Tema 4: Relacional Avanzado

- Desde el standard query language hasta el SQL3 se ha incorporado:
  - Extensiones procedimentales: Se han añadido a la base nuevas funcionalidades.
  - Comportamiento activo: Capturar eventos de datos, para que cuando ocurra cierto evento se de una determinada acción. Trigger
  - Diseño externo: Control de privilegios, vistas,... Es la propia arquitectura de la base de datos: Conceptual, interno y externo.
  - Diseño físico: Indices, clusters...
- **Vista**: Definición de tabla de lo que el usuario puede ver (no todos los usuarios ven todas las tablas, ni tampoco todas la filas), es una redundancia controlada(ya que es lo mismo que la tabla), en realidad la tabla solo estará una vez y se actualizara si se modifica en la vista.
  - Debe ser operativa, que permita borrar, modificar e insertar, que se realizan sobre las tablas fuente.
  - La BD se hace cargo de la operaciones triviales, pero cuando no lo son el programado debe definir que debe hacer. El grado será entre 1 y el numero de columnas de la tabla original, y la cardinalidad al menos un fila.
  - Una vista puede ser física, que se halla materializado, aunque ocupa espacio se accede más rápido que si se hace sobre otra tabla e ira actualizando la tabla de la que proviene.
  - **Creación**:
    - **CREATE [MATERIALIZED] VIEW <nombre de tabla>**
      - Materialized si es física, y el nombre es el de la vista.
    - **[(<nombre de columna> [, <nombre de columna>]...)**
      - Podemos cambiar el nombre de las columnas de la expresión de consulta.
    - **AS <expresión de consulta> [WITH CHECK OPTION]**
      - Tabla de donde se sacan los datos, indicando columnas y condiciones.
      - El check option es para controlar que puedan o no salir filas de la vista, si no pueden salir no permitirá realizar esa modificación.
- **Clases de relación**:
  - **Persistentes**: Solo se borran con una acción del usuario.
    - **Relaciones base:** Tabla con todos los datos. Es a nivel logico.
    - **Vistas**: Redefinición lógica de una tabla sin datos propios. Es un esquema externo.
    - **Vistas materializadas**: Redefinición lógica de una tabla con datos propios. La redundancia esta controlada y los datos actualizados. Es a nivel interno.
    - **Instantáneas**: Copia de los datos de una tabla en un determinado momento, tabla congelada, para gestionar los datos de ese instante y cuando se acaba la eliminamos.
  - **Temporales**: Desaparecen al ocurrir determinado evento, como una transacción o cierre de sesión. Se borra cuando se cierra la sesión.
- **Gestión de privilegios:**
  - **Elementos**:
    - **Usuarios**:
      - **CREATE USER <username> IDENTIFIED BY <password>**
      - **[DEFAULT TABLESPACE <tablespace>]**
      - **[QUOTA <size> ON <tablespace>]**
      - **[PROFILE <profile name>]**
      - **[PASSWORD EXPIRE]**
      - **[ACCOUNT {LOCK | UNLOCK}];**

- **Perfiles:**
  - **CRÉATE PROFILE <profile name> LIMIT <resources>;**
- **Roles:**
  - **CRÉATE ROLE <rolename>**
  - **{NOT IDENTIFIED | IDENTIFIED BY <password>};**

- **Privilegios :**
  - **Conceder:**
    - **GRANT {<rolename> | <sys_privileges | ALL PRIVILEGES}**
    - **TO <users/roles> [WITH ADMIN OPTION];**
    - **GRANT { <object_privileges | ALL PRIVILEGES}**
    - **[(column [, ...])] ON [schema.] <object>TO <users/roles>**
    - **[WITH HIERARCHY OPTION][WITH GRANT OPTION];**
  - **Revocar:**
    - **REVOKE <privileges> [ON <object>] FROM <users/roles>;**
- **Estructura de un bloque:** Declaraciones, Cuerpo y Excepciones.
  - **[DECLARE**
  - **varname type; [...]]**
  - **BEGIN**
  - **<codigo procedimental>**
  - **[EXCEPTION**
  - **WHEN ... THEN ...; [...]]**
  - **END;**
- **Procedimientos**: Omite el DECLARE
  - **CRÉATE OR REPLACE PROCEDURE name(params) IS**
  - **Bloque de código;**
- **Funciones**: Omite el DECLARE
  - **CRÉATE OR REPLACE FUNCTION name(params) RETURN CHAR IS**
  - **Bloque de código;**
- **Invocaciones a procedimientos**: Se deben hacer dentro de un bloque o con la instrucción EXEC.
  - **BEGIN my_proc(“”); END;**
  - **EXEC my_proc(“”);**
- **Paquete:**
- **Disparador:** Son procedimientos que se ejecutan ante un determinado evento, definidas por el usuario. Hay dos maneras esta es la sencilla, la compuesta en la 17 del tema 4.
  - **CRÉATE OR REPLACE TRIGGER [<nombre>]**
  - **<tiempo activación acción>**
    - Cuando se lleva a cabo: AFTER, BEFORE o INSTEAD OF. El ultimo en vistas.
  - **<evento disparador>**
    - Porque se dispara: INSERT, DELETE o UPDATE [OF <columnas>]
  - **ON <nombre tabla>**
    - Tabla en la que se aplica.
  - **<nivel de activación>**
    - Si se hace para cada línea o para la operación. FOR EACH ROW o STATEMENT.
      - Se usa :old y :new para referirnos a la fila antes o después de la operación.

    - **<bloque definiendo la acción disparada>**
- **Error tabla mutante:** Cuando una tabla no esta estable, durante una operación, y se opera sobre ella. Para observarlo hay que eliminar o insertar varias filas. Ocurre con FOR EACH ROW, pero no con FOR EACH STATEMENT.
    - **Soluciones**:
        - Almacenar en una tabla temporal las operaciones que se van a tener que realizar, para que cuando termine y este estable se realicen, con un disparador de fila.
        - Con disparadores complejos.
- **Desactivar constraints y triggers**: Hay mas maneras.
    - **ALTER TABLE <table-name> {DISABLE | ENABLE} CONSTRAINT <c_name>;**
    - **ALTER TRIGGER <table-name> {DISABLE | ENABLE} ALL TRIGGER;**
- **Disparadores DDL:** Sobre las tablas de metadatos.
- **Disparadores DB**: Para crear tablas de auditorias, para almacenar eventos.