Ingeniería en Sistemas ula

Bases de datos

Lenguajes de consulta

SQL básico

Ingeniería en Sistemas ula

SQL

- Structured Query Language
- Lenguaje de consulta estructurado
- Lenguaje declarativo de alto nivel

- Lenguaje estándar para los SGBD relacionales comerciales
  - 1 sentencia la entienden varios SGBDR

SEQUEL y SQL

- **SEQUEL**: Structured English QUEry Language
- Diseñado e implementado por IBM Research como la interfaz para un sist. de BD relacional experimental llamado SYSTEM R.

- Luego se convirtió en el estándar SQL gracias al esfuerzo conjunto de la ANSI y la ISO
  - SQL-86 o SQL1
  - SQL-92 o SQL2
  - **SQL-99 o SQL3**
  - SQL:2003
  - SQL:2006
  - SQL:2008

Estándar SQL

- **Núcleo de especificaciones**: se supone que sea implementado por todos los vendedores de SGBDR que cumplen con el estándar.

- **Extensiones especializadas**: pueden implementarse como módulos opcionales que se venden independientemente para aplicaciones de BD específicas (minería de datos, datos espaciales, datos temporales, OLAP, datos multimedia, etc.)

SQL-89

- Su primera versión se denominó *SQL-89* y estaba compuesta por tres partes:
  - *LDD*: contiene las instrucciones para definir el esquema de una BD (*create, alter y drop*).
  - *LMD*: contempla las instrucciones de gestión de tablas (select, insert, delete y update), y para control de concurrencia (*commit y rollback*).
  - *LCD* (Lenguaje de Control de Datos): tiene instrucciones para dar y revocar permisos de acceso a los datos de la BD (*grant y revoke)*

Ingeniería en Sistemas ula

SQL2

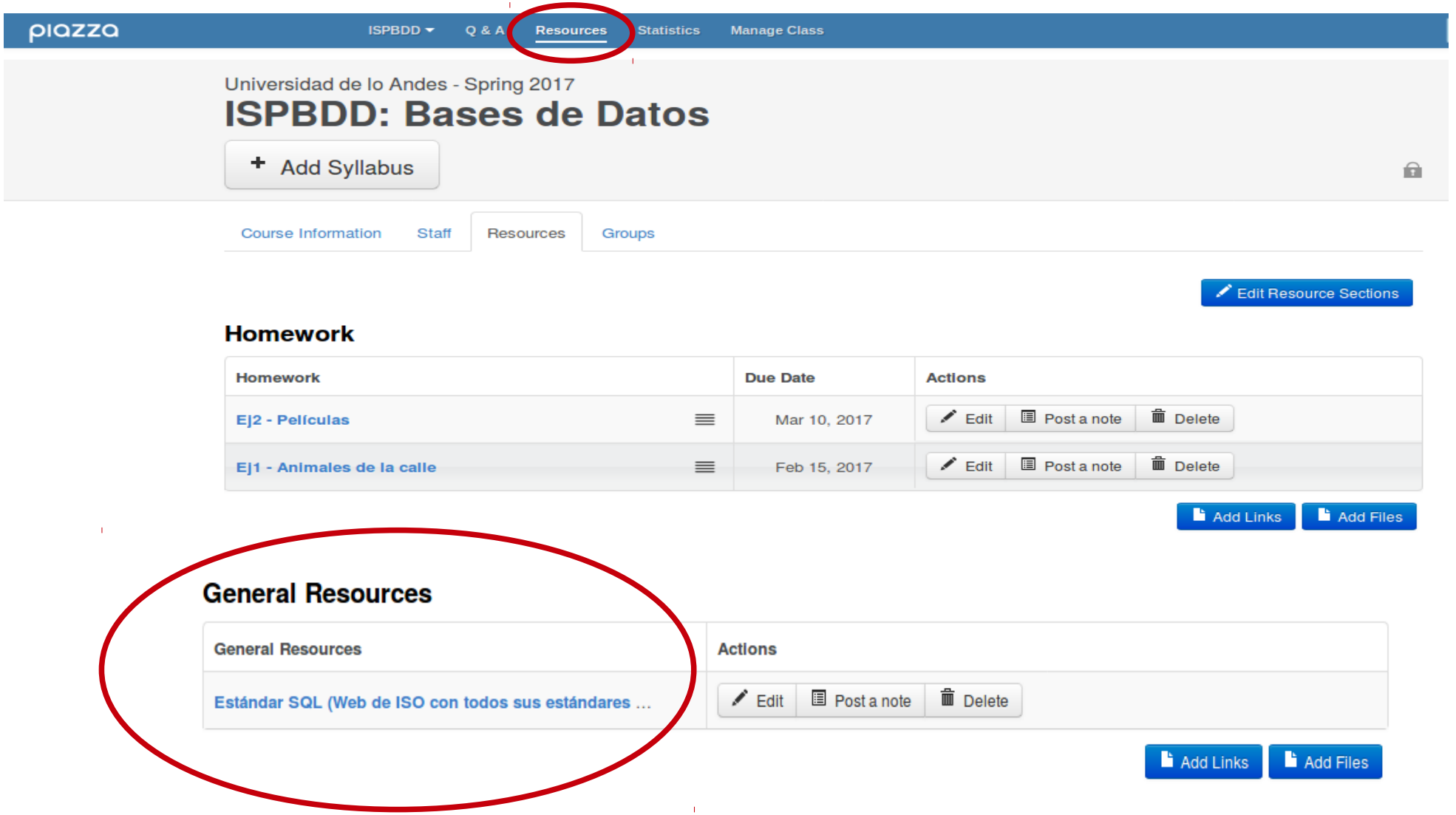
- Segunda versión de SQL, también denominada SQL-92.
  - Incluyó el uso de agentes de software, nuevos tipos básicos de datos como: Date, Time, Timestamp, BLOB, Varchar.
  - Para sesiones concurrentes se establecen conexiones cliente-servidor en sesiones concurrentes
  - Tiene SQL dinámico
  - Aumentó la granularidad a nivel de transacciones
  - Crearon nuevas versiones de la operación producto
  - Usa un catálogo estandarizado, se manejan códigos de error estandarizados
  - Se utilizan nuevos lenguajes de programación como: C, ada y mumps

Ingeniería en Sistemas ula

SQL3

- Versión de SQL que contiene:
  - tipos abstractos de datos definidos por el diseñador de la BD
  - manejo de roles de usuarios
  - consultas recursivas
  - disparadores o triggers
  - procedimientos almacenados
  - encadenamiento tardío
  - manejo de interoperabilidad a través de un API u ODBC para acceso a bases de datos basado en el estándar SAG (SQL Access Group)
  - manejo de transacciones anidadas

Estándar SQL



Ingeniería en Sistemas ula

SQL

- SQL es un lenguaje extensivo: tiene sentencias para definir, consultar y actualizar datos.
- SQL es un LDD y un LMD
- También tiene facilidades para definir vistas en la BD, para especificaciones de seguridad y autorización, para definir restricciones de integridad, y para especificar controles de transacciones
- Incluye reglas para incrustar sentencias SQL en un lenguaje de programación de propósito general (Java, C+++)

LDD en SQL

Términos en SQL		**Términos modelo relacional**
Tabla	⟺	Relación
Fila		Tupla
Columna		Atributo

Comando SQL para crear objetos:

CREATE

Esquemas, tablas, dominios

Vistas, aserciones, disparadores

DROP: destruir objetos

ALTER: modificar objetos

Esquema

Esquema SQL:

Está identificado con un nombre

Incluye un identificador de autorización para indicar el usuario que es propietario del esquema

Incluye descriptores para cada elemento del esquema (tablas, restricciones, vistas, dominios y otros constructos que describen al esquema)

CREATE SCHEMA nombreesquema **AUTHORIZATION** 'usuarioesquema' ;

Catálogo

Catálogo SQL:

Colección de esquemas que tiene un nombre

Siempre contiene un esquema especial llamado INFORMATION_SCHEMA, que proporciona información sobre todos los esquemas en el catálogoy todos los descriptores de elementos en estos esquemas.

Las restricciones de integridad pueden definirse solo si las relaciones involucradas existen en esquemas dentro del mismo catálogo.

Los esquemas dentro del mismo catálogo también comparten ciertos elementos como la definición de dominios.

Tablas

Crear una nueva relación dándole un nombre y especificando sus atributos y restricciones iniciales. Los atributos se consideran ordenados en la secuencia que se especifican en la sentencia SQL de creación.

CREATE TABLE esquema.nombretabla...

CREATE TABLE nombretabla...

Restricciones

- **Restricciones de atributos y valores por omisión de los atributos**

Valor **NOT NULL**, implícitamente específicado para claves primarias y de forma explícita para cualquier otro atributo

DEFAULT <valor>, define un valor por omisión para un atributo.

CHECK <definición de atributo o dominio>, para restringir los valores de un atributo o dominio

Restricciones

- **Restricciones de claves e integridad referencial**

PRIMARY KEY, uno o mas atributos que forman la clave primaria de la relación

UNIQUE, permite especificar claves secundarias, la relación no tendrá dos tuplas con el mismo valor en ese atributo

FOREIGN KEY, para especificar integridad referencial. Ante una violación de esta restricción la acción por omisión es rechazar la operación de actualización que la ocasiona (opción RESTRICT).

Restricciones

- **Restricciones de claves e integridad referencial**

FOREIGN KEY, para especificar integridad referencial. Ante una violación de esta restricción la acción por omisión es rechazar la operación de actualización que la ocasiona (opción RESTRICT).

Pueden especificarse *acciones alternativas* adjuntando una cláusula de acción referencial disparada a cualquier restricción de clave foránea.
Operaciones: **ON DELETE** o **ON UPDATE**
Opciones: **SET NULL**, **CASCADE** o **SET DEFAULT**

Restricciones

- **Restricciones basadas en tuplas**

Se aplican a cada tupla de forma individual y se chequean cada vez que se inserta o modifica una tupla.

CHECK (fecha_creacion_empresa <= fecha_inicio_empleado);

Ingeniería en Sistemas ula

Tablas

```
CREATE TABLE Empleado (
    Nombre VARCHAR(15) NOT NULL,
    Cedula VARCHAR(10) NOT NULL,
    Sexo char DEFAULT 'F',
    Sueldo DECIMAL(10,2),
    Fechanac DATE,
    Cargo INT UNIQUE,
    PRIMARY KEY (cedula),
    FOREIGN KEY (cargo) REFERENCES Cargo(id) ON
DELETE SET NULL ON UPDATE CASCADE,
    CHECK (Sueldo>0 and Sueldo<900000000),
    CHECK (Fechanac < TODAY())
);
```

Tipos de datos básicos en SQL

- **Numeric**: INTEGER o INT, SMALLINT, FLOAT o REAL, DOUBLE PRECISION, DECIMAL(i,j), NUMERIC(i,j)
- **Character string**: CHAR(n), VARCHAR(n), CLOB (character large object)
- **Bit string**: BIT(n), BIT VARYING(n), BLOB (binary large object)
- **Boolean**
- **Date**: formato YYYY-MM-DD
- **Time**: formato HH:MM:SS

Tipos de datos en SQLite

- INTEGER
- REAL
- TEXT
- BLOB

https://www.sqlite.org/datatype3.html

Dominios en SQL

CREATE DOMAIN nombredominio **AS** tipodato;

CREATE DOMAIN numerox **AS** INTEGER
CHECK (numerox > 0 **AND** numerox < 21) ;

Consultas

Recuperar información de la BD: **SELECT**
SELECT <lista de atributos>
FROM <lista de tablas>
WHERE <condición>

Operadores de comparación lógicos: =, <, <=, >, >=, <>

SELECT fechanac, direccion
FROM Empleado
WHERE nombre='María' **AND** apellido='González';

Consultas

```
SELECT fechanac, direccion, Empleado.cargo,
Proyecto.nombre
FROM Empleado, Proyecto
WHERE nombre='María' AND apellido='González' AND
Empleado.id_proyecto=Proyecto.id;

SELECT peli.titulo, secuela.titulo
FROM Pelicula AS peli, Pelicula AS secuela
WHERE peli.id_secuela=Secuela.id;
```

Consultas

```
SELECT *
FROM Empleado
WHERE nombre='María' AND 'González' ;

SELECT titulo, año
FROM Pelicula;
```

SELECT

Elimina las tuplas duplicadas

SELECT UNIQUE A1, A2, A3, ..., An
FROM nombreTabla

$$\Pi_{A1,\ A2,\ldots,\ An}\ (\text{nombreTabla})$$

SELECT

SELECT *
FROM nombreTabla
WHERE condicion

Puede usar los operadores: <, >, <=, >=, =, <>, and, or, not, between

SELECT A1, A2,..., An
FROM nombreTabla
WHERE condicion

Proyección y restricción

$$\Pi_{A1,\ A2,\ldots,\ An}\ (\sigma_{condicion}\ (nombreTabla))$$

SELECT con resultados ordenados

SELECT A1, A2, A3,..., An
FROM R1, R2, ..., Rm
WHERE condicion
ORDER BY A2 **ASC**, A5 **DESC**

SELECT

SELECT *
FROM R1, R2, ..., Rm

(R1 x R2 x ... x Rm)

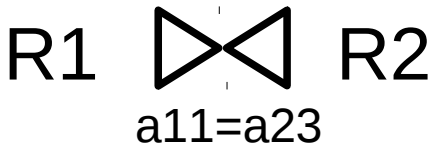
SELECT

SELECT A1, A2, A3,..., An
FROM R1, R2, ..., Rm
WHERE condicion

$$\Pi_{A1,\ A2,\ldots,\ An}\ (\sigma_{condicion}\ (R1 \times R2 \times \ldots \times Rm))$$

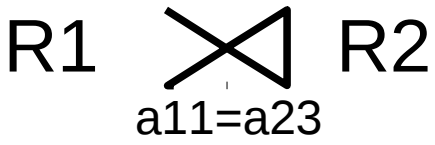
SELECT

```
SELECT *
FROM R1, R2
WHERE a11=a23
```



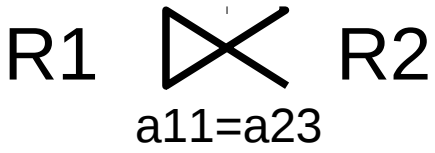
LEFT JOIN

```
SELECT *
FROM R1 LEFT JOIN R2
WHERE a11=a23
```



RIGHT JOIN

```
SELECT *
FROM R1 RIGHT JOIN R2
WHERE a11=a23
```



SELECT

```
SELECT {*| listaDeAtributos}
FROM listaDeTablas
[WHERE condiciones ]
[GROUP BY listaDeAtributos1]
[HAVING condicion ]
[ORDER BY listaDeAtributos2]
```

- GROUP BY agrupa los resultados por los atributos de la listaDeAtributos1
- HAVING selecciona los resultados que cumplan con la condición especificada

INSERT

INSERT INTO nombreDeTabla
[(listaDeAtributos)]
VALUES (listaDeValores | consulta)

- *consulta* es una sentencia SELECT que obtiene los valores que se insertan en la tabla

INSERT INTO Empleado
VALUES
(‘Ricardo’, ‘J’, ‘Marin’, ‘653293’, ‘1972-10-05’, ‘98
La Hechicera, Mérida’, ‘M’, 37000, ‘8653’, 4);

DELETE

DELETE FROM nombreDeTabla
[**WHERE** condicion]

DELETE FROM Empleado
WHERE
apellido =‘Suárez’;

UPDATE

UPDATE nombreTabla
SET {listaDeExpresiones}
[**WHERE** Q]

- ListaDeExpresiones es una lista separadas por coma de nombre_de_atributo = expresion

UPDATE Empleado
SET Sueldo = Sueldo * 1.1
WHERE id_departamento = 5;

UNION

```
SELECT {*| listaDeAtributos}
FROM listaDeTablas
WHERE condicion1

UNION

SELECT {*| listaDeAtributos}
FROM listaDeTablas
WHERE condicion2
```

- En el resultado se eliminan las tuplas duplicadas
- Es equivalente al operador del algebra relacional $\cup$, si los resultados de ambas consultas tienen el mismo esquema

INTERSECT

```
SELECT {*| listaDeAtributos}
FROM listaDeTablas
WHERE condicion1

INTERSECT

SELECT {*| listaDeAtributos}
FROM listaDeTablas
WHERE condicion2
```

- En el resultado se eliminan las tuplas duplicadas
- Es equivalente al operador del algebra relacional $\cap$, si los resultados de ambas consultas tienen el mismo esquema

EXCEPT

```
SELECT {*| listaDeAtributos}
FROM listaDeTablas
WHERE condicion1

EXCEPT

SELECT {*| listaDeAtributos}
FROM listaDeTablas
WHERE condicion2
```

- En el resultado se eliminan las tuplas duplicadas
- Es equivalente al operador del álgebra relacional **-** , si los resultados de ambas consultas tienen el mismo esquema

Funciones en SQL

- **COUNT(atributo)**: cuenta la cantidad de valores o tuplas
- **SUM(atributo)**: suma valores numéricos
- **AVG(atributo)**: calcula el promedio de valores numéricos.
- **MIN(atributo)**: calcula el valor más pequeño
- **MAX(atributo)**: calcula el valor más grande

Encadenamiento de consultas

SELECT {*| listaDeAtributos}
FROM listaDeTablas
WHERE atributo1 {IN | [NOT]EXIST | operadorDeComparacion {ALL | ANY }} (**SELECT** {*| listaDeAtributos} ...)

Subconsultas

IN: operador de membresía equivalente a pertenece
EXIST: cuantificador existencial
ALL: cuantificador universal
ANY: indica que un atributo “es al menos” >, <, >=, <=, = o <> que cualquier valor de otro atributo

Ejercicio

Base de datos de una biblioteca

Libro(id, título, *nombre_publicista*)
AutorLibro(*id_libro*, nombre_autor)
Publicista(nombre, direccion, telefono)
CopiaLibro(*id_libro*, *id_rama*, num_copias)
PrestamoLibro(*id_libro*, *id_rama*, *num_tarjeta*, fecha_salida, fecha_tope_prestamo)
RamaBiblioteca(id, nombre, direccion)
UsuarioPrestamo(num_tarjeta, nombre, direccion, telefono)

Ingeniería en Sistemas ula

Ejercicio

Cree el modelo usando sentencias SQL.

Escoja la restricción para integridad referencial apropiada (RESTRICT, CASCADE, SET TO NULL, SET TO DEFAULT) para el borrado de una tupla y para la modificación de un atributo clave primaria en una tupla referenciada.