Modelado mediante acs objects

Modelado mediante acs objects

. .: Objetivos :: .

- comprender las problemáticas existentes en un servidor de aplicaciones
- entender cómo se solucionan estas problemáticas en OpenACS
- evaluar dichas soluciones con las que se ofrecían en aLF2

Modelado mediante acs objects

. .: ¿Qué es ACS? :: .

- la aplicación que tenemos entre manos en sus diferentes sabores: aLF2, ACES, OpenACS, dotLRN, el propio ACS, etc.
- es un servidor de aplicaciones para usuarios y grupos de usuarios.

Modelado mediante acs objects

. .: ¿Qué aplicaciones? :: .

- Aplicaciones finales: foro, área de almacenamiento, encuestas, calendario, noticias, etc.
- Aplicaciones reutilizables: categorías, comentarios, avisos, etc.
- También hay aplicaciones que se parecen a otras. Ejemplo: los exámenes tipo test, se parecen a las encuestas.

. .: ¿Quién usa las aplicaciones? :: .

- la aplicación es usada por actores en diferentes dominios. En ACES los dominios y actores son los siguientes:
  - público: usuario no registrado
  - usuario: usuarios registrado y site-wide-admin
  - comunidad: miembro, miembro en un rol y admin
  - curso: estudiante, profesor, admin y profesor tutor

Modelado mediante acs objects

. .: ¿Qué grupos? :: .

- el grupo por defecto en ACS es la comunidad
- para personas que quieren tener páginas de usuario destinadas a mantener una revista electrónica en aLF2 se creó el tipo de grupo revista y así podrían éstos aprovechar las aplicaciones de ACS.

Modelado mediante acs objects

. .: ¿Qué relaciones hay entre los grupos? :: .

- ACES es una adaptación de ACS para usar sus aplicaciones en educación
- En la práctica lo distintivo era tener en cuenta una serie de relaciones entre grupos

Modelado mediante acs objects

. .: Permisos :: .

- una de las piezas clave en un servidor de aplicaciones para usuarios y grupos de usuarios es saber quién puede hacer qué en tal contexto.

Modelado mediante acs objects

. .: ¿Qué es modelar :: .

- saber cuáles son nuestros usuarios
- definir cuáles son nuestros grupos de usuarios
- entender de qué manera se relacionan unos grupos de usuarios con otros
- saber qué aplicaciones necesitan nuestros usuarios
- entender qué permisos serán más adecuados

Modelado mediante acs objects

. .: Usuarios en ACS 3.x :: .

- Los usuarios en ACS 3.x se implementan mediante la tabla users

```
create table users (
        user_id                         integer not null primary key,
        first_names                     varchar(100) not null,
        last_name                       varchar(100) not null,
        screen_name                     varchar(100),
        constraint users_screen_name_unique unique(screen_name),
        priv_name                       integer default 0,
        email                           varchar(100) not null unique,
....
);
```

- Era una tabla con muchos atributos.

Modelado mediante acs objects

. .: Grupos en ACS 3.x :: .

- Los grupos en ACS 3.x se implementan mediante la tabla groups

```
create table user_groups (
        group_id            integer primary key,
        group_type          not null references user_group_types(group_type),
        group_name          varchar(100),
        short_name          varchar(100) unique not null,
        admin_email         varchar(100),
...
);
```

- Esta tabla tampoco era muy delgada.

Modelado mediante acs objects

. .: Grupos y Usuarios en ACS 3.x :: .

- La relacion entre usuarios y grupos se hacia por medio de una tabla de mapeo.

```
create table user_group_map (
        group_id            not null references user_groups,
        user_id                 not null references users,
        role                varchar(200),
        registration_date           date default sysdate not null,
        mapping_user            not null references users(user_id),
        mapping_ip_address              varchar(50) not null,
        unique (group_id, user_id, role)
);
```

Modelado mediante acs objects

. .: Tipos de Grupos :: .

- En ACS 3.x es posible con campos especificos, y/o que son usados con propositos especificos.

- Estos tipos de grupo se almacenan en:

```
create table user_group_types (
        user_group_types_id integer primary key,
        group_type              varchar(20) unique not null,
        pretty_name             varchar(50) not null,
        pretty_plural           varchar(50) not null,
        approval_policy             varchar(30) not null,
        default_new_member_policy               varchar(30) default 'open' not null,
        group_module_administration             varchar(20),
        has_virtual_directory_p                 char(1) default 'f' check(has_virtual_directory_p in ('t','f')),
        group_type_admin_directory      varchar(200),
        group_public_directory          varchar(200),
        group_admin_directory           varchar(200)
);
```

. .: Crear Tipos de Grupos :: .

- Crear tabla con atributos ampliados.
- Establecer relaciones con otros tipos de grupos si procede.
- Insertar el grupo como un nuevo registro en user_group_types.
- Mapear cada atributo con el grupo en user_group_type_fields.

. .: Relaciones entre grupos :: .

- En education la relacion que habia entre grupos se hacia mediante tablas de mapeo.

. .: Aplicaciones en ACS 3.x :: .

- Las aplicaciones pensadas para ser reutilizadas por el usuario final se incluian en lo que se llamaba core. Ej: general-comments, categories, etc.
- Las aplicaciones finales normalmente se asociaban a usuarios, o grupos de usuarios.

Modelado mediante acs objects

. .: Permisos en ACS 3.x :: .

- El quien puede hacer que y donde se resolvia con la siguiente tabla:

```
create table general_permissions (
        permission_id                   integer not null primary key,
        on_what_id                      integer not null,
        on_which_table                  varchar(30) not null,
      scope                     varchar(20),
        user_id                         references users,
        group_id                  references user_groups,
        role                      varchar(200),
        permission_type                 varchar(20) not null,
        check ((scope = 'user' and user_id is not null
              and group_id is null and role is null) or
               (scope = 'group_role' and user_id is null
              and group_id is not null and role is not null) or
               (scope = 'group' and user_id is null
              and group_id is not null and role is null) or
               (scope in ('registered_users', 'all_users')
              and user_id is null
              and group_id is null and role is null)),
        unique (on_what_id, on_which_table,
              scope, user_id, group_id, role, permission_type)
);
```

. .: Open ACS 4 un servidor de aplicaciones web para grupos y usuarios orientado a objetos :: .

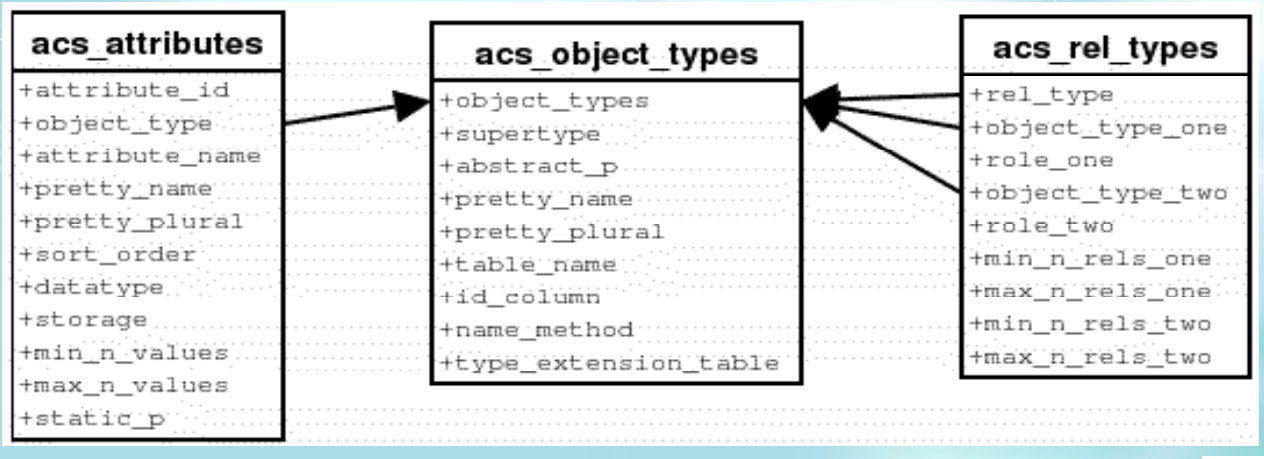
- Ahora que ya sabemos qué es modelar aprendamos a hacerlo con Open ACS 4 y, por tanto, con dotLRN.

- Debemos tener en cuenta que Open ACS 4 es un entorno orientado a objetos con todo lo que ello supone (clases, herencia, métodos, etc.)

Modelado mediante acs objects

. .: ¿Como se implementa un diseño de clases en OpenACS :: .

- Los atributos de cada clase se guardan en acs_attributes
- El nombre y los modificadores de clase en acs_object_types
- Las relaciones con otras clases en acs_rel_types.



Modelado mediante acs objects

. .: ¿Donde se guardan las instancias del diseño? :: .

- Las tablas para almacenar valores son distintas que las que usamos para almacenar la definicion de clases.

Modelado mediante acs objects

. .: ¿Como se implementan los metodos de una clase? :: .

- Las herramientas necesarias para ello ya estan implementadas en la api:

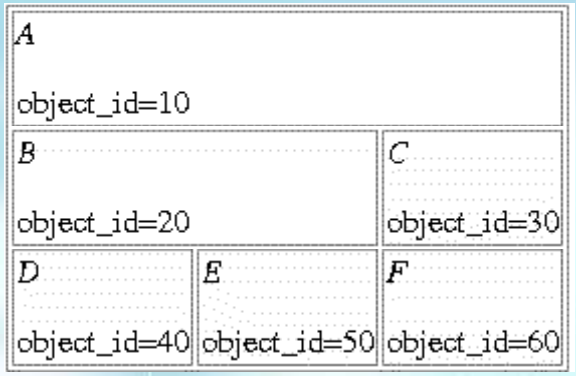
Tipos de Objetos y Atributos	Objetos	Tipos de Relaciones
create_type	new	create_role
create_attribute	name	drop_role
drop_attribute	default_name	create_type
add_description	get_attribute_storage	drop_type
drop_description	get_attribute	new
	set_attribute	delete

- A partir de estas herramientas ya podemos crear los metodos especificos de nuestra clase. Esto se hace via pl/sql.

Modelado mediante acs objects

. .: Jerarquia de Objetos :: .

- Con la siguiente jerarquia de objetos:



- Tendriamos la siguiente relacion de atributos:

object_id	context_id
20	10
30	10
40	20
50	20
60	30

- Si configuramos un objeto con el campo security_inherit_p a 'f', no hay herencia de permisos de unos objetos a otros.

Modelado mediante acs objects

. .: ¿Que son las parties :: .

- Una party es una abstraccion de lo que son los grupos y la personas. Se implementa del siguiente modo:

```
create table parties (
 party_id
    not null
    constraint parties_party_id_fk references acs_objects (object_id)
    constraint parties_pk primary key,
    email          varchar2(100)   constraint parties_email_un unique,
    url            varchar2(200)
);
```

Modelado mediante acs objects

. .: Jerarquia de parties :: .

- Algunas clases importantes que heredan de parties

Modelado mediante acs objects

```
create table parties (
    party_id  not null  constraint parties_party_id_fk references acs_objects (object_id)
        constraint parties_pk primary key,
    email           varchar2(100)   constraint parties_email_un unique,
    url             varchar2(200)
);
create table persons (
    person_id  not null  constraint persons_person_id_fk references parties (party_id)
        constraint persons_pk primary key,
    first_names         varchar2(100)    not null,
    last_name           varchar2(100)     not null
);

create table users (
    user_id  not null  constraint users_user_id_fk references persons (person_id)
        constraint users_pk primary key,
    password   char(40),

create table groups (
    group_id       not null  constraint groups_group_id_fk references parties (party_id)
        constraint groups_pk primary key,
    group_name  varchar2(100) not null
);
```

Modelado mediante acs objects

. .: Relaciones entre Parties :: .

- Hay 2 relaciones importantes: composicion y membresia

```
create table composition_rels (
    rel_id
            constraint composition_rel_rel_id_fk references acs_rels (rel_id)
        constraint composition_rel_rel_id_pk primary key
);

 create table membership_rels (
      rel_id
          constraint membership_rel_rel_id_fk references acs_rels (rel_id)
          constraint membership_rel_rel_id_pk primary key,
      -- null means waiting for admin approval
      member_state        varchar2(20)
          constraint membership_rel_mem_ck
           check (member_state in ('approved', 'banned', 'rejected', 'deleted'))
 );
```

. .: Relaciones entre Parties :: .

- La diferencia entre estas 2 relaciones es que la composicion permite la transitividad y la de membresia no.
- Si A se compone de B y C y B se compone de D y E, entonces uno de los componente de A es D.
- Si A tiene como miembros a B y C y B tiene los miembros D y E, entonces D puede ser miembro de A o no.
- Estas relaciones se usan en parties y sobre todo en grupos.

Modelado mediante acs objects

. .: Jerarquia de Privilegios :: .

- Si, por ejemplo, miramos a los foros podemos desglosar una jerarquia de privilegios asociada.

Modelado mediante acs objects

. .: Jerarquia de Privilegios :: .

- Los privilegios y su jerarquia se implementan con la siguiente estructura:

```
create table acs_privileges (
        privilege       varchar(100) not null constraint acs_privileges_pk  primary key,
        pretty_name     varchar(100),
        pretty_plural   varchar(100)
);

create table acs_privilege_hierarchy (
        privilege           varchar(100) not null  constraint acs_priv_hier_priv_fk  references acs_privileges (privilege),
        child_privilege  varchar(100) not null  constraint acs_priv_hier_child_priv_fk  references acs_privileges
(privilege),
        constraint acs_privilege_hierarchy_pk   primary key (privilege, child_privilege)
);
```

Modelado mediante acs objects

. .: Permisos :: .

- La tabla acs_permissions responde a la ya famosa pregunta clave quién (grantee_id) puede hacer qué (privilege) en tal objeto (object_id), teniendo en cuenta las jerarquias.

```
create table acs_permissions (
 object_id      not null        constraint acs_permissions_on_what_id_fk references acs_objects (object_id),
 grantee_id     not null        constraint acs_permissions_grantee_id_fk references parties (party_id),
 privilege      not null        constraint acs_permissions_priv_fk references acs_privileges (privilege),
 constraint acs_permissions_pk  primary key (object_id, grantee_id, privilege)
);
```

Modelado mediante acs objects

. .: Especializar Usuarios :: .

- Si tuvieramos una comunidad de usuarios de zapatos podria interesar guardar el numero de pie de cada usuario.
- Para ello se podria crear un subtipo de usuarios llamado `shoes_users` .
- La tabla `shoes_users` referenciaria a `users` por su primary_key.
- Si en vez de usuarios fueran grupos actuariamos de forma similar.

. .: Especializar Relaciones :: .

- En una intranet podria interesarnos tener la relacion salario entre una compañia y una persona o usuario registrado.
- Esto se haria especializando la relacion de membresia ya comentada.

. .: Modelado de education con OpenACS: dotlrn :: .

- ¿Como modelariais vosotros el anterior modelo de datos de education con esta nueva filosofia?. Recordad:

Modelado mediante acs objects

. .: Modelado de education con OpenACS: dotlrn :: .

- La implementacion actual es algo decepcionante. No usa acs_rels

. .: Modelado de education con OpenACS: dotlrn :: .

- Una posible mejora con relaciones de composicion.

. .: Conclusiones :: .

- El paradigma orientado a objetos de Open ACS 4:
  - Facilita la abstraccion
  - Facilita la reutilizacion de codigo.
  - Almacena los datos en bases de datos normalizadas.

- Es lo que se llama una refactorizacion de codigo para trabajar mas comodamente :-).