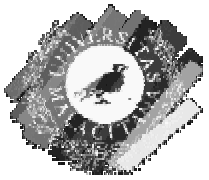


Departamento de Lenguajes y Ciencias de la Computación Universidad de Málaga

Bases de Datos

(Ingeniería Técnica en Informática de Sistemas)

2. Bases de Datos Relacionales: SQL y el PL/SQL de Oracle

E.T.S.I. Informática

J. Galindo Gómez

Modelo de DATOS RELACIONAL

- Introducido por Ted **Codd** de IBM, en 1970.
- Se basa en los conceptos de relación matemática, teoría de conjuntos y en la lógica de predicados de primer orden.
- **Base de Datos Relacional** (**BDR**) **=** Conjunto de Relaciones.
  - **Relación o Tabla** (*relation, table*)**:** **Lista de valores** con un nombre, donde cada valor es una fila (registro), compuesto por 1 o más columnas (campos).
    - **Fila o Tupla** (*row, tuple*): Hecho que corresponde a una entidad o relación en el mundo real. Sin repeticiones.
    - **Columna o Atributo** (*column, attribute*): Valor relacionado con ese hecho, sobre un aspecto particular.
      - Todos los valores de una columna son del **mismo tipo o dominio**.
      - Los valores (y el dominio) deben ser **atómicos o indivisibles** (como información). Ejemplos: Números naturales, reales, cadenas de caracteres...
      - **Formato:** Forma de representar un dato. Ej: Tlfno: +34 999-99-99-99.
        - » Una fecha también puede representarse de muchas formas.

Persona

Nombre	Altura	Peso
Santiago	1.79	78
Jesús	1.82	80
María	Null	61

INTENSIÓN y EXTENSIÓN de una BDR

- **INTENSIÓN o Esquema de una Relación:** Característicascasi invariables de la relación: **$R(A_1, A_2, ... A_n)$**
  - Nombre: R.
  - Grado (número de atributos): n
  - Nombre de los atributos: A_1, A_2, ... A_n.
  - Dominio de cada atributo: $dom(A_1)$, $dom(A_2)$, ... $dom(A_n)$.
    - $\forall i=1,2,...$ N: Null $\in dom(A_j)$
  - Ejemplo: **Persona(Nombre,Altura,Peso).**
- **EXTENSIÓN o RELACIÓN:** Valores de las tuplas, los cuales pueden variar con el tiempo.
  - Una relación r con esquema R, se representa como r(R) y es un conjunto de tuplas de la forma $\{t_1, ... t_m\}$, con $t_i = \{v_1, v_2, ... v_n\}$, tal que, $v_j \equiv t_i[A_j] \in dom(A_j)$.
  - Por tanto: $r(R) \subseteq (dom(A_1) \times dom(A_2) \times ... \times dom(A_n))$.
  - Con dominios finitos, el máximo número de tuplas es: $|dom(A_1)| * |dom(A_2)| * \cdots * |dom(A_n)|$

CARACTERÍSTICAS de las Relaciones

- **No existe orden entre las tuplas**, aunque lógicamente se almacenen y se visualicen en un cierto orden, pero ese orden puede variar sin que suponga alterar la relación.
- **No existe orden entre los atributos** y, por tanto, puede cambiarse el orden sin que suponga un cambio a la relación. Por comodidad suele suponerse un orden en los atributos y así se representarán los valores de las tuplas en ese orden.
- **Primera Forma Normal:** Los valores son atómicos o indivisibles. Valores compuestos o multivaluados no son permitidos.
  - **Atributos Compuestos:** Se representan sólo sus componentes simples.
  - **Atributos Multivaluados:** Se representa cada atributo multivaluado en una relación separada. Dicha relación almacenará todos los valores.
- **Interpretación de una Relación:**
  - El esquema de una relación es un aserto o afirmación que indica los atributos del objeto (entidad o relación) que representa la relación.
  - Cada tupla es un hecho o instancia de ese objeto, que puede representarse como un predicado.

NOTACIÓN del MODELO RELACIONAL

- **Esquema de una Relación de grado n: $R(A_1, A_2, ... A_n)$.**
- **Relación r con el esquema R: r(R).**
- **n-tupla t de r(R) con n valores componentes:**

  $$t = <v_1, v_2, ...v_n>, \ v_i \in A_i.$$

  - Valor v_i en t del atributo A_i: $t[A_j] = t.A_j$.
  - Subtupla con algunos valores de t: $t[A_u, ... A_z] = t.(A_u, ... A_z)$.
- **Atributo A de la relación R: R.A**
  - Así se diferencian atributos con igual nombre de distintas relaciones: R.A, S.A, Q.A...
  - No pueden existir dos atributos con el mismo nombre en una misma relación.
- Para evitar problemas en la implementación, los nombres de Atributos y Relaciones no deben contener espacios en blanco ni caracteres “raros” (acentos, ñ, Ñ...).

RESTRICCIONES RELACIONALES

- **Restricciones de DOMINIO:** Los valores deben pertenecer a su dominio correspondiente y ser atómicos. Puede restringirse el valor **Null.**
- **Restricciones de LLAVE:** En una relación no hay tuplas repetidas. Por tanto, hay un conjunto de atributos SK tal que no existen 2 tuplas con iguales valores en SK: $\forall t_1, t_2 \in r(R): t_1 \neq t_2 \rightarrow t_1[SK] \neq t_2[SK]$
  - **Superllave o Superclave** (*superkey*): Cualquier conjunto de atributos SK que cumple la **restricción de unicidad:** Que no existan dos tuplas con el mismo valor en los atributos de SK.
    - Todos los atributos de una relación forman siempre una superllave.
  - **Llave o Clave:** Superllave mínima, sin atributos superfluos. Si se quita un atributo de la llave, deja de ser superllave.
    - **Llave Candidata:** Llave que existe en una relación (pueden existir varias).
    - **Llave Primaria:** Llave Candidata que se usará para identificar las tuplas de la relación. Suele escogerse la que tenga menos atributos. Se representará subrayada. Ejemplo:

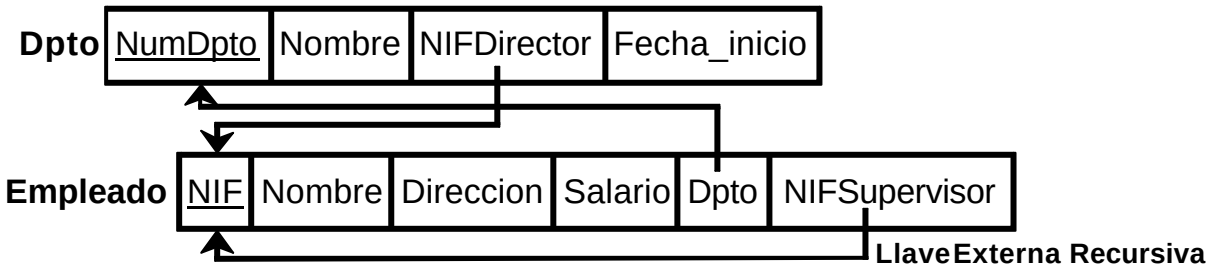
      Estudiante(NIF, Nombre, Fecha_Nacimiento, Direccion, Telefono);
    - **Llave Externa:** Atributos de una relación que son Llave Primaria en otra.
      - Sirve para enlazar una relación con otra.

BASES de DATOS RELACIONALES

- **Esquema de BDR:**
  - Formado por:
    - Conjunto de **Esquemas de Relación S: S={R_1, R_2 , ... R_n}.**
    - Conjunto de **Restricciones de Integridad**, para ser cumplidas.
  - Un esquema puede tener varios **ejemplares, instancias** (*database instance*) o **estados** (*database state*), con valores distintos en sus relaciones. Cada uno de ellos forma una **extensión** de la relación.
- **Restricciones de Integridad:**
  - **Integridad de Entidad:** Una llave primaria no puede ser Null.
  - **Integridad Referencial:** Si en una relación hay una referencia a una tupla de otra relación, esa referencia debe ser a una tupla que exista.
    - **Ejemplo: Dpto** (NumDpto,Nombre,NIFDirector...)
      - NIFDirector indica el NIF del director del departamento número NumDpto. Así, NIFDirector, debe existir en la relación **Empleado**(NIF,Nombre,Direccion...).
      - NIFDirector en la relación **Dpto** es una **LLAVE EXTERNA**.

BASES de DATOS RELACIONALES

- **Las Restricciones de Integridad Referencial (llaves externas...)** pueden representarse gráficamente**:**



- Los **SGBD** tienen mecanismos para forzar a que se cumplan las restricciones de integridad que se establezcan.
- **Restricciones de Integridad Semánticas:** Dependen del contexto.
  - Ejemplos: Salario **>** 0, Número de horas semanales trabajadas en un proyecto **<** 50, SalarioSupervisor **>** SalarioSupervisado...
  - Se controlan mediante disparadores (***triggers***) y condiciones (***assertions***).
- **Rest. de Estado:** Todas las ya vistas (Referencial y de Entidad).
  - Hacen que la BD sea consistente.
- **Rest. de Transición:** Se aplican en las modificaciones y dependen del estado anterior. Se controlan mediante *triggers*.
  - Ejemplo: El salario sólo puede aumentar.

INSERTAR en la BD: Restricciones

- **INSERTAR una tupla t en R (INSERT):**
  - Puede violar las restricciones siguientes:
    - **De Dominio:** Si un valor no pertenece al dominio que debe.
    - **De Llave:** Si la llave de la tupla ya existe en R.
    - **De Entidad:** Si la llave primaria vale Null.
    - **De Integridad Referencial:** Si una llave externa no existe en la relación referenciada y tiene un valor distinto de Null.
  - En ese caso, se pueden efectuar las siguientes acciones:
    - **Rechazar la inserción**, indicando el motivo.
    - **Solicitar** la **información** necesaria para subsanar la violación.

BORRAR de la BD: Restricciones

- **BORRAR una tupla t en R (DELETE):**
  - Sólo puede violar la restricción **de Integridad Referencial:** Si se borra una tupla que es referenciada en otra relación.
  - En ese caso, se pueden efectuar las siguientes acciones y el SGBD debería permitir al usuario elegir cual es la más apropiada para cada caso (cada llave externa):
    - **Rechazar el borrado**, indicando el motivo.
    - **Propagar el borrado**, borrando todas las tuplas que hagan referencia a la que se quiere borrar. El borrado de esas tuplas puede generar el borrado en cascada de otras.
    - **Modificar los valores** que hacen referencia a la tupla borrada: O se ponen otros valores válidos (valores por defecto, por ejemplo) o se establecen a Null (si no son parte de la llave primaria).

ACTUALIZAR en la BD: Restricciones

- **ACTUALIZAR valores de una tupla t en R (UPDATE):**
  - Puede violar las restricciones siguientes:
    - **De Dominio:** Si el nuevo valor no pertenece al dominio que debe.
    - **De Llave:** Si al modificar la llave de la tupla ya existe en R.
    - **De Entidad:** Si se actualiza la llave primaria al valor Null.
    - **De Integridad Referencial:**
      - Si al modificar una llave externa con un valor distinto de Null, ese valor no existe en la relación referenciada.
      - También, si modificamos una llave primaria que es referenciada como llave externa en otra relación.
  - En ese caso, existen las siguientes acciones:
    - **Rechazar la actualización**, indicando el motivo.
    - **Solicitar** la **información** necesaria para subsanar la violación.
    - Si se incumple la restricción de integridad referencial porque estamos modificando una llave primaria podemos optar por **propagar dicha modificación, o bien establecer los valores** de las llaves externas a **Null** o a un **valor por defecto**.

CONSULTAS: Álgebra Relacional

- **Álgebra Relacional:** Sistema para efectuar consultas a una BDR usando un conjunto de operaciones básicas sobre las relaciones que generan una relación como resultado. Las relaciones resultantes pueden también usarse para otras operaciones.
  - **Expresión de Álgebra Relacional:** Una secuencia de operaciones cuyos operandos básicos son relaciones de la BD.
    - El Álgebra Relacional es un lenguaje procedimental (se especifica la forma de obtener los resultados), al contrario que otros lenguajes como SQL o Cálculo Relacional.
  - **Tipos de Operaciones:**
    - **De Conjuntos:** Unión (*Union*, $\cup$), Intersección (*Intersection*, $\cap$), Diferencia (*Difference*, $-$) y Producto Cartesiano (*Cartesian Product*, $\times$).
    - **Específicas para BDR:** Selección (*Select*, σ), Proyección (*Project*, π) y Reunión (*Join*, $\bowtie$).
    - **Otras:** División (*Division*, $\div$), Clausura Transitiva (*Recursive Closure*), Reunión y Unión Externa (*Outer Join* y *Outer Union*)...

Operación de SELECCIÓN (s)

- **Objetivo: Seleccionar** el conjunto de tuplas de una relación **R** que satisfacen una condición **C**.
- **Representación:** Se utiliza la letra griega sigma: $\sigma_{C}(R)$.
- **Resultado:** Es una relación con los mismos atributos que **R**, pero en la que todas sus tuplas satisfacen la condición **C**.
  - **Condición C:** Es "*booleana*": Para cada tupla de **R** toma el valor **Verdad** o **Falso**.
    - Valores de atributos con **NULL** no cumplirán ninguna condición.
    - Cada condición simple o cláusula tiene el formato:
      <Atrib.> <Comparador> <Atrib.|Cte.del Dominio>
      - **<Comparador>** Î **{=, >, <, ³, £, ¹ }** (pueden considerarse otros).
      - Las cláusulas pueden conectarse con: **NOT**, **AND** y **OR**.
- **Proceso:** Se aplica la condición a cada tupla de **R**. Si la condición es Verdad (*true*), dicha tupla pertenecerá al resultado y si es Falsa (*false*), dicha tupla no será seleccionada.
- **Ejemplo:** $\sigma_{(Salario>2.5)\ AND\ (Nombre\ ^{1}\ 'Fulano')}(Empleado);$

PROYECCIÓN (p) y Renombrado (r)

- **Objetivo de la Proyección: Seleccionar** el conjunto de atributos o columnas que se indiquen en una lista de atributos L, mostrando sólo esa información.
- **Representación:** Se representa por la letra griega pi: $\pi_{L}(R)$.
- **Resultado:** El resultado es una relación con los atributos de L.
  - Si L no incluye una llave, podrán producirse tuplas repetidas en el resultado, las cuales serán eliminadas.
  - Si L es una superllave, el número de tuplas del resultado será el mismo que el de R. En otro caso será menor o igual.
- **Ejemplos:** $\pi_{NIF,Nombre,Dpto}(Empleado);$

  $\pi_{NIFDirector}(\sigma_{Fecha\_inicio<1/1/2002}(Dpto));$

  → { **Dpto2002** ¬ $\sigma_{Fecha\_inicio>1/1/2002}(Dpto);$
  $\pi_{NIFDirector}(Dpto2002);$
- **Renombrar (r):** El nombre de una relación, $\rho_{S}(R)$, el nombre de sus atributos, $\rho_{(B1,B2...Bn)}(R)$ o ambas cosas, $\rho_{S(B1,B2...Bn)}(R)$.

Operaciones de Conjuntos: È, Ç, - y ´

- **Unión R È S :** Tuplas que están en R, en S, **o** en ambas.
- **Intersección R Ç S:** Tuplas que están en R **y** en S.
- **Diferencia R - S:** Tuplas que están en R **y no** en S.
- Para esas 3 operaciones R y S deben ser **compatibles respecto a la unión:** Tener igual número y tipo de atributos.
- **Producto Cartesiano R(A_1, ... A_n) ´ S(B_1, ... B_m):** Relación **Q** de grado n+m, **Q(A_1, ... A_n,B_1, ... B_m)**, con todas las combinaciones posibles de tuplas de **R** y **S**.
  - El número de tuplas resultante es **|R| * |S|** (producto de ambos).
  - La operación del **Producto Cartesiano** no es muy útil por sí misma, pero permite combinar dos relaciones, para luego poder efectuar una **Selección** sobre esa combinación. Esas dos operaciones son tan comunes que tienen nombre propio:
    - **REUNIÓN** (*Join*, ⋈): **$R \bowtie_C S = \sigma_C(R \times S)$.**

REUNIÓN (⋈): $R \bowtie_C S = \sigma_C(R \times S)$

- **Ejemplo:** Con el esquema conceptual siguiente, hallar los nombres de los directores de cada departamento:

  Dpto (NumDpto, Nombre, NIFDirector, Fecha_inicio)
  Empleado (NIF, Nombre, Direccion, Salario, Dpto, NIFSupervisor)

  $\pi_{\text{Dpto.Nombre,Empleado.Nombre}}$ **(Dpto** $\bowtie_{\text{NIFDirector=NIF}}$ **Empleado);**
- **Tuplas con Null** en los "Atributos de la Reunión", no se incluyen en el resultado.
- **EQUI–REUNIÓN** (*Equijoin*): Reunión con el comparador **=**.
- **REUNIÓN NATURAL (R * S):** Es una equi-reunión usando los atributos de ambas relaciones que tengan el mismo nombre y eliminando uno de ellos, ya que ambos tendrán el mismo valor.
  - Si hay varios atributos con el mismo nombre, las condiciones irán enlazadas con el operador lógico AND.
  - En general, las llaves externas suelen tener el mismo nombre de atributo que los atributos a los que referencian. Así, la Reunión Natural evita la necesidad de indicar explícitamente la condición de reunión.

DIVISIÓN (¸): R(X,Y) ¸ S(Y) = Q(X)

- **DIVISIÓN R(X,Y) ¸ S(Y):** El resultado es una relación de esquema **Q(X)**, con todas las tuplas **t[X]** que están relacionadas en **R** con **TODAS** las tuplas **t[Y]** de **S**.
  - **X** e **Y** pueden ser atributos simples o conjuntos de atributos.
  - Observe que el conjunto de atributos de **S** es un subconjunto del conjunto de atributos de **R**.
- **Ejemplos:** Tengamos el esquema conceptual de la siguiente BD**:**

  Suministrador (S#, NombreS, Dirección, Ciudad);
  Pieza (P#, NombreP, Peso, Cantidad);
  Suministros (S#,P#);

  S# y P# son códigos numéridos para Suministrador y Pieza.

  - "Hallar el número de todos los suministradores que suministran **TODAS** las piezas":

    Suministros ¸ $\pi_{P\#}$ (Pieza);

  - "Hallar el número de todos los suministradores que suministran **TODAS** las piezas que pesan más de 20 grs":

    Suministra ¸ $\pi_{P\#}$ ($\sigma_{Peso>20}$(Pieza));

Álgebra Relacional: EJEMPLOS

Suministrador (S#, NombreS, Dirección, Ciudad);
Pieza (P#, NombreP, Peso, Cantidad);
Suministros (S#,P#);

- **E1:** "Nombres de Suministradores que suministren una Pieza de 15 grs.":
  $\pi_{NombreS}$ (($\sigma_{Peso=15}$(**Pieza**) ∗ **Suministros**) ∗ **Suministrador**);
- **E2:** "Nombres de los Suministradores de TODAS las Piezas":
  π_{Nombre} ((**Suministros** ¸ $\pi_{P\#}$ (**Pieza**)) ∗ **Suministrador**);
- **E3:** "Extraer los números de todos los Suministradores que suministren alguna Pieza que no sea la 8": $\pi_{S\#}$ ($\sigma_{P\#\neq 8}$(**Suministros**));
- **E4:** "Números de todos los Suministradores que NO suministren la Pieza 8": $\pi_{S\#}$ (**Suministrador**) – $\pi_{S\#}$ ($\sigma_{P\#=8}$(**Suministros**));
- **E5:** "Nombres de la consulta **E4**": $\pi_{NombreS}$ (**E4** ∗ **Suministrador**);
- **E6:** "Números de las Piezas suministradas por los Suministradores que NO suministran la Pieza 8: $\pi_{P\#}$ (**E4** ∗ **Suministros**);
- **E7:** "Números de Suministradores que NO provean una Pieza de 1 kilo": $\pi_{S\#}$ (**Suministrador**) – $\pi_{S\#}$ ($\sigma_{Peso=1000}$(**Pieza**) ∗ **Suministros**);
- **E8:** "Nombres de Piezas suministradas desde Málaga":
  $\pi_{NombreP}$ **(** ($\sigma_{Ciudad='Málaga'}$ (**Suministrador**) ∗ **Suministros**) ∗ **Pieza)**;

OPERACIONES PRIMITIVAS (básicas)

- **Conjunto de Operaciones COMPLETO:** Es el conjunto de operaciones tales que cualquier otra operación puede efectuarse con operaciones de ese conjunto: $\{\sigma, \pi, \cup, -, \times\}$.
  - $R \cap S = (R \cup S) - ((R - S) \cup (S - R))$;
  - $R \bowtie_C S = \sigma_C(R \times S)$;
  - $R(X,Y) \div S(Y) = \pi_X(R) - \pi_X((\pi_X(R) \times S) - R)$;
- Hay **Consultas Sin Solución** con las operaciones vistas:
  - **Funciones de Agregación o de Grupo** (también existen en SQL)**:** Se aplican sobre un grupo de valores.
    - **Ejemplos:** Suma (**SUM**), Media (**AVG**), Máximo (**MAX**), Mínimo (**MIN**), Contar (**COUNT**), Desviación Típica (**STDEV**)...
    - **Formato:** $\mathcal{F}_{\text{<Atribs. para Agrupar...>}\ \text{<F(Atrib)...>}}(R)$**;**
      - Si no aparece ningún atributo para agrupar se calcula la función de grupo **F** a TODOS los valores del atributo **A1** de TODAS las filas de **R**.
    - **Resultado:** Tiene atributos con los que se agrupa y un valor por cada función aplicada a un atributo. Ej.: Peso medio de las piezas de cada suministrador**:** $S\#\ \mathcal{F}_{AVG(Peso)}(\text{Pieza} * \text{Suministra})$**;**

REUNIÓN EXTERNA *(Outer Join)*

- En la **operación de Reunión** se eliminan las tuplas del producto cartesiano que **NO** cumplen la condición de reunión o tienen **NULL** en tales atributos. **Esas tuplas, pueden interesar, aunque no cumplan la condición.**
- **REUNIÓN EXTERNA IZQUIERDA** $(R \supset\bowtie_C S)$**:** Mantiene las tuplas de la primera relación **R**, aunque no cumplan la condición **C,** con los valores de **S**. En las tuplas que no están relacionadas con ninguna tupla de **S**, los atributos de **S** se ponen a **NULL**.
  - **Ej.: Empleado**(NIF,Nombre...), **Dpto**(NumDpto,NombreD,NIFDirector...). Listar los empleados, indicando si son directores de algún Depto.: $\pi_{NIF,Nombre,NombreD}(\text{Empleado} \supset\bowtie_{NIF=NIFDirector} \text{Dpto})$;
- **REUNIÓN EXTERNA DERECHA** $(R \bowtie\subset_C S)$**:** Similar a la Izquierda pero mantiene las tuplas de la segunda relación **S**.
- **REUNIÓN EXTERNA COMPLETA** $(R \supset\bowtie\subset_C S)$**:** Similar a las anteriores, manteniendo las tuplas de ambas relaciones.
- Esas tres operaciones existen en el lenguaje estándar **SQL2**.

CÁLCULO RELACIONAL: Introducción

- **Cálculo Relacional:** Es un Lenguaje de Consulta NO PROCEDURAL, escribimos una expresión indicando qué queremos y no cómo conseguirlo (al revés que en Álgebra Relacional, que es Procedural).
  - **Cálculo y Álgebra Relacional son lenguajes equivalentes:** Tienen idéntico poder expresivo.
  - **Lenguaje RELACIONALMENTE COMPLETO:** Si puede expresar todas las consultas que permite el Cálculo. Puede incluir operaciones adicionales (agregación, agrupar, ordenar...).
  - **Formato de las Expresiones en Cálculo: {t | COND(t)}**
    - **Variable tupla t:** Expresa la tupla de valores a recuperar.
    - **COND(t):** Expresa la condición (o Fórmula Bien Formada) que deben cumplir las tuplas para pertenecer al resultado.
    - **Ejemplos:** **Empleado**(NIF,Nombre,Nivel,Sueldo);
      Trabaja (NIF, Proyecto, Horas)
      - {e **| Empleado**(e) **AND** e.Sueldo>2.5}
      - {e.Nombre **| Empleado**(e) **AND** (e.Sueldo>2.5 **OR** e.Nivel='Jefe')}
      - {e.Nombre **| Empleado**(e) **AND**
        $ t (**Trabaja**(t) **AND** t.NIF=e.NIF **AND** t.Proyecto=12))}

QBE (Query By Example): Introducción

- **Lenguaje de Consulta GRÁFICO** presentado por Zloof en 1977 y desarrollado por IBM.
- **Características de QBE:**
  - **No tiene sintaxis:** El usuario debe rellenar unas plantillas (*templates*) en las que aparece el esquema de las relaciones.
    - Así, el usuario no tiene que recordar nombres de relaciones ni atributos.
  - En las plantillas de los esquemas, el usuario **construye un EJEMPLO** con lo que desea obtener.
    - Suele usarse una marca (P.) en los atributos de proyección.
    - Pueden establecerse condiciones adicionales escritas de forma normal.
  - Es similar al **Calculo Relacional de Dominios** y es **Relacionalmente Completo**.
  - Ejemplos:
    - Consulta con **AND:**

Empleado	NIF	Nombre	Nivel	Sueldo
		P.	**Jefe**	**>2.5**

    - Consulta con **OR:**

Empleado	NIF	Nombre	Nivel	Sueldo
		P.		**>2.5**
		P.	**Jefe**	

SQL (Structured Query Language)

- **SQL está en continua evolución:** Es una evolución del lenguaje SEQUEL de D.D. Chamberlin y R.F. Boyce (1974) y fue implementado por primera vez por IBM en su BDR llamado SYSTEM R.
  - **ISO** (*International Standards Organization*) y **ANSI** (*American National Standards Institute*) desarrollaron una versión estándar en 1986, llamada **SQL86** o **SQL1**. Posteriormente, se desarrolló **SQL92** o **SQL2**. Actualmente se desarrolla **SQL3**, que incluye conceptos de BD orientadas a objetos.
- **SQL es un lenguaje estándar para GESTIÓN de BDR:**
  - **Está incluido en muchos SGBD (DBMS),** como DB2 (de IBM), Oracle, Ingres, Informix, Sybase, Access, SQL Server...
    - Las mismas sentencias sirven en distintos SGBD.
    - Si se usan sólo las características estándares facilita la tarea de migrar de SGBD ® Hay funciones no estándar en algunos SGBD.
  - **Fácil de usar y aprender:** Tiene similitudes con el Álgebra Relacional, aunque se parece más al Cálculo y es más fácil e intuitivo que ambos lenguajes formales.
  - Aquí se incluye una introducción a SQL estándar con algunos comentarios sobre el SGBD Oracle.

SQL (Structured Query Language)

- **SQL es un lenguaje COMPLETO:** Incluye sentencias para
  - **DDL y DML:** Permite definir esquemas, consultar, borrar, actualizar...
  - **Definición de vistas:** Para ver la BD de distintas formas.
  - **Seguridad:** Permisos de acceso distintos para cada usuario.
  - **Definir restricciones de integridad:** Integridad referencial...
  - **Especificar control de transacciones:** Para grandes empresas, recuperación de errores, archivos históricos...
  - **Puede incrustarse en lenguajes de alto nivel** (C, C++, Pascal, COBOL...).
  - Permite **operaciones tan complejas** que su total definición es complicada: Sólo veremos un subconjunto de las operaciones posibles.
- **ESQUEMA** (*schema*)**:** Conjunto de elementos (tablas, vistas, permisos...) que pertenecen a la misma BD. Cada esquema tiene un nombre y un usuario propietario del esquema**:**

```
CREATE SCHEMA <Nombre> AUTHORIZATION <Usuario>;
```

- **CATÁLOGO** (*catalog*)**:** Conjunto de esquemas. Tiene un esquema especial llamado `INFORMATION_SCHEMA` que provee información sobre los demás esquemas, usuarios autorizados, definiciones de dominio, restricciones de integridad referencial (sólo entre tablas del mismo catálogo)...

SQL (Structured Query Language)

- **TIPOS de DATOS** de los atributos de las relaciones**:**
  - **Enteros** de distintos tamaños**: INTEGER** o **INT** y **SMALLINT**.
  - **Reales** de distinta precisión**: FLOAT(**p**)**, **REAL**, **DOUBLE PRECISION**, o el más genérico **DECIMAL(**precisión, escala**)** (o **NUMBER** en Oracle), donde precisión="número total de dígitos" (de 1 a 38 en Oracle) y escala="número de decimales" (de -84 a 127 en Oracle, con valor 0 por defecto). También admite **DEC(**p,e**)** o **NUMERIC(**p,e**)**.
  - **Caracteres: CHAR(**n**)** o **CHARACTER(**n**)**, n=longitud fija (por defecto n=1). También **VARCHAR(**n**)**, n=longitud máxima (Oracle aconseja usar **VARCHAR2** con 4000 de máximo). Para cadenas muy largas usar **LONG** (máximo 2GB) o **CLOB** (*Character Large OBject*, máx. 4GB).
    - **NCHAR** y **NVARCHAR2** usan el juego de caracteres **N**acional definido al crear la BD.
  - **Cadenas de bits** (para gráficos, sonidos, ficheros binarios...)**: BIT(**n**)**, n=longitud fija o **BIT VARYING(**n**)**, con n=longitud máxima. Por defecto n=1. En Oracle se prefiere usar **RAW(**tamaño_fijo_máx_2000bytes**)** o **LONG RAW** (sin argumento y con un tamaño máximo de 2GB). Últimamente Oracle aconseja usar **LOB** o **BLOB** (*Binary Large OBject*, máximo 4GB), o también **BFILE** para almacenar la localización de un fichero binario.
  - **Fecha y Hora: DATE** (año, mes, día: YYYY-MM-DD). **TIME** (horas, minutos, segundos: HH:MM:SS). **TIMESTAMP** incluye ambos (sinónimo a **DATE** en Oracle).
    - Se usan las funciones **TO_CHAR** y **TO_DATE** para convertir un dato de tipo fecha a texto y viceversa.
- **Definiciones de DOMINIOS:** Crear nuevos tipos de datos:

```
CREATE DOMAIN <Nombre> AS <Tipo>;
```

  - **Ejemplo:** `CREATE DOMAIN NIF_TYPE AS CHAR(12);`
  - **Utilidades:** Hacer las definiciones más legibles y hacer más fáciles los cambios de dominio de ciertos atributos. En Oracle se traduce como **CREATE TYPE**.

DDL de SQL: CREATE TABLE

- **Comando CREATE TABLE:** Crea una nueva relación/tabla **base** con su nombre, atributos (nombres y dominios) y restricciones:

```
CREATE TABLE <NombreTabla> (
      <NombreA1> <TipoA1> <RestricA1>, ...
      <RestriccionesTabla>);
```

- **Restricciones de Atributos** (puede haber varias por cada uno):
  - **NOT NULL** | - **UNIQUE** | - **PRIMARY KEY**
  - **DEFAULT** <Valor> | - **CHECK (**<Condición>**)** | - **REFERENCES...**
- **Restricciones de Tabla:** Pueden tener un **Nombre**, que se asigna al principio, con el formato: **CONSTRAINT <nombreR>** (las rest. de Atrib. también)
  - **PRIMARY KEY (**<Atributos de la Llave Primaria>**)** | - **CHECK (**<Condición>**)**
  - **UNIQUE (**<Llave Candidata o Secundaria>**)** | → **Tabla de Restricción**
  - **FOREIGN KEY (**<Llave Externa>**) REFERENCES** <Tabla>**(**<Atributos>**)**
    [ON DELETE {CASCADE | SET NULL | SET DEFAULT}]
    [ON UPDATE {CASCADE | SET NULL | SET DEFAULT}]

Si se **borra** la llave referenciada, se borran las tuplas que la referencian
Si se **actualiza** la llave referenciada, se actualizan las tuplas que la referencian

Si se **borra/actualiza** la llave referenciada, se ponen a NULL los valores que la referencian (llave externa).

Si se **borra/actualiza** la llave referenciada, se ponen los valores que la referencian a su valor por defecto.

Las Restricciones en Oracle

- **Propiamente, Oracle no distingue entre restricciones de tabla y restricciones de Atributo (o de Columna):**
  - Oracle las almacena todas en la vista **USER_CONSTRAINTS** del Diccionario de Datos, con atributos como:
    - **CONSTRAINT_NAME:** Nombre de la restricción. Si no se le ha dado uno, Oracle le asigna uno con un código.
    - **TABLE_NAME:** Nombre de la tabla con dicha restricción.
    - **CONSTRAINT_TYPE:** Es un carácter (**P** para **PRIMARY KEY**, **U** para **UNIQUE**, **R** para una restricción de integridad referencial, **C** para una restricción de tipo **CHECK** (o **NOT NULL**) con la condición almacenada en el atributo **SEARCH_CONDITION**…)
    - **STATUS:** Estado de la restricción (**ENABLE** o **DISABLE**).
  - La diferencia entre restricciones de tabla y de atributo es sólo a nivel sintáctico sobre dónde y cómo deben escribirse:
    - Las restricciones que involucren varios atributos deben ser consideradas forzosamente como restricciones de tabla.
    - Las restricciones **NOT NULL** y **DEFAULT** son forzosamente restricciones de atributo, aunque estrictamente hablando **DEFAULT** no es una restricción.
- **Oracle 8 no implementa las opciones de ON UPDATE ni la opción ON DELETE SET DEFAULT.**
  - Por defecto, Oracle no permite borrar una tupla si existe una o varias tuplas que estén haciendo referencia a algún valor de la tupla que se intenta borrar (ORA-2292).
- **Por defecto, las restricciones de tipo CHECK sólo se exigen si los atributos involucrados tienen valores distintos de NULL.**

DDL de SQL estándar: **DROP** *y* **ALTER**

- **Borrar Esquemas y Tablas: DROP**
  - **DROP SQUEMA <NombreEsquema [CASCADE | RESTRICT]**
    - **CASCADE** borra el esquema totalmente y **RESTRICT** sólo si está vacío.
  - **DROP TABLE <NombreTabla> [CASCADE | RESTRICT]**
    - **CASCADE:** Borra la tabla (y su contenido). Si hay referencias sobre ella (llaves externas en otras tablas), dichas restricciones son borradas.
    - **RESTRICT:** Borra la tabla si no hay referencias sobre ella.
    - En Oracle **RESTRICT** no existe y es la opción por defecto. Para borrar las restricciones que hacen referencia a la tabla se usa **CASCADE CONSTRAINTS**.
- **Modificar Tablas: ALTER TABLE <NombreTabla> <ACCIÓN>**
  - **Añadir columna: ADD (<NombreA, Tipo, Restric_de_Columna>);**
    - En Oracle, para modificar un atributo se usa **MODIFY** en vez de **ADD**.
  - **Borrar columna: DROP <NombreA> [CASCADE|RESTRICT];**
  - **Añadir restric. de Atributo: ALTER <NombreA> SET <RestricA>;**
  - **Borrar restric.: ALTER <NombreA> DROP <TipoRA: DEFAULT...>;**
  - **Borrar restric. de tabla** (debe tener un nombre)**:**
    DROP CONSTRAINT <NombreC> CASCADE;
  - **Añadir restric. de tabla:** **ADD (<Restric_de_Tabla>);**

ALTER TABLE *en Oracle 8*

- **Modificar Tablas: ALTER TABLE <NombreTabla> <ACCIÓN>**
  - **Añadir Columna: ADD (<NombreA> <Tipo> [<Restric_Columna>]);**
  - **Añadir Restricción de Tabla: ADD (<Restric_de_Tabla>);**
  - **Modif. Col.: MODIFY (<NombreA> [<Tipo>] [DEFAULT <expr>][[NOT] NULL]);**
  - **Estado de una Restricción: MODIFY CONSTRAINT <NombreR> <Estado>;**
    - donde **<Estado>** es: (el **<Estado>** puede seguir a toda restricción)
      - **[NOT] DEFERRABLE:** Indica si el chequeo de la restricción se aplaza al final de la transacción o no se aplaza (por defecto), comprobándola tras la sentencia DML.
      - **ENABLE [VALIDATE|NOVALIDATE] :** Activa la restricción para los nuevos datos (opción por defecto). **VALIDATE** (opción por defecto) exige que se compruebe si la restricción es *válida* en los datos antiguos (lo que tenía previamente la tabla).
      - **DISABLE**: Desactiva la restricción.
    - Si a una restricción no se le ha asignado ningún nombre, Oracle le asigna un nombre. Puede consultarse en la vista **USER_CONSTRAINTS** del diccionario.
  - **Borrar R. por Tipo: DROP {PRIMARY KEY | UNIQUE(<Cols>)} [CASCADE];**
    - No pueden borrarse esas dos restricciones si existe una llave externa que referencie sus atributos**:** Con **CASCADE** se borran todas esas llaves externas.
  - **Borrar Restricción por Nombre: DROP CONSTRAINT <NombreR>;**
    - Obligatorio para las restricciones de tipo **CHECK** y **REFERENCES**.
  - **Borrar Columna: DROP COLUMN <NombreA> [CASCADE CONSTRAINTS];**
  - **Renombrar Tabla: RENAME TO <Nuevo_Nombre_Tabla>;**

DDL de SQL: Ejemplos

```
- CREATE TABLE CLIENTE (
    NIF    CHAR(12) NOT NULL,
    Nombre CHAR(80) NOT NULL,
    Edad   NUMBER(2) CONSTRAINT Edad_Pos CHECK (Edad>0),
    CONSTRAINT ClientePK PRIMARY KEY(NIF));
- CREATE TABLE MASCOTA (
    NIF_Owner CHAR(12) NOT NULL,
    Nombre CHAR(30) NOT NULL,
    Fecha_Nac DATE,
    Especie CHAR(30) DEFAULT 'Perro' NOT NULL,
    PRIMARY KEY(NIF_Owner,Nombre),
    CONSTRAINT Sin_Propietario FOREIGN KEY (NIF_Owner)
      REFERENCES CLIENTE(NIF) ON DELETE CASCADE);
- ALTER TABLE CLIENTE ADD Telefono CHAR(20);
- ALTER TABLE CLIENTE DROP COLUMN Edad CASCADE CONSTRAINTS;
- ALTER TABLE MASCOTA DROP CONSTRAINT Sin_Propietario;
- ALTER TABLE MASCOTA MODIFY Especie NULL;
- ALTER TABLE MASCOTA DROP PRIMARY KEY;
- ALTER TABLE MASCOTA ADD PRIMARY KEY (NIF_Owner,Nombre);
```

DML de SQL: Consultas con SELECT

- **SELECT <Lista_Atributos>**
  FROM <Lista_Tablas>
  WHERE <Condición>;

Esquema Conceptual:
Suministrador (S#, NombreS, Dirección, Ciudad);
Pieza (P#, NombreP, Peso, Cantidad);
Suministros (S#,P#);

- **Ejemplos:**
  - "Nombres de Piezas que pesen más de 15 gramos y que su Cantidad sea menor o igual que 30":
    ```
    SELECT NombreP FROM Pieza
    WHERE Peso>15 AND Cantidad<=30;
    ```
  - "Piezas de las que se ignore el Peso":
    ```
    SELECT * FROM Pieza
    WHERE Peso IS NULL;
    ```
    No usar Peso=NULL. Cada NULL es distinto a otro.
  - "Números de Suministradores que Suministren una Pieza de 15 gramos":
    ```
    SELECT S# FROM Pieza, Suministros
    WHERE Pieza.P# = Suministros.P# AND Peso=15;
    ```
  - "Nombres de Suministradores que Suministren una Pieza de 15 gramos":
    ```
    SELECT NombreS FROM Suministrador, Pieza, Suministros
    WHERE Suministros.P# = Pieza.P# AND Peso=15
      AND Suministros.S# = Suministrador.S#;
    ```
  - "Nombres de Piezas suministradas desde Málaga": Usando **alias** de tablas.
    ```
    SELECT NombreP
    FROM Suministrador S, Pieza P, Suministros SP
    WHERE SP.P# = P.P# AND SP.S# = S.S# AND Ciudad='Málaga';
    ```


Consultas con SELECT: Observaciones

- **Cláusula SELECT:**
  - Pueden **renombrarse** los nombres de las columnas del resultado, poniendo el nuevo nombre justo después del atributo (usar comillas si son varias palabras).
  - Puede ponerse un ***** para indicar que se recuperen **todos los atributos** de todas las tablas de la cláusula FROM.
  - Puede usarse también el formato **<Tabla>.*** para referirse a **todos** los atributos de esa tabla.
  - Pueden seleccionarse **tuplas repetidas** (con todos sus atributos iguales), ya que SQL no establece esa condición (no son conjuntos de tuplas sino multi-conjuntos). Pueden eliminarse tuplas repetidas usando: **SELECT DISTINCT**.
- **Cláusula FROM:** Para especificar tablas, vistas o instantáneas (*snapshot*).
  - Pueden ponerse alias a las tablas: **<Tabla> [AS] <Alias>**.
  - Pueden renombrarse todos los atributos de las tablas después de establecer el alias: **<Tabla> [AS] <Alias(Nuevos Nombres de Atributos)>**
- **Cláusula WHERE:** La condición puede usar los op. lógicos **NOT**, **AND** y **OR**.
  - **Si no existe cláusula WHERE:** Se supone que se recuperan todas las tuplas (como si la condición fuese siempre verdad).
    - Si no existe cláusula **WHERE** y en la clásula **FROM** hay varias tablas, se obtiene el **Producto Cartesiano:** Por tanto, si hay varias tablas la condición establece una selección de tuplas del Producto Cartesiano.
    - Las condiciones de una **operación de REUNIÓN** hay que explicitarlas.

Consultas con SELECT*: Ejemplos*

- **Ejemplos:** **Empleado** (NIF, Nombre, Salario, Dpto, NIFSupervisor);
  - "Nombres de los empleados supervisores y sus empleados supervisados":
    ```
    SELECT E1.Nombre Supervisor, E2.N Supervisado
    FROM   Empleado AS E1, Empleado AS E2(NIF,N,S,D,NS)
    WHERE  E1.NIF = E2.NS;
    ```
    Esta consulta no puede plantearse para todos los niveles de supervisores/supervisados en SQL2 (en SQL3 sí hay consultas recursivas).
  - "Datos de Suministradores de Piezas de 5 gramos o cuya cantidad sea menor a 9":
    ```
    SELECT S.*  FROM Suministrador S, Pieza P, Suministros SP
    WHERE SP.P#=P.P# AND SP.S#=S.S# AND(Peso=5 OR Cantidad<9);
    ```
  - "Mostrar todas las posibles combinaciones de pares (Suministrador, Pieza), mostrando el número y el nombre para cada Suministrador y cada Pieza":
    ```
    SELECT S#, NombreS, P#, NombreP  FROM Suministrador, Pieza;
    ```
  - "Mostrar todas las combinaciones de pares (Suministrador, Pieza), tales que ese suministrador provea esa pieza":
    ```
    SELECT S.S#, S.NombreS, P.P#, P.NombreP
    FROM Suministrador S, Pieza P, Suministros SP
    WHERE SP.P#=P.P# AND SP.S#=S.S#;
    ```
  - "Números de Suministrador y Pesos (si se conocen) de todas las piezas que suministren":
    ```
    SELECT DISTINCT S#, Peso  FROM Suministros SP, Pieza P
    WHERE SP.P#=P.P# AND Peso IS NOT NULL;
    ```
    Si hay varias piezas con el mismo peso suministradas por el mismo suministrador, se eliminan, para que no aparezcan dos veces.

Consultas con SELECT*: Multiconjuntos*

- **Tuplas Repetidas:** Las tablas en SQL no son conjuntos de tuplas sino **Multiconjuntos**, pues admiten duplicados. En las consultas:
  - Eliminar duplicados es una tarea costosa (ordenar primero es mejor).
  - Los duplicados pueden interesar al usuario.
  - En funciones de agregación no se desean eliminar duplicados.
  - Una tabla con PRIMARY KEY/UNIQUE no admite duplicados
- **Operaciones de Conjuntos:** Pueden usarse entre varias subconsultas, actuando sobre los conjuntos de tuplas de cada una:
  - **UNION: Unión SIN duplicados** de las tuplas de las consultas.
  - **UNION ALL: Unión CON duplicados**.
  - **INTERSECT: Intersección** de conjuntos
  - **MINUS (o EXCEPT): Diferencia** (resta) de conjuntos.
- **Ejemplo:** "Números de Suministradores que NO suministren la Pieza 8":
  ```
  SELECT S# FROM Suministrador
  MINUS
  SELECT S# FROM Suministros WHERE P#=8;
  ```

Consultas con SELECT*: Operaciones*

- **Comparación de cadenas:** Puede usarse el comparador **[NOT] LIKE** con los caracteres **"%"** (para cualquier número de caracteres) y **"_"** (para un único carácter). "_" y "\%" son sendos símbolos. El operador **"||"** concatena cadenas.
  - "Seleccionar Empleados que tengan un apellido López":
    ```
    SELECT * FROM Empleado WHERE Nombre LIKE '%López%';
    ```
  - "Seleccionar Valores que hayan bajado un porcentaje acabado en 5":
    ```
    SELECT * FROM Valores WHERE Variacion LIKE '-_5\%';
    ```
- **Operaciones Aritméticas** (+,-,*,/)**:**
  - "Productos con su porcentaje de descuento posible aplicado si el descuento aplicado no supera los 5 Euros": Esquema **Producto**(Nombre, Precio, Descuento...).
    ```
    SELECT Precio, Descuento "Porcentaje Descuento",
           Precio-(Precio*Descuento/100) "Precio Final"
    FROM Producto WHERE (Precio*Descuento)/100 < 5;
    ```
- **Ordenar Resultados:** Cláusula **ORDER BY** seguida de una lista de atributos o de posiciones en la lista de atributos del **SELECT**.
  - Tras cada atributo puede ponerse:
    - **ASC** (ordenación ascendente, por defecto).
    - **DESC** (ordenación descendente).
  - "Seleccionar Empleados y sus sueldos aumentados un 10% si su sueldo inicial está entre 1 y 3 millones de euros, ordenando descendentemente según su sueldo incrementado"**:**
    ```
    SELECT Nombre, 1.1*Sueldo FROM Empleado
    WHERE Salario BETWEEN 1 AND 3
    ORDER BY 2 DESC;
    ```


SELECT*: Pseudocolumnas en Oracle*

- **Pseudocolumnas:** Son valores que se comportan como si fueran columnas, pero no tienen valores almacenados en la BD. No se pueden insertar, actualizar o borrar sus valores.
  - **ROWID:** Es la dirección de una fila concreta en una tabla concreta. Permite averiguar cómo se almacenan las filas de una tabla. Además, es la forma más rápida de acceder a una fila particular.
    - No debe usarse como llave primaria, pues el SGBD puede reorganizarlas (al borrar o insertar una fila, por ejemplo).
    - Ejemplo: `SELECT ROWID,NIF FROM Empleado WHERE Dpto=20;`
  - **ROWNUM:** Es el número de orden de cada fila en una consulta particular.
    - Se asigna en el orden en el que las tuplas van siendo seleccionadas.
    - Ejemplo: "Selecciona los 10 empleados con mayor salario"
      ```
      SELECT NIF FROM (SELECT * FROM Empleado ORDER BY Salario DESC)
             WHERE ROWNUM <= 10;
      ```
      - La subconsulta es necesaria para que el **ROWNUM** se aplique después de ordenar.
    - Consulta que no devuelve nada: `SELECT * FROM Empleado WHERE ROWNUM>1;`
      - La primera fila recuperada es la 1 y la condición es falsa, por lo que NO se recupera. La segunda tupla es ahora la primera y también hace que la condición sea falsa.
    - Se puede utilizar para actualizar un atributo de cierta tabla, con valores únicos y correlativos: `UPDATE Tabla SET AtributoNum = ROWNUM;`
  - Secuencia**.CURRVAL** y Secuencia**.NEXTVAL:** Permiten obtener el valor actual y el siguiente de una Secuencia previamente creada (**CREATE SEQUENCE**).
    - Una secuencia permite generar una sucesión de números únicos (sin repetirse), ideal para generar llaves primarias artificiales.
  - **LEVEL:** Es el nivel en una consulta jerárquica (usando **START WITH** y **CONNECT BY**).

SELECT*: Funciones en Oracle*

- **Funciones:** Oracle permite utilizar funciones ya definidas, que simplifican ciertas operaciones. Además, también permite crear nuestras propias funciones. Algunos Ejemplos:
  - **Funciones Numéricas: ABS** (valor absoluto), **SIN** (seno), **SINH** (seno hiperbólico), **COS** (coseno), **TAN** (tangente), **SQRT** (raíz cuadrada), **POWER(base,exp)** (potencia), **EXP** (exponencial con e = 2.71828183), **LN** (logaritmo neperiano), **LOG(b,n)** (logaritmo en base **b** de **n**), **ROUND** (redondear números), **TRUNC** (truncar números)...
  - **Funciones de Caracteres: CHR** (carácter que corresponde a un número), **LOWER** (devuelve la cadena del argumento con todas las letras en minúsculas), **UPPER** (pasa a mayúsculas), **LPAD**/**RPAD** (ajusta a la izquierda/derecha una cadena), **LTRIM**/**RTRIM** (quita ciertos caracteres de la izquierda/derecha de una cadena), **SUBSTR** (extrae una subcadena de una cadena), **REPLACE** (reemplaza subcadenas),
  - **Funciones de Caracteres que devuelven números: ASCII** (código ASCII de un carácter), **LENGTH**/**LENGTHB** (longitud de una cadena en caracteres/bytes), **INSTR** (buscar caracteres en una cadena)...
  - **Funciones de Fecha: ADD_MONTHS** (suma meses a una fecha), **LAST_DAY** (devuelve el último día del mes de una fecha dada), **SYSDATE** (fecha y hora actual del sistema)...
  - **Funciones de Conversión: TO_CHAR** (convierte fechas y números a cadenas de caracteres), **TO_DATE** (convierte una cadena a un dato de tipo fecha **DATE**), **TO_LOB** (convierte datos de tipo **LONG**/**LONG RAW** a tipo **LOB**), **TO_NUMBER** (convierte una cadena con un número a un número)...
    - Ej.: **SELECT TO_CHAR(SYSDATE, 'MM-DD-YYYY HH24:MI:SS') "Fecha" FROM DUAL;** (**DUAL** es una tabla usada cuando se necesita poner algo en el **FROM**)
  - **Otras funciones: GREATEST/LEAST** (devuelve el mayor/menor valor de una lista de valores), **NVL**/**NVL2** (para cambiar valores **NULL** por otros valores), **USER**/**UID** (devuelven el nombre/número del usuario actual de la sesión), **USERENV** (para obtener diversa información sobre la sesión actual: Lenguaje, si es DBA, número de sesión...)...

Subconsultas o Consultas Anidadas

- **Subconsulta o Consulta anidada** (*nested query*)**:** Sentencia **SELECT** incluida dentro de otra **SELECT** (llamada consulta externa, *outer query*).
  - Una **subconsulta** genera un **multiconjunto** de valores: Podemos ver si un valor **pertenece** al multiconjunto (**IN**), si es mayor, igual... que **todos** (**ALL**) o **alguno** (**SOME**, **ANY**) de los valores del multiconjunto:

Las **subconsultas** suelen ser **ineficientes.**

```
SELECT * FROM Pieza, Suministrador
WHERE (S#,P#) IN (SELECT * FROM Suministros);
```

Sustituye a una doble Reunión Natural.

```
SELECT * FROM Pieza
WHERE P# IN (SELECT P#
             FROM Suministros SP, Suministrador S
             WHERE SP.S#=S.S# AND Ciudad='Málaga')
   OR P# IN (SELECT P# FROM Pieza
             WHERE Peso IS NOT NULL AND Cantidad IN (15,20));
```

OJO: No es la mejor manera de efectuar esta consulta.

```
SELECT Nombre FROM Empleado
WHERE Salario > ALL (SELECT Salario FROM Empleado
                     WHERE Dpto IN (1,3,5,7)
                        OR Dpto IN (SELECT Dpto FROM Empleado
                                    WHERE NIFSupervisor=999));
```

```
SELECT Nombre FROM Empleado E
WHERE  Dpto IN (SELECT Dpto FROM Empleado
                WHERE Salario > E.Salario );
```

SUBCONSULTA correlacionada: Usa atributo de la consulta externa. **Se evalúa una vez por cada tupla de la consulta externa.**

Cuantificador Existencial: EXISTS

- **Función [NOT] EXISTS:** Comprueba si una subconsulta (normalmente correlacionada) es o no **vacía** (si recupera o no alguna tupla).
  - **Ejemplos:**
    - "Empleados tales que **NO existen** compañeros en el mismo Dpto.":
```
SELECT Nombre FROM Empleado E
WHERE  NOT EXISTS (SELECT * FROM Empleado
                   WHERE Dpto = E.Dpto AND NIF<>E.NIF);
```
    - "Empleados tales que **existen** compañeros de Dpto. que cobran más que ellos":
```
SELECT Nombre FROM Empleado E
WHERE  EXISTS (SELECT * FROM Empleado
               WHERE Salario > E.Salario AND Dpto = E.Dpto);
```
    - "Piezas de las que haya menos de 100 y **NO exista** actualmente ningún suministrador":
```
SELECT * FROM Pieza P
WHERE Cantidad<100 AND NOT EXISTS
  (SELECT * FROM Suministros SP WHERE SP.P#=P.P#);
```
    - **División Relacional, Suministra** ÷ ρ_{NumPieza} **(Pieza):** "Números de suministradores para los que **NO existen** piezas que **NO** sean suministradas por ellos":
```
SELECT S# FROM Suministros SP1
WHERE NOT EXISTS (SELECT * FROM Pieza P
                  WHERE NOT EXISTS
                    (SELECT * FROM Suministros SP2
                     WHERE SP2.S#=SP1.S# AND SP2.P#=P.P#));
```


Tablas de Reunión: JOIN

- **Tablas de Reunión** (*joined tables*)**:** Permite especificar, en la cláusula **FROM**, una relación que sea el resultado de una operación de **reunión** (**JOIN**) entre dos tablas. **Formato:**

```
...FROM (<Tabla1> JOIN <Tabla2> ON <Condic.>)
```

  - **Reunión con una Reunión:** Una tabla de un **JOIN**, puede ser un **JOIN:**
```
((<T1> JOIN <T2> ON <Cond1>) JOIN <T3> ON <Cond2>)
```
  - **Reunión Natural:** Si los atributos se llaman igual, puede usarse **NATURAL JOIN** y eliminar la **condición**.
    - Los atributos con igual nombre son igualados y aparecen sólo una vez en el resultado. Si no se llaman igual puede usarse también **NATURAL JOIN**, **cambiando los nombres** de los atributos:
```
<T1> NATURAL JOIN (<T2> AS <Nuevat2>(<Nuevos Nombres>))
```
  - **Tipos de Reunión:** SQL2 permite las expresiones: (**OUTER** es opcional)

INNER JOIN	®	**Reunión (equivalente a JOIN)**
NATURAL JOIN	®	**Reunión Natural**
LEFT OUTER JOIN	®	**Reunión Externa Izquierda**
RIGHT OUTER JOIN	®	**Reunión Externa Derecha**
FULL OUTER JOIN	®	**Reunión Externa Completa**

Ejemplos de Reuniones con **JOIN**

- "Números de **Suministradores** que Suministren una Pieza de 15 gramos":
```
SELECT S# FROM(Pieza P JOIN Suministros SP ON P.P#=SP.P#)
WHERE Peso=15;  ó  SELECT S# FROM (Pieza NATURAL JOIN
                                   Suministros) WHERE Peso=15;
```
(equivale a)

- "Nombres de los **empleados** y sus **supervisores** (si los tienen)":
```
SELECT E1.Nombre Empleado, E2.Nombre Supervisor
FROM (Empleado E1 LEFT OUTER JOIN Empleado E2
      ON E1.NIFSupervisor=E2.NIF);
```

- "Nombres **departamentos** (a la izda.) y los nombres de sus **directores** (a la dcha.), si los tienen (suponemos que pueden existir deptos. sin director)":
```
SELECT Dpto.Nombre, Empleado.Nombre
FROM (Empleado RIGHT OUTER JOIN Dpto ON NIF=NIFDirector);
```

- "**Empleados** y los nombres del **Dpto.** que dirigen (si dirigen algún Dpto.)":
```
SELECT NIF,Empleado.Nombre, Dpto.Nombre
FROM (Empleado LEFT OUTER JOIN Dpto ON NIF=NIFDirector);
```

- "Nombres de **Piezas** suministradas desde Málaga":
```
SELECT NombreP FROM ((Suministros NATURAL JOIN Pieza)
                     NATURAL JOIN Suministrador)
WHERE Ciudad='Málaga';
```


Funciones de Grupo o de Agregación

- **Funciones de Grupo o de Agregación:** Son funciones que se aplican sobre un grupo de valores del mismo dominio. Se aplican sobre un grupo de tuplas (o de atributos concretos de esas tuplas).
  - **COUNT:** Cuenta el número de tuplas del grupo (indicadas por *****).
    - "¿Cuántas piezas hay que pesen más de 15 gramos?":
```
SELECT COUNT(*) FROM Pieza WHERE Peso>15;
```
    - "¿Cuántos empleados hay en el Departamento de I+D?":
```
SELECT COUNT(*) FROM Empleado E, Dpto D
WHERE E.Dpto=D.NumDpto AND D.Nombre='I+D';
```
    - "¿Cuántos salarios **distintos** hay entre los empleados?":
```
SELECT COUNT(DISTINCT Salario) FROM Empleado;
```
→ **COUNT(Salario)** es lo mismo que **COUNT(*)** si no hay Nulls
    - "¿Cuántas piezas hay de las que se ignore su peso?":
```
SELECT COUNT(*) FROM Pieza WHERE Peso IS NULL;
```
  - **Observaciones:**
    - Los **NULL** se ignoran, excepto por **COUNT(*).**
    - **DISTINCT** elimina los valores duplicados (y no cuenta los **NULL**).
    - En el lugar de **DISTINCT** se puede usar **ALL** (opción por defecto).
    - **DISTINCT** y **ALL** pueden usarse en todas las funciones de grupo.

Funciones de Grupo o de Agregación

- **SUM, MAX, MIN, AVG, STDEV, VARIANCE:** Calcula la **suma** de los valores del grupo, el valor **mayor**, el valor **menor**, la **media aritmética** (*average*), la **desviación típica** (*standard deviation*) y la **varianza** (el cuadrado de la desviación típica).
  - ```
    SELECT
          SUM(Salario), MAX(Salario), MIN(Salario),
          AVG(Salario), VARIANCE(Salario)
          STDDEV(Salario), '=', SQRT(VARIANCE(Salario))
    FROM Empleado;
    ```
  - ```
    SELECT AVG(DISTINCT Salario) "Media"
    FROM Empleado
    WHERE Dpto NOT BETWEEN 5 AND 9;
    ```
  - ```
    SELECT SUM(Salario), AVG(Salario)
    FROM Empleado E, Dpto D
    WHERE E.Dpto=D.NumDpto AND D.Nombre='I+D';
    ```
- **Observación: DISTINCT** y **ALL** no tienen efecto en las funciones **MAX** y **MIN**.
- **Oracle incluye otras Funciones de Agregación más complejas: CORR** (coeficiente de correlación de un conjunto de pares de números), **COVAR_POP/COVAR_SAMP** (covarianza), Funciones **REGR_** (regresión lineal)...

Funciones de Grupo: *Ejemplos en subconsultas*

- **Funciones de Grupo en Subconsultas** (correlacionadas o no):
  - "Nombre de los suministradores que suministran más de 5 piezas":
    ```
    SELECT NombreS FROM Suministrador S
    WHERE 5 < (SELECT COUNT(*) FROM Suministros
               WHERE S#=S.S#);
    ```
  - "Nombre de las Piezas que pesan más que la media":
    ```
    SELECT NombreP FROM Pieza
    WHERE Peso > (SELECT AVG(Peso) FROM Pieza);
    ```
  - "Nombre de la pieza o piezas que pesen más":
    ```
    SELECT NombreP FROM Pieza
    WHERE Peso = (SELECT MAX(Peso) FROM Pieza);
    ```
  - "Nombre de piezas que son suministradas por varios suministradores":
    ```
    SELECT NombreP FROM Pieza
    WHERE 2 <= (SELECT COUNT(*) FROM Suministros
                WHERE P#=Pieza.P#);
    ```
  - "Nombre de piezas que provengan de 3 suministradores de Málaga":
    ```
    SELECT NombreP FROM Pieza P WHERE 3 =
           (SELECT COUNT(*)
            FROM Suministros SP, Suministradores S
            WHERE P#=P.P# AND SP.S#=S.S#
              AND Ciudad='Málaga');
    ```
    ```

Agrupación con GROUP BY

- **Las Funciones de Grupo se pueden aplicar a subgrupos de entre las tuplas recuperadas** (no sólo al grupo formado por TODAS las tuplas recuperadas).
- La **Cláusula GROUP BY** seguida de una lista de atributos permite agrupar las tuplas en grupos que tengan los mismos valores en todos los atributos de esa lista.
- En una consulta con **GROUP BY todos** los elementos seleccionados deben ser:
  - Expresiones de la cláusula **GROUP BY**,
  - Expresiones con funciones de grupo, o
  - Constantes.
- Ejemplos:
  - "Para cada departamento, recuperar el número de empleados que tiene, su salario medio, y su mayor y menor salario":

```
SELECT  Dpto, COUNT(*), AVG(Salario), MAX(Salario),
        MIN(salario)
FROM Empleado  GROUP BY Dpto;
```

  - "Para cada pieza, recuperar el número de suministradores que tiene":

```
SELECT P.P#, NombreP, COUNT(*)
FROM Pieza P, Suministros SP
WHERE P.P#=SP.P#  GROUP BY P.P#, NombreP;
```


Agrupación con GROUP BY *y* HAVING

- **Cláusula HAVING:** Establece una condición sobre los grupos, para que sólo se recuperen los grupos que la cumplen:
  - "Para cada pieza, indicar cuántos suministradores tiene, si son varios":

```
SELECT P.P#, NombreP, COUNT(*) FROM Pieza P, Suministros
WHERE P.P#=Suministros.P#
GROUP BY P.P#, NombreP HAVING COUNT(*)>1;
```

  - "Departamentos y el número de sus empleados de aquellos que tienen más de 5 empleados y la media de su salario es mayor que 6":

```
SELECT Dpto, COUNT(*) FROM Empleado
GROUP BY Dpto HAVING COUNT(*)>5 AND AVG(Salario)>6;
```

  - "Recuperar los departamentos y el número de sus empleados que cobran más de 2.5 de aquellos que tienen más de 5 empleados que cobren más que esta cantidad":

```
SELECT Dpto, COUNT(*) FROM Empleado
WHERE Salario>2.5 GROUP BY Dpto HAVING COUNT(*)>5;
```

  - "Recuperar los departamentos y el número de sus empleados que cobran más de 2.5 de aquellos que tienen más de 5 empleados":

```
SELECT Dpto, COUNT(*) FROM Empleado
WHERE Salario>2.5
  AND Dpto IN (SELECT Dpto FROM Empleado
               GROUP BY Dpto HAVING COUNT(*)>5)
GROUP BY Dpto;
```

Resumen del Comando SELECT

- **SELECT** <Select_list>
  - Expresiones a recuperar (atributos, funciones, operaciones...).
- **FROM** <Table_list>
  - Tablas necesarias para la consulta (incluyen tablas de reunión, subconsultas...).
- **[WHERE** <Condición> **]**
  - Condición de selección de tuplas, incluyendo condiciones de reunión.
- **[GROUP BY** <Atributos_para_Agrupar> **]**
  - Atributos por los que agrupar el resultado (cada grupo tendrá los mismos valores en estos atributos). Las funciones de grupo o de agregación se aplicarán sobre cada uno de estos grupos. Se aplicarán sobre todas las tuplas si no existe esta cláusula.
- **[HAVING** <Condición_de_Grupo> **]**
  - Condición sobre los grupos (no sobre las tuplas).
- **[ORDER BY** <Atributos_para_Ordenar> **]**
  - Atributos por los que ordenar. El orden de estos atributos influye.

Insertar Tuplas: Comando INSERT

- **INSERTA** una o varias tuplas en una relación. Formato:

```
INSERT INTO <Tabla> VALUES (a1, a2, ..., an);
```

  - **Lista de valores:** En el mismo orden en el que fue creada la tabla con **CREATE TABLE**.
  - **Ej.:** `INSERT INTO pieza VALUES (4,'Teja',50,1000);`

- Puede especificarse sólo un **subconjunto de atributos**. Formato:
  `INSERT INTO <Tabla>(<Lista_Atribs>) VALUES (...);`
  - **Lista de atributos:** Nombres de los atributos en los que se desea insertar algún valor. Los valores deben estar en ese orden.
    - Atributos ausentes: Se les asigna su valor por defecto o NULL.
    - Deben estar todos los atributos que no admiten NULL y que además no tienen valor por defecto (restr. **NOT NULL** y **DEFAULT**).
- **Insertar varias tuplas:** Separadas por comas y entre paréntesis.
  - Se puede sustituir la cláusula **VALUES**(. . .) por una subconsulta: Se insertarán TODAS las tuplas recuperadas por esa subconsulta.
- **SGBD:** Debe gestionar que se cumplan las restricciones (especialmente la de integridad referencial). Si no, lo tendrá que hacer el usuario.

Comando INSERT *con Subconsultas*

- **Insertar el resultado de una subconsulta:** Formato:

```
INSERT INTO <Tabla> <Subconsulta>;
```

  - **Ej.:** Suponemos que la tabla TOTAL está creada correctamente.

```
INSERT INTO TOTAL
    SELECT S.S#, NombreS, COUNT(*)
    FROM Suministros SP, Suministrador S
    WHERE SP.S#=S.S#
    GROUP BY S.S#, NombreS
    ORDER BY 2;
```

  - **Observaciones:**
    - La tabla TOTAL es una tabla normal: Puede consultarse, modificarse y borrarse como cualquier otra tabla.
    - La tabla TOTAL no cambiará si se producen cambios en las tablas de las que se ha extraído su información. Sus datos pertenecerán a la fecha en la que se ejecutó la sentencia **INSERT**.
    - Si deseamos que la tabla esté actualizada, crear una **VISTA** (con **CREATE VIEW**).

Borrar Tuplas: Comando DELETE

- **BORRA** una o varias tuplas en una relación. Formato:

```
DELETE [FROM] <Tabla> [<Alias>] [WHERE <Cond>];
```

  - **Borra** las tuplas que cumplan la condición**.**
  - **Puede implicar el borrado de otras tuplas en otras tablas,** para que se cumpla una restricción de integridad referencial.
  - **Si se omite la cláusula WHERE** (y su condición), se borran todas las tuplas de la tabla: La tabla quedará vacía (pero sigue existiendo).
  - **Ejemplos:**
    -
```
DELETE Suministrador WHERE S# IN (2,4,8);
```
    -
```
DELETE Suministros SP
WHERE S# IN (SELECT S# FROM Suministrador
             WHERE Ciudad='Tegucigalpa');
```
    -
```
DELETE Pieza
WHERE Peso-250 > (SELECT AVG(Peso) FROM Pieza
                  WHERE Peso > (SELECT AVG(Peso)
                                FROM Pieza) );
```

Actualizar Tuplas: Comando **UPDATE**

- **ACTUALIZA** o **MODIFICA** tuplas de una relación. Formato:

```
UPDATE <Tabla> [<Alias>] SET <Actualizaciones>
                        [ WHERE <Cond> ];
```

  - <Actualizaciones> puede ser**:**
    - **1.** Una lista de asignaciones separadas por coma del tipo:
      <Atributo> = <Expresión>
      (Una expresión puede ser una subconsulta que recupere sólo un valor).
    - **2.** Una lista de atributos entre paréntesis a los que se le asigna una subconsulta: (<A1>, ..., <Am>) = (<Subconsulta>)
  - La cláusula **WHERE** selecciona las tuplas a modificar.
  - Modificar la **llave primaria** puede acarrear la modificación de llaves externas en otras tablas.

- Ejs.:

```
• UPDATE Pieza SET Nombre='Eje', Peso=9 WHERE P#=5;
• UPDATE Pieza SET Cantidad=Cantidad+100
      WHERE P# IN (SELECT P# FROM Suministros
                   WHERE S# IN (3,7));
• UPDATE Pieza SET (Peso,Cantidad)=(SELECT Peso,
      Cantidad FROM Pieza WHERE P#=6) WHERE P#=5;
```


Creación de Vistas: **CREATE VIEW**

- **VISTA:** Es una **tabla virtual** cuyas tuplas derivan de otras tablas (que pueden ser tablas base o también otras vistas).
  - Sus tuplas no se almacenan sino que se calculan a partir de las tablas de las que dependa.
  - Son útiles para usar, como si fueran tablas, consultas que se efectúan frecuentemente, y también para cuestiones de **seguridad**.
  - **Formato:**

```
CREATE [OR REPLACE] [[NO] FORCE] VIEW
<NombreV> [(<Lista_Atrbs>)] AS (<Subconsulta>)
[WITH READ ONLY];
```

    - Crea la vista <NombreV>, asociada a la subconsulta especificada.
    - La **lista de atributos** es el nombre de los atributos de la vista: Por defecto toma los nombres de los atributos de la subconsulta. Son necesarios si los atributos son calculados (funciones de grupo...).
    - **OR REPLACE:** Permite modificar una vista ya existente sin borrarla.
    - **WITH READ ONLY:** Indica que no se permitirán borrados, inserciones o actualizaciones en la vista.
    - **FORCE:** Fuerza a que la vista se cree aunque no existan los objetos que se usan en ella (tablas, otras vistas...) o no se tengan privilegios suficientes. Esas condiciones serán necesarias para usar la vista. La opción contraria es **NO FORCE**, que es la opción por defecto.

Vistas: Ejemplos y Observaciones

– **Ejemplos:**

```
• CREATE OR REPLACE VIEW SumiNombres
      AS (SELECT NombreS, NombreP
          FROM Suministros SP,Suministrador S,Pieza P
          WHERE SP.S#=S.S# AND SP.P#=P.P#);

• CREATE OR REPLACE VIEW Cantidad(NombreS,NumPiezas)
      AS (SELECT NombreS, COUNT(*)
          FROM Suministros SP, Suministrador S
          WHERE SP.P#=P.P#
          GROUP BY NombreS);
```

OJO: ¿Qué ocurre si varios Suministradores tienen el mismo nombre?

– **Observaciones:**

- Las vistas pueden **consultarse como si fueran tablas**.
- Una vista está **siempre actualizada** (*up to date*): Si se modifican las tablas de las que depende, la vista reflejará esos cambios.
- Para que la vista **NO se actualice,** no hay que crear una vista, sino una "instantánea", "foto" o vista materializada (*materialized view*) de la BD (con **CREATE SNAPSHOT**, o bien con **CREATE MATERIALIZED VIEW**).
- Para **borrar una vista** que ya no es útil: **DROP VIEW** <NombreV>;

Vistas: Implementación y Modificación

– **<u>Implementación de vistas</u>:** Dos técnicas principales:

- **Modificación a consulta:** Se modifica la vista, como si fuera una subconsulta, cada vez que se usa. Es muy **ineficiente**, especialmente si la vista es compleja y se usa frecuentemente.
- **Materialización de la vista:** Consiste en crear físicamente la vista la primera vez que se usa, suponiendo que se va a usar varias veces.
  - La vista **se borrará** si no se usa en cierto tiempo.
  - Técnicas para mantener la **vista actualizada:** Si se borran, modifican o insertan tuplas en las tablas base hay que hacerlo también en la vista.

– **<u>Modificar directamente una vista puede ser complicado</u>:**

- Es **fácil** si es una vista sobre una tabla, sin funciones de agregación e incluye la llave primaria (o una candidata). En general, se consideran vistas **NO actualizables** si están definidas sobre varias tablas (en una reunión) o si usan agrupamiento y/o funciones de agregación.
- A veces, una **modificación en una vista** puede traducirse de varias formas en las relaciones base, lo cual implica que tal modificación no debe admitirse por ser **ambigua**. A veces, se escoge la más probable.
  - **Ej.:** Si en la vista SumiNombres se modifica el nombre de una Pieza: ¿Queremos modificar el nombre de la pieza o queremos modificar la pieza que es suministrada por el correspondiente suministrador?

Control de Transacciones

- **SQL permite controlar la ejecución de una transacción:**
  - **Una transacción empieza con** la primera orden SQL después de conectarse a la base de datos o con la primera orden SQL después de terminar la transacción anterior.
  - **Una transacción termina con: `COMMIT`** o **`ROLLBACK`**.
- **Órdenes SQL de Control de Transacciónes:**
  - **`COMMIT`:** Si la transacción terminó bien y se deben **guardar los cambios** efectuados a la base de datos.
    - Además, **`COMMIT`** hace que los cambios sean visibles por otras sesiones y que se liberen los bloqueos establecidos por la transacción.
    - Hasta que no se usa **`COMMIT`** los cambios no son permanentes, sino temporales y sólo pueden ser vistos por la sesión que los hizo.
  - **`ROLLBACK`:** Si la transacción no terminó bien y se deben **deshacer los cambios** efectuados a la base de datos.
    - **`ROLLBACK`** también libera los bloqueos establecidos.
    - Esta es la opción por defecto si una sesión se desconecta de la base de datos SIN terminar la transacción.
  - **`SAVEPOINT <Nombre>`:** Para deshacer sólo parte de la transacción, se pueden poner varios **puntos de salvaguarda** con distinto nombre cada uno.
    - Tras esto, la orden **`ROLLBACK TO SAVEPOINT <Nombre>`**, deshace todo lo hecho desde el punto **<Nombre>** y libera los bloqueos establecidos tras ese punto de salvaguarda.
    - La transacción no termina con este tipo de **`ROLLBACK`**.

Otras Características: Permisos y SDL

- **Permisos: Concederlos** (**`GRANT`**) y **Revocarlos** (**`REVOKE`**).
  - Pueden usar estas órdenes los **propietarios** (*owner*) de los objetos (tablas, vistas...) o el **DBA** (administrador de la BD).
  - **Permisos sobre Objetos:** Para **`SELECT`**, **`INSERT`**, **`DELETE`**, **`UPDATE`**, **`ALTER`**, **`REFERENCES`**, **`INDEX`** y **`EXECUTE`** (para programas).
  - **Permisos del Sistema:** Existen **permisos** de usuarios para crear, modificar, borrar... **objetos**, como tablas (**`CREATE ANY TABLE`**, **`ALTER`**...), vistas, usuarios, *triggers*, sesiones, roles (**`CONNECT`**...), secuencias, *snapshots*...
  - **Formato:**
    ```
    GRANT <Lista_Permisos> ON <Objeto> TO <Usuario>
     [WITH GRANT OPTION];
    ```
    - La opción **`WITH GRANT OPTION`** permite al **<Usuario>** conceder dicho privilegio a otro usuario (si son permisos del **sistema** se usa **`WITH ADMIN OPTION`** y se elimina la cláusula **`ON`**).
  - **Formato:**
    ```
    REVOKE <Lista_Permisos> ON <Objeto> FROM <Usuario>;
    ```
    - Si son permisos del sistema se quita la cláusula **`ON`**.
- **Lenguaje de Definición del Almacenamiento, SDL** (*Storage Definition Languaje*)**:**
  - Son comandos para definir ciertos parámetros para el **almacenamiento Físico**, estructura de los ficheros, índices de acceso...
  - **No son de SQL** porque no son del nivel del esquema conceptual.
  - Cada **SGBD** puede tener distintos comandos.