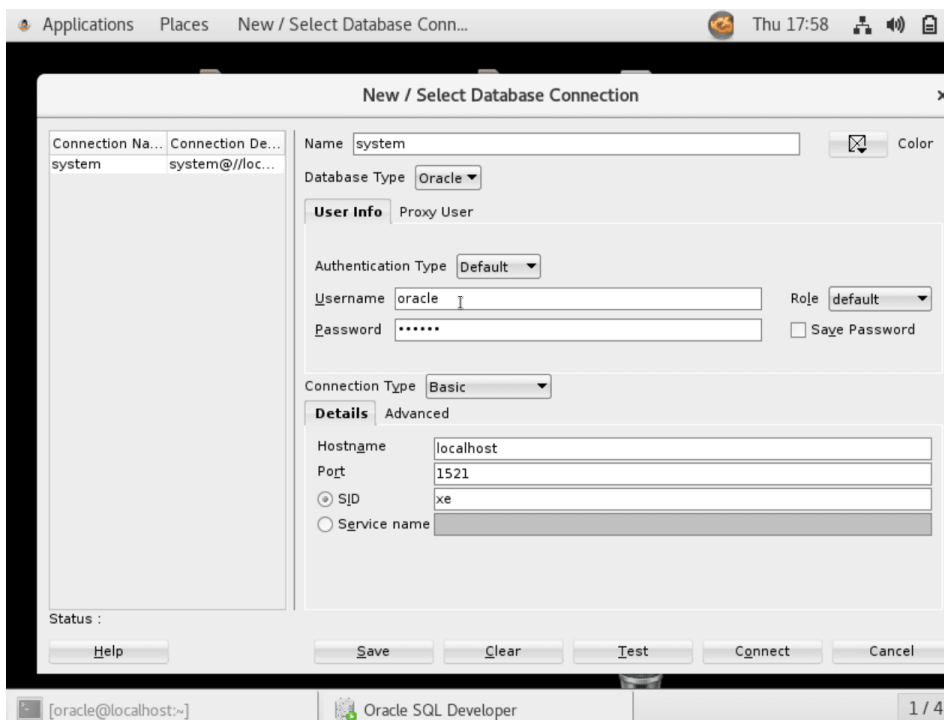
Recordem que per a crear una connexió al nostre servidor Oracle Database en SQL Developer, ho farem des del menú "New Database Connection" o directament fent clic sobre el botó "+"



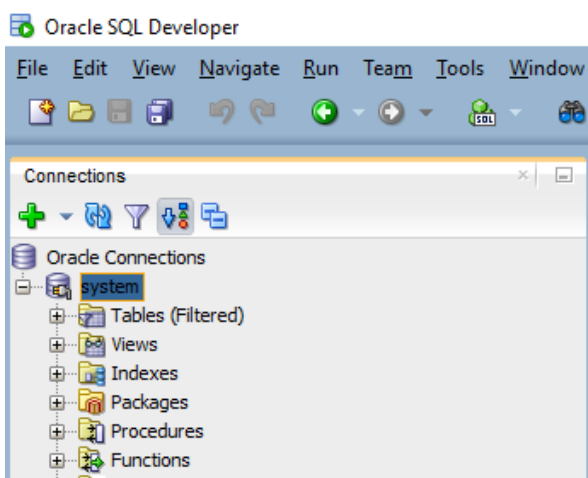
A continuació, completarem tota la informació necessària per a connectar-nos al servidor de base de dades, en aquest cas, la nostra primera connexió, utilitzarem l'usuari `system` (el que generalment direm usuari DBA)



En cas d'estar utilitzant la màquina virtual basada en Linux, és probable que l'usuari siga "oracle" i la contrasenya siga "oracle" (com se'ns mostraria en un document de text en llançar la màquina).



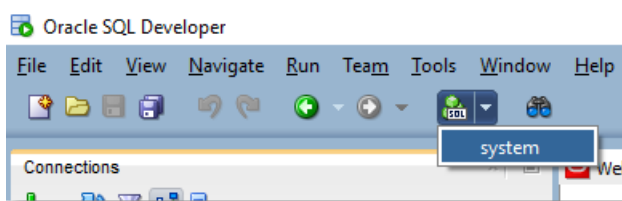
Després de guardar aquesta nova connexió, podrem treballar amb ella a través de l'arbre d'objectes.



Una vegada que tinguem aquesta connexió amb privilegis de DBA (usuari `system`), ens centrarem en crear aquells elements que ens permeten treballar contra aqueixa instància de base de dades que ja està funcionant i puguem crear taules, inserir dades en elles, etc.

4.2.8. Creació del nostre tablespace

En primer lloc crearem un tablespace que serà el nostre espai de treball. Per a això obrirem el que SQLDeveloper flama "SQL Worksheet" (una fulla de treball SQL)



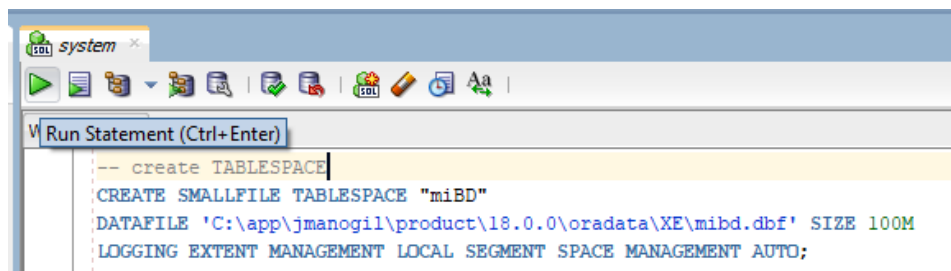
Copiarem en la nova pestanya la següent sentència, modificant la ubicació de l'arxiu perquè coincidisca amb els directoris d'instal·lació de Oracle en la nostra màquina:

```
CREATE SMALLFILE TABLESPACE "miBD"  
DATAFILE 'C:\app\jmanogil\product\18.0.0\oradata\XE\mibd.dbf'  
SIZE 100M LOGGING EXTENT MANAGEMENT LOCAL SEGMENT  
SPACE MANAGEMENT AUTO;
```

En la màquina virtual Linux no seria necessari indicar ruta per a l'arxiu:

```
CREATE SMALLFILE TABLESPACE "miBD"  
DATAFILE 'mibd.dbf'  
SIZE 100M LOGGING EXTENT MANAGEMENT LOCAL SEGMENT  
SPACE MANAGEMENT AUTO;
```

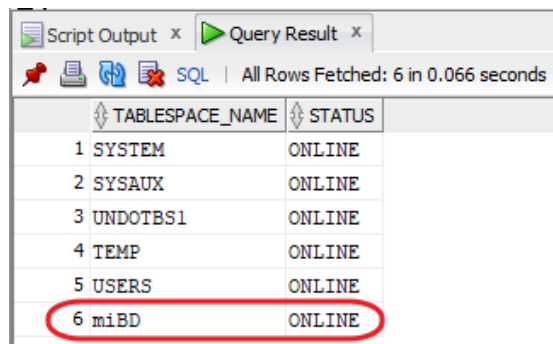
Ho executem



Amb aquesta sentència hem creat un tablespace, compost per un únic arxiu amb una grandària de 100M. Podem comprovar que s'ha creat correctament executant la sentència següent en la "SQL Worksheet"

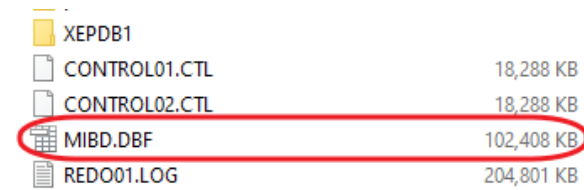
```
SELECT tablespace_name, estatus FROM dba_tablespaces;
```

Obtenint en la pestanya "Query result"



	TABLESPACE_NAME	STATUS
1	SYSTEM	ONLINE
2	SYS_AUX	ONLINE
3	UNDOTBS1	ONLINE
4	TEMP	ONLINE
5	USERS	ONLINE
6	miBD	ONLINE

Si utilitzem l'explorador d'arxius, veurem que un nou arxiu s'ha creat amb el nom mibd.dbf



Arxiu	mida
XE_PDB1	
CONTROL01.CTL	18,288 KB
CONTROL02.CTL	18,288 KB
MIBD.DBF	102,408 KB
REDO01.LOG	204,801 KB

Podem augmentar la grandària del tablespace amb la sentència

```
ALTER DATABASE
DATAFILE 'C:\app\jmanogi1\product\18.0.0\oradata\XE\mibd.dbf'
RESIZE 200M;
```

També **podríem** esborrar-ho

```
DROP TABLESPACE "miBD";
```

La sentència anterior no eliminarà l'arxiu físic del disc, per a això haurem de fer un esborrat complet amb la sentència

```
DROP TABLESPACE "miBD" INCLUDING CONTENTS AND DATAFILES;
```

Aquestes últimes sentències són irreversibles, per la qual cosa cal tindre molt clar el que s'està fent.

4.2.9. Creació del nostre usuari

Anem ara a crear un usuari de base de dades l'espai de treball de la qual per defecte siga el tablespace que hem creat. Per a això, primer executarem la següent sentència SQL (necessària des de la versió 12c per a crear aquest tipus d'usuaris)

```
ALTER SESSION SET "_ORACLE_SCRIPT"=true;
```

amb el que ja podrem crear l'usuari.

```
CREATE USER usuari  
IDENTIFIED BY contrasenya  
DEFAULT TABLESPACE "miBD"  
QUOTA UNLIMITED  
ON "miBD"  
TEMPORARY TABLESPACE "TEMP";
```

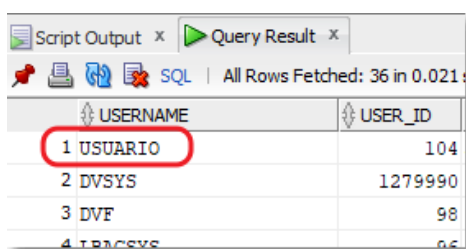
Podem usar el nom i contrasenya que vulguem. La modificació de la contrasenya es pot fer també per comandes.

```
ALTER USER usuari  
IDENTIFIED BY usuari_nou;
```

Si volem veure la llista d'usuaris, incloent el que acabem de crear, ho podem fer amb la sentència

```
SELECT * FROM all_users ORDER BY created DESC;
```

Que mostraria una pestanya "Query result"



	USERNAME	USER_ID
1	USUARIO	104
2	DVSY	1279990
3	DVF	98
4	TRCSYS	95

L'esborrat d'un usuari també és molt senzill

```
DROP USER usuari;
```

4.2.10. Rols d'usuari.

Una vegada creat el nostre primer usuari, haurem d'assignar-li una sèrie de permisos d'ús i creació sobre uns certs objectes de la base de dades, és el que anomenarem privilegis. Aquests privilegis se solen agrupar en el que es coneix com a rols, de tal manera que podrem organitzar millor aqueixa assignació de permisos als usuaris.

Quan un usuari es connecta a la base de dades, en primer lloc es comprovarà que l'usuari existeix i si la seua contrasenya és correcta. Després, es recolliran aquells privilegis que se li han concedit i sobre la base d'ells se li permetran o no fer determinades accions sobre la base de dades.

Oracle ens ofereix uns rols predefinitos

Rol	Privilegis
CONNECT	Tots els permisos necessaris per a iniciar una sessió en Oracle
RESOURCE	Tots els permisos necessaris per a la creació d'objectes (menys vistes i sinònims)
DBA	Tots els permisos per a un administrador de base de dades (DBA)

Assignarem al nostre usuari els rols CONNECT i RESOURCE utilitzant la sentència GRANT. A més, afegirem els privilegis per a crear vistes (CREATE VIEW) i sinònims (CREATE SYNONYM), ja que curiosament, no estan inclosos en RESOURCE. Si volem assignar tots els privilegis utilitzarem (ALL PRIVILEGES)

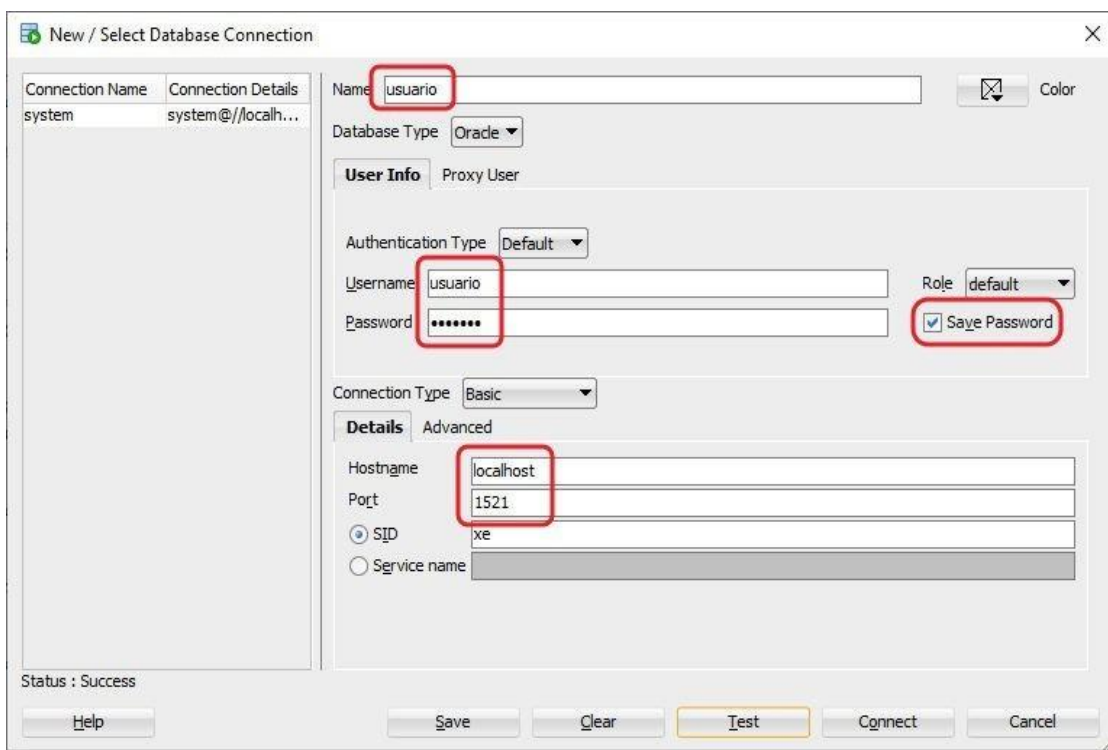
GRANT CONNECT, RESOURCE, CREATE VIEW, CREATE SYNONYM TO usuari;

Per a eliminar el rol assignat podem utilitzar la sentència REVOKE

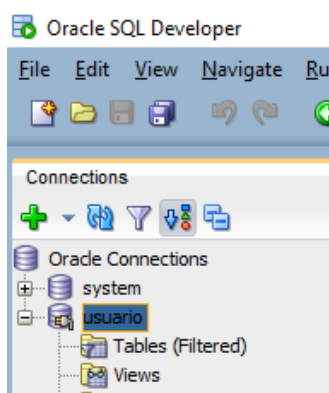
REVOKE RESOURCE, CONNECT, CREATE VIEW, CREATE SYNONYM FROM usuari;

4.2.11. Primers passos amb el nostre usuari

En aquest punt ja tenim tot el necessari per a poder treballar contra Oracle Database utilitzant el nostre propi usuari i el seu espai de treball propi. Ens toca ara comprovar que podem connectar-nos i crear objectes en la base de dades. Primer crearem una nova connexió

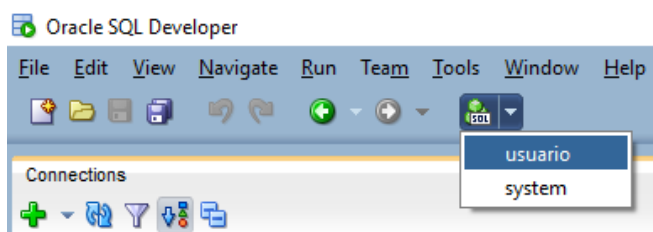


Fent clic sobre el botó "Test" podem comprovar que aqueix usuari pot connectar-se a la base de dades. Guardem la nova connexió i veurem que ara tenim dues connexions disponibles en el nostre SQL Developer



Encara que estem usant el mateix nom per a les connexions que l'usuari utilitzat en elles, no és necessari que es faça així, però sí que és cert que pot ser molt còmode fer-ho d'aquesta manera. Cal tindre en compte que les operacions que puguem realitzar amb la connexió, estaran supeditades als permisos (privilegis) que l'usuari utilitzat tinga.

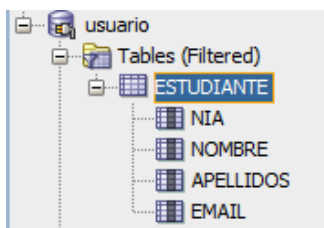
Obrim una nova "SQL Worksheet", però que utilitze la nova connexió



Crearem la nostra primera taula, per a això copiarem la següent sentència SQL i la executarem

```
CREATE TABLE estudiant (  
  nia NUMBER NOT NULL,  
  nom VARCHAR2(25),  
  cognoms VARCHAR2(50),  
  email VARCHAR2(100),  
  CONSTRAINT pk_estudiant PRIMARY KEY (nia)  
);
```

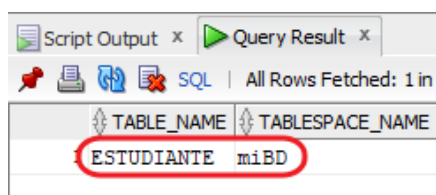
En l'arbre d'objectes podrem veure la nostra primera taula creada



També podem comprovar que s'ha creat amb la sentència

```
SELECT * FROM USER_ALL_TABLES;
```

On veurem que la nova taula s'ha creat en el tablespace que té per defecte el nostre usuari



Per a esborrar una taula usem, com en el cas dels usuaris, la sentència DROP

```
DROP TABLE estudiant;
```

4.3. SQL i el seu llenguatge de Definició de Dades (DDL)

SQL és un llenguatge per a organitzar, gestionar i recuperar dades emmagatzemades en una base de dades informàtica. El nom "SQL" és una abreviatura de Structured Query Language (Llenguatge Consultes Estructurades). És un llenguatge informàtic que es pot utilitzar per a interaccions amb una base de dades de tipus relacional.

SQL s'utilitza per a controlar totes les funcions que un sistema de gestió de base de dades (SGBS, DBMS en anglés) proporciona als seus usuaris, incloent:

- Definició de dades: permet a un usuari definir l'estructura i organització de les dades emmagatzemades i les relacions existents entre ells.
- Recuperació de dades: permet a un usuari o aplicació recuperar les dades emmagatzemades de la base de dades i utilitzar-los.
- Manipulació de dades: permet a un usuari o aplicació actualitzar la base de dades afegint noves dades, suprimint dades antigues i modificant dades prèviament emmagatzemades.
- Control d'accés: pot ser utilitzat per a restringir la capacitat d'un usuari per a recuperar, afegir i modificar dades, protegint així les dades emmagatzemades enfront de accessos no autoritzats.
- Compartició de dades: s'utilitza per a coordinar la compartició de dades per part d'usuaris concurrents, assegurant que no interfereixen els uns amb els altres.
- Integritat de dades: defineix restriccions d'integritat en la base de dades, protegint-la contra corrupcions degudes a actualitzacions inconsistents o a fallades del sistema.

4.3.1. Llenguatges SQL

Tradicionalment les sentències SQL s'han agrupat en quatre grups principals:

- DDL (llenguatge de definició de dades): possibilita el crear, modificar i eliminar objectes. Els més importants són CREATE, ALTER i DROP
- DML (llenguatge de manipulació de dades): permeten recuperar, inserir, actualitzar i eliminar files d'una taula. Les quatre sentències DML bàsiques serien SELECT, INSERT, UPDATE i DELETE
- TCL (llenguatge de control de transaccions): permeten controlar el resultat d'una transacció en una base de dades. Tindríem les sentències COMMIT i ROLLBACK

- DCL (llenguatge de control de dades): utilitzat per a controlar l'accés dels usuaris als objectes de la base de dades. Els més comuns són GRANT i REVOKE.
- Hi ha autors que distingeixen el DQL (Data Query Language), en comptes de considerar-ho part del DML.

En aquest document ens centrarem en el primer d'ells, el DDL. Per a poder comprovar que les sentències DDL s'han executat correctament, a vegades utilitzarem algunes sentències senzilles del llenguatge DML, no obstant això, seran en el tema següent on els tractarem en profunditat.

4.3.2. Tipus de dades

A l'hora de crear les taules, haurem d'indicar el tipus de dada de cadascuna de les seues columnes, per la qual cosa necessitarem conèixer els diferents tipus de dades que SQL ens ofereix. Ens centrarem en el cas de *Oracle, però existeixen xicotetes diferències entre uns gestors i altres, per la qual cosa ací tens les referències oficials dels gestors més habituals:

- Oracle: <https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlqr/Data-Types.html>
- MySQL: <https://dev.mysql.com/doc/refman/8.0/en/data-types.html>
- SQL Server: <https://docs.microsoft.com/en-us/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-ver16>
- PostgreSQL: <https://www.postgresql.org/docs/current/datatype.html>
- SQLite: <https://www.sqlite.org/datatype3.html>

Veurem els tipus de dades que es consideren més importants i que pràcticament tots els sistemes de gestió de bases de dades utilitzen, amb xicotetes variacions.

4.3.2.1. Textos

Sintaxi	Descripció	Rang
CHAR(n)	Emmagatzema textos de longitud fixa indicat pel valor de n.	Fins a 2000 bytes
NCHAR(n)	Emmagatzema textos de longitud fixa indicat pel valor de n. Pot utilitzar diferents jocs de caràcters, com UTF8	Fins a 2000 bytes
NVARCHAR2(n)	Emmagatzema textos de longitud variable, fins al nombre indicat de caràcters. Pot utilitzar diferents jocs de	Fins a 4000 bytes

	caràcters, com UTF8	
VARCHAR2(n)	Emmagatzema textos de longitud variable, fins al nombre indicat de caràcters.	Fins a 4000 bytes i 32KB en PL/SQL
LONG	Emmagatzema textos de longitud variable. Es manté per compatibilitat amb versions anteriors (preferible usar tipus LOB)	Fins a 2GB.

Cal tindre en compte que si definim una columna de tipus VARCHAR2(1000), però introduïm un text que continga solament 10 caràcters, la columna ocuparà 10 bytes, això és, la qual cosa ocupen els caràcters introduïts. No ocorre el mateix amb el tipus CHAR, ja que en aqueix cas, la resta de caràcters no utilitzats s'emplenarien amb espais.

Es recomana l'ús de NCHAR i NVARCHAR, quan es vulguen emmagatzemar textos en diferents llenguatges, en la mateixa columna o quan es vulga permetre que l'usuari introduísca el text que desitge sense cap restricció.

4.3.2.2. Números

Sintaxi	Descripció	Rang
NUMBER(p, e)	On p és la precisió i e la escala	Precisió d'1 a 38 i escala de -84 a 127.
INTEGER	Equival a NUMBER(38, 0)	
DECIMAL(p, e)	Equival a NUMBER(p, e)	

El tipus de dada NUMBER és el que normalment utilitzarem per a representar valors numèrics que poden ser positius i negatius. És cert que Oracle ofereix més varietat de tipus de dades numèriques, però quasi tots acaben representant-se internament com del tipus NUMBER.

Per a aquesta mena de dada podem especificar precisió (nombre de dígitos d'un número) i escala (nombre de dígitos a la dreta del punt decimal). Per exemple, el número 123.45 té una precisió de 5 i una escala de 2, per la qual cosa per a emmagatzemar aquest número necessitem un NUMBER(5, 2). Si no indiquem precisió i/o escala, prendrien els valors màxims possibles.

En el cas d'inserir un número que excedeix la precisió definida, Oracle llançarà un error. No obstant això, si el número excedeix l'escala, s'arredoneix. Per exemple, si la columna és del tipus NUMBER(5,1) i inserim el valor 123.45, el valor finalment inserit és 123.5.

També podem utilitzar escales amb valor negatiu, amb l'efecte d'un arrodoniment cap a la part sencera. Per exemple, amb una columna del tipus NUMBER(4, -1) en inserir el valor 123.5, estarem inserint 120.

El tipus de dada BOOLEAN no existeix en Oracle, en el seu lloc usarem NUMBER(1).

4.3.2.3. Dates i hores

Sintaxi	Descripció	Rang
DATE	Emmagatzema data i hora: segle, any, mes, dia, hora, minuts i segons	Des d'1 de Gener - 4712 fins a 31. de Desembre 9999
TIMESTAMP(p)	Ampliació del tipus DATE, però emmagatzema també fraccions de segons. El valor p ens indica el nombre de dígitos en la part fraccional dels segons, pot prendre el valor entre 0 i 9, per defecte 6.	
INTERVAL YEAR (p) TO MONTH	Període de temps en termes de anys i mesos, on p indica el nombre de dígitos de l'any	
INTERVAL DAY (pd) TO SECOND	Període de temps en termes de dies, hores, minuts i segons, on pd indica el nombre de dígitos del dia i ps el dels segons	

El format per defecte per a les dates en Oracle SQL és 'DD-MES-YYYY', per exemple: '01-DEC-2015'. No obstant això recomanem utilitzar el format de dates ISO: 'YYYY-MM-DD'

Un valor de tipus DATE sense el component d'hora, se li assignarà l'hora per defecte 12.00:00 AM. Si per contra no s'inclou el component data, això és, solament l'hora, se li assigna el primer dia del mes actual.

El tipus INTERVAL resulta molt útil quan es vol emmagatzemar un període concret de temps entre dues dates i hores. Podem treballar amb intervals any-mes, expressant períodes de temps en termes d'anys i un nombre de mesos, i amb intervals dia-hora, que ens permeten una major precisió emmagatzemant períodes en termes de dies, hores, minuts i segons. Cal tindre en compte que els intervals pots ser positius o negatius.

4.3.2.4. Objectes de gran grandària

Oracle ofereix tres tipus de dades per a emmagatzemar objectes de gran grandària, són els tipus de dades LOB (Large Object)

Sintaxi	Descripció	Rang
BLOB	Emmagatzema dades binàries sense estructura	Fins a 4GB.
CLOB	Emmagatzema text	Fins a 4GB.
NBLOB	Emmagatzema text amb diferents jocs de caràcters.	Fins a 4GB.
BFILE	Emmagatzema un localitzador que apunta a un arxiu binari fora de la base de dades	Fins a 4GB.

Aquests tipus de dades ens permeten emmagatzemar grans estructures d'informació com a text, gràfics, àudio i vídeo, d'aquesta manera, arxius multimèdia poden residir en la mateixa base de dades.

4.3.2.5. Conversió automàtica des de ANSI SQL

Alguns tipus de dades es pot indicar també seguint l'estàndard ANSI SQL i es convertiran automàticament a la mena de dades equivalent de Oracle:

ANSI SQL Data Type	Oracle Data Type
CHARACTER(n) CHAR(n)	CHAR(n)
CHARACTER VARYING(n) CHAR VARYING(n)	VARCHAR2(n)
NATIONAL CHARACTER(n) NATIONAL CHAR(n) NCHAR(n)	NCHAR(n)
NATIONAL CHARACTER VARYING(n) NATIONAL CHAR VARYING(n) NCHAR VARYING(n)	NVARCHAR2(n)
NUMERIC[(p,s)] DECIMAL[(p,s)]	NUMBER(p,s)
INTEGER INT SMALLINT	NUMBER(38)
FLOAT DOUBLE PRECISION REAL	FLOAT(126) FLOAT(126) FLOAT(63)

4.4. Creació de Taules

Després de conèixer els principals tipus de dades que Oracle SQL ens ofereix, és hora de centrar-nos en la creació de taules, formades per columnes que definirem sobre la base de aqueixos tipus de dades.

Per a crear una taula utilitzarem la sentència CREATE TABLE, amb la següent sintaxi

```
CREATE TABLE nom_taula( nom_columna_1 tipus_date
  [NOT NULL] [UNIQUE] [PRIMARY KEY]
  [DEFAULT valor]
  [REFERENCES nom_taula] [columna[,columna]]
  [ON DELETE CASCADE]] [CHECK condició],
```

```

...
{CONSTRAINT ck_nom CHECK (condició)}
{CONSTRAINT uk_nom UNIQUE(columnes)}
{CONSTRAINT pk_nom PRIMARY KEY(columnes_clau)}
{CONSTRAINT fk_nom FOREIGN KEY(columnes)
REFERENCES taula(columnes_clau)
{ON DELETE CASCADE}}
)

```

Vegem un exemple

```

CREATE TABLE client(
  cif VARCHAR2(10) NOT NULL,
  nom VARCHAR2(20),
  direccio VARCHAR2(50),
  ciutat VARCHAR2(20),
  cp VARCHAR2(5) DEFAULT '00000',
  detalls VARCHAR2(1000),
  CONSTRAINT pk_client PRIMARY KEY(cif)
);

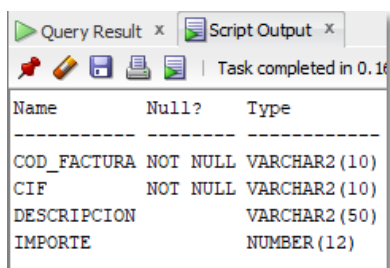
CREATE TABLE factura(
  cod_factura VARCHAR2(10) NOT NULL,
  cif VARCHAR2(10) NOT NULL,
  descripcio VARCHAR2(50),
  import NUMBER(12),
  CONSTRAINT pk_factura PRIMARY KEY(cod_factura),
  CONSTRAINT fk_factura_client FOREIGN KEY(cif)
REFERENCES client(cif)
ON DELETE CASCADE
);

```

Amb la sentència DESCRIBE obtenim l'estructura d'una taula de manera ràpida, per exemple

DESCRIBE factura;

Ens mostra



Name	Null?	Type
COD_FACTURA	NOT NULL	VARCHAR2(10)
CIF	NOT NULL	VARCHAR2(10)
DESCRIPCION		VARCHAR2(50)
IMPORTE		NUMBER(12)

També podem en SQL Developer anar a l'arbre d'objectes, seleccionar la taula i ens apareixerà una nova pestanya amb multitud de detalls

FACTURA						
Columns Data Model Constraints Grants Statistics Triggers Flashback Dependencies Details Partitions Indexes SQL						
Actions...						
	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	COD_FACTURA	VARCHAR2(10 BYTE)	No	(null)	1 (null)	
2	CIF	VARCHAR2(10 BYTE)	No	(null)	2 (null)	
3	DESCRIPCION	VARCHAR2(50 BYTE)	Yes	(null)	3 (null)	
4	IMPORTE	NUMBER(12,0)	Yes	(null)	4 (null)	

Les restriccions o CONSTRAINT es poden definir tant a nivell de columna o taula, amb l'excepció de NOT NULL que solament s'accepta en la definició de la columna.

Els tipus de CONSTRAINT de columna inclouen

- NOT NULL: la columna no accepta valors nuls (NULL)
- UNIQUE: el contingut d'aqueixa columna o columnes no pot repetir valors
- PRIMARY KEY: els valors en aqueixa columna o columnes identifiquen de manera única una fila en la taula, fent que els camps siguin NOT NULL i UNIQUE
- FOREIGN KEY: els valors de la columna o columnes han de correspondre's amb valors en les claus primàries en la taula referenciada
- CHECK: estableix restriccions de validació que han de complir-se en cada fila.

Per exemple

```
CREATE TABLE proveidor(
  codi NUMERIC(4) PRIMARY KEY,
  nom VARCHAR2(50) NOT NULL,
  CONSTRAINT ck_codi CHECK (codi BETWEEN 5000 AND 9999),
  CONSTRAINT ck_nom CHECK (nom = UPPER(nom)),
  CONSTRAINT uk_nom UNIQUE(nom)
);
```

En la taula anterior les restriccions ens impedeixen inserir codis de proveïdor fora del rang entre 5000 i 9999. A més el nom no pot ser NULL, no es pot repetir i ha d'estar en majúscules. Es produirà un error en totes i cadascuna de les següents insercions (la sintaxi de l'orde INSERT es veurà amb més detall en el tema 5):

```
INSERT INTO proveidor VALUES (1, 'ACME');
INSERT INTO proveidor VALUES (5500, NULL);
INSERT INTO proveidor VALUES (5500, 'acme');
```

La següent inserció es farà correctament

```
INSERT INTO proveidor VALUES (5500, 'ACME');
```

Podem comprovar-ho amb la sentència

```
SELECT * FROM proveidor;
```

Però les següents també produeixen un error

```
INSERT INTO proveïdor VALUES (5500, 'TIA');  
INSERT INTO proveïdor VALUES (5501, 'ACME');
```

La restricció UNIQUE, a diferència de la PRIMARY KEY, permet que les columnes que ho formen puguin contindre valors NULL, sempre que la combinació de valors siga única. Oracle no permet crear una PRIMARY KEY i una restricció UNIQUE amb les mateixes columnes.

Les FOREIGN KEYS o claus alienes obliguen al compliment del que en bases de dades relacionals diem integritat referencial. Aquesta integritat, per exemple en el cas de les taules `client` i `factura`, obliga al fet que tota factura que es cree, el CIF que emmagatzeme tinga un valor que existisca en la taula referenciada `client`, en cas contrari es produiria un error. A més, en l'exemple afegim la sentència `ON DELETE CASCADE`, aconseguint que quan un client s'elimine de la seua taula, totes les factures que tinguera aqueix client també s'eliminaran automàticament.

Si no indiquem aqueix `ON DELETE CASCADE` en la FOREIGN KEY `fk_factura_client` i intentem per exemple eliminar un client amb factures associades, es produirà un error. De la mateixa manera que també es produiria un error quan intentem inserir una factura amb un CIF que no pertanga a cap client, en tots dos casos s'està intentant violar la integritat referencial establida en la base de dades.

Inserirem un registre en la taula `client` i un altre en `factura` amb les sentències:

```
INSERT INTO client (cif, nom)  
VALUES ('12345678A', 'Luis Rodriguez');  
INSERT INTO factura (cod_factura, cif, import)  
VALUES ('1A', '12345678A', 1000);
```

A causa de la FOREIGN KEY que associa `factura` i `client`, la següent sentència ens produiria un error

```
INSERT INTO factura (cod_factura, cif, import)  
VALUES ('2A', '87654321A', 1000);
```

És molt útil parar atenció al missatge d'error que Oracle ens retorna, us donarà moltes pistes.

Tampoc podríem modificar el CIF del client associat amb la factura. La següent sentència també produiria un error

```
UPDATE client  
SET cif = '87654321B'  
WHERE nom = 'Luis Rodriguez';
```

Què succeeix si elimine el client amb CIF '12345678A'? Comprovem-ho amb la següent sentència

```
DELETE client WHERE CIF = '12345678A';
```

La factura associada a aqueix client s'ha eliminat automàticament.

Cal tindre en compte que Oracle Database ens ofereix solament dues possibilitats que podem usar amb les claus alienes:

- ON DELETE SET NULL: actualitza a NULL les claus relacionades amb l'esborrada.
- ON DELETE CASCADE: esborra totes les files quan s'eliminen les files de la taula referenciada

A vegades pot interessar crear una taula partint d'una altra taula ja existent, copiant les seues columnes i totes o part de les seues files, per a això utilitzarem la sentència CREATE TABLE AS. Vegem un exemple utilitzant l'anterior taula `proveidor` i creant una nova a partir de ella

```
CREATE TABLE proveidor_5000 AS  
(SELECT codi, nom FROM proveidor WHERE codi >= 5000);
```

En posteriors documents aprofundirem més en la sentència SELECT que, com hem vist, ens permet filtrar les files i columnes d'una taula.

Per a acabar amb la creació de taules, comentarem el cas de les columnes del tipus IDENTITY. Aquest tipus de columnes es van introduir a partir de la versió 12c de Oracle i ens permeten crear fàcilment columnes que emmagatzemen números que automàticament s'incrementen amb cada inserció. Això resulta molt útil quan necessitem crear un camp clau i no necessitem donar-li un valor en concret, és suficient que el SGBD s'encarregue d'això.

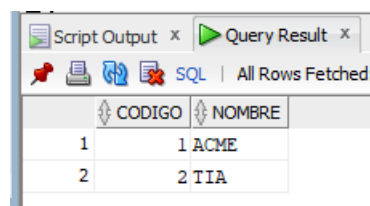
Fins a la versió 12c, podíem aconseguir el mateix resultat emprant el que Oracle denomina SEQUENCES, no obstant això és molt més senzill treballar amb aquesta mena de columna.

Vegem com utilitzar aquesta columna IDENTITY creant una variant de la taula `proveidor` i inserint algunes files

```
CREATE TABLE proveidor_identity(  
  codi NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
  nom VARCHAR2(50) NOT NULL  
);  
INSERT INTO proveidor_identity (nom) VALUES ('ACME');  
INSERT INTO proveidor_identity (nom) VALUES ('TIA');
```

Si consultem el contingut de la taula, veurem que el camp `codi` té un valor que nosaltres no hem assignat, s'ha fet automàticament

```
SELECT * FROM proveidor_identity;
```



CODIGO	NOMBRE
1	1 ACME
2	2 TIA

4.5. Modificació de Taules

A vegades pot ser necessari modificar l'estructura d'una taula ja creada. Si la taula ja s'ha carregat amb dades, l'opció d'esborrar-la i tornar a crear-la no és convenient. Per a això tenim la sentència ALTER TABLE, amb la sintaxi

```
ALTER TABLE nom_taula {  
  ADD [ COLUMN ] definicio_columna |  
  MODIFY [ COLUMN ] definicio_columna  
  ADD CONSTRAINT restriccio |  
  DROP [ COLUMN ] nom_columna  
    [ CASCADE | RESTRICT ]  
  DROP { PRIMARY KEY | FOREIGN KEY nom | UNIQUE nom  
    | CHECK nom | CONSTRAINT nom }  
}
```

Vegem exemples dels canvis més habituals

4.5.1. Canviar de nom la taula

```
ALTER TABLE client RENAME TO client_empresa;
```

4.5.2. Afegir columnes

```
ALTER TABLE client_empresa ADD (data_alta DATE);
```

4.5.3. Modificar columnes

```
ALTER TABLE client_empresa MODIFY (  
  data_alta TIMESTAMP,  
  nom VARCHAR2(100) NULL,  
  direccio VARCHAR2(100) NOT NULL);
```

Lògicament, els canvis que es poden fer en les columnes seran possibles si les dades que ja estan en la taula ho permeten. Per exemple, si intentem modificar una columna tipus VARCHAR(50) a VARCHAR(10) i hi ha valors amb més de 10 caràcters, ens donarà error.

4.5.4. Esborrar columnes

```
ALTER TABLE client_empresa DROP (data_alta);
```

4.5.5. Canviar de nom columnes

```
ALTER TABLE client_empresa  
RENAME COLUMN nom TO nom_complet;
```

4.5.6. Esborrar una CONSTRAINT (també FOREIGN KEY)

```
ALTER TABLE factura  
DROP CONSTRAINT fk_factura_client;
```

4.5.7. Afegir una FOREIGN KEY

```
ALTER TABLE factura  
ADD CONSTRAINT fk_factura_client  
FOREIGN KEY(CIF)  
REFERENCES client_empresa(CIF) ON DELETE CASCADE;
```

4.5.8. Afegir una CHECK CONSTRAINT

```
ALTER TABLE factura ADD CONSTRAINT ck_import CHECK(import > 1000);
```

4.6. Buidar Taules

Disposen d'una sentència per a buidar el contingut d'una taula, això és, tots els seus registres sense afectar l'estructura d'aquesta. El seu ús és molt senzill

```
TRUNCATE TABLE factura;
```

4.7. Índexs

Els índexs són objectes que acceleren les operacions de consulta i ordenació sobre determinades columnes en unes determinades taules. Són estructures de dades que s'emmagatzemen a part de la taula a la qual referencien, per la qual cosa es poden crear i esborrar en qualsevol moment.

Es basen a crear i mantindre una llista ordenada, que és la que Oracle utilitzarà per a fer les cerques i ordenacions. Una cerca en una llista ordenada sempre és més ràpida que si les dades no segueixen cap ordre.

Cada vegada que s'afeg un nou registre, els índexs involucrats s'actualitzen, d'aquí ve que quants més índexs tinguem, més li costa a Oracle (o el SGBD amb el qual treballem) afegir registres, però més ràpides es realitzen les consultes.

Les restriccions PRIMARY KEY, UNIQUE i FOREIGN KEY, creen automàticament índexs associats. Es tracta d'índexs obligatoris que el SGBD necessita, però podem crear nous índexs són la sintaxi

```
CREATE [UNIQUE] INDEX nom_index  
ON nom_taula (columna_1, columna_2, ... columna_n)
```

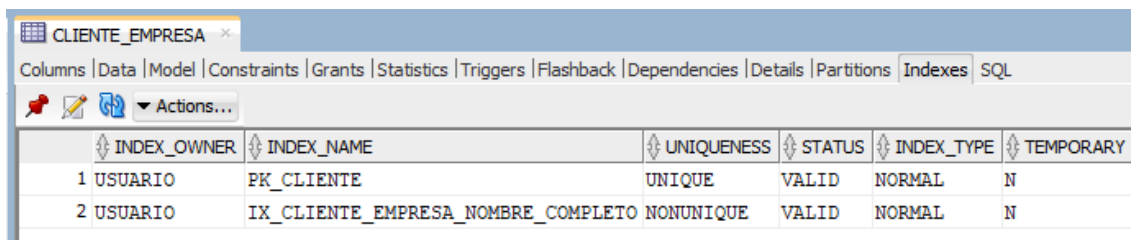
Un índex pot fer referència a valors que es puguem repetir, cal no confondre'l amb una clau primària. A més pot estar format per més d'una columna.

Per exemple

```
CREATE INDEX ix_client_empresa_nom_complet  
ON client_empresa (nom_complet);
```

Amb aquest índex aconseguirem que les cerques en la taula, a través del camp nom_complet, siguem més ràpides.

En la fitxa de la taula en SQL Developer, pestanya "Indexes", podem comprovar que l'índex s'ha creat.



	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY
1	USUARIO	PK_CLIENTE	UNIQUE	VALID	NORMAL	N
2	USUARIO	IX_CLIENTE_EMPRESA_NOMBRE_COMPLETO	NONUNIQUE	VALID	NORMAL	N

Una vegada creat un índex, Oracle no permet modificar-lo afegint-li més columnes, haurem d'eliminar-lo i tornar-lo a crear de nou. L'esborrat es realitza amb la sentència DROP, per exemple

```
DROP INDEX ix_client_empresa_nom_complet;
```

4.8. Vistes

Una vista o VIEW, no és més que una consulta SELECT emmagatzemada en la base de dades amb un nom, de tal manera que podrem obtenir el resultat d'aqueixa consulta, tantes vegades com es desitge, fent referència a aqueix nou objecte VIEW creat.

Per exemple , en l'apartat dedicat a la creació de taules, hem creat una taula a partir de una consulta

```
CREATE TABLE proveidor_5000 AS  
(SELECT codi, nom FROM proveidor WHERE codi >= 5000);
```

Crear una vista a partir de aqueixa SELECT hauria sigut una opció també

```
CREATE VIEW v_proveidor_5000 AS  
(SELECT codi, nom FROM proveidor WHERE codi >= 5000);
```

I la consulta d'aqueixa vista seria molt senzilla

```
SELECT * FROM v_proveidor_5000;
```

Mentre que la taula `proveidor_5000` conté les dades que hagueren en la taula `proveidor` just quan es va crear, la vista `v_proveidor_5000` ens retorna les dades que hagueren en `proveidor` en el moment just de fer la consulta, contra la vista.

Una vista no conté dades, sinó la instrucció `SELECT` necessària per a generar els resultats, per la qual cosa les vistes ocupen molt poc espai en disc. Se solen emprar per a

- Realitzar consultes complexes més fàcilment, ja que permeten dividir la consulta en diverses parts
- Proporcionar resultats amb dades complexes
- Utilitzar visions especials de les dades
- Server com a taules que resumeixen de totes les dades

Cal tindre en compte que una vista pot ser tan complexa com la `SELECT` que la forme, per la qual cosa podria estar formada per dades de diferents taules, agrupacions, etc.

La sintaxi d'una `VIEW` és

```
CREATE OR REPLACE VIEW nom_vista AS consulta;
```

L'esborrat és com en la resta d'objectes, amb la sentència `DROP`

```
DROP VIEW v_proveidor_5000;
```

Aprofundirem més el tema de les vistes quan estudiem les consultes avançades en el llenguatge `DML`.

4.9. Sinònims

Un sinònim o SYNONYM és un nom alternatiu per a un objecte de la base de dades . Se sol utilitzar per a taules o vistes, el que es puga accedir a elles a través d'un altre nom pot ser útil a vegades .

La sintaxi seria

```
CREATE SYNONYM nom_sinònim FOR objecte;
```

Per exemple, hi ha gent que està acostumat a utilitzar plurals per a nomenar les taules, podríem crear un sinònim per a la taula `proveïdor`

```
CREATE SYNONYM proveïdors FOR proveïdor;
```

Amb això la consulta següent, en realitat està buscant les dades en `proveïdor`

```
SELECT * FROM proveïdors;
```

4.10. Comentaris

Per a marcar un text com a comentari en SQL usarem el doble guió "--", per a un comentari d'una única línia i "/" "/" per a una o diverses línies.

```
-- Això és un comentari  
/ Això també /  
/  
  Això també  
/
```

4.11. Comprovar objectes creats

Oracle ens ofereix unes vistes predefinides que ens permeten consultar els objectes que anem creant en la base de dades . De fet, ja usem en un apartat anterior la següent consulta que ens mostra les taules que hem creat:

```
SELECT * FROM USER_ALL_TABLES;
```

Restriccions

```
SELECT * FROM USER_CONSTRAINTS;
```

Índexs

```
SELECT * FROM USER_INDEXES;
```

Tots els objectes

```
SELECT * FROM USER_OBJECTS;
```

Sinònims

```
SELECT * FROM USER_SYNONYMS;
```

Columnes de taules i vistes

```
SELECT * FROM USER_TAB_COLUMNS;
```

Vistes

```
SELECT * FROM USER_VIEWS;
```