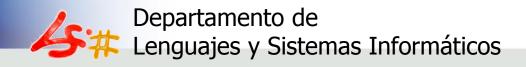
## Diseño físico – Ejercicio del hotel

Diseño de Bases de Datos Grado en Ingeniería Informática



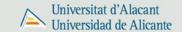


#### Diseño de una BD

Diseño conceptual
Diseño lógico
Diseño físico

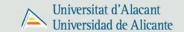
## Objetivo del diseño físico

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.



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.



1.1. Diseñar las relaciones base para el SGBD específico.

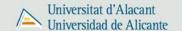
1.2. Diseñar las reglas de negocio para el SGBD específico.

- 1. Traducir e. lógico para el SGBD específico.
  - 1.1. Diseñar las relaciones base para el SGBD específico.

SGBD: Oracle.

**Consideraciones:** 

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



#### 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

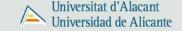
C. Ajena: categoría → CATEGORIA

V.N.N.: categoría

#### **CALENDARIO** (fecha, temporada)

C.P.: fecha

C. Ajena: temporada → TEMPORADA



#### 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

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

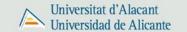
C.Ajena: nif → EMPLEADO

#### **EMPRECEPCION**(nif)

C.P.: nif

C.Ajena: nif → EMPLEADO

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



#### Revisamos el diseño lógico obtenido

- Se decide cambiar las claves primarias de las tablas TEMPORADA y CATEGORIA añadiendo una clave primaria numérica que puede evitar posibles problemas posteriores. Se debe entonces crear clave UNICA para que no se puedan repetir los valores
- 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(**

idtemporada number(1) constraint pk\_temporada primary key,

nombre varchar(5) constraint un\_nomtemporada unique not null constraint <a href="mailto:ch\_nomtemporada">ch\_nomtemporada</a> check (nombre in ('BAJA','MEDIA','ALTA'))

#### **CREATE TABLE CATEGORIA**(

idcategoria number(1) constraint pk\_categoria primary key,

Nombre varchar(2) constraint un\_noncategoria unique not null constraint ch\_nomcategoria check (nombre in ('l','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.

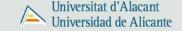
## RESERVA(código, cliente)

C.P.: código

C.Ajena: cliente → CLIENTE

V.N.N.: cliente

¿Cómo hacerlo en Oracle?



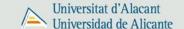
## Revisamos el diseño lógico obtenido

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)**;



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** 

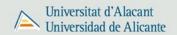
SELECT seq\_reserva.NEXTVAL INTO :new.código

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

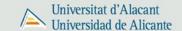
En prácticas: las actividades del hotel si son de NIÑOS no son de ADULTO ni para TODOS los públicos, si son de ADULTO no son ...



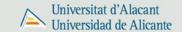
## 1.1. Diseñar las reglas de negocio para el SGBD específico.

- 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'))

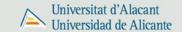
- Definición de disparadores (hechos algunos en prácticas)
  - para controlar generalizaciones disjuntas,
  - controlar que una actividad no sea sustituta de ella misma,
  - controlar que los precios sean más altos según categoría alimenticia



- 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.



- 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.
- 0 ....

## 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.

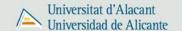


## 2.3 Considerar la introducción de algunas modificaciones en el diseño lógico obtenido en 3FN y la introducción de vistas para facilitar consultas.

 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

