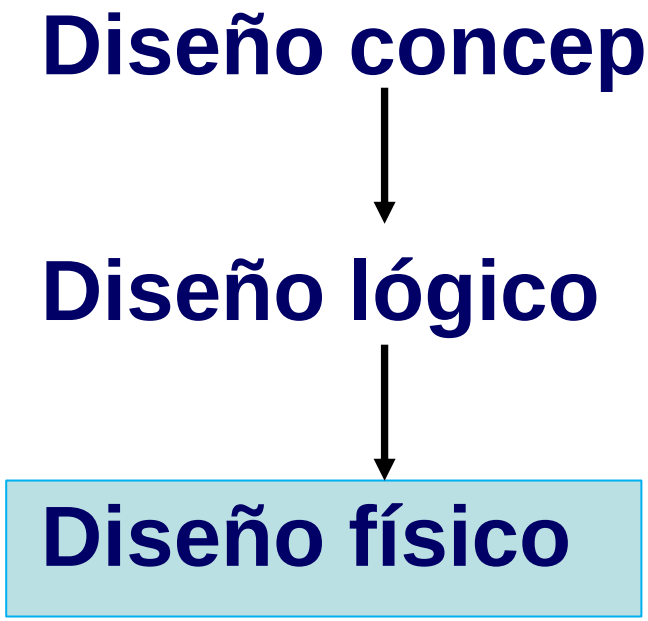
Diseño físico – Ejercicio del hotel

Diseño de Bases de Datos
Grado en Ingeniería Informática

Departamento de
Lenguajes y Sistemas Informáticos

Universitat d'Alacant
Universidad de Alicante

Diseño de una BD



Objetivo del diseño físico

En el diseño físico debe conseguir definir las estructuras de almacenamiento *de entre las que permita el SGBD elegido* para que las aplicaciones que accedan a la BD obtengan un buen rendimiento.

Fases del diseño físico

Continuamos con el ejercicio del HOTEL

1. **Traducir el esquema lógico para el SGBD específico.**
2. **Diseñar la representación física.**
3. **Diseñar los mecanismos de seguridad.**
4. **Monitorizar y afinar el sistema.**

1. Traducir e. lógico para el SGBD específico.

- **1.1. Diseñar las relaciones base para el SGBD específico.**
- **1.2. Diseñar las reglas de negocio para el SGBD específico.**

1. Traducir e. lógico para el SGBD específico.

- **1.1. Diseñar las relaciones base para el SGBD específico.**

SGBD: Oracle.

Consideraciones:

- **Admite definición de C.P., C.Ajena, NOT NULL, UNIQUE, DELETE CASCADE …**
- **No admite autoincremento (a partir de la versión 12 sí)**

1. Traducir e. lógico para el SGBD específico.

Revisamos el diseño lógico obtenido

TEMPORADA(nombre)
- C. Primaria: nombre

CATEGORIA(nombre, descripción, supMin, supMax)
- C. Primaria: nombre

PVPTEMPORADA(categoría, temporada, pSA, pAD, pMP, pPC)
- C. Primaria: (categoría, temporada)
- C. Ajena: categoría → CATEGORIA
- C. Ajena: temporada → TEMPORADA

HABITACIÓN (número, categoría)
- C. Primaria: número
- C. Ajena: categoría → CATEGORIA
- V.N.N.: categoría

CALENDARIO (fecha, temporada)
- C.P.: fecha
- C. Ajena: temporada → TEMPORADA

1. Traducir e. lógico para el SGBD específico.

Revisamos el diseño lógico obtenido

HORA (hora)

- C.P.: hora consideraremos 2 dígitos

CLIENTE(nif, nombre, dirección, población, teléfono, país)

- C.P.: nif
- V.N.N.:nombre
- V.N.N.:teléfono

RESERVA(código, cliente)

- C.P.: código
- C.Ajena: cliente → CLIENTE
- V.N.N.: cliente
- Una reserva debe estar vinculada al menos a la reserva de una habitación para una fecha

CALENDRESERVAS(habitacion, fecha, camasup, alimentacion, reserva)

- C. P.: (habitacion, fecha)
- C.Ajena: habitacion → HABITACION
- C.Ajena: fecha → CALENDARIO
- C.Ajena: reserva → RESERVA
- V.N.N.:reserva

1. Traducir e. lógico para el SGBD específico.

Revisamos el diseño lógico obtenido

CONSUMIR(habitación, fecha, servicio, cantidad)
- **C.P.:(habitación, fecha, servicio)**
- **C.Ajena: servicio → SERVICIO**
- **C.Ajena:habitación, fecha → CALENDRESERVAS**

EMPLEADO(nif, nombre, dirección, población, teléfono, estudios)
- **C.P.: nif**

EMPANIMACION(nif)
- **C.P.: nif**
- **C.Ajena: nif → EMPLEADO**

EMPSERVICIOS(nif)
- **C.P.: nif**
- **C.Ajena: nif → EMPLEADO**

EMPLIMPIEZA(nif)
- **C.P.: nif**
- **C.Ajena: nif → EMPLEADO**

EMPRESTAURANTE(nif)
- **C.P.: nif**
- **C.Ajena: nif → EMPLEADO**

EMPRECEPCION(nif)
- **C.P.: nif**
- **C.Ajena: nif → EMPLEADO**

Habrá que controlar que la generalización de EMPLEADO es TOTAL y DISJUNTA.

1. Traducir e. lógico para el SGBD específico.

Revisamos el diseño lógico obtenido

- Nos damos cuenta de que las tablas TEMPORADA y CATEGORÍA tienen como CP la columna “nombre” que es una cadena de caracteres (podemos tener problemas posteriores de rendimiento cuando el SGBD internamente haga las comprobaciones).

TEMPORADA(nombre)

C. Primaria: nombre

CATEGORIA(nombre, descripción, supMin, supMax)

C. Primaria: nombre

1. Traducir e. lógico para el SGBD específico.

Revisamos el diseño lógico obtenido

- Nos damos cuenta de que las tablas TEMPORADA y CATEGORÍA tienen como CP la columna “nombre” que es una cadena de caracteres.

TEMPORADA(nombre)

C. Primaria: nombre

¿LAS MANTENEMOS ASÍ O HACEMOS ALGÚN CAMBIO EN EL DISEÑO FÍSICO?

CATEGORIA(nombre, descripción, supMin, supMax)

C. Primaria: nombre

1. Traducir e. lógico para el SGBD específico.

Revisamos el diseño lógico obtenido

- Nos damos cuenta de que las tablas TEMPORADA y CATEGORÍA tienen como CP la columna “nombre” que es una cadena de caracteres.

¿LAS MANTENEMOS ASÍ O HACEMOS ALGÚN CAMBIO EN EL DISEÑO FÍSICO?

Aspectos a tener en cuenta y a analizar:

1) Tamaños y posibles valores para esas columnas

- No tendrá el mismo coste para el SGBD almacenar un varchar2(3) que varchar2(100).
- Tampoco tendrá el mismo coste si los posibles valores para cada una de esas columnas son limitados (por ejemplo: “alta”, “media”, y “baja”) o cualquier valor.

1. Traducir e. lógico para el SGBD específico.

Revisamos el diseño lógico obtenido

- Nos damos cuenta de que las tablas TEMPORADA y CATEGORÍA tienen como CP la columna “nombre” que es una cadena de caracteres.

¿LAS MANTENEMOS ASÍ O HACEMOS ALGÚN CAMBIO EN EL DISEÑO FÍSICO?

Aspectos a tener en cuenta y a analizar:

2) Podemos decidir dejar estas columnas como columnas de la tabla sin que sean CP, y por temas de optimización del rendimiento, crear una nueva columna de tipo numérico que sea la CP

TEMPORADA(idtemporada, nombre)

C. Primaria: idtemporada

CATEGORIA(idcategoria, nombre, descripción, supMin, supMax)

C. Primaria: idcategoria

1. Traducir e. lógico para el SGBD específico.

Revisamos el diseño lógico obtenido

- Nos damos cuenta de que las tablas TEMPORADA y CATEGORÍA tienen como CP la columna “nombre” que es una cadena de caracteres.

¿LAS MANTENEMOS ASÍ O HACEMOS ALGÚN CAMBIO EN EL DISEÑO FÍSICO?

Aspectos a tener en

2) Podemos decidir dejar
sin que sean CP, y por te
una nueva columna de ti

¿Y qué pasa entonces con la restricción original de CP para nombre de temporada y nombre de categoría?

TEMPORADA(idtemporada, nombre)

C. Primaria: idtemporada

CATEGORIA(idcategoria, nombre, descripción, supMin, supMax)

C. Primaria: idcategoria

1. Traducir e. lógico para el SGBD específico.

Revisamos el diseño lógico obtenido

- Nos damos cuenta de que las tablas TEMPORADA y CATEGORÍA tienen como CP la columna “nombre” que es una cadena de caracteres.

¿Y qué pasa entonces con la restricción original de CP para nombre de temporada y nombre de categoría?

TEMPORADA(idtemporada, nombre)

C. Primaria: idtemporada

CATEGORIA(idcategoria, nombre, descripción, supMin, supMax)

C. Primaria: idcategoria

- Podríamos definirlas como VNN y crear clave UNICA para que no se puedan repetir los valores

1. Traducir e. lógico para el SGBD específico.

Revisamos el diseño lógico obtenido

¿Y qué pasa entonces con la restricción original de CP para nombre de temporada y nombre de categoría?

TEMPORADA(idtemporada, nombre)

C. Primaria: idtemporada

CATEGORIA(idcategoria, nombre, descripción, supMin, supMax)

C. Primaria: idcategoria

Podríamos definirlas como VNN y crear clave UNICA para que no se puedan repetir los valores

```
CREATE TABLE TEMPORADA(
idtemporada number(1) constraint pk_temporada primary key,
nombre varchar(5) constraint un_nomtemporada unique not null)

CREATE TABLE CATEGORIA(
idcategoria number(1) constraint pk_categoria primary key,
Nombre varchar(2) constraint un_noncategoria unique not null,
descripción varchar(30), supMin number(4,2), supMax number(4,2))
```

1. Traducir e. lógico para el SGBD específico.

Revisamos el diseño lógico obtenido

¿Y qué pasa entonces con la restricción original de CP para nombre de temporada y nombre de categoría?

TEMPORADA(

C. Primaria

CATEGORIA(i

C. Primaria:

¿Algún problema o pega con esta propuesta?

Podríamos definirlas como VNN y crear clave UNICA para que no se puedan repetir los valores

CREATE TABLE TEMPORADA(

idtemporada number(1) constraint pk_temporada primary key,

nombre varchar(5) constraint un_nomtemporada unique not null)

CREATE TABLE CATEGORIA(

idcategoria number(1) constraint pk_categoria primary key,

Nombre varchar(2) constraint un_noncategoria unique not null, descripción varchar(30), supMin number(4,2), supMax number(4,2))

1. Traducir e. lógico para el SGBD específico.

Revisamos el diseño lógico obtenido

- Si decidimos que ponemos una CP numérica para facilitar la comprobación, pero luego añadimos la restricción UNIQUE, estamos empeorando a nivel de rendimiento, y podría ser también difícil y costoso para el SGBD igual que si hubiéramos definido nombre como CP, porque de nuevo tendrá que hacer las comprobaciones pertinentes para verificar que sea único y no nulo (cómo lo haría si fuera CP).

CREATE TABLE TEMPORADA(
idtemporada number(1) constraint pk_temporada primary key,
nombre varchar(5) constraint un_nomtemporada unique not null)

CREATE TABLE CATEGORIA(
idcategoria number(1) constraint pk_categoria primary key,
Nombre varchar(2) constraint un_noncategoria unique not null,
descripción varchar(30), supMin number(4,2), supMax number(4,2))

1. Traducir e. lógico para el SGBD específico.

Revisamos el diseño lógico obtenido

- Si decid
  facilitar
  restricci
  rendimie
  el SGBD
  CP, por
  comprol
  y no nul

¿Cuál sería entonces una alternativa más adecuada que no empeorara el rendimiento?

CREATE TABLE TEMPORADA(
idtemporada number(1) constraint pk_temporada primary key,
nombre varchar(5) constraint un_nomtemporada unique not null)

CREATE TABLE CATEGORIA(
idcategoria number(1) constraint pk_categoria primary key,
Nombre varchar(2) constraint un_noncategoria unique not null,
descripción varchar(30), supMin number(4,2), supMax number(4,2))

1. Traducir e. lógico para el SGBD específico.

Revisamos el diseño lógico obtenido

- Si dejamos como CP la nueva columna numérica, entonces las columnas “nombre” las definiríamos sólo con VNN y luego, mediante procesos externos lanzados periódicamente (por ejemplo, cada noche) se comprobaría si hay duplicidades y en caso de haberlas, se detectarían y arreglarían de otra manera.

CREATE TABLE TEMPORADA(
idtemporada number(1) constraint pk_temporada primary key,
nombre varchar(5) constraint un_nomtemporada not null)

CREATE TABLE CATEGORIA(
idcategoria number(1) constraint pk_categoria primary key,
Nombre varchar(2) constraint un_noncategoria not null, descripción varchar(30), supMin number(4,2), supMax number(4,2))

1. Traducir e. lógico para el SGBD específico.

Revisamos el diseño lógico obtenido

- Otra opción es dejar la columna “nombre” como CP, tal y como estaba originalmente → SOLUCIÓN ADOPTADA FINALMENTE (ver script de BD Hotel de prácticas)
- Además conforme a los valores marcados en el esquema conceptual se deberán utilizar restricciones CHECK para controlar los valores que toman los nombres tanto de temporada como de categoría.

CREATE TABLE TEMPORADA(
nombre varchar(5) constraint pk_temporada primary key constraint ch__nomtemporada check (nombre in (‘BAJA’,’MEDIA’,’ALTA’))

CREATE TABLE CATEGORIA(
nombre varchar(2) constraint pk_categoria primary key constraint ch__nomcategoria check (nombre in (‘I’,’D’,’DT’,’S’), descripción varchar(30), supMin number(4,2), supMax number(4,2))

Revisamos el diseño lógico obtenido

- Se decide que en la tabla de RESERVA el código de reserva debe ser un número autoincrementado. Además al borrar un cliente se deben de borrar todas sus reservas.

RESERVA(código, cliente)

C.P.: código

C.Ajena: cliente → CLIENTE

V.N.N.: cliente

¿Cómo hacerlo en Oracle?

1. Traducir e. lógico para el SGBD específico.

Revisamos el diseño lógico obtenido

```
CREATE TABLE RESERVA(
código number(10)
constraint pk_reserva primary key,
cliente char(9) not null
constraint fk_reserva_cliente references
cliente ON DELETE CASCADE);
```

Revisamos el diseño lógico obtenido

A partir de la versión 12:

```
CREATE TABLE RESERVA(
código number(10) GENERATED BY
DEFAULT ON NULL AS IDENTITY constraint
pk_reserva primary key,
cliente char(9) not null constraint
fk_reserva_cliente references cliente ON
DELETE CASCADE);
```

1. Traducir e. lógico para el SGBD específico.

Revisamos el diseño lógico obtenido

Nosotros trabajamos con la versión 10 de Oracle

```
CREATE SEQUENCE seq_reserva;

CREATE OR REPLACE TRIGGER reserva_cod
BEFORE INSERT ON reserva
FOR EACH ROW
 BEGIN
SELECT seq_reserva.NEXTVAL INTO :new.código
FROM dual;
END;
```

Oracle no garantiza que sea correlativo, sin saltos, sí garantiza que siempre será único.

1.2. Diseñar las reglas de negocio para el SGBD específico

- **Algunas se incorporan con el uso de CHECK en CREATE TABLE,**

- **otras a través de disparadores**

En prácticas: las actividades del hotel si son de NIÑOS no son de ADULTO ni para TODOS los públicos, si son de ADULTO no son …

1. Traducir e. lógico para el SGBD específico.

1.1. Diseñar las reglas de negocio para el SGBD específico.

- **CHECK: algunas ya consideradas en las restricciones de la definición de tablas**
  - ***constraint ch__nomtemporada check (nombre in ('BAJA','MEDIA','ALTA'))***
  - ***ALTER TABLE habitación ADD CONSTRAINT check_tipo_hab CHECK (tipo IN ('I', 'D','DT','S'))***

- **Definición de disparadores** *(hechos algunos en prácticas)*
  - para controlar generalizaciones disjuntas,
  - controlar que una actividad no sea sustituta de ella misma,
  - controlar que los precios sean más altos según categoría alimenticia

Fases del diseño físico

1. Traducir el esquema lógico para el SGBD específico.
2. **Diseñar la representación física.**
3. Diseñar los mecanismos de seguridad.
4. Monitorizar y afinar el sistema.

Fases del diseño físico

2. Diseñar la representación física

2.1 Analizar las transacciones

2.2 Escoger índices primarios y secundarios

2.3 Considerar la introducción de algunas modificaciones en el diseño lógico obtenido en 3FN

2.1 Analizar las transacciones

Insertar/Borrar/Modificar/Consultar empleados

Insertar/Borrar/Modificar/Consultar clientes

Insertar/Borrar/Modificar/Consultar reservas

Insertar/Borrar/Modificar/Consultar actividades

Planificar calendario de actividades

Consultar disponibilidad/ precios

Generar facturas

***Regla del 20-80**: el 20% de las transacciones suponen el 80% de la carga*

2.2 Escoger índices primarios y secundarios

Automáticamente Oracle define un índice por cada clave primaria.

Teniendo en cuenta las transacciones más destacadas convendría definir, por ejemplo,

- **un índice para la columna del código de reserva del calendario de reservas (calendreservas) para facilitar la facturación,**
- **un índice por CLIENTE en RESERVA (para saber las reservas de un cliente),**
- **un índice por fecha en CALENDRESERVAS para saber la disponibilidad dada una fecha.**
- **….**

2. Diseñar la representación física.

2.3 Considerar la introducción de algunas modificaciones en el diseño lógico obtenido en 3FN

Situaciones candidatas para introducir redundancias controladas:

- ***Introducción de atributos calculados y mantenerlos mediante disparadores.***
- **Considerar el tratamiento de generalizaciones como una única tabla donde se agrupa objeto general y subtipos.**

Considerar el uso de vistas materializadas.

2.3 Considerar la introducción de algunas modificaciones en el diseño lógico obtenido en 3FN y la introducción de vistas para facilitar consultas.

- Estudiar la generalización de empleado para establecer si es necesario considerar una única tabla que reúna todos los subtipos.

Es mejor mantener la estructura actual puesto que cada tipo de empleado juega un papel distinto, hay más claves ajenas a los tipos particulares que al general.

- Dado que se va a trabajar mucho con la disponibilidad de habitaciones, y suponiendo que el cálculo fuese muy costoso y que no fuese crítico tener la disponibilidad real al instante, sería bueno crear una vista materializada que se refrescase todos los días, en la que para cada fecha de los próximos 3 meses se muestre el número de habitaciones de cada categoría (I,D,DT,S) que quedan disponibles.
- Puesto que la tabla calendario de reservas puede ser muy grande, si llegase a tener millones de filas y problemas de rendimiento convendría particionarla por fechas.

3.- Diseñar los mecanismos de seguridad

Próximo tema

4.- Monitorizar y afinar el sistema

Se realizan pruebas, auditorias (Oracle tiene su propia utilidad aunque también se pueden usar disparadores para auditar) y en base a los resultados se realizan los cambios oportunos.

Diseño físico – Ejercicio del hotel

Diseño de Bases de Datos
Grado en Ingeniería Informática

Departamento de
Lenguajes y Sistemas Informáticos

Universitat d'Alacant
Universidad de Alicante