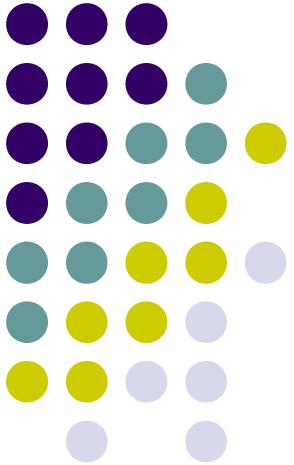
Bases de datos

Tema IV – El comando SELECT

Universidad Nacional del Litoral
FACULTAD DE INGENIERÍA
Y CIENCIAS HÍDRICAS



SQL - Select

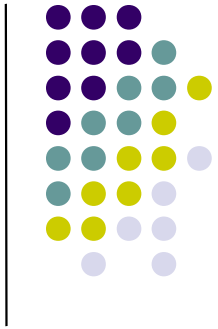
Formato

SELECT *select_list* ***Clausulas;***

Cláusulas

Cláusula	Finalidad
FROM	Nombres de las tablas de las que se seleccionan filas
WHERE	Especifica las condiciones que deben satisfacer las filas seleccionadas
GROUP BY	Agrupa las filas seleccionadas en grupos especificados
HAVING	Indica la condición que ha de satisfacer cada grupo mostrado en el **GROUP BY**
ORDER BY	Especifica el orden en que se mostrarán las filas seleccionadas

SQL – Select - Cláusulas

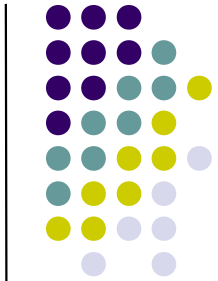


FROM

- Especifica una o varias tablas a partir de las cuales se recuperarán las filas que se desean.
- Si no hay expresiones optativas en la consulta (where, group by, having u order by), entonces la tabla que se ha recuperado es la tabla completa compuesta por las columnas de la lista objeto solamente (select_list).
- Si la lista objeto contiene columnas de más de una tabla, entonces la cláusula FROM deberá nombrar todas las tablas en cualquier orden, independientemente del orden de las columnas en la lista objeto.

```
SELECT id_provincia, nom_provincia
FROM persona.provincia;
```

SQL - Select - Cláusulas



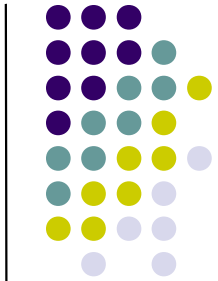
WHERE

- Especifica una tabla obtenida por la aplicación de una condición de búsqueda a las tablas que se listan en la cláusula FROM.

```
SELECT columna1, columna2, ....., columnaN
    FROM [esquema.]nombre_de_la_tabla
    WHERE condicion;
```

- Esta cláusula puede contener una o más subconsultas. Si es así, cada subconsulta que sigue a la cláusula WHERE se ejecuta para cada fila recuperada por la cláusula FROM.

SQL - Select - Cláusulas



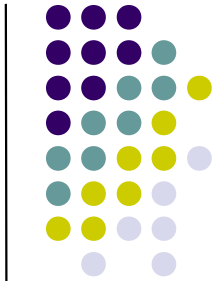
WHERE

```
SELECT * FROM persona.provincia
WHERE nom_provincia LIKE 'SAN%';
```

```
SELECT * FROM persona.provincia
WHERE nom_provincia LIKE 'S_N%';
```

```
SELECT * FROM persona.provincia
WHERE id_provincia > 1;
```

SQL - Select - Cláusulas

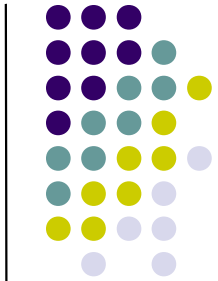


GROUP BY

- **Hace referencia de manera específica a una columna o varias columnas de la tabla nombrada en la cláusula FROM y agrupa las filas sobre la base de los valores de esas columnas.**

- **El resultado de una cláusula GROUP BY divide el resultado de la cláusula FROM en un conjunto de grupos, de forma que para cada grupo de más de una fila, los valores de la columna agrupada son idénticos.**

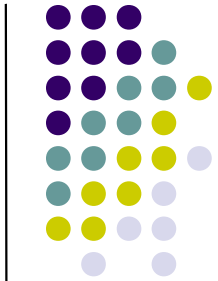
SQL - Select - Cláusulas



GROUP BY

```
SELECT columna1, columna2, ....., columnaN
    FROM [esquema.]nombre_de_la_tabla
    GROUP BY columna_de_agrupacion1, …
        columna_de_agrupacionN;
```

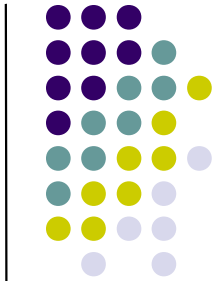
SQL - Select - Cláusulas



GROUP BY

- **TODAS LA COLUMNAS de la select_list deben estar en GROUP BY.**
- **Si el comando no contiene una cláusula WHERE, entonces la cláusula GROUP BY se coloca inmediatamente detrás de la cláusula FROM. Si el comando tiene la cláusula WHERE, entonces la cláusula GROUP BY va detrás de aquella.**
- **Las filas devueltas, estarán ordenadas al azar dentro de cada grupo porque esta cláusula no realiza ningún tipo de ordenamiento.**

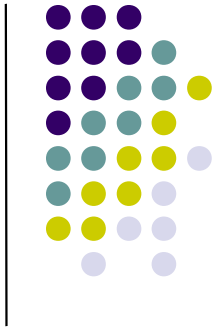
SQL - Select - Cláusulas



HAVING

- Especifica una restricción en la tabla agrupada que resulta de la cláusula anterior GROUP BY y elimina los grupos que no satisfagan la condición especificada.
- Si se especifica HAVING en una consulta, entonces también se tendrá que haber especificado GROUP BY.
- Se utiliza para especificar la cualidad que debe poseer un grupo para que éste sea devuelto.

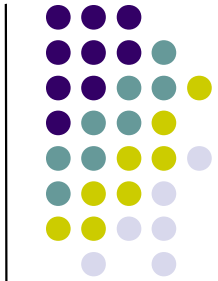
SQL - Select - Cláusulas



HAVING

- **Compara una propiedad del grupo con un valor específico. Realiza la misma función con los grupos que WHERE realiza con las filas individuales eliminando los grupos que no poseen la cualidad, de la misma manera que WHERE elimina las filas que no poseen esa cualidad.**

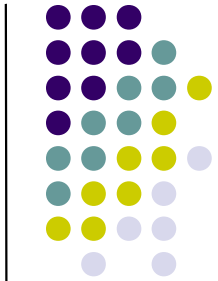
SQL - Select - Cláusulas



HAVING

```
SELECT columna1, columna2, ....., columnaN
    FROM [esquema.]nombre_de_la_tabla
    GROUP BY columna/s_de_agrupacion
    HAVING propiedad_especificada_del_grupo;
```

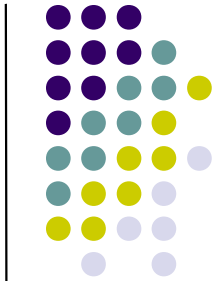
SQL - Select - Cláusulas



ORDER BY

- Especifica el orden en que aparecerán las filas en la recuperación haciendo una lista de las filas que hay en un grupo especificado de acuerdo con el valor creciente o decreciente.
- Si se usa la cláusula ORDER BY, ésta deberá ser la última cláusula del comando SELECT.
- Puede especificarse el orden ascendente (ASC) o descendente (DESC).
- Si la columna ordenada consta de letras SQL utilizará el orden alfabético ascendente (comenzando con la A) si no se especifica DESC.

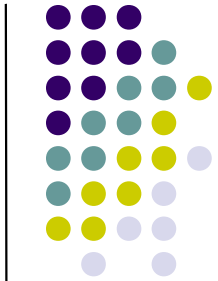
SQL - Select - Cláusulas



- **El orden en que los elementos de la lista se especifican (select_list) en la consulta determina el orden en que aparecerán las columnas en la recuperación. *No determina el orden en que aparecerán las filas*.**

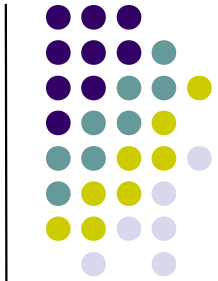
- **En caso de desear un ordenamiento específico, debe emplearse a continuación del WHERE la cláusula ORDER BY**

SQL - Select - Predicados



- **Los predicados son condiciones que se indican en la cláusula WHERE de una consulta SQL. Los predicados que permite SQL son:**
  - **la comparación**
  - **los cuantificadores**
  - **los existenciales**

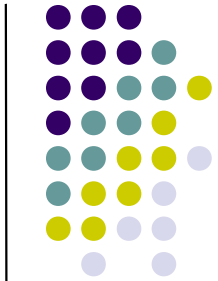
SQL - Select - Predicados



Comparación

- **Compara dos valores. Consta de una expresión de valor, seguida de un operador de comparación, seguido, a su vez, ya sea de otra expresión de valor o de una subconsulta de valor único.**
- **Los tipos de datos de las dos expresiones de valores, o la expresión de valor y la subconsulta, deben ser comparables.**

SQL - Select - Predicados



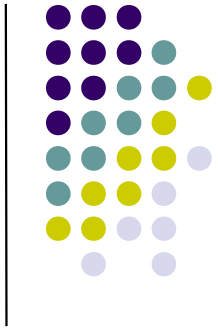
Comparación

- Los operadores de comparación incluidos en SQL son

  =, <> ó !=,<, >, <=, >=

- Si los valores que hay a ambos lados del operador de la comparación no son NULL, entonces el predicado de la comparación es verdadero o falso.

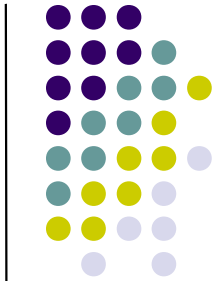
SQL - Select - Predicados



Comparación

- Si alguna de las dos expresiones de valores es un valor NULL o si la subconsulta está vacía, entonces el resultado del predicado de la comparación es desconocido;
- Cuando se usa GROUP BY, ORDER BY o DISTINCT junto con un predicado de comparación, un valor NULL es idéntico a otro valor NULL o es un duplicado del mismo.

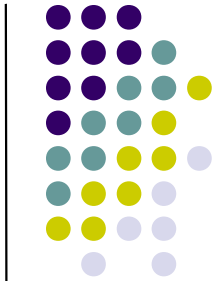
SQL - Select - Predicados



Comparación

- **Las cadenas de caracteres se pueden comparar por medio de los operadores de comparación mencionados antes. Esto se consigue comparando los caracteres que se encuentran en las mismas posiciones ordinales de la cadena. Así, dos cadenas de caracteres son iguales si todos los caracteres con la misma posición ordinal, son iguales.**

SQL - Select - Predicados



BETWEEN

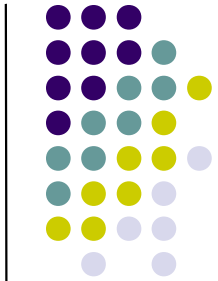
- Especifica la comparación dentro de un intervalo. La sintaxis es:

  . . . [NOT] BETWEEN *valor* AND *valor*

- Los tipos de datos de los valores, deben ser comparables.
- Para seleccionar las personas nacidas entre 01/01/2000 y el 01/01/2005 sería:

```
SELECT apenom, fenaci FROM persona.persona
    WHERE fenaci BETWEEN '2000-01-01' AND '2005-01-01';
```

SQL - Select - Predicados



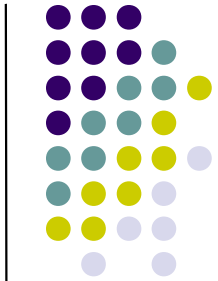
BETWEEN

- La respuesta que se obtendrá, viene dada en cualquier orden. Si se efectúa nuevamente la misma consulta, puede ser que el orden, sea totalmente distinto al obtenido en el primer intento.

- El término NOT se puede utilizar con BETWEEN para recuperar la información que hay fuera de un intervalo en lugar de la que hay dentro del mismo:

```
SELECT apenom, fenaci FROM persona.persona
    WHERE fenaci NOT BETWEEN '2000-01-01' AND '2005-01-01';
```

SQL - Select - Predicados



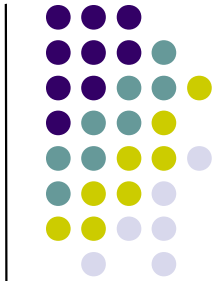
IN

- **Es equivalente al uso de OR. Especifica una comparación cuantificada. Hace una lista de un conjunto de valores y prueba si un valor está en esa lista.**

- **La lista debe ir entre paréntesis. Por ejemplo, para recuperar personas que tienen alguno de los tipo de documento LE, LC :**

```
SELECT apenom, nrodoc FROM persona.persona
    WHERE tipodoc IN ('LE','LC');
```

SQL - Select - Predicados



IN

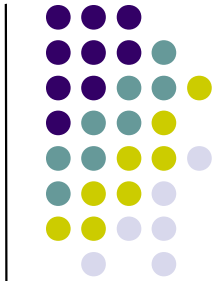
- **Es equivalente a:**

```
SELECT apenom, nrodoc FROM persona.persona
    WHERE tipodoc = 'LE' OR tipodoc = 'LC';
```

- **La consulta también se puede escribir usando ANY:**

```
SELECT apenom, nrodoc FROM persona.persona
    WHERE tipodoc = ANY ('LE','LC');
```

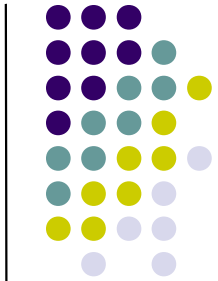
SQL - Select - Predicados



LIKE

- El predicado LIKE especifica una comparación de caracteres pudiendo usarse substrings y comodines.
- El *guión bajo* (“_”) representa un único carácter.
- El porcentaje (“%”) representa una cadena de caracteres de longitud arbitraria (incluyendo cero caracteres).
- Otro elemento más que se utiliza ya propio del motor en cuestión, es el símbolo arroba (“@”) usado como carácter de escape para poder utilizar símbolos especiales (caso de “_” o “%” o “).

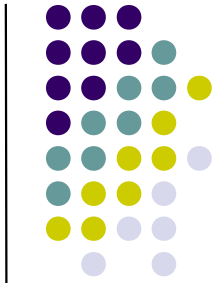
SQL - Select - Predicados



LIKE

Comodín	Significado
%	Cualquier string de cero o más caracteres
_	Cualquier carácter simple
[ ]	Cualquier carácter simple dentro del rango especificado entre corchetes
[^]	Cualquier carácter simple no incluido dentro del rango especificado entre corchetes

Igualdad entre caracteres



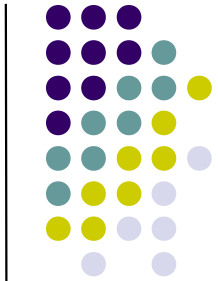
- Este ejemplo usa un conjunto de valores para encontrar aquellos nombres de persona que comiencen con las letras B, D, E, or R:

```
select * from persona.persona
        where apenom LIKE '[BDER]%';
```

- Este ejemplo usa un rango de caracteres para encontrar las personas cuyo nombre comienza con cualquier letra mayúscula entre la D y la H inclusive:

```
select * from persona.persona
        where apenom LIKE '[D-H]%';
```

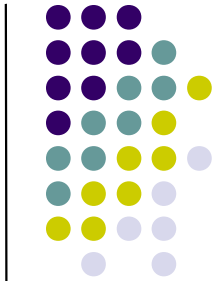
Igualdad entre caracteres



- **Este ejemplo usa un conjunto de valores, un comodín de posición, y un rango de caracteres para encontrar todas las salas cuyo nombre tenga seis caracteres y que los cuatro primeros sean ‘SALA‘, el quinto puede ser cualquier caracter y el sexto sea un número:**

```
select * from estructura.sala
        where nom_sala LIKE ‘SALA_[0-9]’;
```

SQL - Select - Predicados

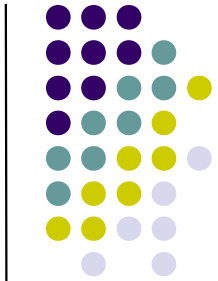


NULL

- Especifica la prueba que se lleva a cabo con un valor NULL. Por ejemplo, si se buscan los datos de las personas que no tienen un domicilio asignado y que debido a la falta de información se ha puesto NULL, NO se puede especificar:
  -
  - SELECT * FROM persona.persona
    WHERE domicilio = NULL;

… porque nada, ni incluso el mismo NULL, es igual al valor de NULL.

SQL - Select - Predicados

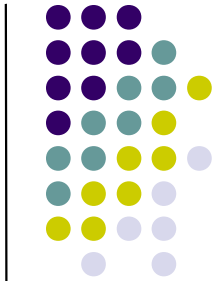


NULL

- **El único predicado que se puede utilizar cuando se busca un valor NULL es:**

```
WHERE
    especificacion_de_columna IS [NOT] NULL;
```

SQL - Select - Predicados

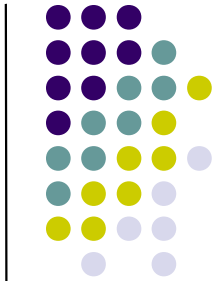


ALL, SOME, ANY

- Exigen que se use el predicado de la comparación aplicado a los resultados de una ***subconsulta***.
- Permiten probar un valor único frente a todos los elementos de un conjunto. Por ejemplo, podría desearse encontrar los proveedores que no pertenecen a la ciudad de Santa Fe y cuya fecha de inicio de actividades sea inferior a la de todos los proveedores que son de la ciudad de Santa Fe:

```
SELECT nom_provee FROM proveedor
      WHERE Ciudad <> “Santa Fe”
         AND fecha_inicio < ALL
               (SELECT  fecha_inicio  FROM  proveedor
                  WHERE Ciudad = “Santa Fe”);
```

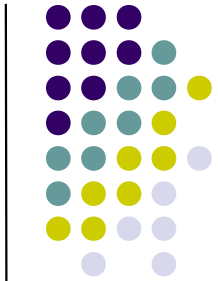
SQL - Select - Predicados



ALL, SOME, ANY

- Todos los otros operadores de comparación se pueden combinar con SOME, ANY y ALL.
- Cuando el operador de comparación que se está usando es igual, el término ANY se puede intercambiar con IN, e incluso algunas veces puede parecer más lógico usar el IN.
- En muchos casos, ANY tiene el mismo significado que SOME.

SQL - Select - Predicados



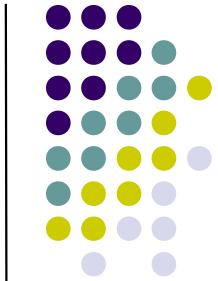
ALL, SOME, ANY

- Proveedores que no pertenecen a la ciudad de Santa Fe y cuya fecha de inicio de actividades sea inferior a la de algún proveedor de la ciudad de Santa Fe, se puede escribir de la siguiente forma:

```
SELECT nom_provee  FROM proveedor
  WHERE  ciudad <> "Santa Fe"
      AND fecha_inicio < ANY
          (SELECT fecha_inicio FROM proveedor
              WHERE ciudad = "Santa Fe");
```

- La comparación < ANY de la cláusula WHERE del SELECT exterior es verdadera, si la fecha de inicio es menor que al menos un elemento del conjunto formado por todos los proveedores de Santa Fe.

SQL - Select - Predicados

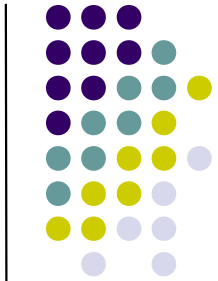


EXISTS

- Indica las condiciones de un conjunto vacío.
- Contiene una subconsulta que junto a EXISTS se puede evaluar como verdadera o falsa. Si el resultado de la subconsulta no existe, entonces el conjunto descrito por la subconsulta estará vacío.
- Representa el *cuantificador existencial de la lógica formal*.
- El predicado EXISTS se puede usar siempre que se pueda usar una consulta con el predicado IN aunque no siempre se puede usar IN en lugar de EXISTS.
- Sintaxis:

```
SELECT select_list FROM nombre_de_la_tabla
  WHERE [NOT] EXISTS(subconsulta);
```

Recuperación Simple: select / from



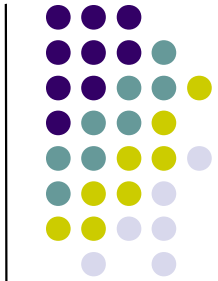
- **Lista únicamente los nombres de la tabla persona**

```
select apenom from persona.persona;
```

- **Este ejemplo, recupera más de una columna, listando documento y nombre de la persona:**

```
select tipodoc, nrodoc, apenom from persona.persona;
```

Recuperación Simple: select / from

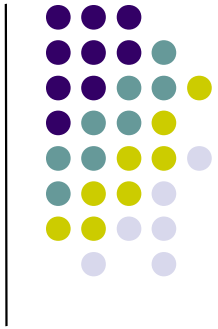


- Este ejemplo devuelve todas las columnas de la tabla persona. "*" es equivalente a "todas las columnas". El orden de las columnas es el de la definición de la tabla cuando fue creada.

```
select * from persona.persona;
```

- El orden de los nombres de las columnas en la select list determina el orden de las columnas en la salida.

Recuperación Simple: select / from

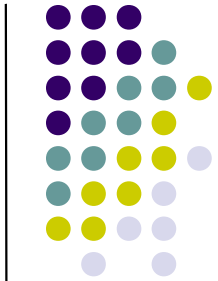


- El nombre de las columnas puede ser cambiado en el resultado para mayor claridad, por ejemplo:

```
select 'nombre' = apenom ó
select apenom as 'nombre' ó
select apenom nombre
    from persona.persona;
```

- No en todos los softwares de base de datos se permiten todas las formas mostradas.

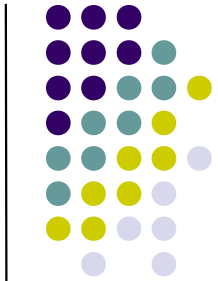
Funciones Agregadas



Función	Significado
count(*)	Cantidad de filas seleccionadas
count(col_name)	Cantidad de valores no nulos en la col.
max(col_name)	Mayor valor en la columna
min(col_name)	Menor valor en la columna
sum(col_name)	Sumatoria de valores de la columna
avg(col_name)	Promedio de valores de la columna

- **Ignoran los valores NULL (excepto count (*))**
- **sum y avg sólo trabajan con datos numéricos**
- **Si la cláusula group by no es usada, sólo una fila es devuelta**
- **NO pueden ser usadas en la cláusula where**
- **Pueden ser aplicadas a todas las filas o a un** subconjunto de ellas

Función Agregada: count



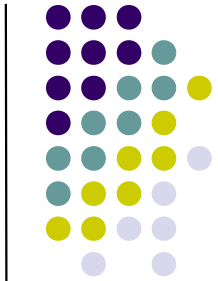
- La función count cuenta el número de filas que cumplen con una condición. El ejemplo devuelve la cantidad de personas de la tabla:

```
select count(*) from persona.persona;
```

- El ejemplo cuenta el número de filas en que la columna especificada (domicilio) no es NULL:

```
select count(domicilio) from persona.persona;
```

Funciones Agregadas: max / min



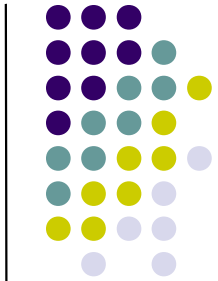
- La función **max** encuentra el valor más grande en la columna (la mayor fecha de un nacimiento):

```
select max(fenaci) from persona.persona;
```

- La función **min** encuentra el valor más pequeño en la columna (la menor fecha de un nacimiento):

```
select min(fenaci) from persona.persona;
```

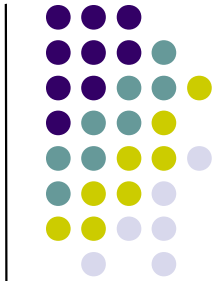
Funciones Agregadas: sum / avg



- La función sum suma todos los valores de la columna

- La función avg suma todos los valores de la columna y lo divide por el número de filas que ha encontrado

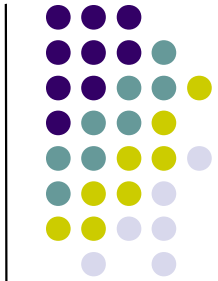
Funciones Agregadas: distinct



- La palabra clave distinct elimina duplicados antes que la función agregada sea aplicada
- La palabra clave distinct es:
  - Permitida con sum, avg, count, y count(col_name)
  - No permitida con min, max, y count(*)
  - Usada únicamente con nombres de columnas, no con expresiones aritméticas
- Ejemplo de cuántos tipos distintos de documento hay en la tabla persona:

```
select count(distinct tipodoc) from persona.persona;
```

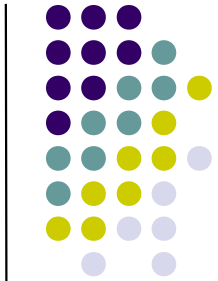
select / group by



- La cláusula group by organiza los datos en conjuntos que los agrupan tomando como base el contenido de una o varias columnas
  - La cláusula generalmente contiene una función agregada en la *select_list*
  - La función agregada es calculada para cada grupo
  - Todos los valores NULL en la columna del group by son tratados como un grupo
- El ejemplo agrupa las filas por provincia y calcula la cantidad de localidades de cada una:

```
select id_provincia, count(*) from persona.localidad
        group by id_provincia;
```

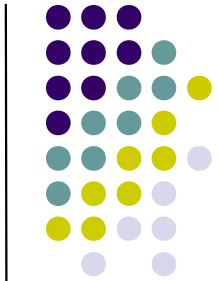
select / group by



- El agrupamiento puede hacerse por una columna o por una expresión que no contiene una función agregada

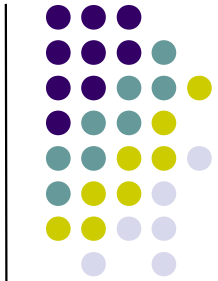
- La cláusula group by generalmente contiene todas las columnas y expresiones que aparecen en la *select_list* no afectadas por una función agregada

group by con una cláusula where y having



- La cláusula where:
  - Elimina las filas antes que vayan a los grupos
  - Aplica una condición a la tabla antes que se formen los grupos
  - No acepta funciones agregadas (las condiciones de búsqueda son evaluadas de a una fila por vez)
- La cláusula having:
  - Restringe los grupos
  - Aplica una condición a los grupos después que han sido formados

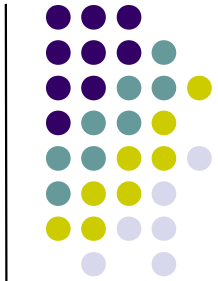
group by con la cláusula where y having



- **Devuelve por provincia la cantidad de localidades cuyo nombre comienza con la letra S pero solamente de aquellas provincias que tengan más de 5 localidades con esta característica.**

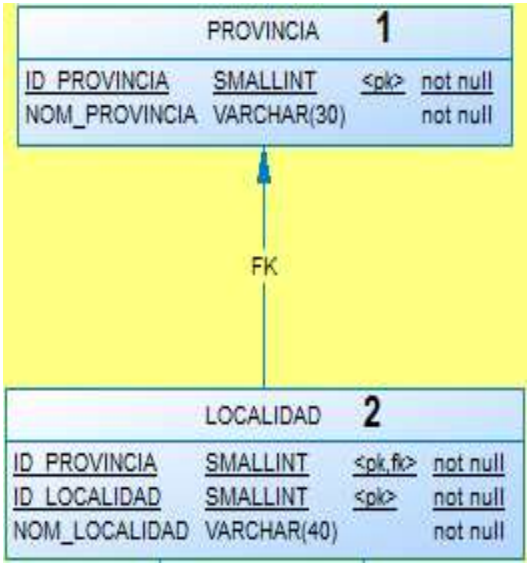
```
select id_provincia, count(*)
        from persona.localidad
        where nom_localidad like 'S%'
        group by id_provincia
        having count(*) > 5;
```

Por qué un Join?



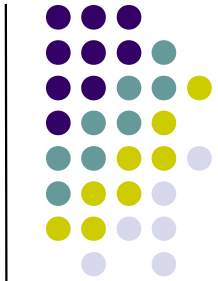
- **Es la manera de poder vincular las distintas tablas del modelo.**
- **Para resolver las consultas se debe decidir cuáles columnas se desean presentar**
- **Usar el modelo físico donde aparezcan las columnas de datos y consecuentemente las tablas que los contienen**
- **Seguir las líneas de referencia entre las tablas para encontrar el *join* que las conecta.**
- **Si se quiere ver una salida en que aparezcan nombre de localidad y nombre de provincia a la que pertenecen, serán necesarias las tablas localidad y provincia.**

Cuáles tablas deben utilizarse?



- **Las columnas en la condición de join no necesitan estar presentes en la select_list**
- **Las columnas con el mismo nombre en varias tablas deberán estar precedidas del nombre de la tabla (o un alias)**
- **La lista de columnas de la cláusula select no necesitan estar presentes en la condición de join**

Relacionando Tablas

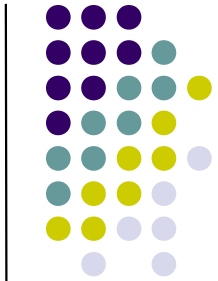


- **Listar nombre de localidad y nombre de provincia a la que pertenecen:**

```
select loca.nom_localidad localidad,
       prov.nom_provincia provincia
  from persona.provincia prov, persona.localidad loca
  where loca.id_provincia = prov.id_provincia;
```

- **El ejemplo utiliza alias en nombres de columnas y alias en el nombre de tablas.**
- **El JOIN es loca.id_provincia = prov.id_provincia**
- **Una fila en una tabla se empareja con una fila en otra mediante la igualdad del contenido de una o más columnas.**

Producto Cartesiano

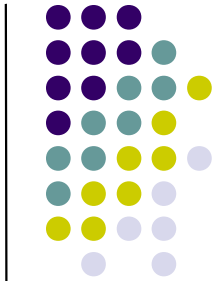


- Si no se especifica cómo realizar el join de las filas de diferentes tablas, el sistema asume que se desea realizar un join entre todas las filas de ambas tablas

- Este tipo de join es denominado Producto Cartesiano o producto cruzado

- Este tipo de join puede ser muy costoso (en tiempo de CPU) y puede demorar bastante hasta retornar un resultado muy grande y probablemente sin sentido alguno

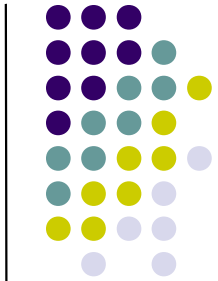
Joins con Order by y Group by



- Obtener un listado ordenado por nombre de provincia que contenga el nombre de la provincia y la cantidad de localidades que posee cuyo nombre comience con la letra ‘S’ pero solamente de aquellas provincias que tengan más de 5 localidades con esta condición:

```
select nom_provincia provincia, count(*) localidades
  from persona.provincia p, persona.localidad l
  where nom_localidad like ‘S%’
      and p.id_provincia = l.id_provincia
  group by 1
  having count(*)>5
  order by 1;
```

Uso de alias en los joins

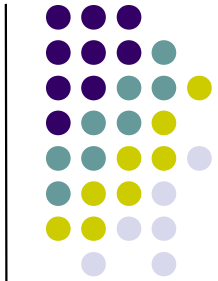


- **Un alias permite temporariamente utilizar un nombre de tabla definido por el usuario para reducir la longitud de las consultas**

- **Los alias de tabla deben ser usados siempre que se refiera a una columna que aparece en dos tablas distintas con el mismo nombre para evitar ambigüedades**

- **Cuando *n* tablas son unidas se requieren al menos *n-1* joins para evitar el producto Cartesiano.**

Join sobre más de dos tablas

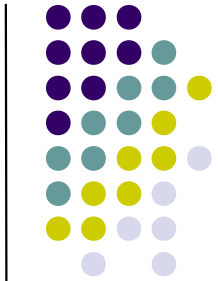


- La cláusula from debe listar todas las tablas involucradas en la consulta (aunque no se recuperen datos de una de ellas)

- La cláusula where debe listar alguna condición de join para conectar todas las tablas

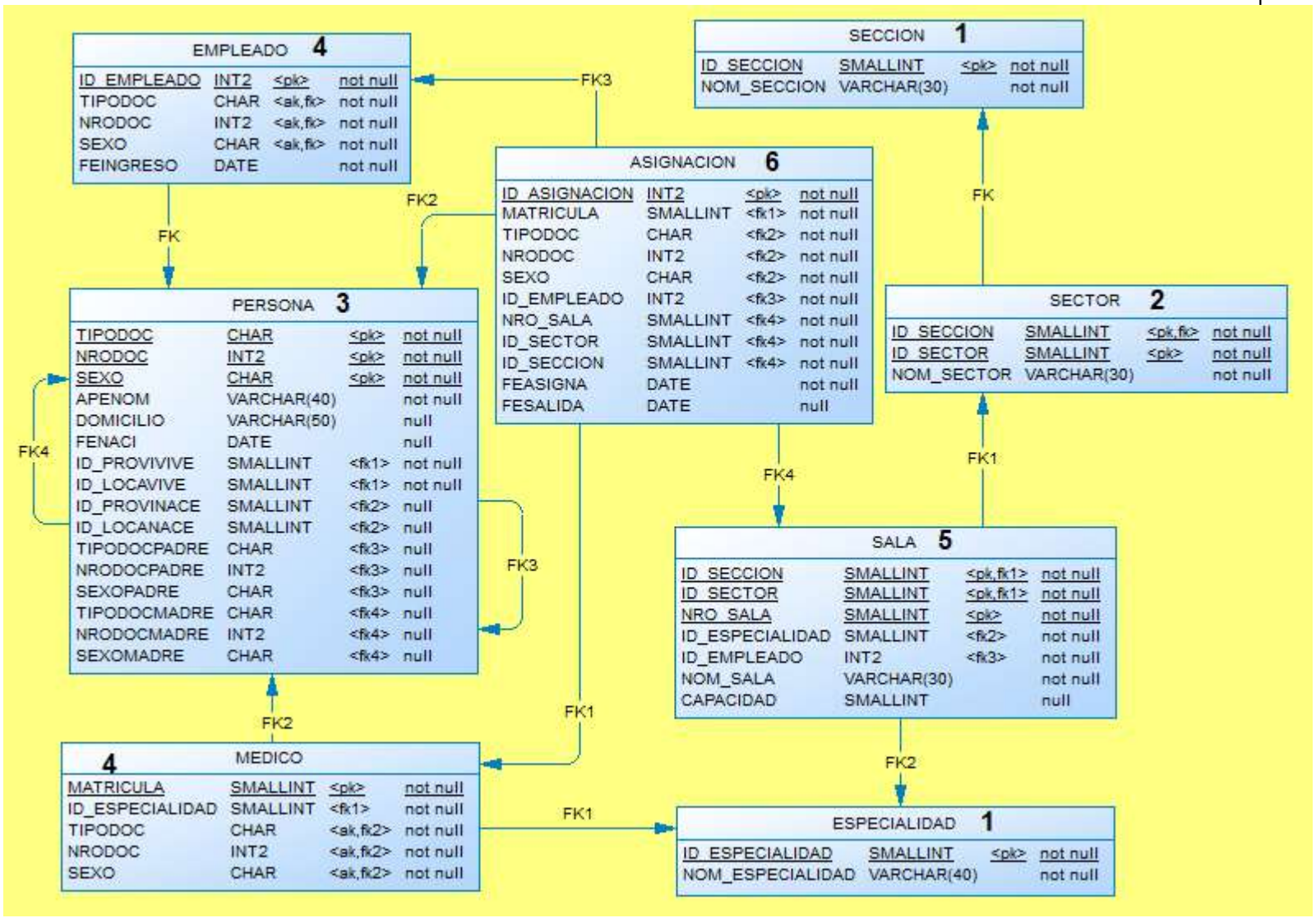
- No necesariamente aparecerán las columnas que están involucradas en el join

Joins sobre más de dos tablas

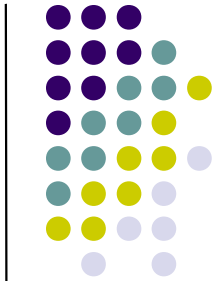


- **Ejemplo: obtener un listado de las asignaciones del día ordenado por nombre del internado mostrando:**
  - ***Número de asignación,***
  - ***Documento (tipo, número y sexo) y nombre del internado,***
  - ***Nombre del medico interviniente, matrícula y especialidad que tiene,***
  - ***Identificativo y nombre del empleado que realizó la carga,***
  - ***Nombre de la sección, del sector y de la sala donde la persona se interna***

Joins sobre más de dos tablas – Tablas que participan



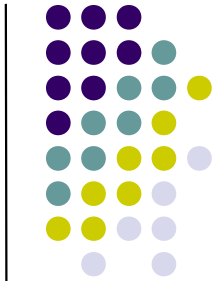
Joins sobre más de dos tablas



- **Ejemplo: obtener un listado de las asignaciones del día ordenado por nombre del internado mostrando:**
  - ***Número de asignación,***
  - ***Documento (tipo, número y sexo) y nombre del internado***

```
SELECT asig.id_asignacion, internado.tipodoc,
       internado.nrodoc, internado.apenom
FROM gestion.asignacion asig, persona.persona internado
WHERE asig.tipodoc = internado.tipodoc
   AND asig.nrodoc = internado.nrodoc
   AND asig.sexo = internado.sexo
   AND asig.feasigna = current_date
ORDER BY internado.apenom;
```

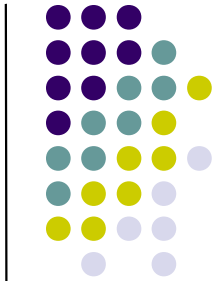
Joins sobre más de dos tablas



- **Ejemplo: obtener un listado de las asignaciones del día ordenado por nombre del internado mostrando:**

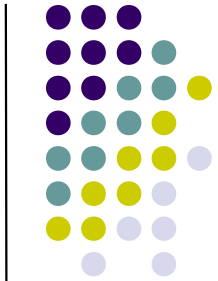
  - ***Número de asignación,***
  - ***Documento (tipo, número y sexo) y nombre del internado,***
  - ***Nombre del medico interviniente, matrícula y especialidad que tiene,***

Joins sobre más de dos tablas



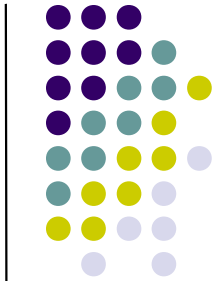
```
SELECT asig.id_asignacion, internado.tipodoc,
       internado.nrodoc, internado.sexo,
       internado.apenom internado, med.matricula,
       permed.apenom, esp.nom_especialidad
FROM gestion.asignacion asig, persona.persona internado,
     persona.medico med, persona.persona permed,
     gestion.especialidad esp
WHERE asig.tipodoc = internado.tipodoc
      AND asig.nrodoc = internado.nrodoc
      AND asig.sexo = internado.sexo
      AND asig.matricula = med.matricula
      AND med.tipodoc = permed.tipodoc
      AND med.nrodoc = permed.nrodoc
      AND med.sexo = permed.sexo
      AND med.id_especialidad = esp.id_especialidad
      AND asig.feasigna = current_date
ORDER BY internado.apenom;
```

Joins sobre más de dos tablas



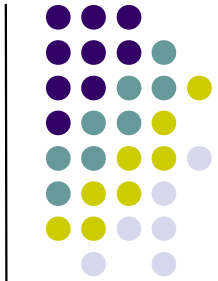
- **Ejemplo: obtener un listado de las asignaciones del día ordenado por nombre del internado mostrando:**
  - ***Número de asignación,***
  - ***Documento (tipo, número y sexo) y nombre del internado,***
  - ***Nombre del medico interviniente, matrícula y especialidad que tiene,***
  - ***Identificativo y nombre del empleado que realizó la carga,***

Joins sobre más de dos tablas



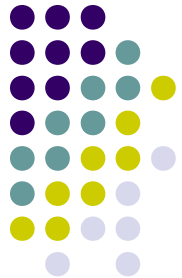
```
SELECT asig.id_asignacion, internado.tipodoc, internado.nrodoc, internado.sexo,
       internado.apenom internado, med.matricula, permed.apenom,
       esp.nom_especialidad, asig.id_empleado, peremp.apenom empleado
FROM gestion.asignacion asig, persona.persona internado,
     persona.medico med, persona.persona permed,
     gestion.especialidad esp, persona.empleado emp, persona.persona peremp
WHERE asig.tipodoc = internado.tipodoc
      AND asig.nrodoc = internado.nrodoc
      AND asig.sexo = internado.sexo
      AND asig.matricula = med.matricula
      AND med.tipodoc = permed.tipodoc
      AND med.nrodoc = permed.nrodoc
      AND med.sexo = permed.sexo
      AND med.id_especialidad = esp.id_especialidad
      AND asig.id_empleado = emp.id_empleado
      AND emp.tipodoc = peremp.tipodoc
      AND emp.nrodoc = peremp.nrodoc
      AND emp.sexo = peremp.sexo
      AND asig.feasigna = current_date
ORDER BY internado.apenom;
```

Joins sobre más de dos tablas



- **Ejemplo: obtener un listado de las asignaciones del día ordenado por nombre del internado mostrando:**
  - ***Número de asignación,***
  - ***Documento (tipo, número y sexo) y nombre del internado,***
  - ***Nombre del medico interviniente, matrícula y especialidad que tiene,***
  - ***Identificativo y nombre del empleado que realizó la carga,***
  - ***Nombre de la sección, del sector y de la sala donde la persona se interna***

Joins sobre más de dos tablas



```
SELECT asig.id_asignacion, internado.tipodoc, internado.nrodoc, internado.sexo,
       internado.apenom internado, med.matricula, permed.apenom,
       esp.nom_especialidad, asig.id_empleado, peremp.apenom empleado,
       secc.nom_seccion seccion, sect.nom_sector sector, sa.nom_sala sala
FROM gestion.asignacion asig, persona.persona internado,
     persona.medico med, persona.persona permed,
     gestion.especialidad esp, persona.empleado emp, persona.persona peremp,
     estructura.seccion secc, estructura.sector sect, estructura.sala sa
WHERE asig.tipodoc = internado.tipodoc
      AND asig.nrodoc = internado.nrodoc AND asig.sexo = internado.sexo
      AND asig.matricula = med.matricula AND med.tipodoc = permed.tipodoc
      AND med.nrodoc = permed.nrodoc AND med.sexo = permed.sexo
      AND med.id_especialidad = esp.id_especialidad
      AND asig.id_empleado = emp.id_empleado AND emp.tipodoc = peremp.tipodoc
      AND emp.nrodoc = peremp.nrodoc AND emp.sexo = peremp.sexo
      AND asig.id_seccion = sa.id_seccion AND asig.id_sector = sa.id_sector
      AND asig.nro_sala = sa.nro_sala AND sa.id_seccion = secc.id_seccion
      AND sa.id_sector = sect.id_sector AND sa.id_seccion = sect.id_seccion
      AND asig.feasigna = current_date
ORDER BY internado.apenom;
```