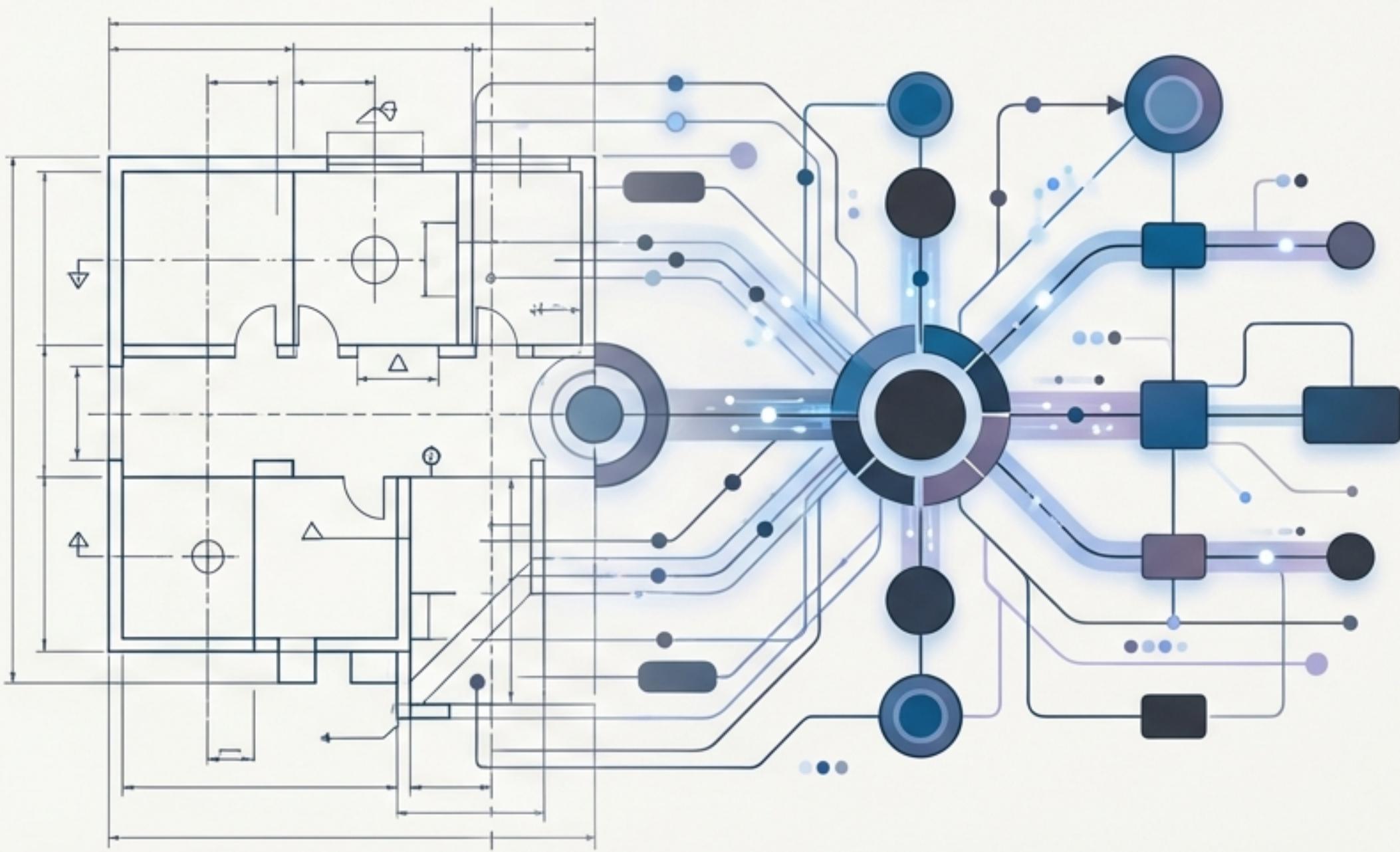
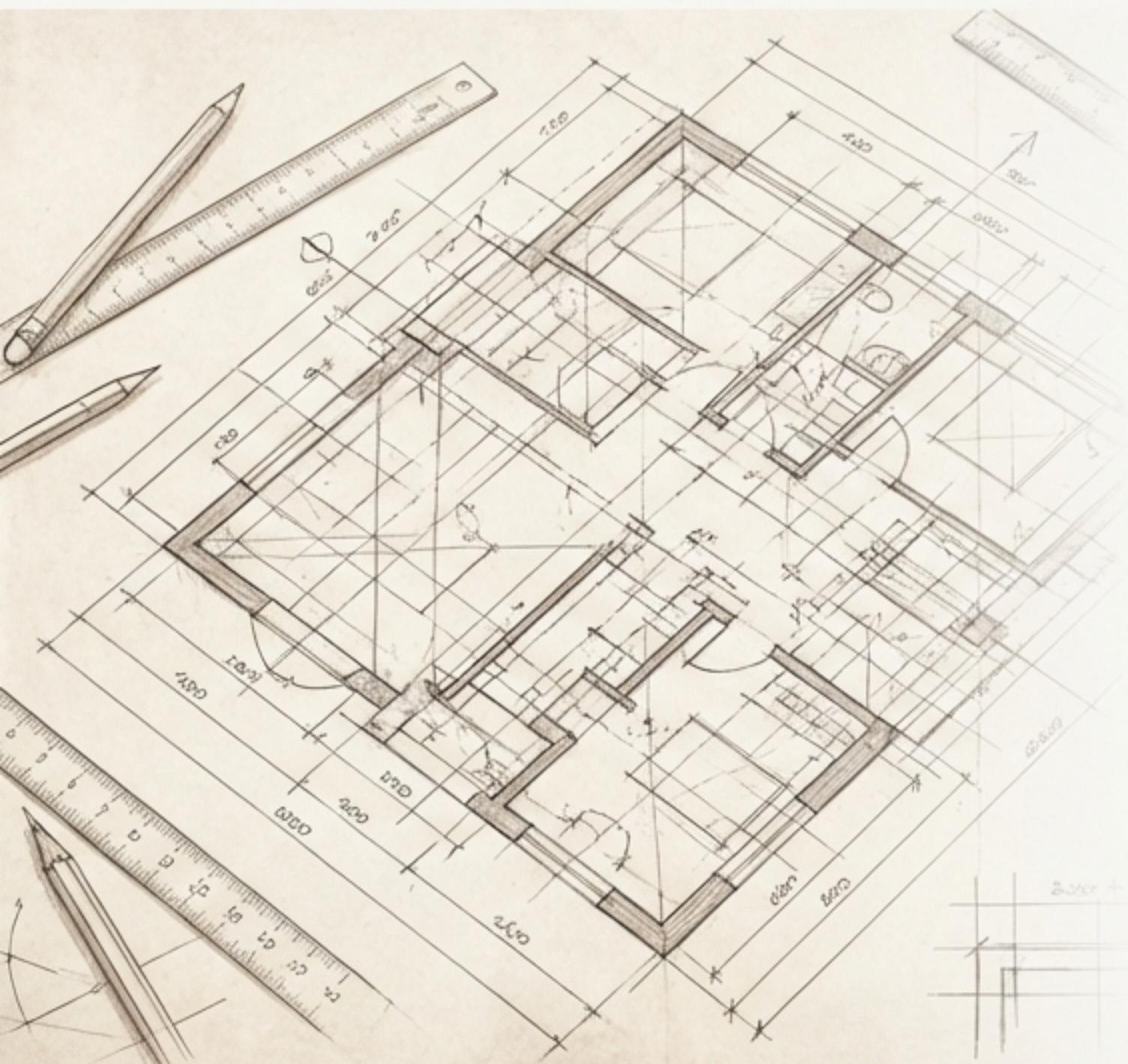


SAP CDS: Do Projeto à Realidade

Modelando o Blueprint de Aplicações Empresariais Modernas



O CDS é o Ecossistema, não Apenas a Linguagem

O SAP Core Data Services (CDS) é a espinha dorsal do SAP Cloud Application Programming Model (CAP). Ele fornece os meios para capturar declarativamente definições de serviço, modelos de dados, consultas e expressões. Um modelo CDS serve como a única fonte da verdade.

Linguagens de Origem



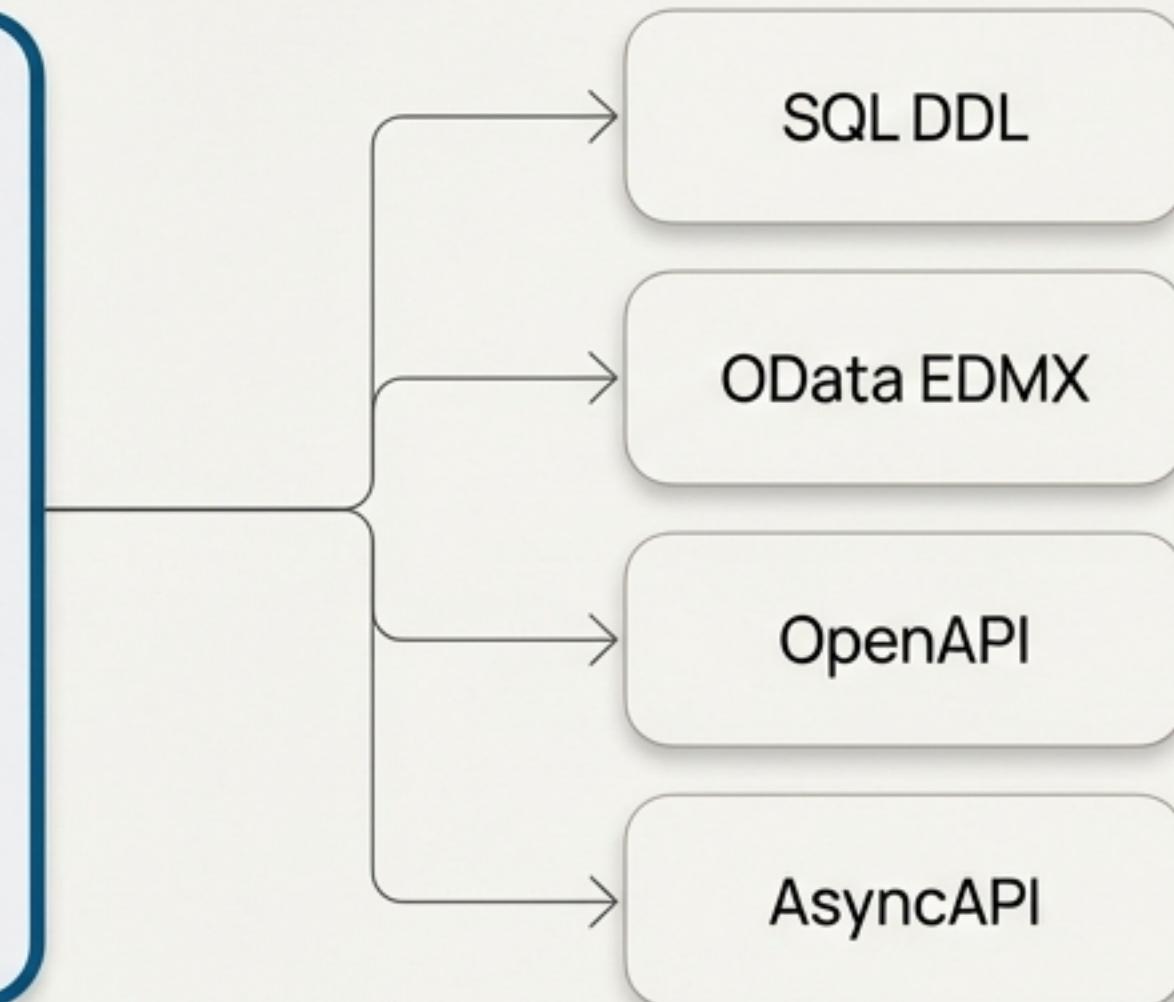
parse/compile

{ CSN }

Formato Canônico /
Fonte Única da Verdade

Em tempo de execução, os modelos CDS são
objetos JavaScript simples que cumprem a Core
Schema Notation (CSN), uma especificação
aberta derivada do JSON Schema.

Linguagens de Destino / Reflexões



A Linguagem do Projeto: CDL (Conceptual Definition Language)

O Conceptual Definition Language (CDL) é uma linguagem legível por humanos para definir modelos CDS. As fontes são comumente fornecidas em arquivos com a extensão .cds e compiladas em representações CSN.

- **Sintaxe Fundamental:** Palavras-chave (`entity`, `type`, `service`) são usadas para declarações. Identificadores são usados para referências.
- **Sensibilidade a Maiúsculas/Minúsculas:** Palavras-chave são insensíveis (`entity` é o mesmo que `ENTITY`), mas identificadores são sensíveis (`Foo` é diferente de `foo`).
- **Organização:** `namespace` para prefixar definições; `using` para importar de outros modelos, semelhante ao `import` do ES6.

```
```cds
// Definir um namespace para organizar o modelo
namespace capire.bookshop;

// Importa definições reutilizáveis
using { managed, cuid } from '@sap/cds/common';

// Define uma entidade 'Books'
entity Books : cuid, managed {
 title : String(111);
 stock : Integer;
 author : Association to Authors;
}

// Define uma entidade 'Authors'
entity Authors : cuid, managed {
 name : String;
}
```

# Construindo com Entidades e Tipos de Dados Essenciais

Entidades são tipos estruturados que representam conjuntos de dados persistentes, geralmente com uma ou mais chaves primárias. Elas são construídas usando tipos embutidos e personalizados.

## Definindo Entidades

```
```cds
entity Books {
    key ID : UUID;
    title : String(111);
    stock : Integer;
    price : Price;
    author : Association to Authors;
}

type Price : Decimal(10, 2);
```

Tipos de Dados Embutidos Comuns

Tipo CDS	Descrição	ANSI SQL (Exemplo)
UUID	Identificador único universal (RFC 4122)	NVARCHAR(36)
String(len)	Texto com comprimento definido	NVARCHAR
Integer, Int64	Números inteiros	INTEGER, BIGINT
Decimal(p,s)	Números decimais de precisão fixa	DECIMAL
Boolean	Valor verdadeiro/falso	BOOLEAN
Date, Time	Data ou hora	DATE, TIME
Timestamp	Data e hora com precisão de microssegundos	TIMESTAMP
LargeString	Texto de comprimento ilimitado	NCLOB

Conectando os Pontos: Associações e Composições

Associações capturam relacionamentos entre entidades, funcionando como 'joins pré-declarados'. Composições definem uma relação de 'contido em', ideal para estruturas de documentos como cabeçalho-item.

Associações Gerenciadas

O CDS resolve automaticamente as chaves estrangeiras. Ideal para relacionamentos entre entidades independentes.

```
```cds
entity Books {
 author : Association to Authors; // Chave estrangeir
}

entity Authors {
 // O backlink permite a navegação inversa
 books : Association to many Books on books.author =
}
```

## Composições Gerenciadas

Reflete estruturas de documentos sem a necessidade de entidades de ligação explícitas. O ciclo de vida do item é ligado ao do cabeçalho.

```
```cds
entity Orders {
    key ID : Integer;
    Items : Composition of many {
        key pos : Integer;
        product : Association to Products;
        quantity : Integer;
    }
};
```

A Interpretação da Máquina: Do CDL para o CSN (Core Schema Notation)

O CSN é a representação compacta e canônica de um modelo CDS, otimizada para ser compartilhada e interpretada com o mínimo de dependências.

Todo CDL é compilado para CSN antes de ser processado.

CDL: Legível por Humanos

```
// contexts.cds
namespace foo.bar;
entity Foo {};
context scoped {
    entity Bar : Foo {};
    context nested {
        entity Zoo {};
    }
}
```

CSN: Legível por Máquina

```
// contexts.json
{
    "definitions": {
        "foo.bar.Foo": { "kind": "entity" },
        "foo.bar.scoped": { "kind": "context" },
        "foo.bar.scoped.Bar": {
            "kind": "entity",
            "includes": [ "foo.bar.Foo" ]
        },
        "foo.bar.scoped.nested": { "kind": "context" },
        "foo.bar.scoped.nested.Zoo": { "kind": "entity" }
    }
}
```

Moldando a Realidade: Views e Projeções

Use `as select from` ou `as projection on` para derivar novas entidades a partir das existentes, de forma muito parecida com as views em SQL. Elas são traduzidas para views SQL em bancos de dados relacionais, mas também são usadas para projeções lógicas.

`as select from`: Poder SQL Completo

Para utilizar todos os recursos que um banco de dados relacional suportaria, incluindo `JOIN`s complexos, `UNION`s e subconsultas.

```
entity BooksWithAuthorDetails as select from Books {  
    ID,  
    title,  
    author.name as authorName, // Navegação via path expression  
    author.dateOfBirth  
};
```

`as projection on`: Projeções Estruturadas

Indica que você não está usando todo o poder do SQL. Ideal para expor entidades a partir de serviços OData externos, com restrições como 'sem JOINs manuais explícitos'.

```
entity PublicBookInfo as projection on Books {  
    title,  
    author.name as author  
} excluding { stock, price }; // Exclui campos sensíveis
```

Consultando o Modelo com CQL (CDS Query Language)

O CDS Query Language (CQL) é baseado no SQL padrão, aprimorando-o com recursos poderosos como expressões de caminho e projeções aninhadas, que simplificam a navegação em modelos de dados complexos.

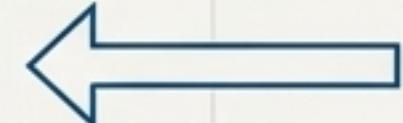
Expressões de Caminho (Path Expressions)

Use expressões de caminho para navegar através de associações e elementos estruturados em qualquer cláusula SQL ('select', 'from', 'where'). O CQL desdobra-os automaticamente nos 'JOIN's apropriados.

Concisa e Intuitiva

```
SELECT title, author.name, author.address.town.name  
FROM Books  
WHERE author.name = 'Emily Brontë';
```

Simplificado por CQL



Verboso e Complexo

```
-- plain SQL  
SELECT  
    Books.title,  
    author.name,  
    author_address_town.name  
FROM Books  
    LEFT JOIN Authors author ON author.ID = Books.author_ID  
    LEFT JOIN Addresses author_address ON  
        author_address.ID = author.address_ID  
    LEFT JOIN Towns author_address_town ON  
        author_address_town.ID = author_address.town_ID  
WHERE author.name = 'Emily Brontë';
```

Expondo o Blueprint: Definições de Serviço e APIs

O CDS permite definir interfaces de serviço como coleções de entidades expostas dentro de um bloco `service`. Essas definições são a base para a geração geração automática de serviços OData.

Exposição de Entidades

Use projeções (as `projection on`) para expor uma visão curada e segura do seu modelo de dados subjacente.

Redirecionamento Automático

Ao expor entidades relacionadas, as associações são automaticamente redirecionadas para apontar para as projeções dentro do serviço.

Ações e Funções

Defina lógica de negócios personalizada com `actions` (com efeitos colaterais) e `functions` (somente leitura).

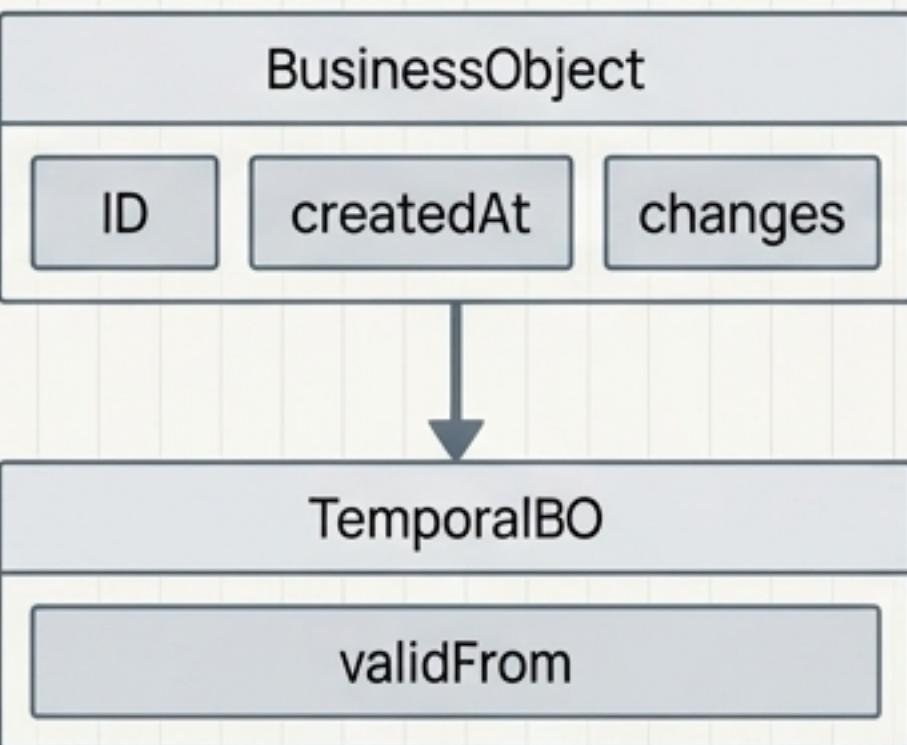
```
// Define a service to expose curated views of  
the data model  
service CatalogService {  
  
    // Expose a read-only projection of Products  
    @readonly  
    entity Products as projection on db.Products;  
  
    // Expose Authors and their books  
    entity Authors as projection on db.Authors;  
  
    // Define a custom action bound to the  
    // Products entity  
    entity Products actions {  
        action addRating (stars: Integer);  
    }  
}
```

A Filosofia da Modelagem Orientada a Aspectos

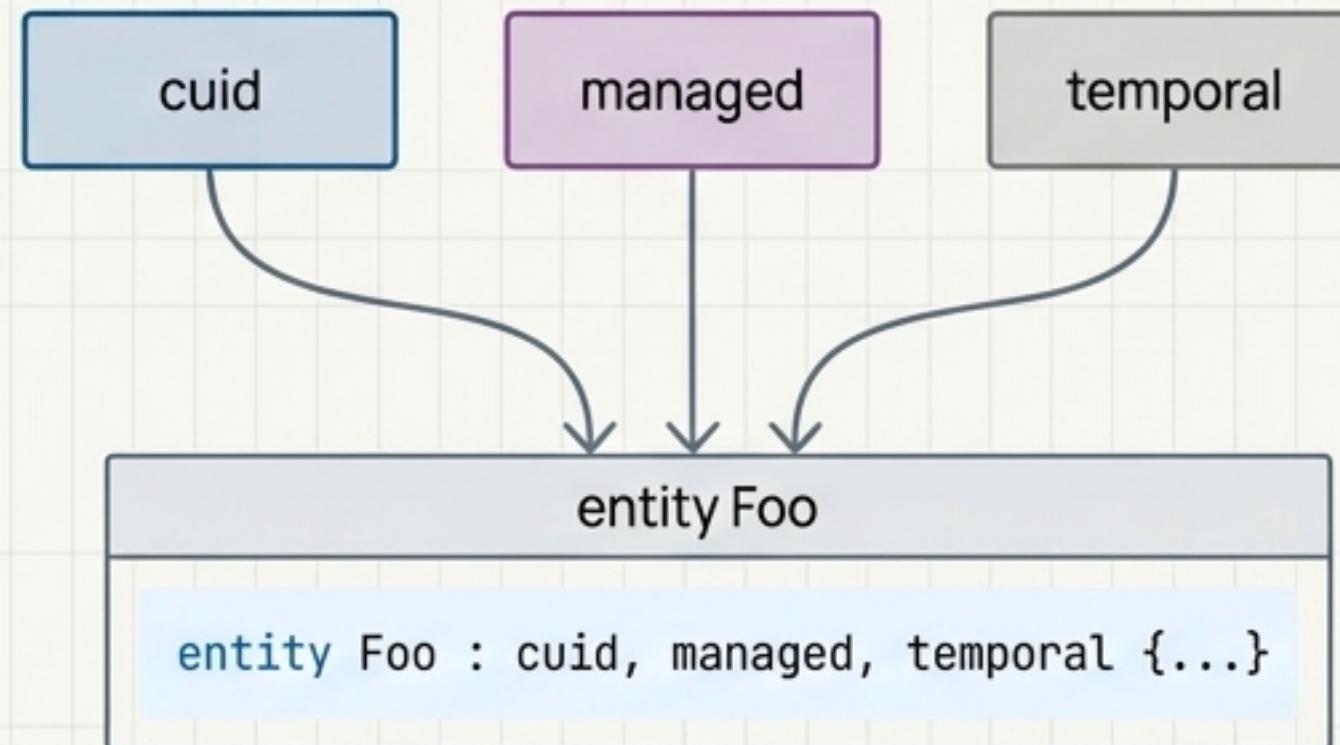
A técnica de Aspectos permite organizar seus modelos de forma a manter o domínio principal conciso, separando preocupações secundárias, reutilizando aspectos comuns e adaptando definições.

É semelhante à Programação Orientada a Aspectos.

A Abordagem Clássica (Evitar)



A Abordagem CDS (Preferir)



Devido às limitações da herança única, essas classes base frequentemente têm que combinar vários aspectos independentes em uma única definição, e os consumidores têm que aceitar todos eles.

Permite uma separação mais clara das preocupações e, portanto, liberdade de escolha sobre quais combinações de aspectos selecionar. Também permite a propriedade distribuída de tais aspectos de reutilização.

Reutilização Inteligente com @sap/cds/common

O CDS vem com um modelo pré-construído, @sap/cds/common, que fornece tipos e aspectos comuns para reutilização, capturando as melhores práticas e promovendo a interoperabilidade.

Antes: Aspecto `cuid`

Para chaves primárias UUID canônicas.

Conciso

```
cds entity Foo : cuid { ... }
```

Verboso

```
entity Foo {  
    key ID : UUID;  
    ...  
}
```

Aspecto `managed`

Para informações de auditoria (criado/modificado por/em).

Conciso

```
cds entity Foo : managed { ... }
```

Verboso

```
entity Foo {  
    createdAt : Timestamp @cds.on.insert : $now;  
    createdBy : User @cds.on.insert : $user;  
    modifiedAt : Timestamp @cds.on.insert : $now @cds.on.update : $now;  
    modifiedBy : User @cds.on.insert : $user @cds.on.update : $user;  
    ...  
}
```

Os runtimes preenchem esses campos automaticamente, simplificando a lógica da aplicação.

Acelerando com Tipos e Listas de Códigos Padronizados

`@sap/cds/common` fornece tipos de reutilização para `Country`, `Currency` e `Language`. Estes são definidos como associações a entidades de 'lista de códigos' (Code List), que atuam como tabelas de referência com códigos únicos e textos localizáveis.

1. Aspecto Base: `CodeList` →
2. Entidade Concreta: →
3. Tipo de Reutilização: →
4. Uso Final no Modelo de Domínio

```
aspect sap.common.CodeList {  
    name : localized String(111);  
    descr : localized String(1111);  
}
```

```
entity sap.common.Countries : CodeList {  
    key code : String(3);  
}
```

```
type Country : Association to  
sap.common.Countries;
```

```
using { Country } from '@sap/cds/common';  
  
entity Addresses {  
    street : String;  
    country : Country; // Simples e poderoso  
}
```

Adicionando Metadados e Comportamento com Anotações

Anotações, prefixadas com `@`, adicionam informações personalizadas às definições. Elas são usadas por compiladores e runtimes para diversos fins, como gerar UIs, validar entradas ou controlar o comportamento do serviço.

Categoria	Anotação de Exemplo	Propósito
Interface do Usuário (UI)	@title: 'Título do Livro'	Fornece um rótulo legível por humanos para um campo em uma UI Fiori. Equivalente a `@Common.Label`.
Validação de Entrada	@readonly: true @mandatory: true	Torna um campo somente leitura ou de preenchimento obrigatório.
Comportamento do Serviço	@odata.draft.enabled	Habilita o suporte a rascunho (draft) para uma entidade em um serviço OData.
Persistência	@cds.on.insert: \$now	Define que o campo deve ser preenchido com o timestamp atual (`\$now`) durante uma operação de inserção.



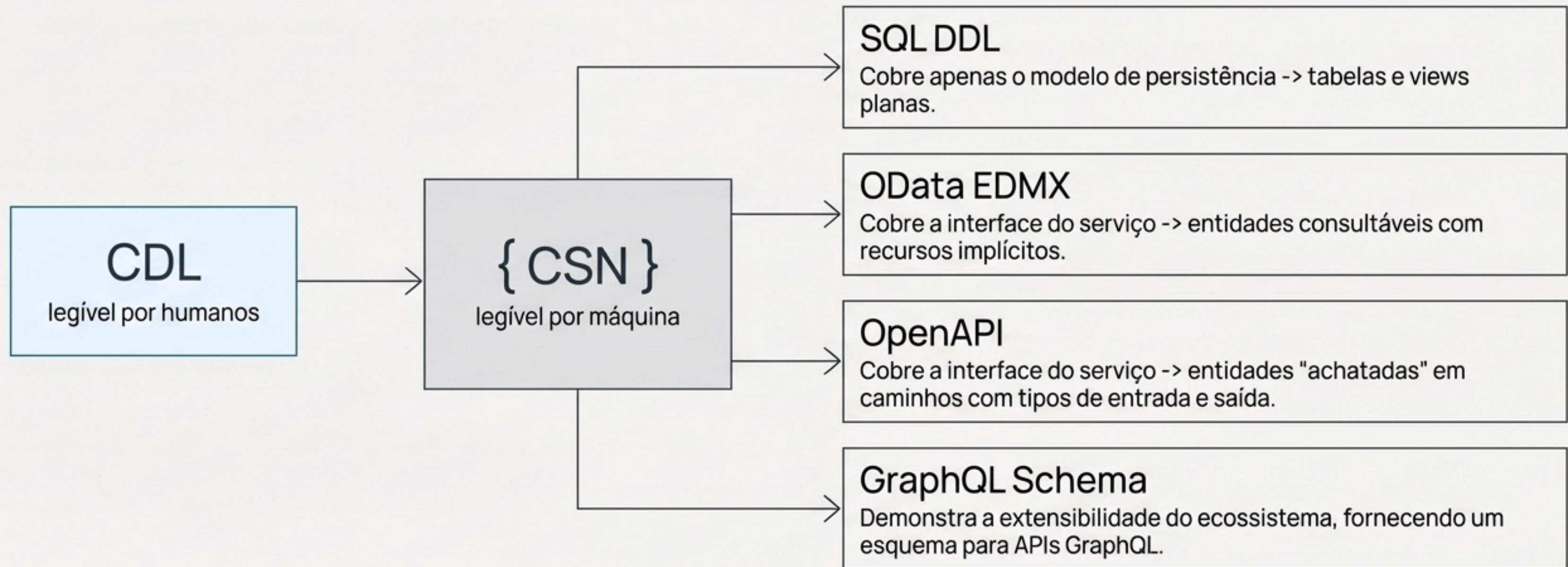
Dica Pro

Use a diretiva `annotate` para aplicar anotações a definições existentes em arquivos separados, mantendo seu modelo de domínio principal limpo e focado.

```
// Em um arquivo fiori-annotations.cds
using { CatalogService } from '../srv/cat-service';
annotate CatalogService.Books with @(
    UI.LineItem: [ {Value: title} ]
);
```

O Ciclo Completo: Do CDL a Múltiplas "Reflexões"

Um modelo CDS, em sua forma canônica CSN, é a representação central. A partir dele, o toolkit do CDS gera diferentes "reflexões" - representações que cobrem aspectos específicos da aplicação, como a interface de persistência ou a API de serviço, com alguma perda de informação, mas perfeitamente adaptadas ao seu propósito.



CDS é um Paradigma Completo para o Desenvolvimento de Aplicações

A jornada 'Do Projeto à Realidade' revela que o CDS é mais do que apenas uma coleção de linguagens. É um paradigma integrado que combina:

1. Definição Declarativa



Usando **CDL**, os desenvolvedores descrevem o *quê*, não o *como*. Isso resulta em modelos de domínio concisos e compreensíveis que formam o blueprint da aplicação.

2. Representação Canônica



O **CSN** atua como uma linguagem intermediária universal, a fonte única da verdade que desacopla os modelos de qualquer sintaxe de origem ou formato de destino específico.

3. Modularidade e Reutilização



Através de **Aspectos e Anotações**, o CDS promove a separação de preocupações, permitindo que funcionalidades transversais (como auditoria, UI, autorização) sejam aplicadas de forma limpa e reutilizável.

Ao adotar o CDS, as equipes constroem aplicações com mais rapidez, consistência e com um design fundamentalmente mais sólido e adaptável.