

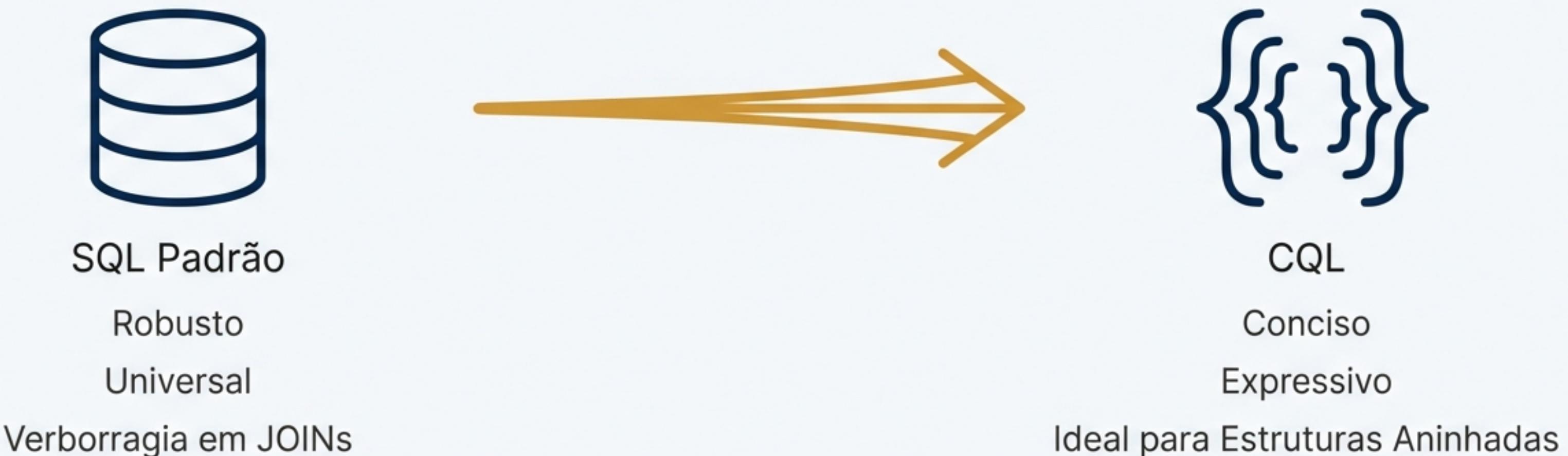
Dominando a CDS Query Language (CQL)

Uma jornada da sintaxe SQL a recursos expressivos para dados modernos.



A Evolução da Consulta: Do SQL Padrão à Expressividade da CQL

O SQL é a base universal, mas as aplicações modernas exigem mais. A CQL estende o SQL para lidar de forma nativa com estruturas de dados complexas e associações, resultando em um código mais limpo e poderoso.



O Primeiro 'Power-Up': A Inversão com Projeções Pós-fixadas

A CQL permite que a cláusula `SELECT` (a projeção) venha **após** a cláusula `FROM`, envolvida por chaves `{}`. Isso abre a porta para uma manipulação de dados muito mais intuitiva.

SQL Padrão

```
```sql
SELECT name, address.street
from Authors
```
```

CQL Elegante

```
```sql
SELECT
from Authors {
 name, address.street
}
```
```

Expandindo Horizontes: Consultando Estruturas Profundas com `Nested Expands`

Navegue por associações para criar documentos JSON aninhados diretamente na consulta. Uma funcionalidade poderosa, especialmente para bancos de dados NoSQL, sem equivalente direto em SQL padrão.

Código CQL

```
SELECT from Authors {  
    name,  
    address {  
        street,  
        town {  
            name  
        }  
    }  
};
```



Resultado JSON

```
[  
  {  
    "name": "Victor Hugo",  
    "address": {  
      "street": "6 Place des Vosges",  
      "town": {  
        "name": "Paris"  
      }  
    }  
  }, ...  
]
```

Atenção: `Nested Expands` em associações 'to-many' não são suportados.

Moldando Seus Dados: Aliases e Expressões

Use `as` para renomear campos ou associações. Crie novas estruturas com campos calculados, onde um alias para a nova estrutura é mandatório e posicionado *após* as chaves `{...}`.

```
SELECT from Books {  
    title,  
    author as creator { name },  
    { stock, stock * price as value } as inventory  
};
```

Renomeando uma associação

Criando uma nova estrutura com um campo calculado

A consulta não apenas busca, mas também **molda** os dados no formato exato que a aplicação precisa.

Menos Digitação, Mais Clareza com `Nested Inlines`

Coloque um `.` antes da chave de abertura para 'achatar' (inline) os elementos do alvo, evitando listas extensas de caminhos para ler múltiplos elementos do mesmo alvo.

Sem Inline (mais verboso)

```
SELECT from Authors {  
    name,  
    address.street,  
    address.town.name  
};
```

Com Inline (conciso e claro)

```
SELECT from Authors {  
    name,  
    address.{ street, town.{ name } }  
};
```

O Seletor Inteligente: O Poder Reimaginado do `*`

Em CQL, colunas definidas explicitamente após um `*` substituem colunas de mesmo nome que foram inferidas anteriormente pelo `*`. Isso evita a duplicação de colunas comum em `JOIN`'s SQL.

```
SELECT from Books {  
    *, author.name as author  
}
```

Explicação: Neste caso, o campo `author` (uma string com o nome) substitui a associação `author` que seria incluída pelo `*`, resultando em uma estrutura de dados mais limpa.

[ID,
title,
price,
~~author_assoc~~
→ author_string

O Poder da Exclusão: Selecione Tudo, *Exceto*...

Use a cláusula `excluding` em combinação com `SELECT *` para selecionar todos os elementos, exceto aqueles listados na lista de exclusão. Flexibilidade sem precedentes.

```
SELECT from Books  
{ * }  
excluding { author, price }
```

A cláusula `excluding` também pode ser usada dentro de `Nested Expands` e `Nested Inlines`.

Obtenha todos os campos de `Books`, mas omita a associação `author` e o campo `price`. Simples e direto.

Navegação Universal: A Versatilidade dos `Path Expressions`

`Path Expressions` (ex: `author.address.town.name`) não se limitam ao `SELECT`. Use-os de forma intuitiva nas cláusulas `FROM`, `WHERE` e em expressões.

| Cláusula | Exemplo CQL | Desdobramento em SQL |
|---------------|-----------------------------|------------------------|
| FROM | Authors[name='...'].books | SEMI JOIN |
| WHERE | ...where author.name='...' | LEFT OUTER JOIN |
| SELECT | SELECT author.name from ... | LEFT OUTER JOIN |

Nota Importante: Todas as referências de coluna são qualificadas, eliminando o risco de nomes ambíguos ou conflitantes comuns em `JOIN`s SQL.

Filtragem Cirúrgica com 'Infix Filters'

Aplique filtros diretamente no meio de um `Path Expression` para refinar a condição do `JOIN`, em vez de filtrar o resultado final.

Código CQL

```
SELECT books[genre='Mystery'].title  
from Authors  
WHERE name='Agatha Christie'
```

SQL Gerado

```
SELECT books.title from Authors  
LEFT JOIN Books books ON ( books.author_ID = Authors.ID )  
AND ( books.genre = 'Mystery' ) ← //--> do Infix Filter  
WHERE Authors.name='Agatha Christie';
```

Conceito Relacionado

Use o predicado `EXISTS` para testar se algum elemento da coleção associada corresponde a um filtro.
Ex: `WHERE EXISTS books[year = 2000]`.

Técnicas Avançadas: Garantindo Clareza com Tipagem e Enums

Casts em Estilo CDL

Casts em estilo CDL (`:Decimal`) não geram um `CAST` no SQL final, mas ajudam o compilador e frameworks (como OData) a deduzir tipos, especialmente em expressões como `foo+1`.

```
SELECT from Foo { foo+1 as bar : Decimal };
```

Uso de Enums

Em consultas, use símbolos de enum (#symbol`) em vez de literais de string. Isso torna o código mais legível e menos propenso a erros de digitação.

```
... where status = #in_progress;
```

Técnicas Avançadas: Criando Associações Dinâmicas

Defina associações 'unmanaged' em tempo de execução com `mixin...into` para adicionar logicamente relações a uma fonte de dados e usá-las imediatamente na mesma consulta.

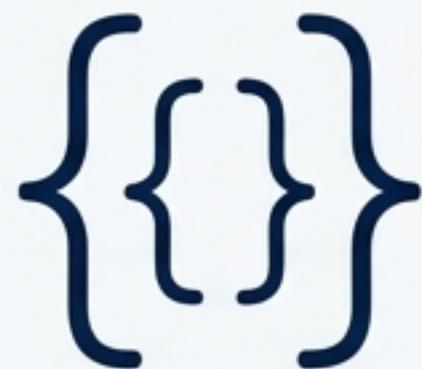
```
SELECT from Books
  mixin {
    localized : Association to LocalizedBooks on localized.ID = ID;
  }
  into {
    ID, localized.title
  };

```

****Texto de Apoio:****

"Você também pode definir associações diretamente na lista 'SELECT', úteis para projeções e views."

Sua Nova Caixa de Ferramentas para Consultas



Expressividade

Crie resultados aninhados complexos de forma intuitiva (Nested Expands).



Inteligência

Use recursos como o * inteligente e a cláusula excluding para evitar código repetitivo.



Concisão

Escreva menos código e faça mais (Nested Inlines, Path Expressions).



Flexibilidade

Molde seus dados e até mesmo as associações em tempo de execução (Aliases, Mixins).

CQL: A Evolução do SQL. Projetada para o Desenvolvedor Moderno.

A CQL não substitui o SQL; ela o eleva. Ao adotar esses padrões, você escreve consultas mais limpas, mais legíveis e mais alinhadas com a natureza estruturada dos dados nas aplicações de hoje.

Continue Explorando



Documentação Oficial da CDS Query Language (CQL)

A referência completa para sintaxe, funcionalidades e exemplos de uso da CQL.



Documentação Relacionada: Schema Notation (CSN)

Entenda como os esquemas de dados são definidos, a base para consultas CQL.



Documentação Relacionada: Query Notation (CQN)

Explore a representação de objeto de consulta intermediária usada pelo CAP.