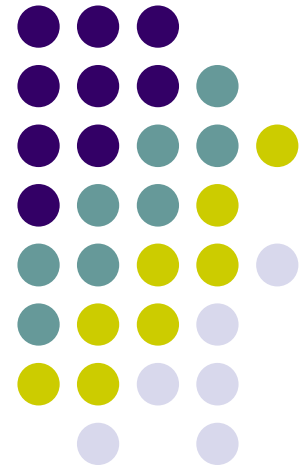


Structure Query Language





Concepto de Base de datos

Es una colección de datos relacionados que representa una porción del mundo real. Tiene una estructura lógica de almacenamiento que facilita la recuperación de los datos en forma segura y eficiente.

Problemas al Almacenar la información en archivos



- Alto costo de mantenimiento
- Redundancia de datos
- Rigidez en la recuperación de datos
- Estructura dependiente de la aplicación
- Falencias de seguridad

Modelos de bases de datos



- **Modelo de Red**

- Este modelo representa los datos mediante colecciones de registros y sus relaciones se representan por medio de ligas o enlaces, los cuales pueden verse como punteros.
- Los registros se organizan en un conjunto de gráficas arbitrarias.

- **Modelo Jerárquico**

- Es similar al modelo de red en cuanto a las relaciones y datos, ya que estos se representan por medio de registros y sus ligas.
- La diferencia radica en que están organizados por conjuntos de arboles en lugar de gráficas arbitrarias.

- **Modelo Relacional**

- En este modelo se representan los datos y las relaciones entre estos, a través de una colección de tablas, en las cuales los **renglones (tuplas)** equivalen a los cada uno de los registros que contendrá la base de datos y las columnas corresponden a las **características (atributos)** de cada registro localizado en la tupla.

Ejemplo: Modelo Relacional



| Libros | | | | | | | | | | | | | | | | |
|----------------|--------------------|------------------|--|--|-----------|--|------|-------|-------------|-----------------|------------|-----------------|------------|---------------|------------|--------------------|
| ISBN | Nombre | Autor | Editorial | | | | | | | | | | | | | |
| 844814063 X | Red Hat Linux | Richard Petersen | McGraw-Hill | | | | | | | | | | | | | |
| 0596006403 | Linux Cookbook | Carla Schroder | <table><tr><th colspan="2">Préstamos</th></tr><tr><th>ISBN</th><th>Socio</th></tr><tr><td>844814063 X</td><td>Juan José Perez</td></tr><tr><td>0596006403</td><td>Juan José Perez</td></tr><tr><td>0596005830</td><td>Andrés García</td></tr><tr><td>1874416656</td><td>Maria Julia Torres</td></tr></table> | | Préstamos | | ISBN | Socio | 844814063 X | Juan José Perez | 0596006403 | Juan José Perez | 0596005830 | Andrés García | 1874416656 | Maria Julia Torres |
| Préstamos | | | | | | | | | | | | | | | | |
| ISBN | Socio | | | | | | | | | | | | | | | |
| 844814063 X | Juan José Perez | | | | | | | | | | | | | | | |
| 0596006403 | Juan José Perez | | | | | | | | | | | | | | | |
| 0596005830 | Andrés García | | | | | | | | | | | | | | | |
| 1874416656 | Maria Julia Torres | | | | | | | | | | | | | | | |
| 059600583 | Linux Unwired | Roger Weeks | | | | | | | | | | | | | | |
| Socios | | | | | | | | | | | | | | | | |
| Documento | Nombre | Domicilio | | | | | | | | | | | | | | |
| DNI 25.902.327 | Juan Jose Perez | L.N. Alem 25 | | | | | | | | | | | | | | |
| DNI 18.987.234 | Andres Garcia | Córdoba 34 | | | | | | | | | | | | | | |
| CI 12.098.485 | Maria Julia Torres | Junin 231 2do A | | | | | | | | | | | | | | |
| DNI 12.345.678 | Carlos Rodriguez | Caballito 1234 | | | | | | | | | | | | | | |

Diseño de Bases de Datos



- Diseñar una base de datos consiste en definir la estructura lógica sobre la cual se almacenará la información. Un buen diseño es fundamental para que la recuperación de datos sea eficiente.
- Para diseñar una base de datos es necesario conocer en detalle las características de lo que se quiere representar y la información que los futuros usuarios esperan obtener de la base



Bases de Datos: Entidades

Cualquier persona u objeto que existe en forma independiente en el mundo se considera una **ENTIDAD**

Ejemplo: En la base de datos de una biblioteca son entidades:

- Los socios
- Los libros
- Los autores de libros
- Los bibliotecarios



Bases de Datos: Atributos

Las entidades se describen a través de sus características o **atributos**.

Ejemplo: La entidad libro tiene como atributos:

- ISBN
- Nombre
- Autor
- Fecha de publicación
- Editorial



Bases de Datos: Valores

Los atributos pueden tomar diferentes valores de un dominio.

Ejemplo: El atributo nombre de la entidad libro puede tomar los valores:

- "El principito"
- "Linux Unleashed"
- "Desarrollo del talento humano"
- etc.



Bases de Datos: Relaciones

Una asociación entre dos entidades se denomina **relación**. Una relación puede tener atributos

Ejemplos:

- La entidad libro y la entidad socio se asocian través de la relación "es prestado". Un atributo de la relación es la fecha del préstamo.
- La entidad autor y la entidad libro se asocian través de la relación "escribió".



Bases de Datos: Claves

Se denomina **clave** al conjunto de atributos que permite identificar en forma unívoca a una entidad dentro del conjunto de entidades del mismo tipo

Ejemplos:

- La clave de la entidad libro es el atributo ISBN.
- La clave de la entidad socio puede ser el conjunto de atributos tipo y número de documento, o bien el atributo número de socio.

Bases de Datos Relacionales: Tablas



En las bases de datos relacionales tanto las entidades como las relaciones entre entidades se almacenan en **tablas**.

- Una tabla contiene un conjunto de entidades o un conjunto de relaciones del mismo tipo.
- Cada entidad o relación del conjunto está representada por un registro.
- Un registro está compuesto por campos que almacenan los valores de cada uno de los atributos de la entidad o relación.

Ejemplo:

Diseño incorrecto



| Libros | | | |
|----------------|---------------|------------------|-------------|
| <u>ISBN</u> | Nombre | Autor | Editorial |
| 844814063 X | Red Hat Linux | Richard Petersen | McGraw-Hill |

| | | |
|----------------|----------------|----------------|
| 059600640 3 | Linux Cookbook | Carla Schroder |
| 059600583 | Linux Unwired | Roger Weeks |

| Socios | | |
|--------|--|--|
|--------|--|--|

| <u>Documento</u> | Nombre | Domicilio |
|-------------------|-----------------|-----------|
| DNI 25.902.327 | Juan Jose Perez | L.N. Alem |

| | | |
|-------------------|---------------|-----------|
| DNI 18.987.234 | Andres Garcia | Córdoba 3 |
|-------------------|---------------|-----------|

| | | |
|---------------|--------------------|-----------------|
| CI 12.098.485 | Maria Julia Torres | Junin 231 2do A |
|---------------|--------------------|-----------------|

| Préstamos | |
|----------------|--------------------|
| <u>ISBN</u> | <u>Socio</u> |
| 844814063 X | Juan José Perez |
| 0596006403 | Juan José Perez |
| 0596005830 | Andrés García |
| 1874416656 | Maria Julia Torres |



Base de Datos: Consideraciones al momento del diseño

- Una tabla no debe contener datos duplicados
- Una tabla no debe contener datos calculables
- Los campos de una tabla deben contener datos simples
- Cuando un campo de la tabla X hace referencia a la tabla Y, debe ser a través de la clave de Y.
- No es conveniente utilizar campos de texto como clave, se debe asociar un código a cada texto y utilizar dicho código como clave.

Ejemplo:

Diseño correcto



| Libros | | | | | |
|-------------|----------------|--------------------|-----------------|-----------|--|
| ISBN | Nombre | Autor | | Editorial | |
| 844814063 X | Red Hat Linux | Préstamos | | | |
| 059600640 3 | Linux Cookbook | | | | |
| 059600583 | Linux Unwired | | | | |
| Soc | | | | | |
| TipoDoc | NroDoc | Nombre | | | |
| DNI | 25.902.32 7 | Juan J Perez | | | |
| DNI | 18.987.23 4 | Andre | | | |
| CI | 12.098.48 5 | Maria Julia Torres | Junin 231 2do A | | |
| DNI | 25.902.32 7 | Fernando | Comodoro 1256 | | |

| Préstamos | | | |
|-----------|----------------|----------------|----------------|
| TipoDoc | NroDoc | ISBN | Retiro |
| DNI | 25.902.32 7 | 844814063 X | 21/04/200 5 |
| DNI | 25.902.32 7 | 059600640 3 | 21/04/200 5 |
| DNI | 18.987.23 4 | 059600583 0 | 23/04/200 5 |
| CI | 12.098.48 5 | 187441665 6 | 30/04/200 5 |



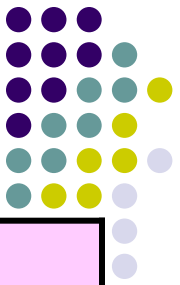
Structured Query Language

Es un lenguaje de consulta estándar para bases de datos

Normalizado por ANSI

Permite:

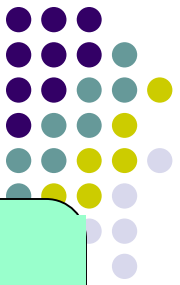
- Crear, modificar y eliminar tablas.
- Insertar, modificar y eliminar registros.
- Seleccionar registros que cumplen ciertos criterios



Tipos de campos

| Nombre | Descripción |
|----------|--|
| CHAR(n) | Tipo caracter que almacena hasta 255 caracteres (2 bytes por caracter) |
| MEMO | Tipo caracter que almacena hasta 2.14 gigabytes (2 bytes por caracter) |
| BIT | Valores de verdad (0 ó -1) |
| SMALLINT | Enteros de 2 bytes |
| INTEGER | Enteros de 4 bytes |
| REAL | Reales de precisión simple |
| FLOAT | Reales de precisión doble |
| DATETIME | Fechas |

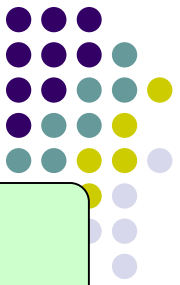
Creación de tablas



```
CREATE TABLE nombreTabla
( nombreCampo1    tipoCampo1
      [PRIMARY KEY | NOT NULL | UNIQUE] ,
  nombreCampo2    tipoCampo2
      [PRIMARY KEY | NOT NULL | UNIQUE] ,
  ....
  nombreCampoN    tipoCampoN
      [PRIMARY KEY | NOT NULL | UNIQUE]
)
```

```
CREATE TABLE TipoPelicula
( Tip_Codigo      smallint PRIMARY KEY,
  Tip_Descr       char(10) NOT NULL
)
```

Creación de tablas



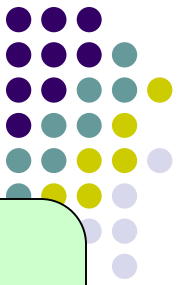
```
CREATE TABLE Generos
(  Gen_Codigo          smallint PRIMARY KEY,
   Gen_Nombre          char(20) NOT NULL
)
```

```
CREATE TABLE Categorias
(  Cat_Codigo          smallint PRIMARY KEY,
   Cat_Nombre          char(20) NOT NULL
)
```

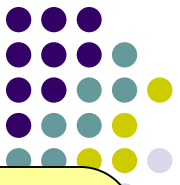
```
CREATE TABLE Directores
(  Dir_Codigo          integer PRIMARY KEY,
   Dir_Nombre          char(50) NOT NULL
)
```

```
CREATE TABLE Doc
(  Doc_Codigo          smallint PRIMARY KEY,
   Doc_Descr           char(20) NOT NULL
)
```

Creación de tablas



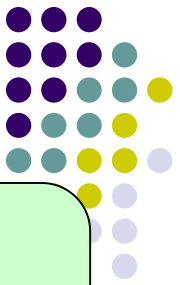
```
CREATE TABLE Peliculas
( Pel_Codigo          integer PRIMARY KEY,
  Pel_Nombre          char(50) NOT NULL,
  Tip_Codigo          smallint,
  Gen_Codigo          smallint,
  Cat_Codigo          smallint,
  Dir_Codigo          integer,
  Pel_Resumen         memo,
  Pel_Duracion        integer
)
```



Creación de tablas

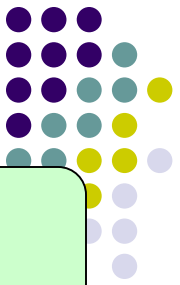
```
CREATE TABLE nombreTabla
( nombreCampo1    tipoCampo1
    [PRIMARY KEY | NOT NULL | UNIQUE] ,
  nombreCampo2    tipoCampo2
    [PRIMARY KEY | NOT NULL | UNIQUE] ,
  ....
  nombreCampoN    tipoCampoN
    [PRIMARY KEY | NOT NULL | UNIQUE] ,
  [[CONSTRAINT nombre1]
    PRIMARY KEY (nombreCampo1, ..., nombreCampoN) ] ,
  [[CONSTRAINT nombre2]
    UNIQUE (nombreCampo1, ..., nombreCampoN) ] ,
  [[CONSTRAINT nombre3]
    FOREIGN KEY (nombreCampo1, ..., nombreCampoN)
    REFERENCES nombreTablaX
        (nombreCampo1, ..., nombreCampoN) ]
)
```

Creación de tablas



```
CREATE TABLE Socios
( Soc_Codigo          integer PRIMARY KEY,
  Soc_Apellido        char(30) NOT NULL,
  Soc_Nombre          char(30) NOT NULL,
  Doc_Codigo          smallint NOT NULL,
  Soc_Doc             integer NOT NULL,
  Soc_Calle           char(30) NOT NULL,
  Soc_Numero          integer NOT NULL,
  Soc_Depto           char(20) ,
  Soc_Telefono        char(20) ,
  Soc_FechaAlta       datetime NOT NULL,
  Soc_FechaNac        datetime,
  CONSTRAINT uqDoc UNIQUE
                        (Doc_Codigo, Soc_Doc) ,
  CONSTRAINT fkDoc FOREIGN KEY (Doc_Codigo)
                        REFERENCES Doc(Doc_Codigo)
)
```

Creación de tablas



```
CREATE TABLE Alquileres
( Alq_Codigo          integer PRIMARY KEY,
  Pel_Codigo          integer NOT NULL,
  Soc_Codigo          integer NOT NULL,
  Alq_Entrega        datetime NOT NULL,
  Alq_Retorno         datetime NOT NULL,
  Alq_Precio          real,
  CONSTRAINT fka1 FOREIGN KEY (Pel_Codigo)
    REFERENCES Peliculas (Pel_Codigo),
  CONSTRAINT fka2 FOREIGN KEY (Soc_Codigo)
    REFERENCES Socios (Soc_Codigo)
)
```



Eliminación de tablas

```
DROP TABLE nombreTabla
```

```
DROP TABLE Alquileres
```


Creación de índices



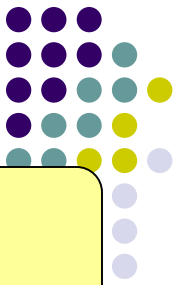
```
CREATE [UNIQUE] INDEX nombreIndice
      ON nombreTabla
( campo1 [ASC | DESC] ,
  campo2 [ASC | DESC] ,
  . . .
  campoN [ASC | DESC]
)
```

```
CREATE INDEX IndAlqPelicula
      ON Alquileres (Pel_Codigo)
```

```
CREATE INDEX IndAlqSocio
      ON Alquileres (Soc_Codigo)
```

```
CREATE INDEX IndAlqFechaSocio
      ON Alquileres (Alq_Entrega, Soc_Codigo)
```

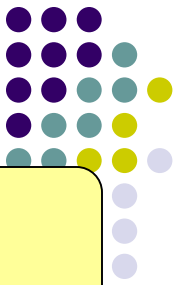
Modificación de tablas



```
ALTER TABLE nombreTabla  
    [ADD nombreCampoX tipoX |  
    DROP nombreCampoX |  
    ADD CONSTRAINT nombre ... |  
    DROP CONSTRAINT nombre]
```

```
ALTER TABLE Peliculas  
    ADD Pel_Estreno datetime
```

```
ALTER TABLE Peliculas ADD CONSTRAINT  
    FOREIGN KEY fkP1(Tip_Codigo)  
REFERENCES  
    TipoPelicula(Tip_Codigo)
```



Modificación de tablas

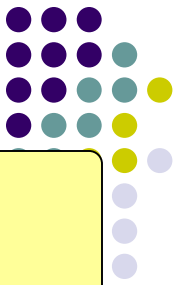
```
ALTER TABLE nombreTabla  
[ADD nombreCampoX tipoX |  
DROP nombreCampoX |  
ADD CONSTRAINT nombre ... |  
DROP CONSTRAINT nombre]
```

```
ALTER TABLE Peliculas ADD CONSTRAINT fkP2  
FOREIGN KEY (Gen_Codigo) REFERENCES  
Generos (Gen_Codigo)
```

```
ALTER TABLE Peliculas ADD CONSTRAINT fkP3  
FOREIGN KEY (Cat_Codigo) REFERENCES  
Categorias (Cat_Codigo)
```

```
ALTER TABLE Peliculas ADD CONSTRAINT fkP4  
FOREIGN KEY (Dir_Codigo) REFERENCES  
Directores (Dir_Codigo)
```

Insertión de registros

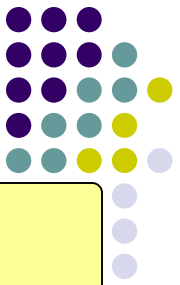


```
INSERT INTO nombreTabla  
[ ( nombreCampo1,...,nombreCampoN) ]  
VALUES (valorCampo1,..., valorCampoN)
```

```
INSERT INTO TipoPelicula  
(Tip_Codigo,Tip_Descr) VALUES (1,"Video")  
INSERT INTO TipoPelicula  
(Tip_Codigo,Tip_Descr) VALUES (2,"DVD")
```

```
INSERT INTO Peliculas  
(Pel_Codigo, Pel_Nombre) VALUES (1,"Shrek")  
INSERT INTO Peliculas  
(Pel_Codigo, Pel_Nombre) VALUES (2,"Blade")  
INSERT INTO Peliculas  
(Pel_Codigo, Pel_Nombre) VALUES (3,"EdTV")
```

Borrado de registros



```
DELETE FROM nombreTabla  
[WHERE condición]
```

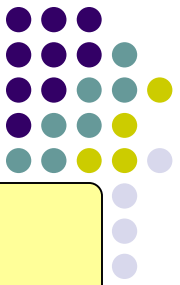
```
DELETE FROM Directores
```

```
DELETE FROM Directores  
WHERE Dir_Codigo > 100
```

```
DELETE FROM Directores  
WHERE Dir_Nombre like 'Ma*'
```

```
DELETE FROM Alquileres  
WHERE Alq_Precio is null
```

Actualización de registros



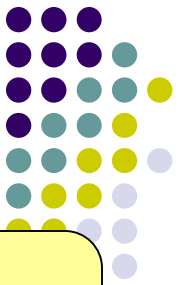
```
UPDATE nombreTabla  
SET nombreCampo = dato [,nombreCampo=dato]  
[WHERE condición]
```

```
UPDATE Peliculas  
SET Dir_Codigo = 1
```

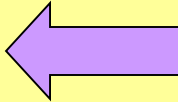
```
UPDATE Alquileres  
SET Alq_Precio = Alq_Precio + 0.5  
WHERE Alq_Codigo between 100 and 150
```

```
UPDATE Socios  
SET Soc_Calle = "Pte. Peron"  
WHERE Soc_Calle = "Cangallo"
```

Consultas - SELECT

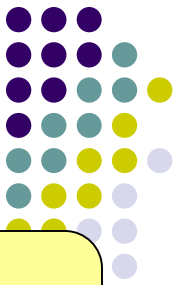


```
SELECT [DISTINCT]
campo1 [,campo2]
FROM tabla1 [,tabla2]
[WHERE condición]
[GROUP BY campos]
[HAVING condición]
[ORDER BY campos]
```

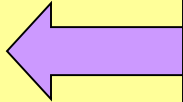


Lista de campos que deben aparecer en el resultado de la consulta

Consultas - SELECT

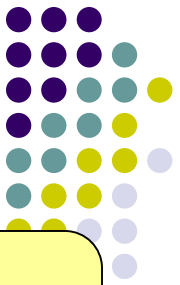


```
SELECT [DISTINCT]
campo1 [,campo2]
FROM tabla1 [,tabla2]
[WHERE condición]
[GROUP BY campos]
[HAVING condición]
[ORDER BY campos]
```

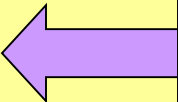


Tablas que intervienen
en el procesamiento de
la consulta

Consultas - SELECT



```
SELECT [DISTINCT]
campo1 [,campo2]
FROM tabla1 [,tabla2]
[WHERE condición]
[GROUP BY campos]
[HAVING condición]
[ORDER BY campos]
```



Condiciones que deben cumplir los registros para aparecer en el resultado



Consultas - SELECT

Si no se pone condición se muestra los atributos pedidos de todas las tuplas

- Seleccionar todos los directores

```
SELECT *  
FROM Directores
```

- Seleccionar los nombres de todos los directores

```
SELECT Dir_Nombre  
FROM Directores
```



Consultas - SELECT

Al agregar la cláusula WHERE estamos limitando a que muestre solo las tuplas que cumplen la condición

```
SELECT [DISTINCT]
campo1 [,campo2]
FROM tabla1 [,tabla2]
[WHERE condición]
```

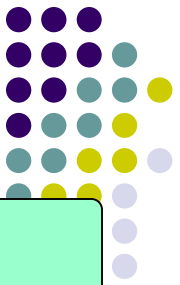
- Seleccionar los directores cuyos nombres comienzan con 'S'

```
SELECT Dir_Nombre
FROM Directores
WHERE Dir_Nombre like "S*"
```

- Seleccionar los socios nacidos en el mes de marzo

```
SELECT Soc_Apellido, Soc_Nombre
FROM Socios
WHERE month(Soc_FechaNac)=3
```

Consultas - SELECT



En la cláusula WHERE se pueden poner operadores lógicos

- Seleccionar las películas estrenadas en febrero de 2005.

```
SELECT Pel_Nombre, Pel_Estreno
FROM Películas
WHERE month(Pel_Estreno) = 2 and
      year(Pel_Estreno) = 2005
```



Consultas - SELECT

- Seleccionar las películas estrenadas entre el 2003 y 2006.

```
SELECT Pel_Nombre, Pel_Estreno  
FROM Películas  
WHERE year(Pel_Estreno) between 2003  
and 2006
```

- Seleccionar los socios que alquilaron películas en marzo de 2005

```
SELECT S.Soc_Apellido, S.Soc_Nombre  
FROM Socios as S, Alquileres as A  
WHERE S.Soc_Codigo = A.Soc_Codigo and  
month(A.Alq_Entrega) = 3 and  
year(A.Alq_Entrega) = 2005
```

Consultas - SELECT



Al agregar más de una tabla en la cláusula FROM se genera una “pseudo-tabla” con el **producto cartesiano** de esas tablas, que no persiste más allá de la consulta.

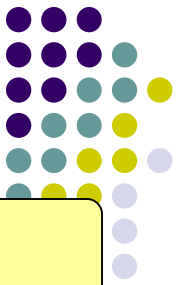
A partir de estas “**juntas**” se puede obtener información relacionada que está en varias tablas.

➤ Seleccionar las comedias estrenadas en febrero de 2005.

```
SELECT P.Pel_Nombre, P.Pel_Estreno  
FROM Películas as P, Generos as G  
WHERE P.Gen_Codigo = G.Gen_Codigo and  
      G.Gen_Nombre = "Comedia" and  
      month(P.Pel_Estreno) = 2 and  
      year(P.Pel_Estreno) = 2005
```

**Del producto cartesiano se excluyen los registros
que no coinciden en el campo relacionado**

Consultas - SELECT

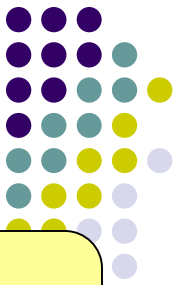


```
SELECT [DISTINCT]
campo1 [,campo2]
FROM tabla1 [,tabla2]
[WHERE condición]
```

➤ Seleccionar los socios que alquilaron estrenos del mes en marzo de 2005

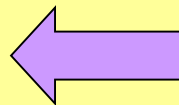
```
SELECT S.Soc_Apellido, S.Soc_Nombre
FROM Socios as S, Alquileres as A,
      Peliculas as P
WHERE S.Soc_Codigo = A.Soc_Codigo and
      P.Pel_Codigo = A.Pel_Codigo and
      month(A.Alq_Entrega) = 3 and
      year(A.Alq_Entrega) = 2005 and
      month(P.Pel_Estreno) = 3 and
      year(P.Pel_Estreno) = 2005
```

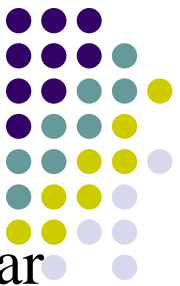
Consultas - SELECT



```
SELECT [DISTINCT]
campo1 [,campo2]
FROM tabla1 [,tabla2]
[WHERE condición]
[GROUP BY campos]
[HAVING condición]
[ORDER BY campos]
```

Orden en que deben aparecer
los registros en el resultado





Consultas - SELECT

- Seleccionar los socios nacidos en el mes de marzo. Mostrar los resultados ordenados por apellido del socio

```
SELECT Soc_Apellido, Soc_Nombre, Soc_Calle,  
Soc_Numero, Soc_Depto, Soc_FechaNac  
FROM Socios  
WHERE month(Soc_FechaNac) = 3  
ORDER BY Soc_Apellido
```

| | Soc_Apellido | Soc_Nombre | Soc_Calle | Soc_Numero | Soc_Depto | Soc_FechaNac |
|---|--------------|------------|------------|------------|-----------|--------------|
| | Avalos | Maria | Nueva York | 3702 | | 04/03/1972 |
| | Garcia | Jose | Callao | 1527 | 2do "A" | 10/03/1994 |
| | Gonzalez | Alberto | Boyaca | 3586 | | 12/03/1970 |
| | Portal | Carina | Talcahuano | 2212 | PB 1 | 04/03/1942 |
| | Valenzuela | Maria | Pepirí | 3948 | | 21/03/1960 |
| | Zapata | Jimena | Honduras | 3452 | | 10/03/1950 |
| ▶ | | | | | | |



Consultas - SELECT

➤ Seleccionar los estrenos del mes de abril de 2005. Listar nombre, género y fecha de estreno, ordenados por nombre.

```
SELECT P.Pel_Nombre, G.Gen_Nombre,  
P.Pel_Estreno  
FROM Peliculas as P , Generos as G  
WHERE P.Gen_Codigo = G.Gen_Codigo and  
      month(P.Pel_Estreno) = 4 and  
      year(P.Pel_Estreno) = 2005  
ORDER BY P.Pel_Nombre
```

| | Pel_Nombre | Gen_Nombre | Pel_Estreno |
|---|------------|------------|-------------|
| | Alma Mia | Comedia | 10/04/2005 |
| | Blade | Acción | 02/04/2005 |
| | EdTV | Comedia | 03/04/2005 |
| ▶ | Shrek | Infantil | 02/04/2005 |



Consultas - SELECT

➤ Seleccionar los estrenos del mes de abril de 2005. Listar nombre, género y fecha de estreno, ordenados por fecha de estreno y luego por nombre.

```
SELECT P.Pel_Nombre, G.Gen_Nombre,  
P.Pel_Estreno  
FROM Peliculas as P , Generos as G  
WHERE P.Gen_Codigo = G.Gen_Codigo and  
      month(P.Pel_Estreno) = 4 and  
      year(P.Pel_Estreno) = 2005  
ORDER BY P.Pel_Estreno, P.Pel_Nombre
```

| | Pel_Nombre | Gen_Nombre | Pel_Estreno |
|---|------------|------------|-------------|
| | Blade | Acción | 02/04/2005 |
| | Shrek | Infantil | 02/04/2005 |
| | EdTV | Comedia | 03/04/2005 |
| ▶ | Alma Mia | Comedia | 10/04/2005 |



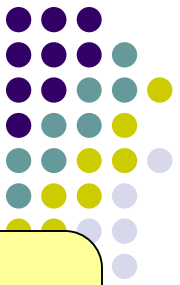
Consultas - SELECT

➤ Seleccionar los estrenos del mes de abril de 2005. Listar nombre, género y fecha de estreno, ordenados por género y luego por nombre.

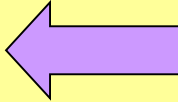
```
SELECT P.Pel_Nombre, G.Gen_Nombre,  
P.Pel_Estreno  
FROM Peliculas as P , Generos as G  
WHERE P.Gen_Codigo = G.Gen_Codigo and  
      month(P.Pel_Estreno) = 4 and  
      year(P.Pel_Estreno) = 2005  
ORDER BY G.Gen_Nombre, P.Pel_Nombre
```

| | Pel_Nombre | Gen_Nombre | Pel_Estreno |
|---|------------|------------|-------------|
| | Blade | Acción | 02/04/2005 |
| | Alma Mia | Comedia | 10/04/2005 |
| | EdTV | Comedia | 03/04/2005 |
| ▶ | Shrek | Infantil | 02/04/2005 |

Consultas - SELECT



```
SELECT [DISTINCT]
campo1 [,campo2]
FROM tabla1 [,tabla2]
[WHERE condición]
[GROUP BY campos]
[HAVING condición]
[ORDER BY campos]
```



Define agrupaciones sobre las que luego se pueden aplicar funciones.

En la cláusula GROUP BY se indica a partir de que columnas (que deben figurar en SELECT) se deben agrupar los datos. Aquellas columnas de SELECT que no son incluidas en GROUP BY deben aparecer afectadas por una función de agregación.

Consultas – SELECT

Funciones de agregación



| | |
|------------------|---------------------------------|
| AVG(expresión) | Promedia los valores |
| COUNT(expresión) | Cuenta las tuplas no nulas |
| COUNT(*) | Cuenta las tuplas |
| MIN(expresión) | Devuelve el mínimo valor |
| MAX(expresión) | Devuelve el máximo valor |
| SUM(expresión) | Devuelve la suma de los valores |



Consultas - SELECT

- Contar la cantidad de películas por género disponibles.

Se agrupan las películas según el género al que pertenecen y luego se cuenta la cantidad de películas incluidas en cada grupo.

```
SELECT G.Gen_Nombre, COUNT(P.Pel_Nombre)
FROM Peliculas as P, Generos as G
WHERE P.Gen_Codigo = G.Gen_Codigo
GROUP BY G.Gen_Nombre
ORDER BY G.Gen_Nombre
```

| | Gen_Nombre | Expr1001 |
|---|------------|----------|
| | Acción | 1 |
| | Comedia | 2 |
| ▶ | Infantil | 1 |

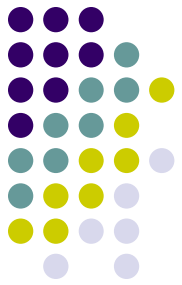


Consultas - SELECT

- Indicar el consumo (en dinero) realizado por cada socio.

Se agrupan todos los alquileres según el socio que los realizó y luego se acumulan los importes de cada grupo.

```
SELECT S.Soc_Apellido, S.Soc_Nombre,  
       SUM(A.Alq_Precio)  
FROM Socios as S , Alquileres as A  
WHERE S.Soc_Codigo = A.Soc_Codigo  
GROUP BY S.Soc_Apellido, S.Soc_Nombre  
ORDER BY S.Soc_Apellido, S.Soc_Nombre
```

Consultas - SELECT

➤ Indicar el consumo (en dinero) realizado por cada socio y por género de película.

Se agrupan todos los alquileres según el socio que los realizó y a que género pertenecen y luego se acumulan los importes de cada grupo.

```
SELECT S.Soc_Apellido, S.Soc_Nombre,  
       G.Gen_Nombre, SUM(A.Alq_Precio)  
FROM Socios as S , Alquileres as A ,  
     Películas as P, Generos as G  
WHERE S.Soc_Codigo = A.Soc_Codigo and  
       A.Pel_Codigo = P.Pel_Codigo and  
       P.Gen_Codigo = G.Gen_Codigo  
GROUP BY S.Soc_Apellido, S.Soc_Nombre,  
         G.Gen_Nombre
```



Resumen - *Structured Query Language*

Es un lenguaje de consulta estándar para bases de datos

Permite:

- Crear, modificar y eliminar tablas.

CREATE TABLE - ALTER TABLE - DROP TABLE

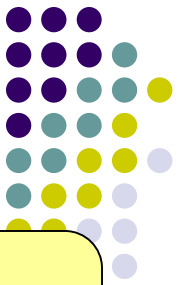
- Insertar, modificar y eliminar registros.

INSERT - UPDATE - DELETE

- Seleccionar registros que cumplen ciertos criterios

SELECT

Consultas - SELECT



```
SELECT [DISTINCT]
campo1 [,campo2]
FROM tabla1 [,tabla2]
[WHERE condición]
[GROUP BY campos]
[HAVING condición]
[ORDER BY campos]
```

Se definen condiciones que deben cumplir los resultados que surgen al aplicar funciones a grupos de registros



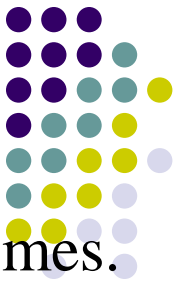
Consultas - SELECT

- Listar los géneros que cuentan con más de 500 películas.

Se agrupan las películas según el género al que pertenecen y luego se cuenta la cantidad de películas incluidas en cada grupo.

Finalmente se seleccionan aquellos grupos de más de 500 registros.

```
SELECT G.Gen_Nombre, COUNT(P.Pel_Nombre)
FROM Peliculas as P , Generos as G
WHERE P.Gen_Codigo = G.Gen_Codigo
GROUP BY G.Gen_Nombre
HAVING COUNT(P.Pel_Nombre) > 500
```



Consultas - SELECT

- Listar los socios que hayan gastado más de 100 pesos en un mes.

Se agrupan los alquileres según el socio que lo realizó y luego se suman los importes de cada grupo.

Finalmente se seleccionan aquellos grupos que suman más de 100.

```
SELECT S.Soc_Apellido, S.Soc_Nombre,  
       SUM(A.Alq_Precio)  
FROM Socios as S , Alquileres as A  
WHERE S.Soc_Codigo = A.Soc_Codigo  
GROUP BY S.Soc_Apellido, S.Soc_Nombre  
HAVING SUM(A.Alq_Precio) > 100
```

Combinación de los registros involucrados en la cláusula FROM



➤ Producto cartesiano

...

From TablaA, TablaB

...

Where TablaA.Campo1 = TablaB.Campo2

➤ Combinación de los registros de ambas tablas que coinciden en cierto/s campo/s

...

From TablaA INNER JOIN TablaB

ON TablaA.Campo1 = TablaB.Campo2

[AND TablaA.Campo3 = TablaB.Campo4]

...

Combinación de los registros involucrados en la cláusula FROM



➤ Combinación de los registros de ambas tablas que coinciden en cierto/s campo/s, **agregando** los registros de la primera tabla que no tienen coincidencia en la segunda

...

```
From TablaA LEFT JOIN TablaB
    ON TablaA.Campo1 = TablaB.Campo2
    [ AND TablaA.Campo3 = TablaB.Campo4 ]
```

...

Combinación de los registros involucrados en la cláusula FROM



➤ Combinación de los registros de ambas tablas que coinciden en cierto/s campo/s, agregando los registros de la **segunda** tabla que no tienen coincidencia en la **primera**

...

```
From TablaA RIGHT JOIN TablaB
    ON TablaA.Campo1 = TablaB.Campo2
    [ AND TablaA.Campo3 = TablaB.Campo4 ]
```

...

Combinación de los registros involucrados en la cláusula FROM



- Seleccionar las comedias estrenadas en febrero de 2005.

```
SELECT P.Pel_Nombre, P.Pel_Estreno
FROM Películas as P, Generos as G
WHERE P.Gen_Codigo = G.Gen_Codigo and
      G.Gen_Nombre = "Comedia" and
      month(P.Pel_Estreno) = 2 and
      year(P.Pel_Estreno) = 2005
```

```
SELECT P.Pel_Nombre, P.Pel_Estreno
FROM Películas as P INNER JOIN Generos as G
  ON P.Gen_Codigo = G.Gen_Codigo
WHERE G.Gen_Nombre = "Comedia" and
      month(P.Pel_Estreno) = 2 and
      year(P.Pel_Estreno) = 2005
```

Combinación de los registros involucrados en la cláusula FROM



- Listar el nombre y categoría de cada película (incluyendo aquellas que no tienen categoría asignada) estrenada en 2005.

```
SELECT P.Pel_Nombre, C.Cat_Nombre  
FROM Peliculas as P LEFT JOIN Categorías as C  
      ON P.Cat_Codigo = C.Cat_Codigo  
WHERE year(P.Pel_Estreno) = 2005
```

```
SELECT P.Pel_Nombre, C.Cat_Nombre  
FROM Categorías as C RIGHT JOIN Peliculas as P  
      ON P.Cat_Codigo = C.Cat_Codigo  
WHERE year(P.Pel_Estreno) = 2005
```

Combinación de los registros involucrados en la cláusula FROM



➤ Listar el nombre, categoría y cantidad de alquileres de cada película (incluyendo aquellas que no tienen categoría asignada) estrenada en 2005.

```
SELECT P.Pel_Nombre,C.Cat_Nombre,COUNT(A.Alq_Codigo)
FROM (Películas as P LEFT JOIN CATEGORIAS as C
      ON P.Cat_Codigo = C.Cat_Codigo)
LEFT JOIN Alquileres as A
      ON P.Pel_Codigo = A.Pel_Codigo
WHERE year(P.Pel_Estreno) = 2005
GROUP BY P.Pel_Nombre,C.Cat_Nombre
```

Consultas Anidadas - SELECT

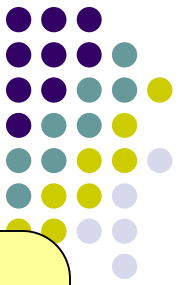


Se utilizan los resultados de una consulta en otra consulta.

Las posibilidades son:

- Incluir una consulta en lugar de una tabla en la cláusula FROM
- Incluir una consulta como parte de una condición de la cláusula WHERE

Consultas Anidadas en FROM



```
SELECT [DISTINCT] campo1 [,campo2]
FROM (SELECT [DISTINCT] campo1 [,campo2]
      FROM tabla1 [,tabla2]
      [WHERE condición]
      [GROUP BY campos]
      [HAVING condición]
      [ORDER BY campos]
      )as nombre [,tabla2]
```

```
[WHERE condición]
```

```
[GROUP BY campos]
```

```
[HAVING condición]
```

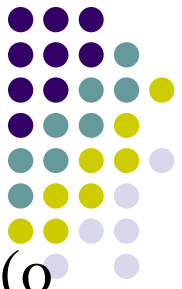
```
[ORDER BY campos]
```



Consultas Anidadas en FROM

- Listar nombre y apellido de los socios que hayan alquilado más de una película el mismo día en el año 2005.

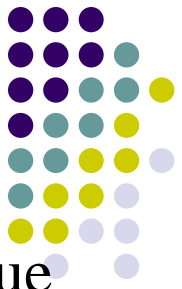
```
SELECT DISTINCT S.Soc_Apellido, S.Soc_Nombre
FROM (
    SELECT Soc_Codigo, Alq_Entrega, count(*)
    FROM Alquileres
    WHERE year(Alq_Entrega)=2005
    GROUP BY Soc_Codigo, Alq_Entrega
    HAVING count(*) > 1
) as T INNER JOIN Socios as S
ON S.Soc_Codigo = T.Soc_Codigo
```



Consultas Anidadas en WHERE

- Utilizando el operador IN: Se indica que un valor debe (o no debe) estar dentro del registros resultantes de una consulta.

```
SELECT [DISTINCT] campo1 [,campo2]
FROM tabla1 [,tabla2]
WHERE valor [not] in
    (SELECT campo
     FROM tabla1 [,tabla2]
     [WHERE condición]
     [GROUP BY campo]
    )
[GROUP BY campos]
[HAVING condición]
[ORDER BY campos]
```



Consultas Anidadas en WHERE

- Listar la cantidad de alquileres realizados en los meses que se estrenó alguna comedia

```
SELECT COUNT(*)  
FROM Alquileres  
WHERE month(Alq_Entrega) in (  
    SELECT distinct month(Pel_Estreno)  
    FROM Peliculas as P INNER JOIN Generos as G  
    ON P.Gen_Codigo = G.Gen_Codigo  
    WHERE G.Gen_Nombre = "Comedia" )
```




Consultas Anidadas en WHERE

➤ Utilizando el operadores relacionales: Se indica que un valor debe (o no debe) guardar cierta relación con el único registro resultante de una consulta.

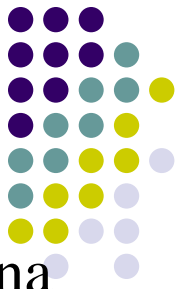
```
SELECT [DISTINCT] campo1 [,campo2]
FROM tabla1 [,tabla2]
WHERE valor operador
      (SELECT campo
        FROM tabla1 [,tabla2]
        [WHERE condición]
        [GROUP BY campo]
      )
[GROUP BY campos]
[HAVING condición]
[ORDER BY campos]
```



Consultas Anidadas en WHERE

- Listar todos los registros de alquileres cuyo valor coincide con el máximo importe cobrado por alquiler en el año 2005.

```
SELECT * FROM Alquileres
WHERE Alq_Precio = (
    SELECT max(Alq_Precio)
    FROM Alquileres
    WHERE year(Alq_Entrega) = 2005 )
```



Consultas Anidadas en WHERE

➤ Utilizando el operador **EXISTS**: Se indica que una subconsulta debe arrojar (o no) como resultado al menos un registro.

```
SELECT [DISTINCT] campo1 [,campo2]
FROM tabla1 [,tabla2]
WHERE [not] exists
      (SELECT [DISTINCT] campo1 [,campo2]
       FROM tabla1 [,tabla2]
       [WHERE condición]
       [GROUP BY campos]
       [HAVING condición]
      )
[GROUP BY campos]
[HAVING condición]
[ORDER BY campos]
```



Consultas Anidadas en WHERE

- Listar las películas que no fueron alquiladas en el año 2005.

```
SELECT P.Pel_Nombre FROM Peliculas as P
WHERE not Exists
      (SELECT * FROM Alquileres as A
       WHERE P.Pel_Codigo = A.Pel_Codigo
        and year(A.Alq_Entrega)=2005)
```

Bibliografía



- **“Microsoft Access Paso a Paso”, 2000, Microsoft Press**

