

PLSQL para completar un esquema relacional

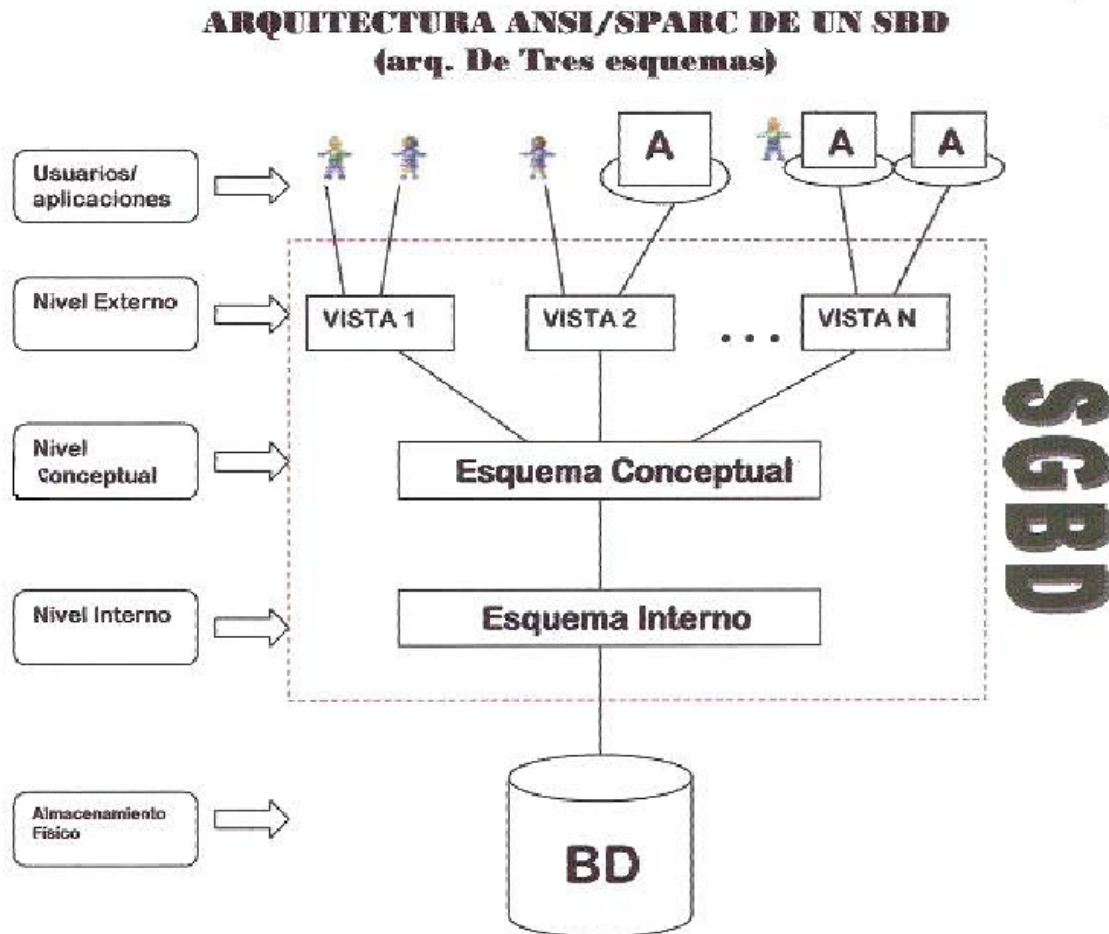
Álvaro González Sotillo

10 de octubre de 2022

Índice

1. Introducción	2
2. Ejemplo: Academia	3
3. Referencias	7

1. Introducción



1.1. Arquitectura ANSI-SPARC

- En la arquitectura ANSI/SPARC se especifica un nivel interno, conceptual y externo
 - El esquema interno corresponde con los ficheros en el disco
 - El esquema de tablas de un SGBDR sería el esquema conceptual
 - El nivel externo puede implementarse mediante
 - Control de acceso (*grant*) a las tablas
 - Vistas (*view*) para la lectura de datos
 - Procedimientos para la inserción, borrado y modificación de datos

1.2. Modelo relacional

- El modelo relacional deja expresar:

-
- Claves primarias
 - Claves extranjeras
 - Restricciones sobre los tipos de datos
 - Restricciones sobre los valores posibles (*check*)
- Pero hay otro tipo de restricciones que no puede expresar:
 - Se pueden implementar con PLSQL en el nivel externo del ANSI-SPARC

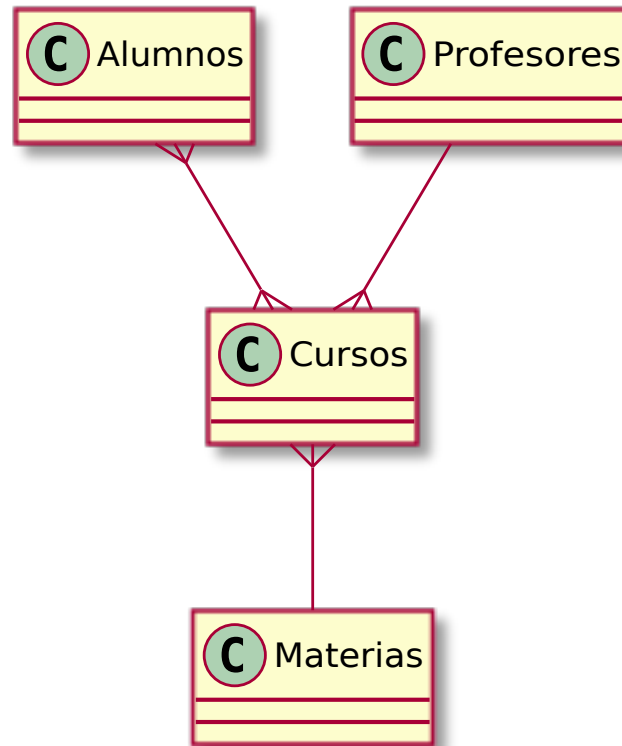
1.3. Limitaciones del SGBD

- No todos los SGBD tienen las mismas funcionalidades
- Por ejemplo, Oracle no tiene claves primarias autoincrementales
 - Se pueden implementar con procedimientos o *triggers*

2. Ejemplo: Academia

- Soportado en el modelo relacional:
 - Cada materia se imparte en uno o más cursos.
 - Cada profesor imparte varios cursos.
 - Los alumnos se matriculan en cursos.
- No soportado en el modelo relacional:
 - Debe haber al menos un profesor en la academia.
 - No puede haber más de 10 alumnos en un curso.

2.1. Esquema relacional



2.2. Tablas

```
create table PROFESORES(  
  profesorid integer primary key, profesornombre varchar(50)  
);  
create table MATERIAS(  
  materianombre varchar(50) primary key  
);  
create table CURSOS(  
  cursoid integer primary key,  
  profesorid integer references PROFESORES, materianombre references MATERIAS  
);  
create table ALUMNOS(  
  alumnoid integer primary key, alumnonombre varchar(50)  
);  
create table CURSOS_ALUMNOS(  
  alumnoid integer references ALUMNOS, cursoid integer references CURSOS,  
  primary key (alumnoid, cursoid)  
);
```

2.3. Operaciones

- Consultas
- Inserciones
- Borrados
- Modificaciones

2.3.1. Consultas

- select sobre las tablas (o vistas que se pudieran hacer)

2.3.2. Inserciones: profesores

- Procedimientos del tipo CREA_XXX
- Con un parámetro de salida que es el identificador de la fila creada

```
create sequence PROFESOR_SECUENCIA;
create or replace procedure CREA_PROFESOR(pprofesor nombre varchar, pprofesorid out integer)
as
begin
    pprofesorid := PROFESOR_SECUENCIA.nextval;
    insert into PROFESORES(profesorid, profesor nombre) values(pprofesorid, pprofesor nombre);
end;
/
```

- Se podría hacer con un *trigger* que calcule automáticamente la clave en un insert
 - El problema sería conseguir la clave de la fila recién insertada

2.3.3. Ejemplo de inserción de profesor

```
declare
    nuevoprofesor integer;
begin
    CREA_PROFESOR('María',nuevoprofesor);
    dbms_output.put_line('Creado profesor:' || nuevoprofesor );
end;
/
```

2.3.4. Inserciones: alumnos

```
create sequence ALUMNO_SECUENCIA;
create or replace procedure CREA_ALUMNO(palumno nombre varchar, palumid out integer)
as
begin
    palumid := ALUMNO_SECUENCIA.nextval;
    insert into ALUMNOS(alumid, alumno nombre) values(palumid, palumno nombre);
end;
/
```

2.3.5. Inserciones: materias y cursos

```
create or replace procedure CREA_MATERIA(pmaterianombre varchar)
as
begin
    insert into MATERIAS values(pmaterianombre);
end;
/
create sequence CURSO_SECUENCIA;
create or replace procedure CREA_CURSO(pmaterianombre varchar, pprofesorid integer, pcursoid out integer)
as
begin
    pcursoid := CURSO_SECUENCIA.nextval;
    insert into CURSOS(cursoid,profesorid,materianombre) values(pcursoid,pprofesorid,pmaterianombre);
end;
/
```

2.3.6. Ejemplo de inserción de materias y cursos

```
declare
    cursoid integer;
    profesorid integer;
begin
    crea_profesor('Juan', profesorid);
    crea_materia('Bases de datos');
    crea_curso('Bases de datos', profesorid, cursoid);
    dbms_output.put_line('Bases de datos: id:' || cursoid );
    dbms_output.put_line('Bases de datos: id de profesor:' || profesorid );
end;
/
```

2.3.7. Inserciones: matriculaciones

- En el procedimiento de inserción, pueden controlarse errores que el modelo relacional no puede expresar
- Ejemplo: No puede haber más de 10 alumnos por curso

```
create or replace procedure MATRICULA_ALUMNO_EN_CURSO(palumnoid integer, pcursoid integer)
as
    yamatriculados number;
begin
    select count(*) into yamatriculados from CURSOS_ALUMNOS where cursoid = pcursoid;
    if yamatriculados = 10 then
        raise_application_error(-20001, 'El curso ' || pcursoid || ' ya tiene 10 alumnos matriculados ');
    end if;
    insert into CURSOS_ALUMNOS(alumnoid, cursoid) values(palumnoid, pcursoid);
end;
/
```

2.3.8. Prueba de inserción de 11 alumnos en un curso

```
declare
    cursoid integer;
    profesorid integer;
    alumnoid integer;
begin
    CREA_MATERIA('Materia popular');
    CREA_PROFESOR('Un profesor', profesorid);
    CREA_CURSO('Materia popular',cursoid);
    for i in 1..11 loop
        CREA_ALUMNO('Alumno' || i, alumnoid );
        dbms_output.put_line( i || ': Voy a matricular al alumno ' || alumnoid || ' en el curso ' || cursoid );
    end loop;
end;
```

```
MATRICULA_ALUMNO_EN_CURSO(alumnoid, cursoid);  
end loop;  
end;  
/
```

2.3.9. Borrados : alumnos

```
create or replace procedure BORRA_ALUMNO(palumnoid integer)  
as  
begin  
  delete from ALUMNOS where alumnoid = palumnoid;  
end;  
/
```

2.3.10. Borrados : profesores

- En el procedimiento de borrado, pueden controlarse errores que el modelo relacional no puede expresar
- Ejemplo: Al menos un profesor en la academia
 - Al principio de la base de datos no hay ninguno, pero tras la primera inserción se garantiza
 - También se podría garantizar con un *trigger*

```
create or replace procedure BORRA_PROFESOR(pprofesorid integer)  
as  
  numeroprofesores number;  
begin  
  select count(*) into numeroprofesores from PROFESORES;  
  if numeroprofesores = 1 then  
    raise_application_error( -20002, 'No se puede borrar el último profesor' );  
  end if;  
  delete from PROFESORES where profesorid = pprofesorid;  
end;  
/
```

2.3.11. Modificaciones

- No es fácil controlar las modificaciones con procedimientos
- Suele ser mejor dejar cambiar las tablas con update y controlar los posibles errores con *triggers*

3. Referencias

- Formatos:
 - [Transparencias](#)
 - [PDF](#)
 - [EPUB](#)
- Creado con:
 - [Emacs](#)
 - [org-re-reveal](#)
 - [Latex](#)
- Alojado en [Github](#)