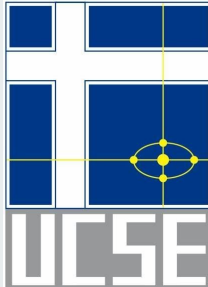


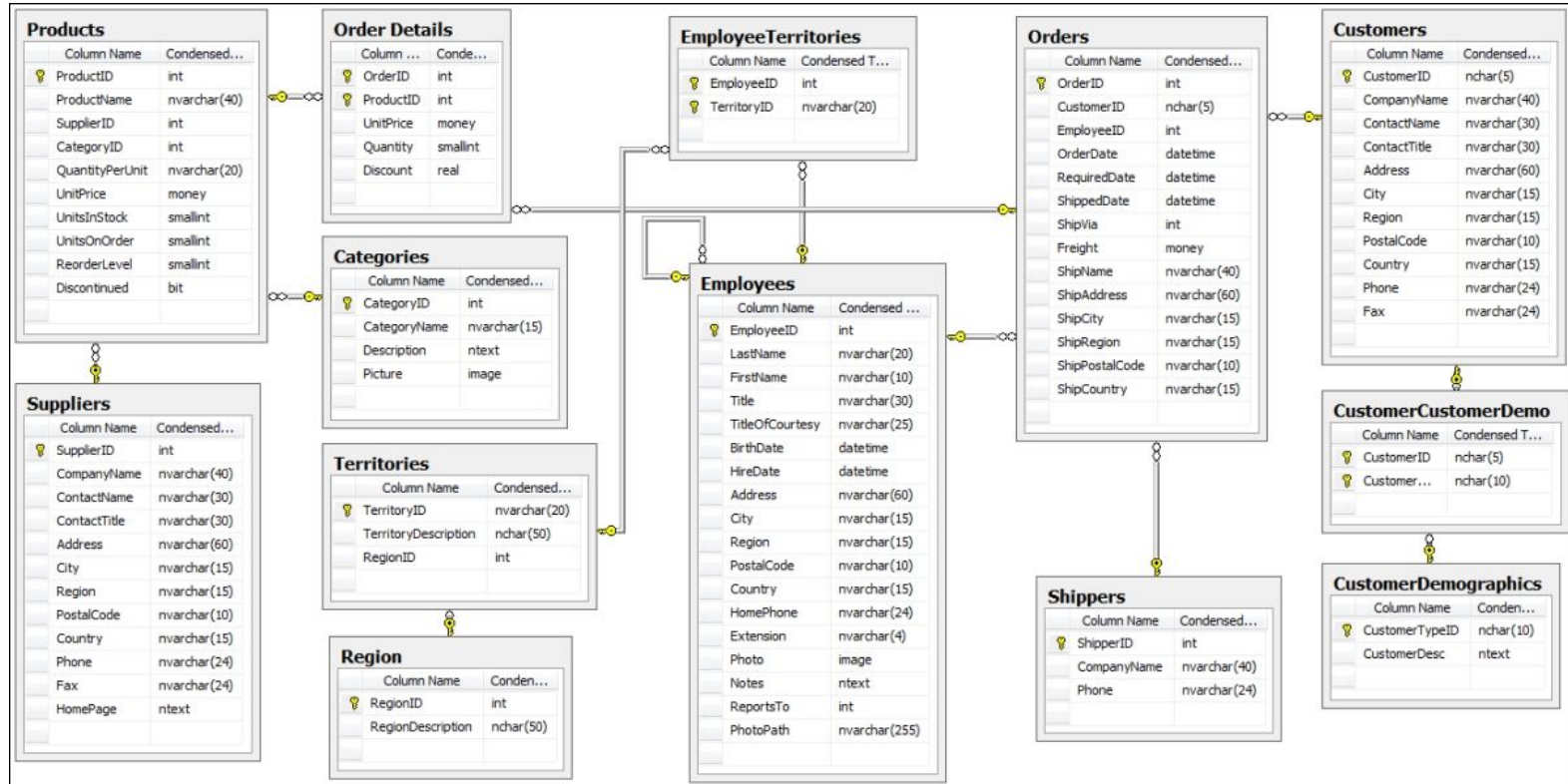


Modelo de datos

UCSE - SEIA



Modelo de datos



Modelo de datos: elementos a incluir



- Tablas
- Campos
- Tipos de datos
- Relaciones (FK)

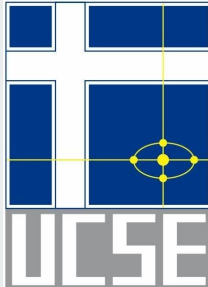
Modelo de datos: elementos a incluir





(des)Normalización

UCSE - SEIA



Normalización en base de datos



- Evitar tener valores repetidos
- Ayuda a mantener consistencia e integridad
- Facilita aplicar controles y restricciones
- ...

No obstante: es la mejor alternativa siempre ?

Ejemplo no tan irreal ...



Posible esquema de nuestra tabla de usuarios al comenzar nuestro desarrollo ...

```
create table users (  
    user_id            integer primary key,  
    first_names        varchar(50),  
    last_name          varchar(50) not null,  
    email              varchar(100) not null unique,  
    -- password HASH !!!  
    password           varchar(30) not null,  
    registration_date   timestamp(0)  
);
```

Ejemplo no tan irreal ...



Luego de un mes, notamos que sería bueno contar con algunos datos extras ...

```
create table users (  
    user_id            integer primary key,  
    first_names        varchar(50),  
    last_name          varchar(50) not null,  
    email              varchar(100) not null unique,  
    -- password HASH !!!  
    password           varchar(30) not null,  
    registration_date   timestamp(0)  
    -- url personal  
    url                varchar(100) null,  
    avatar              varchar(100) null  
);
```



Ejemplo no tan irreal ...



Y luego de un tiempo, este proceso se repite numerosas veces ...

```
create table users (  
    user_id            integer primary key,  
    first_names        varchar(50),  
    last_name          varchar(50) not null,  
    email              varchar(100) not null unique,  
    -- password HASH !!!  
    password           varchar(30) not null,  
    registration_date  timestamp(0)  
    -- url personal  
    url                varchar(100) null,  
    avatar              varchar(100) null,  
    sex                integer null,  
    facebook_url       varchar(100) null,  
    phone_number       varchar(100) null,  
    phone_number2      varchar(100) null,  
    address            varchar(100) null,  
    ...  
);
```

Ejemplo no tan irreal ...



1	x	y	null	null	null	null	null	null	null	null	null	null	null
2	x	y	null	null	null	null	null	null	null	null	null	null	null
3	x	y	null	null	null	null	null	null	null	null	null	null	null
4	x	y	null	null	null	null	null	null	null	null	null	null	null
5	x	y	null	null	null	null	null	null	null	null	null	null	null
6	x	y	null	null	null	null	null	null	null	null	null	null	null
7	x	y	null	null	null	null	null	null	null	null	null	null	null
8	x	y	null	null	null	null	null	null	null	null	null	null	null
9	x	y	null	null	null	null	null	null	null	null	null	null	null

Fat model



Al cabo de un tiempo, nuestro modelo va sumando cada vez más columnas y se transforma en un modelo "fat".

Desventajas:

- Implica realizar múltiples cambios en el modelo de datos y/o código cada vez que uno de estos cambios aparece
- No es muy cómodo cuando tenemos que almacenar datos "dispares" (solo para un número reducido de casos)

Skinny model



Otra alternativa es pensar en otro modelo donde tengamos una tabla principal con datos comunes a todas las filas, y una tabla separada que nos permita resolver los problemas planteados.

```
create table users (  
  user_id          integer primary key,  
  first_names      varchar(50),  
  last_name        varchar(50) not null,  
  email            varchar(100) not null unique,  
  password         varchar(30) not null,  
  registration_date timestamp(0)  
);
```

```
create table users_extra_info (  
  user_info_id     integer primary key,  
  user_id          not null references users,  
  field_name       varchar(100) not null,  
  field_type       varchar(100) not null,  
  -- one of the three columns below will be non-NULL  
  varchar_value    varchar(4000),  
  bool_value       bool,  
  date_value       timestamp(0)  
);
```

Skinny model



Ejemplo:

users table

user_id	first_names	last_name	email	password
1	Wile E.	Coyote	supergenius@yahoo.com	IFUx42bQzgMjE

users_extra_info table

user_info_id	user_id	field_name	field_type	varchar_value	blob_value	date_value
1	1	birthdate	date	--	--	1949-09-17
2	1	biography	blob_text	--	Created by Chuck Jones...	--
3	1	aim_screen_name	string	iq207	--	--
4	1	annual_income	number	35000	--	--

Skinny model



Desventajas:

- Las consultas a la base de datos se vuelven más complejas
- Necesitamos agregar controles extras para garantizar consistencia e integridad entre los datos
- No es lo "estándar"
- No resuelve nada respecto a problemas de almacenamiento !