OpenACS e as Comunidades Virtuais

Eduardo Santos

eduardo.edusantos@gmail.com
eduardosantos@previdencia.gov.br

www.softwarepublico.gov.br
eduardosan.wordpress.com

O que é o OpenACS

- Uma comunidade de desenvolvedores
- Um livro sobre aplicações para a Internet
- Uma tecnologia para desenvolvimento de aplicações Web
- Um repositório de software livre e código aberto bastante antigo
- Um consórcio de instituições e empresas

Comunidade OpenACS

- Conteúdo editado por experts ou voluntários
- Presença constante de meios de colaboração
- Moderação e cultura da comunidade
- Baseada nos ideais de software livre desde o começo

Histórico

- 1996: ACS (ArsDigita Community System)
- 1998: surgimento do “toolkit” ACS
- 1999: Projeto ACS/pg no Source Forge
- 2000: ACS é reescrito em Java
- 2001: ArsDigita prioriza ACS4 (Java) e considera ACS3 obsoleto

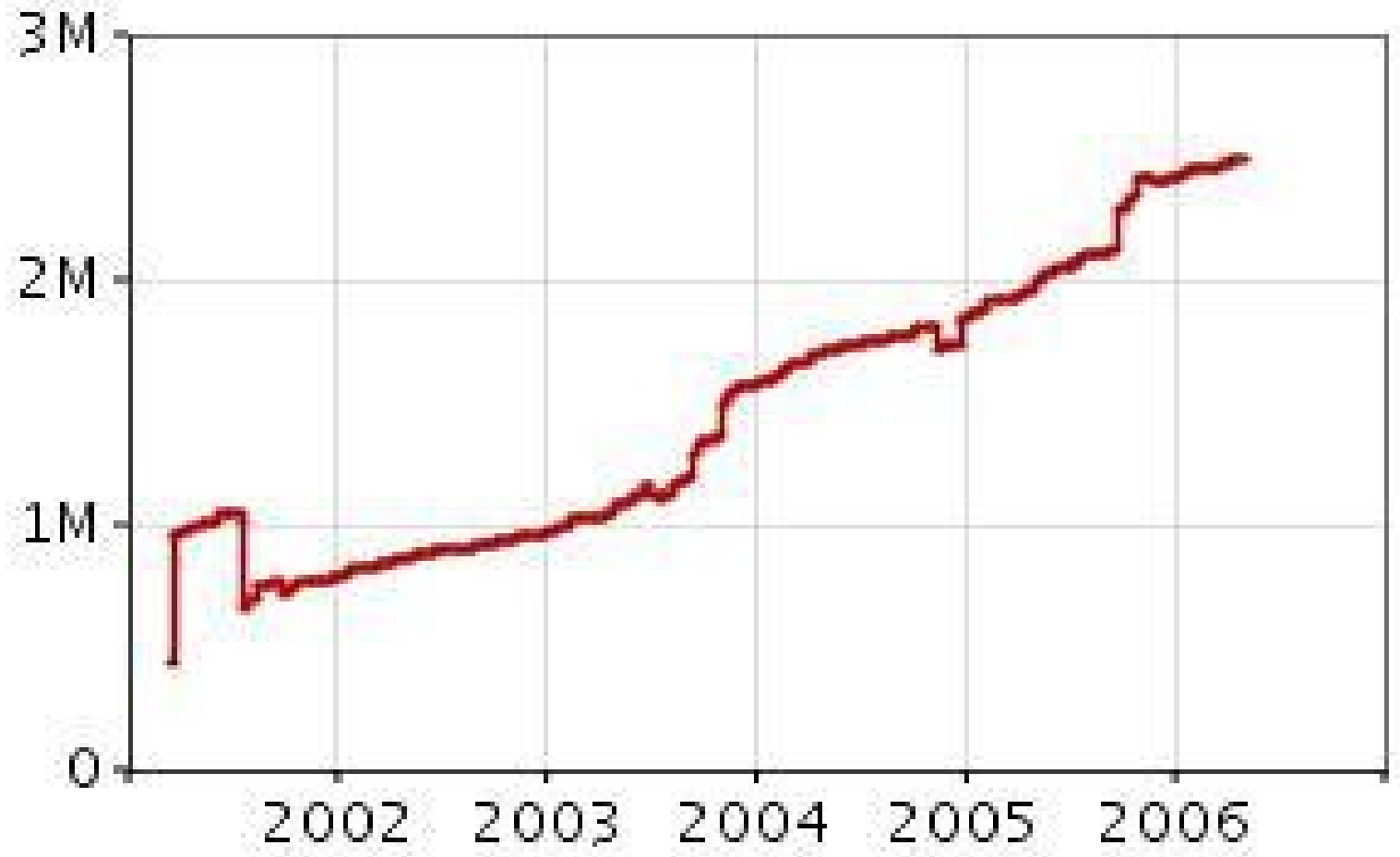
Histórico

- 2001: Fork do ACS3, ainda em TCL (agora OpenACS4) supera ACS4 em desenvolvimento
- ainda 2001: Saída de Greenspun, tentativa de fechar o código, recusa de parceira com Microsoft
- final 2001: ACS4.6 (Java) tem sua licença modificada e deixa de ser livre, excluindo a comunidade OpenACS

Histórico

- Fevereiro de 2002: ArsDigita é comprada pela Red Hat e fecha as portas
- 28 de outubro de 2002: OpenACS 4.6.0 é liberado

Crescimento da comunidade



PANDA

philip.greenspun.com/panda

- Philip and Alex's Guide to Web Publishing
- Publicado pela primeira vez em setembro de 1998
- Interessante histórico sobre o começo da Internet

SEIA

philip.greenspun.com/seia

- Software Engineering for Internet Applications
- Publicado em 2006
- Ótima guia sobre como construir aplicações para a Internet
- Serve para qualquer ambiente de desenvolvimento

Tecnologia

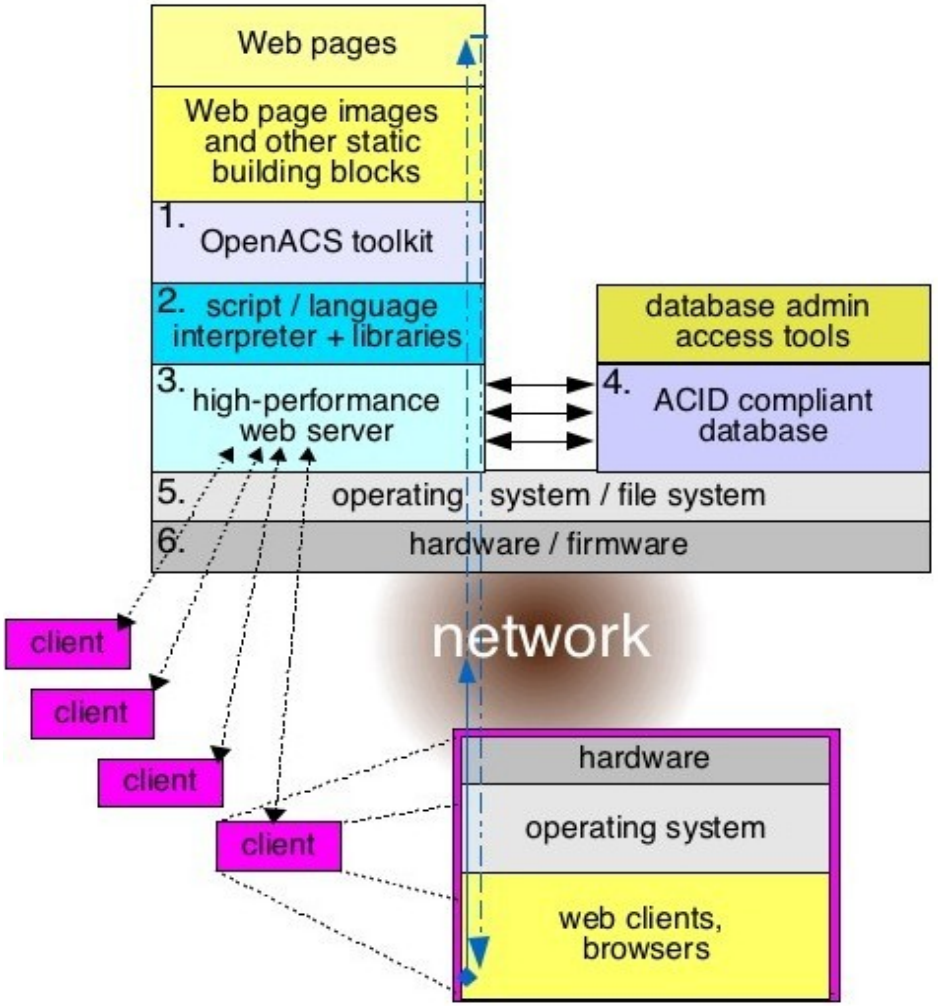
“No desenvolvimento do software, um framework ou arcabouço é uma estrutura de suporte definida em que um outro projeto de software pode ser organizado e desenvolvido.”

Wikipedia

Tecnologia

- O OpenACS é um framework de desenvolvimento Web, em muitas características
- É também um “toolkit” para desenvolvimento de aplicações de comunidades
- É um ambiente agregador nos princípios de modularidade e empacotamento

Tecnologia


Web pages
Web page images and other static building blocks
1. OpenACS toolkit
2. script / language interpreter + libraries
3. high-performance web server
database admin access tools
4. ACID compliant database
5. operating system / file system
6. hardware / firmware
client
client
client
client
network
hardware
operating system
web clients, browsers

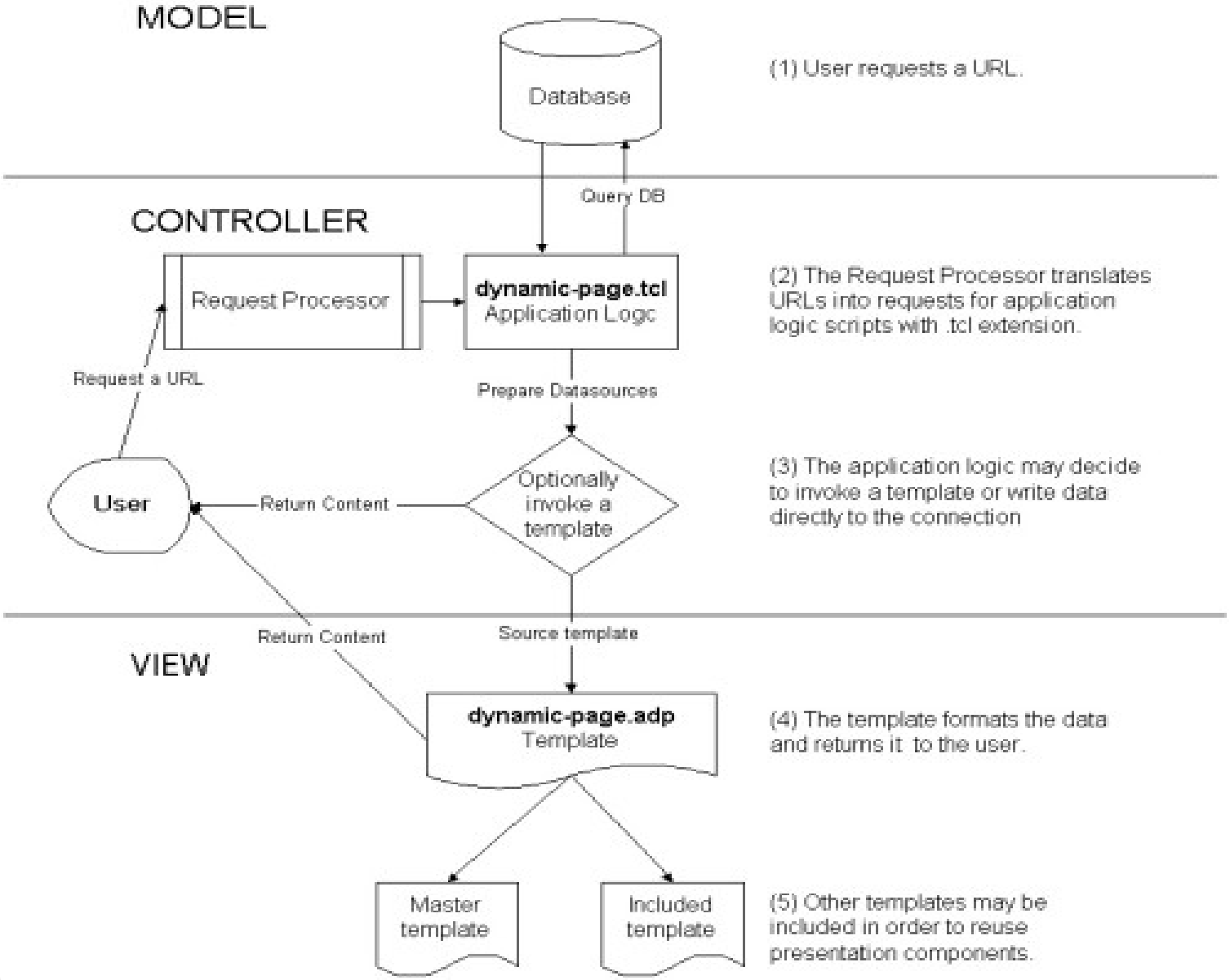
Arquitetura

- Arquitetura unificada entre cliente e servidor (diferente da multi-tier)
- Modelo MVC
- Metadados e geração automática de código
- Gerenciamento de papéis e perfis
- Arquitetura multi-pool do banco de dados

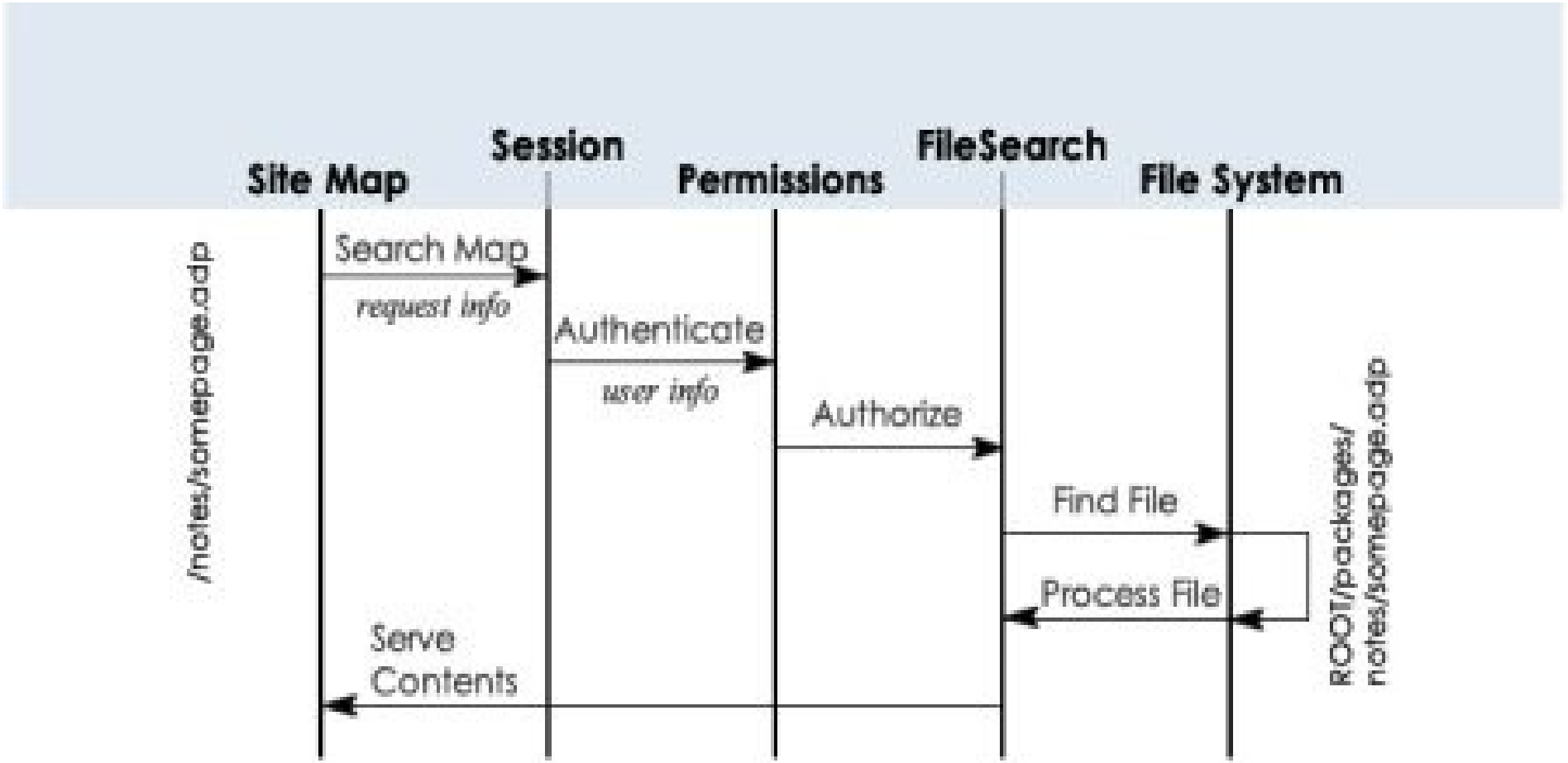
Modelo MVC

- Model-View-Controller
  - Separação entre código e design
  - Geração automática de código
  - Camada de banco de dados separada
  - sistema de cache de queries
- Request Processor
  - Tratamento das requisições de página feitas pelo usuário

Diagrama MVC



Request Processor



Request Processor

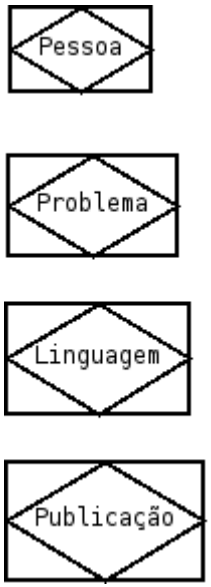
- Possibilidade de criar um mapa virtual do site
- Tratamento de autenticação por usuário e por sessão
- Sistema de permissões por arquivo ou URL
- Tratamento da lógica e conteúdo de maneira eficiente

Metadados

- Dados que geram dados (geração automática de código)
  - Maior facilidade de integrar “atores” no sistema
  - Relacionar vários “atores” sem necessidade de criar novas tabelas específicas

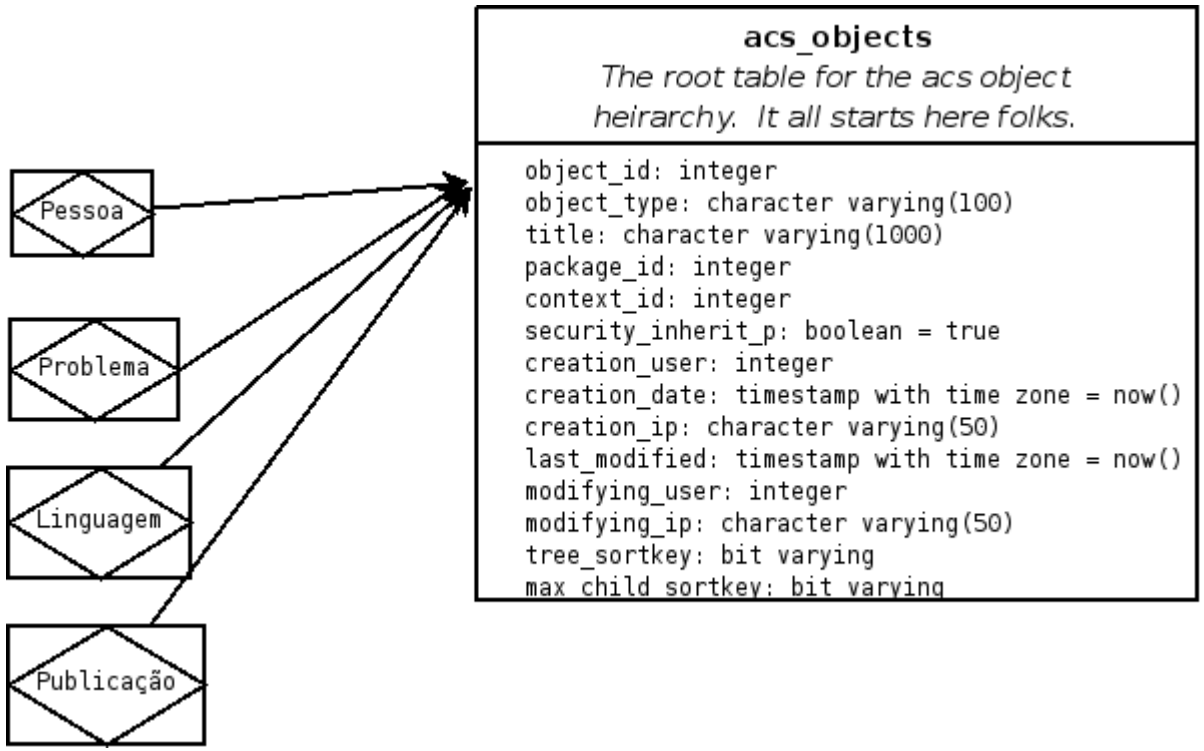
Metadados

- Vamos assumir os seguintes tipos como exemplo:



Metadados

- Definimos todos os tipos como objetos numa tabela

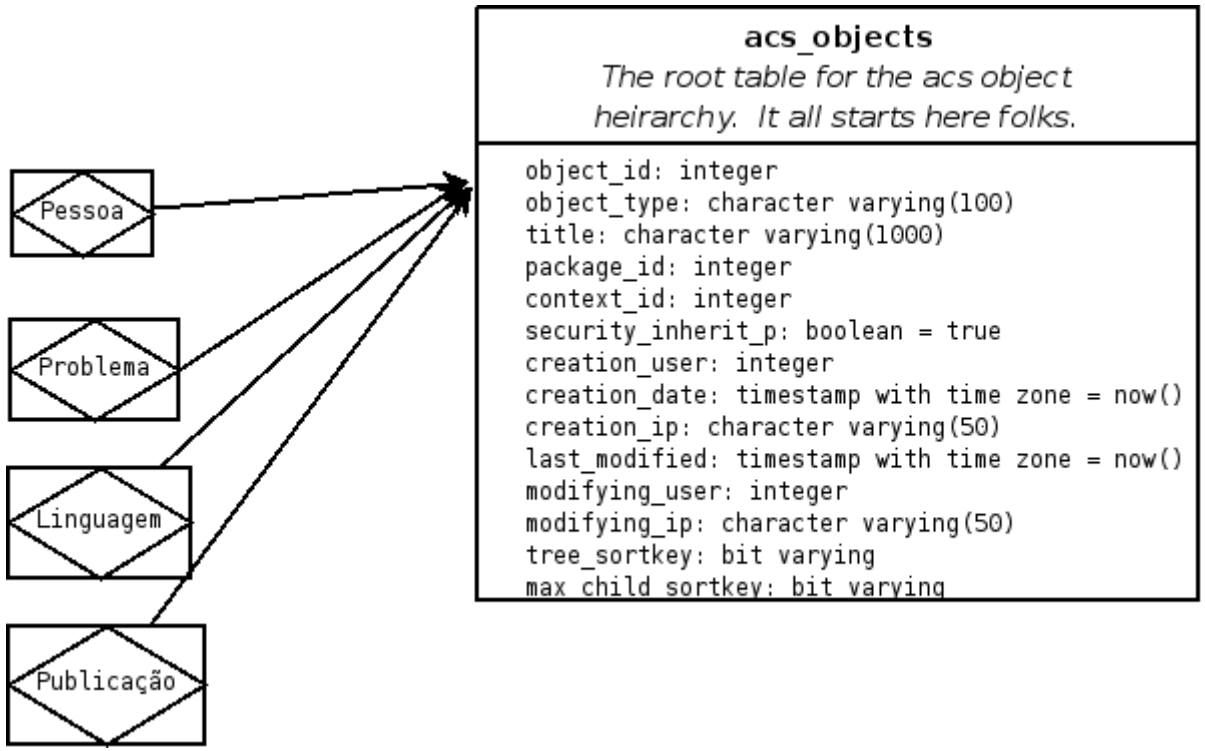


Metadados

- Exemplo: João criou a linguagem LISP
  - Objeto: Linguagem LISP
  - Criador: João
- Pergunta: por que não referenciar a tabela de usuários?
- Resposta: O criador pode não fazer mais parte do sistema

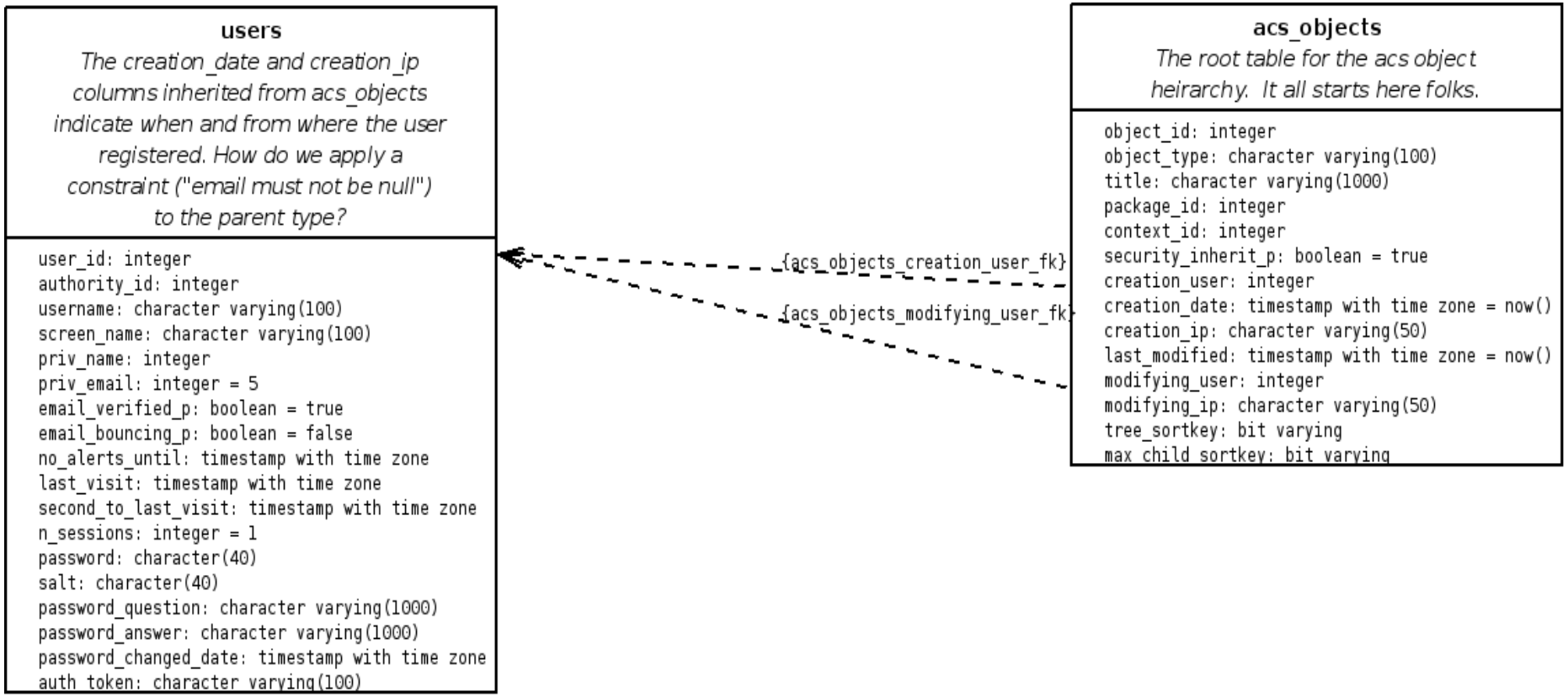
Metadados

- Para cada objeto, precisamos de várias informações



Metadados

- Além disso, precisamos de informações específicas do objeto



Metadados

- A estrutura de metadados me permite gerar códigos genéricos para qualquer tipo de dado
- Através das API's do sistema, toda a estrutura de relacionamentos obedece um padrão

Metadados

- XQL query dispatcher: independência de banco de dados
- Service Contract API
  - Reusabilidade de código
  - Integração de aplicações
  - Extensibilidade de pacotes

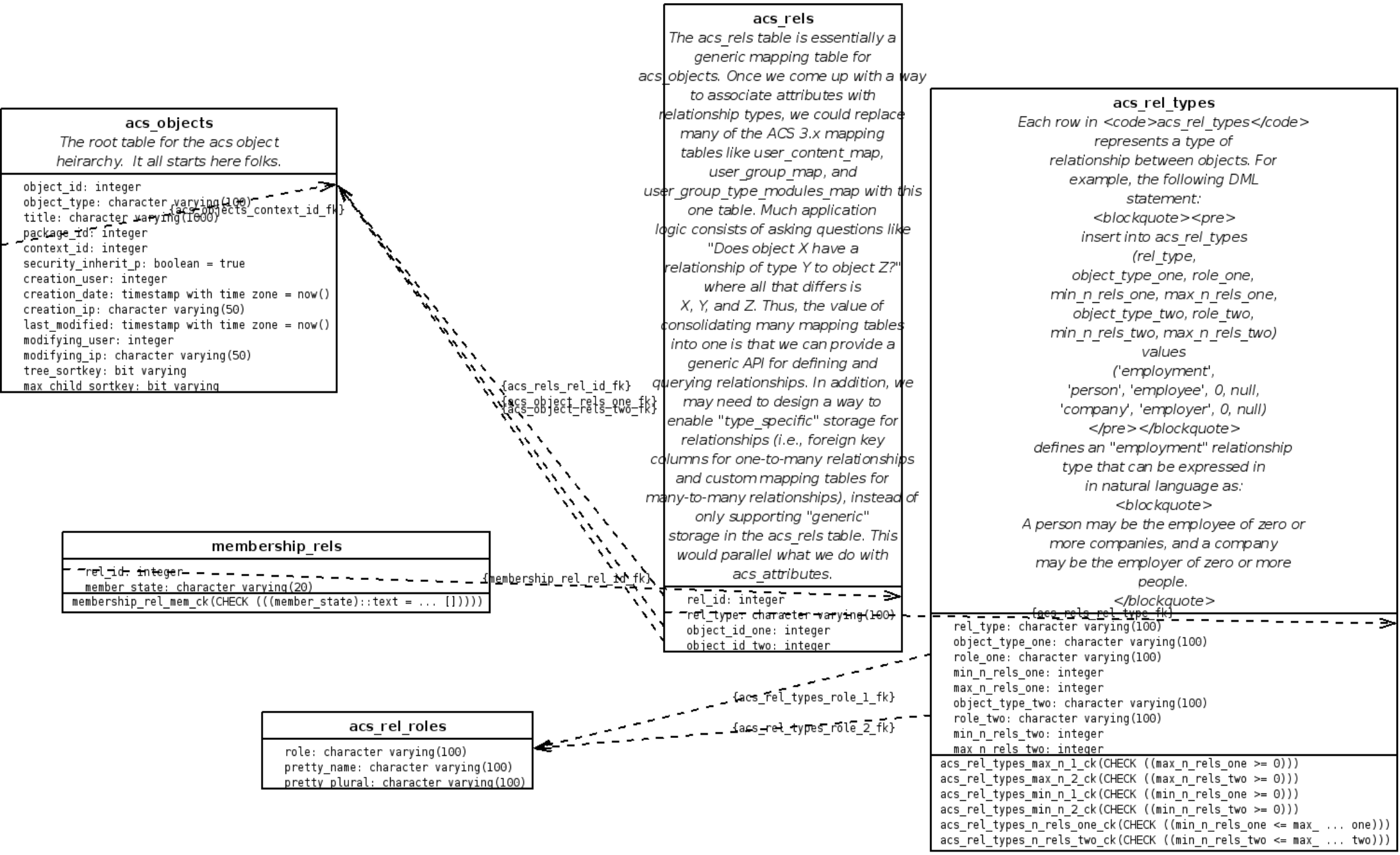
Metadados

```
v_user_id := person__new(
    v_user_id,
    p_object_type,
    p_creation_date,
    p_creation_user,
    p_creation_ip,
    p_email,
    p_url,
    p_first_names,
    p_last_name,
    p_context_id
);
```

```
select acs__add_user(
    :user_id,
    'user',
    now(),
    null,
    :peeraddr,
    :authority_id,
    :username,
    :email,
    :url,
    :first_names,
    :last_name,
    :hashed_password,
    :salt,
    :screen_name,
    :email_verified_p,
    :member_state
);
```

Gerenciamento de papéis e perfis

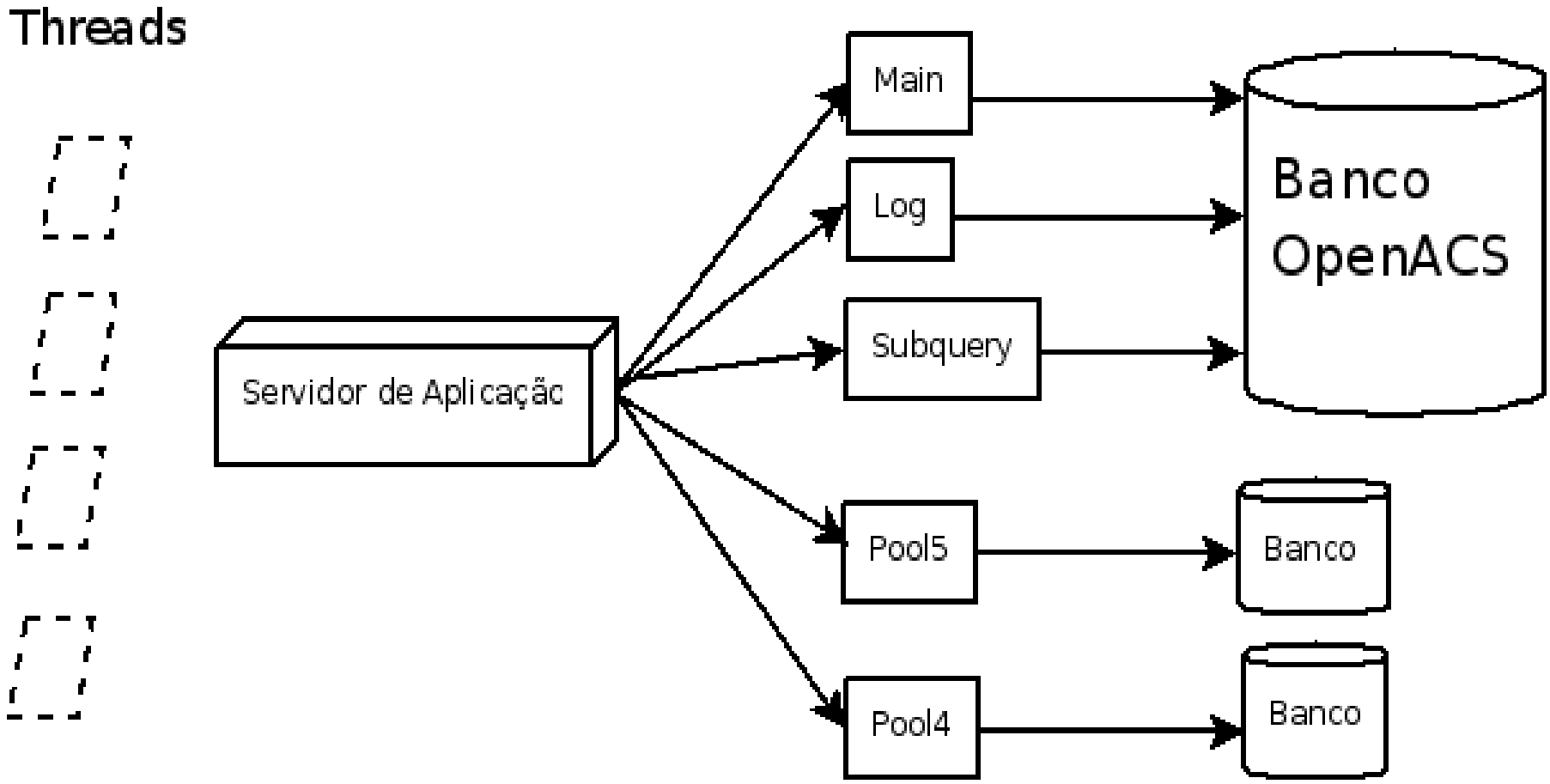
- Relacionamento entre objetos
- Perfis entre grupos de objetos
- Gerenciamento de papéis


acs_objects
The root table for the acs object heirarchy. It all starts here folks.
object_id: integer
object_type: character varying(100)
title: character varying(1000)
package_id: integer
context_id: integer
security_inherit_p: boolean = true
creation_user: integer
creation_date: timestamp with time zone = now()
creation_ip: character varying(50)
last_modified: timestamp with time zone = now()
modifying_user: integer
modifying_ip: character varying(50)
tree_sortkey: bit varying
max_child_sortkey: bit varying
{acs_objects_context_id_fk}
acs_rels
The acs_rels table is essentially a generic mapping table for acs_objects. Once we come up with a way to associate attributes with relationship types, we could replace many of the ACS 3.x mapping tables like user_content_map, user_group_map, and user_group_type_modules_map with this one table. Much application logic consists of asking questions like "Does object X have a relationship of type Y to object Z?" where all that differs is X, Y, and Z. Thus, the value of consolidating many mapping tables into one is that we can provide a generic API for defining and querying relationships. In addition, we may need to design a way to enable "type_specific" storage for relationships (i.e., foreign key columns for one-to-many relationships and custom mapping tables for many-to-many relationships), instead of only supporting "generic" storage in the acs_rels table. This would parallel what we do with acs_attributes.
rel_id: integer
rel_type: character varying(100)
object_id_one: integer
object_id_two: integer
{acs_rels_rel_id_fk}
{acs_object_rels_one_fk}
{acs_object_rels_two_fk}
acs_rel_types
Each row in <code>acs_rel_types</code> represents a type of relationship between objects. For example, the following DML statement:
<blockquote><pre>
insert into acs_rel_types
(rel_type,
object_type_one, role_one,
min_n_rels_one, max_n_rels_one,
object_type_two, role_two,
min_n_rels_two, max_n_rels_two)
values
('employment',
'person', 'employee', 0, null,
'company', 'employer', 0, null)
</pre></blockquote>
defines an "employment" relationship type that can be expressed in in natural language as:
<blockquote>
A person may be the employee of zero or more companies, and a company may be the employer of zero or more people.
</blockquote>
{acs_rels_rel_type_fk}
rel_type: character varying(100)
object_type_one: character varying(100)
role_one: character varying(100)
min_n_rels_one: integer
max_n_rels_one: integer
object_type_two: character varying(100)
role_two: character varying(100)
min_n_rels_two: integer
max_n_rels_two: integer
acs_rel_types_max_n_1_ck(CHECK ((max_n_rels_one >= 0)))
acs_rel_types_max_n_2_ck(CHECK ((max_n_rels_two >= 0)))
acs_rel_types_min_n_1_ck(CHECK ((min_n_rels_one >= 0)))
acs_rel_types_min_n_2_ck(CHECK ((min_n_rels_two >= 0)))
acs_rel_types_n_rels_one_ck(CHECK ((min_n_rels_one <= max_ ... one)))
acs_rel_types_n_rels_two_ck(CHECK ((min_n_rels_two <= max_ ... two)))
membership_rels
rel_id: integer
member_state: character varying(20)
membership_rel_mem_ck(CHECK (((member_state)::text = ... [])))
{membership_rel_rel_id_fk}
acs_rel_roles
role: character varying(100)
pretty_name: character varying(100)
pretty_plural: character varying(100)
{acs_rel_types_role_1_fk}
{acs_rel_types_role_2_fk}

Arquitetura multi-pool

- Sempre que o sistema é iniciado, três pools de conexão são abertos:
  - Main: executa a maior parte das rotinas direto no banco
  - Subquery: permite executar uma query mesmo um um pool já aberto
  - Log: Normalmente grava os registros das transações sendo executadas no momento
  - etc...

Arquitetura multi-pool



Aplicações Verticais

-]Project Open[
  - Módulo ERP (*Enterprise Resource Planning*) do OpenACS;
  - Objetivos: administração de custos e colaboração entre os membros da equipe;
  - Wiki e chat integrados (estilo OpenACS);
  - Módulo de workflow específico;
  - Gerencia “papéis” dentro de uma empresa.

Aplicações Verticais

- Bug-tracker, Ticket-tracker e Workflow
  - Ferramenta para gerência de projetos de software;
  - Exemplo de workflow simplificado;
  - Possibilidade de gerência da agenda de trabalho;
  - Gerência de tickets;
  - Possibilidades de implementação

Aplicações Verticais

- dotLRN (.LRN)
  - Ambiente de Aprendizagem Virtual (AVA)
    - Ambiente virtual para professores e alunos compartilhar tecnologias da informação e aliá-las ao processo de ensino-aprendizagem;
  - *Learning Management System* (LMS)
    - Software que automatiza a administração dos eventos de treinamento;
    - Desenvolvido para lidar com cursos de múltipla publicação.

Aplicações Verticais

- Gerido por um consórcio internacional, responsável por:
  - Fazer o controle de qualidade;
  - Realizar pesquisas educacionais;
  - Promover o uso do software no mundo;
  - Gerenciar a comunidade de desenvolvedores.
- Câmara de diretores:
  - http://www.dotlrn.org/about/board/
- Equipe de liderança:
  - http://www.dotlrn.org/about/leadership/

Aplicações Verticais

- www.catir.sede.embrapa.br

Aplicações Verticais

- www.mda.gov.br

Aplicações Verticais

- www.softwarepublico.gov.br

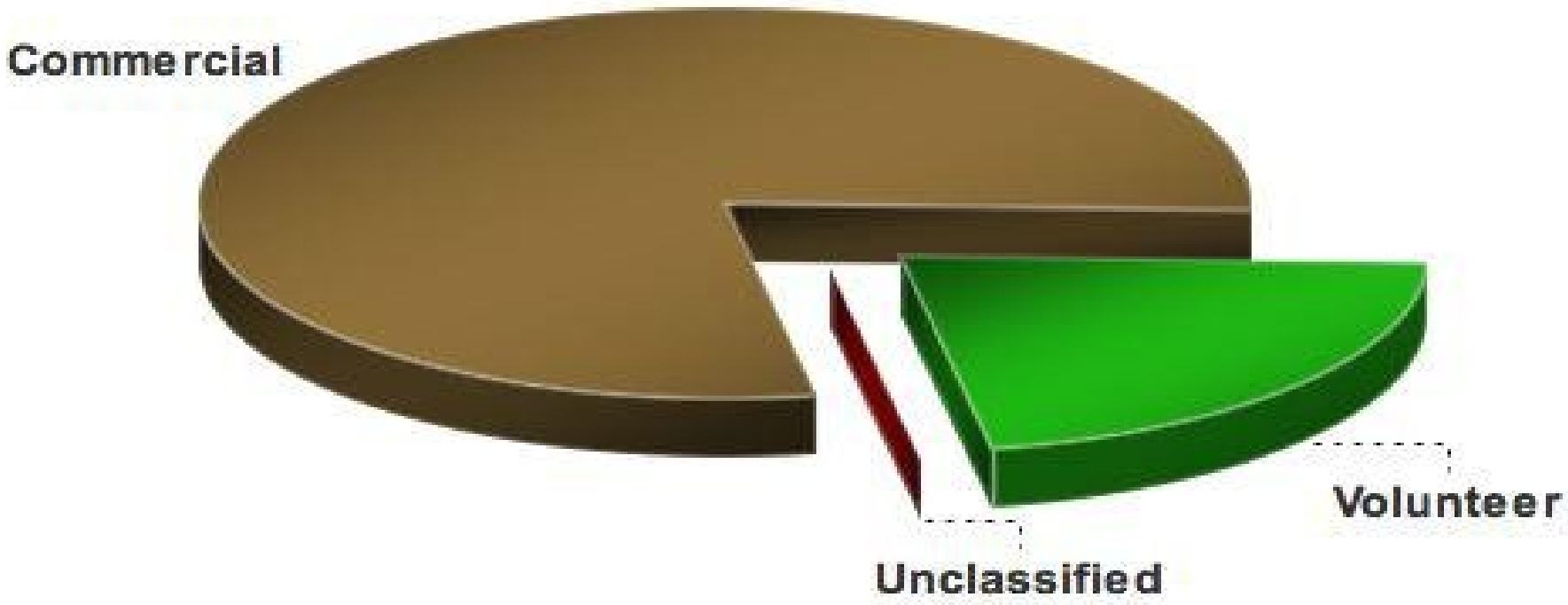
Aplicações Verticais

- http://www.edemocracia.camara.gov.br/

OpenACS: Comunidade

- *“OpenACS is more than a technology, it is a vibrant community”* - Jade Rubick
- Cooperação / moderação / confiança
- Governança
- Componentes testados e maduros
- Suporte comercial / não comercial
- Documentação

OpenACS: Comunidade



OpenACS: Comunidade

- Custo do projeto:
  - Base de código: 2.453.468 linhas de código
  - Esforço estimado: 703 anos de trabalho para uma pessoa
  - Considerando um salário anual médio de US$ 55.000,00
- Custo estimado: US$ 38.683.137,00

Fonte: http://www.ohloh.net/projects/3877?p=OpenACS

Considerações finais

- Links para conhecer mais:
  - www.dotlrn.org
  - www.openacs.org
  - www.softwarepublico.gov.br
- Comunidade brasileira:
  - www.softwarepublico.gov.br/dotlrn/clubs/openacs

Obrigado

Eduardo Santos

eduardo.edusantos@gmail.com
eduardosantos@previdencia.gov.br

www.softwarepublico.gov.br
eduardosan.wordpress.com