Instrucciones SQL

Laboratorio Base de Datos I.

Objetivos de la Clase

- SQL.
- Tipos de sentencias SQL
- Describir los tipos de dato que se pueden utilizar al especificar la definición de columnas.
- Sentencias SQL Select.

Lenguaje SQL

- SQL: Lenguaje de consulta estructurado o SQL (por sus siglas en inglés **structured query language**) es un lenguaje de programación que permite realizar diferentes operaciones sobre bases de datos relacionales.

- En la actualidad el SQL ANSI es el estándar *de facto* de la inmensa mayoría de los SGBD comerciales.

- Otro lenguaje de acceso a datos: LINQ (**Language Integrated Query):** toma la estructura y el modelo de la base de datos relacional y las convierte en un modelo de objetos a través de LINQ to SQL.

- **Transact-SQL** es el lenguaje que se utiliza para administrar instancias del **SQL Server Database Engine** (Motor de base de datos de SQL Server), para crear y administrar objetos de base de datos, y para insertar, recuperar, modificar y eliminar datos.

Lenguaje SQL puro (ANSI)
Clasificación

- DCL: Control de datos
  - Grant
  - Revoke
- DDL: Definición de datos
  - Create
  - Alter
  - Drop
- DML: Manipulación de datos
  - Select
  - Insert
  - Update
  - Delete
  - Manipulación de Transacciones:
    - BEGIN TRANSAC
    - COMMIT,ROLLBACK

Lenguaje SQL extendido

- Transact SQL, PL/SQL
- Variables
  - Declare
  - Set
  - Select
- Operadores
  - Aritméticos
  - Lógicos
  - Comparativos
  - De Concatenación

Lenguaje SQL extendido

- Comentarios
  - De Línea: --
  - De Bloque: /* */
- Control de Flujo
  - Begin / End
  - If / Else
  - Return
  - While
  - Case

Escritura de Sentencias SQL

- Las sentencias SQL no son sensibles a mayúsculas/minúsculas.
- Las sentencias SQL pueden ocupar una o más líneas.
- Las palabras claves no se pueden abreviar ni dividir entre líneas.
- Las cláusulas suelen estar colocadas en líneas separadas.
- Los sangrados se utilizan para mejorar la legibilidad.

Tipos de Datos en SQL Server

- Las tablas tienen columnas, y las columnas se definen en base a un tipo de datos; los tipos de datos acotan el **tipo y tamaño** de la información que se guardará en una columna.
- La importancia de la elección de los tipos de datos reside en el almacenamiento que ocupa; para varios cientos de filas, el tamaño no es tan crucial, pero cuantas más filas se añadan a la tabla, mayor será la repercusión en el rendimiento de las operaciones de E/S.
- **Cada RDBMS maneja sus propios tipos de datos.**

Tipos de Datos en SQL Server

Tipos de datos numéricos exactos que utilizan datos enteros.

Tipo de datos	Intervalo	Almacenamiento
bigint	De -2^63 (-9.223.372.036.854.775.808) a 2^63-1 (9.223.372.036.854.775.807)	8 bytes
int	De -2^31 (-2.147.483.648) a 2^31-1 (2.147.483.647)	4 bytes
smallint	De -2^15 (-32.768) a 2^15-1 (32.767)	2 bytes
tinyint	De 0 a 255	1 byte

Tipos de Datos en SQL Server

Tipos de datos numéricos que tienen precisión y escala fijas.

decimal[(p[,s])] y **numeric[(p[,s])]**

Cuando se utiliza la precisión máxima, los valores permitidos están comprendidos entre - 10^38 +1 y 10^38 - 1. Numeric equivale funcionalmente a decimal.

p (precisión)
El número total máximo de dígitos decimales que se puede almacenar, tanto a la izquierda como a la derecha del separador decimal. 1=< p <=38 . La precisión predeterminada es 18.

s (escala)
El número máximo de dígitos decimales que se puede almacenar a la derecha del separador decimal. La escala debe ser un valor comprendido entre 0 y p. Para especificar la escala es necesario haber especificado la precisión.

Ej: Decimal(8,2). 8 enteros y 2 decimales.

Tipos de Datos en SQL Server

Tipos de datos de caracteres

Char(n) y **varchar(n)**
Los tipos de datos caracter se puede definir de longitud fija y de longitud variable.

Los de longitud fija son **char(n)** y su tamaño lo define el valor que tenga n. Por ejemplo, una columna char(15) ocupa 15 bytes.

Los de longitud variable son **varchar(n)**, y su tamaño lo define la logitud de la columna guardada; por ejemplo una columna varchar(250), que guarda el valor "columna variable" el almacenamiento que ocupa es 16 bytes.

En caso de desear valores Unicode, deberás anteponer al tipo de datos la letra n, siendo los tipos nchar, o nvarchar. La principal diferencia con los tipos de datos no-unicode, es que utilizan el doble de bytes. Por ejemplo, el texto "Tutorial", en una columna varchar(100) ocuparía 8 bytes, mientras que siendo unicode ocuparía 16 bytes.

Ej: char(20) o varchar(20). Permite guardar hasta 20 caracteres.

Tipos de Datos en SQL Server

Tipos de datos fecha

smalldatetime y **datetime**
Son los tipos de datos utilizados para representar la fecha y la hora. El valor internamente se almacena como un valor integer, y dependiendo de la precisión utilizará 4 u 8 bytes

Tipo de datos	Intervalo	Precisión
datetime	Del 1 de enero de 1753 hasta el 31 de diciembre de 9999	3,33 milisegundos
smalldatetime	Del 1 de enero de 1900 hasta el 6 de junio de 2079	1 minuto

Tipos de Datos en SQL Server

Valores NULL

- **Indica que el valor es desconocido**.
- No hay dos valores NULL que sean iguales.
- La comparación entre dos valores NULL, o entre un valor NULL y cualquier otro valor, tiene un resultado desconocido porque el valor de cada NULL es desconocido.
- Cuando se ven los resultados de la consulta en el Editor de código de SQL Server Management Studio, los valores NULL se muestran como **(null)** en el conjunto de resultados.

Tipos de Datos en SQL Server

Otros Tipos de datos

tipos definidos de usuario: suele ayudar para unificar el diseño de las tablas; por ejemplo, se puede crear un tipo llamado NIF que corresponde al tipo de datos CHAR(20), y admite valores nulos.

CREATE TYPE NIF FROM char(20) NULL

XML
A partir de la versión 2005 de SQL Server incorpora el tipo de datos nativo XML. El tipo de datos obliga a que el dato sea por lo menos bien formado (well-formed). Adicionalmente, la columna puede asociarse a un esquema XSD.

Tipos de datos definidos de usuario en .NET
La integración del CLR, permite la posibilidad de definir tipos de datos con cualquier lenguaje .NET.

Instrucciones SQL

SENTENCIAS DDL

Objetivos de la Clase

- Sentencias SQL Create, Insert, Update, Delete, Truncate, Drop.
- Modificar definiciones de tablas.
- Valores nulos
- Restricciones.

Sentencias DDL

- Data Definition Language (DDL)
  - Es el vocabulario (SQL) usado para definir las estructuras de datos en un motor de base de datos.
  - Se utilizan sentencias para CREAR (create), MODIFICAR (alter) y borrar (DROP) estructuras de datos en una BD.

Sentencias DDL

CREATE DATABASE

```
CREATE DATABASE database_name
[ CONTAINMENT = { NONE | PARTIAL } ]
[ ON
     [ PRIMARY ] <filespec> [ ,...n ]
     [ , <filegroup> [ ,...n ] ]
     [ LOG ON <filespec> [ ,...n ] ]
]
[ COLLATE collation_name ]
[ WITH  <option> [,...n ] ]
[;]

<option> ::=
{
     FILESTREAM ( <filestream_option> [,...n ] )
   | DEFAULT_FULLTEXT_LANGUAGE = { lcid | language_name |
language_alias }
   | DEFAULT_LANGUAGE = { lcid | language_name | language_alias }
   | NESTED_TRIGGERS = { OFF | ON }
   | TRANSFORM_NOISE_WORDS = { OFF | ON}
   | TWO_DIGIT_YEAR_CUTOFF = <two_digit_year_cutoff>
   | DB_CHAINING { OFF | ON }
   | TRUSTWORTHY { OFF | ON }
}

<filestream_option> ::=
{
     NON_TRANSACTED_ACCESS = { OFF | READ_ONLY | FULL }
   | DIRECTORY_NAME = 'directory_name'
}

<filespec> ::=
{
(
   NAME = logical_file_name ,
   FILENAME = { 'os_file_name' | 'filestream_path' }
   [ , SIZE = size [ KB | MB | GB | TB ] ]
   [ , MAXSIZE = { max_size [ KB | MB | GB | TB ] | UNLIMITED } ]
   [ , FILEGROWTH = growth_increment [ KB | MB | GB | TB | % ] ]
)
}

<filegroup> ::=
{
FILEGROUP filegroup name [ [ CONTAINS FILESTREAM ] [ DEFAULT ] |
CONTAINS MEMORY_OPTIMIZED_DATA ]
   <filespec> [ ,...n ]
}

<service_broker_option> ::=
{
   ENABLE_BROKER
  | NEW_BROKER
  | ERROR_BROKER_CONVERSATIONS
}
```

Sentencias DDL

- CREATE DATABASE

CREATE DATABASE database_name;

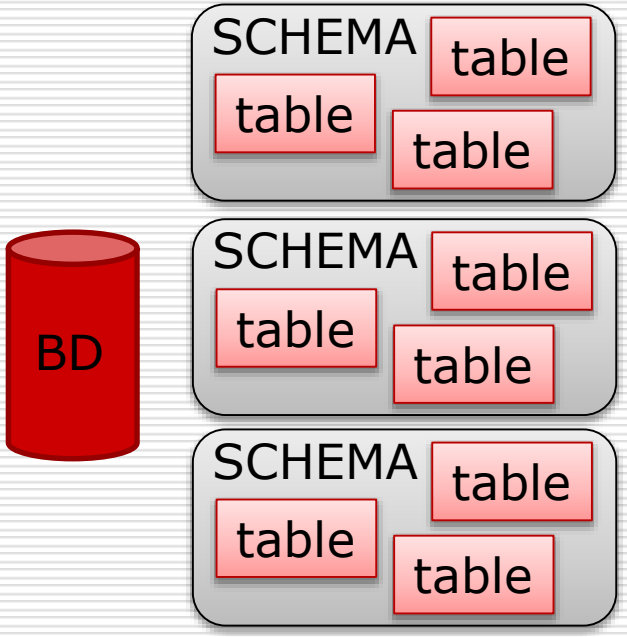
CREATE DATABASE laEmpresa;

Sentencias DDL

CREATE SCHEMA

- Un esquema simplemente es un contenedor de objetos

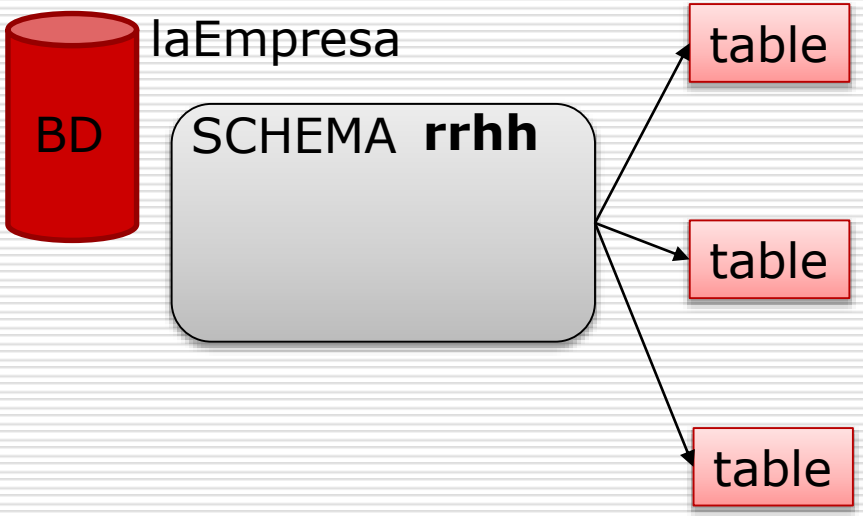
  CREATE SCHEMA nombre_schema

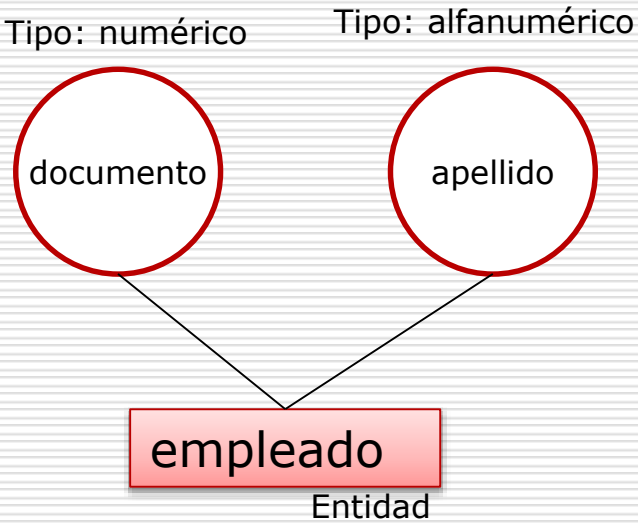



```
USE laEmpresa;
GO
CREATE DATABASE rrhh;
GO
```

Sentencias DDL

- CREATE TABLE





Sentencias DDL

CREATE TABLE

```
CREATE TABLE
    [ database_name . [ schema_name ] . | schema_name . ] table_name
    ( { <column_definition> | <computed_column_definition>
        | <column_set_definition> | [ <table_constraint> ] [ ,...n ] } )
    [ ON { partition_scheme_name ( partition_column_name ) | filegroup
        | "default" } ]
    [ { TEXTIMAGE_ON { filegroup | "default" } ]
    [ FILESTREAM_ON { partition_scheme_name | filegroup
        | "default" } ]
    [ WITH ( <table_option> [ ,...n ] ) ]
[ ; ]

<column_definition> ::=
column_name <data_type>
    [ FILESTREAM ]
    [ COLLATE collation_name ]
    [ NULL | NOT NULL ]
    [
        [ CONSTRAINT constraint_name ] DEFAULT constant_expression ]
      | [ IDENTITY [ ( seed ,increment ) ] [ NOT FOR REPLICATION ]
    ]
    [ ROWGUIDCOL ] [ <column_constraint> [ ...n ] ]
    [ SPARSE ]

<data type> ::=
[ type_schema_name . ] type_name
    [ ( precision [ , scale ] | max |
        [ { CONTENT | DOCUMENT } ] xml_schema_collection ) ]

<column_constraint> ::=
[ CONSTRAINT constraint_name ]
{     { PRIMARY KEY | UNIQUE }
        [ CLUSTERED | NONCLUSTERED ]
        [
            WITH FILLFACTOR = fillfactor
          | WITH ( < index_option > [ , ...n ] )
        ]
        [ ON { partition_scheme_name ( partition_column_name )
            | filegroup | "default" } ]
  | [ FOREIGN KEY ]
        REFERENCES [ schema_name . ] referenced_table_name [ ( ref_column ) ]
        [ ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]
        [ ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]
        [ NOT FOR REPLICATION ]
  | CHECK [ NOT FOR REPLICATION ] ( logical_expression )
}

<computed_column_definition> ::=
column_name AS computed_column_expression
[ PERSISTED [ NOT NULL ] ]
[
    [ CONSTRAINT constraint_name ]
    { PRIMARY KEY | UNIQUE }
        [ CLUSTERED | NONCLUSTERED ]
        [
            WITH FILLFACTOR = fillfactor
          | WITH ( <index_option> [ , ...n ] )
        ]
    | [ FOREIGN KEY ]
        REFERENCES referenced_table_name [ ( ref_column ) ]
        [ ON DELETE { NO ACTION | CASCADE } ]
        [ ON UPDATE { NO ACTION } ]
        [ NOT FOR REPLICATION ]
    | CHECK [ NOT FOR REPLICATION ] ( logical_expression )
    [ ON { partition_scheme_name ( partition_column_name )
        | filegroup | "default" } ]
]

<column_set_definition> ::=……..
```

Sentencias DDL

CREATE TABLE

CREATE TABLE [**schema**.]**table**
(**column** datatype [DEFAULT expr]
[, ...]);

```
USE laEmpresa;
CREATE TABLE rrhh.Empleado (
                documento INT PRIMARY KEY,
                apellido varchar(20) not null
                );
```

Sentencias DDL

ALTER TABLE

- Modifica una definición de tabla, agrega o quita columnas y restricciones.

Agrega nueva columna

```
ALTER TABLE Empleado ADD nombre VARCHAR(20) NULL ;
```

Quita una columna

```
ALTER TABLE Empleado DROP COLUMN column_a;
```

Cambia el tipo de datos

```
ALTER TABLE Empleado ALTER COLUMN column_a DECIMAL (5, 2) ;
```

Agrega una columna con una restricción

```
ALTER TABLE doc_exc ADD column_b VARCHAR(20) NULL
CONSTRAINT exb_unique UNIQUE ;
```

Agrega una restricción no comprobada

```
ALTER TABLE doc_exd WITH NOCHECK
ADD CONSTRAINT exd_check CHECK (column_a > 1)
```

Agrega una restricción default

```
ALTER TABLE doc_exz ADD CONSTRAINT col_b_def
DEFAULT 50 FOR column_b ;
```

Sentencias DDL

ALTER TABLE

```
ALTER TABLE [ database_name . [ schema_name ] . | schema_name . ]
table_name
{
   ALTER COLUMN column_name
   {
      [ type_schema_name. ] type_name [ ( { precision [ , scale ]
         | max | xml_schema_collection } ) ]
      [ COLLATE collation_name ]
      [ NULL | NOT NULL ] [ SPARSE ]
   | {ADD | DROP }
      { ROWGUIDCOL | PERSISTED | NOT FOR REPLICATION | SPARSE }
   }
      | [ WITH { CHECK | NOCHECK } ]

   | ADD
   {
      <column_definition>
     | <computed_column_definition>
     | <table_constraint>
     | <column_set_definition>
   } [ ,...n ]

   | DROP
   {
      [ CONSTRAINT ] constraint_name
      [ WITH ( <drop_clustered_constraint_option> [ ,...n ] ) ]
      | COLUMN column_name
   } [ ,...n ]

   | [ WITH { CHECK | NOCHECK } ] { CHECK | NOCHECK } CONSTRAINT
      { ALL | constraint_name [ ,...n ] }

   | { ENABLE | DISABLE } TRIGGER
      { ALL | trigger_name [ ,...n ] }

   | { ENABLE | DISABLE } CHANGE_TRACKING
      [ WITH ( TRACK_COLUMNS_UPDATED = { ON | OFF } ) ]

   | SWITCH [ PARTITION source_partition_number_expression ]
      TO target_table
      [ PARTITION target_partition_number_expression ]

   | SET ( FILESTREAM_ON = { partition_scheme_name | filegroup |
            "default" | "NULL" } )

   | REBUILD
     [ [PARTITION = ALL]
      [ WITH ( <rebuild_option> [ ,...n ] ) ]
     | [ PARTITION = partition_number
         [ WITH ( <single_partition_rebuild_option> [ ,...n ] )]
      ]
     ]

   | (<table_option>)
}
[ ; ]
```

Sentencias DDL

ALTER TABLE

```
USE laEmpresa;

ALTER TABLE rrhh.Empleado ADD nombre VARCHAR(20) NULL;
GO
ALTER TABLE rrhh.Empleado ADD codEmpleado INT IDENTITY(1,1);
GO
ALTER TABLE rrhh.Empleado ALTER COLUMN column apellido VARCHAR(30)
NOT NULL;
GO
ALTER TABLE rrhh.Empleado DROP COLUMN codEmpleado;
```

Sentencias DDL

- DROP TABLE
  - Elimina la definición de una tabla y todos los datos, índices, restricciones y especificaciones de permisos asociados. Realiza un borrado físico de la tabla.

```
DROP TABLE [ database_name . [ schema_name ] . |
schema_name . ]
      table_name [ ,...n ] [ ; ]
```

```
DROP TABLE rrhh.empleados;
```

Instrucciones SQL

SENTENCIAS DML

Sentencias DML

INSERT

- Agrega una o varias filas nuevas a una tabla o una vista.

INSERT [INTO] objeto [(lista de columnas)]
VALUES (({ DEFAULT | NULL | expresión } [,...n]) [,...n])

INSERT INTO rr.hhEmpleado **VALUES** (20111333,'GOMEZ');

INSERT INTO rrhh.Empleado (documento, apellido) **VALUES** (20111333,'GOMEZ');

Sentencias DML

- **INSERT DESDE UN SELECT**

INSERT [INTO] objeto [(lista de columnas)]
(SELECT [(lista columnas)] FROM Tabla

```
INSERT INTO rrhh.Empleado SELECT
                          ContactID, LastName
                          FROM Person.Contact
                          WHERE EmailPromotion = 2;
```

Sentencias DML

UPDATE

- Cambia los datos existentes en una o varias columnas de una tabla o vista.

```
UPDATE objeto
      SET nombre columna = { expresión | DEFAULT |
NULL } [ ,...n ]
[ FROM{ <tabla> } [ ,...n ] ]
[ WHERE { <condición> ]
```

```
UPDATE Empleado SET apellido ='PEREZ'
WHERE apellido ='Achong';
```

Sentencias DML

La instrucción **DELETE** quita una o varias filas de una tabla o vista.

Ej:

1)
```
DELETE
[FROM] tablax
WHERE col1 = 100
```

2)
```
DELETE FROM Sales.SalesPersonQuotaHistory
WHERE SalesPersonID IN
                    (SELECT SalesPersonID FROM Sales.SalesPerson
                    WHERE SalesYTD > 2500000.00 )
```

La instrucción **TRUNCATE TABLE** es un método rápido y no registrado para eliminar todas las filas de una tabla. TRUNCATE TABLE es funcionalmente equivalente a la instrucción DELETE sin una cláusula WHERE. Sin embargo, TRUNCATE TABLE es más rápida y utiliza menos recursos de registro de sistema y de transacciones.
La instrucción DELETE quita una a una las filas y graba una entrada en el registro de transacciones por cada fila eliminada.

Ej:

```
TRUNCATE table
```

VALORES NULOS

NULL = NULL ?

Funcion **ISNULL()**

Ej: Create table persona (id int, apellido varchar(20))
Insert into persona (id, apellido)
Values (1,null)
Create table historico_persona (id int, apellido varchar(20))
Insert into historico_persona (id, apellido)
Values (2,null)

```
select *
from persona a
inner join historico_persona h on a.apellido = h.apellido
```

¿Qué Son las Restricciones?

- Las restricciones fuerzan las reglas a nivel de tabla.
- Las restricciones evitan la supresión de un registro si hay dependencias.
- Son válidos los siguientes tipos de restricción:
  - `NOT NULL`
  - `UNIQUE`
  - `PRIMARY KEY`
  - `FOREIGN KEY`
  - `CHECK`
  - `DEFAULT`

La Restricción **FOREIGN KEY**

DEPARTMENTS

PRIMARY KEY

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500

EMPLOYEES

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
102	De Haan	90
103	Hunold	60
104	Ernst	60
107	Lorentz	60

FOREIGN KEY

...

INSERT INTO

200	Ford	9
201	Ford	60

No permitido (9 no existe)

Permitido

La Restricción **FOREIGN KEY**

Definida a nivel de tabla o de columna:

```
CREATE TABLE employees(
    employee_id      int,
    last_name        VARCHAR(25) NOT NULL,
    email            VARCHAR(25),
    salary           decimal(8,2),
    commission_pct   decimal(2,2),
    hire_date        DATETIME NOT NULL,
    department_id    int,
    CONSTRAINT emp_dept_fk FOREIGN KEY (department_id)
      REFERENCES departments(department_id),
    CONSTRAINT emp_email_uk UNIQUE(email));
```

Palabras Clave de la Restricción `FOREIGN KEY`

- `FOREIGN KEY`: Define la columna de la tabla secundaria a nivel de restricción de tabla.
- `REFERENCES`: Identifica la tabla y la columna en la tabla principal.

BIBLIOGRAFIA

- **Un Enfoque Practico del SQL.**
  ISBN 9789871076611. Morteo Francisco A. y Bocalandro Nicolas L.E. Editorial COOPERATIVAS
- **ORACLE 11g. SQL, PL/SQL, SQL*PLUS.**
  ISBN 9782746053601. Gabillaud Jerome.
- **Oracle SQL and PL/SQL Handbook.**
  ISBN 9780201752946. John Adolph Palinski
- **SQL: Guia práctica para usuarios.**
  ISBN 9788441519152. Charte Ojeda, Francisco. Editorial ANAYA
- **E. F. Codd*, The Relational Model for Database Management Version 2*** (Reading, Mass.: Addison-Wesley, 1990).
- **Varios Sitios WEB.**

ESPACIO PARA PREGUNTAS