

# API

- [utRequest](#)
- [acuContainers](#)
- [Configuração de Serviço](#)
- [Arquivo de Configuração](#)
- [Manual OQL](#)

# utRequest

utRequest

# acuContainers

# Configuração de Serviço

Para configurar um serviço dentro do ambiente em que o Curió está sendo utilizado, deve primeiramente obter o projeto com o qual você irá trabalhar, obter os código fontes do controle de versão. Neste tutorial utilizaremos o Prestamista como exemplo para configurar o serviço.

1. Criar uma pasta onde irá salvar todos os serviços que irão ser gerados e instalados na máquina. EX: "C:\Users\brunoagrizzi\Documents\SistemasEvológica". Este diretório irá conter os serviços ".exe" além de seus arquivos de configurações respectivos.
2. Ao abrir o Rad Studio (plataforma de desenvolvimento do Delphi), abra o projeto ".dproj" ou ".dpr" do projeto do serviço. Em "Project/Options", após abrir a janela ir em "Directories/Conditionals" e adicionar o caminho que vc criou no passo 1 no campo Output Directory. E no Search Path adicionar o caminho das units do Framework, com o final ...\srv.

# Arquivo de Configuração

O arquivo de configuração é um arquivo XML com o nome igual ao do serviço com ".config" como extensão do arquivo e que precisa estar na mesma pasta onde está instalado o serviço. Abaixo temos um exemplo de um arquivo de configuração:

```
<?xml version="1.0" encoding="UTF-8"?>
<Config>  <Application DefaultSecurityDriver="ConexoSecurityDriver" Timeout="320"
EnabledChannels="AGENCIA,BANSEG,CORRETOR">      <Service Name="svportal" DisplayName="CONEXO -
PORTAL" Description="Portal de Sistemas" Port="10001" />      <Database
ActiveConnectionString="SGS">          <Local>Provider=SQLOLEDB.1;Persist Security Info=True;User
ID=sa;;Initial Catalog=bd_vgbl;Data Source=localhost;Application
Name=Patrimonial;cxsqldialect=SQLServer</Local>          <SGS>Provider=SQLOLEDB.1;Persist
Security Info=True;User ID=sa;Initial Catalog=BD_SGS;Data Source=srplvd1;Application
Name=Portal</SGS>
          </Database>      <Options SessionTimeout="3660" AllowPasswordChange="1"
TestEnvironment="2" URL="" />      <SegurancaCorretorBanseg
ConnectionString="Provider=SQLOLEDB.1;Password=conexo;Persist Security Info=True;Data
Source=srplvd1;Application Name=Portal;cxsqldialect=SQLServer" />      <LDAP
Host="srpl.conexops.com.br" Port="389" UserNameTemplate="%username%@conexops.com.br" />
<UseXMLEncode version="1" />
    </Application>
    <Systems>      <System Code="50" Name="Patrimonial" ServerName="localhost" ServicePort="5361"
ClientBPLName="patrimonial_acessoexterno.bpl" EnabledChannels="CORRETOR,AGENCIA" />      <System
Code="53" Name="Automóvel" ServerName="localhost" ServicePort="5563"
ClientBPLName="automovel.bpl" EnabledChannels="BANSEG,CORRETOR,AGENCIA" />      <System
Code="50" Name="VGBL" ServerName="localhost" ServicePort="5355" ClientBPLName="vgbl_cli.bpl"
EnabledChannels="AGENCIA,BANSEG" />
    </Systems></Config>
```

Existem duas tags principais: *Application* e *Systems*. A primeira tem várias "subtags" para configurar

o serviço em si e a segunda tem a lista de serviços que são disponibilizados aos clientes.

## Tag "Application"

A tag “*Application*” engloba várias outras tags e ela possui os seguintes atributos:

| Atributo              | Função                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DefaultSecurityDriver | Informar qual é o driver de segurança que server utiliza. Pode assumir os seguintes <code>NativeSecurityDriver</code> , <code>ConexoSecurityDr</code><br><code>BanestesLDAPSecurityDriver</code> ,<br><code>CorretorBansegSecurityDriver</code> .                                                                                                                                                                                                      |
| Timeout               | Alterar o timeout (em segundos) de ses aplicação. O valor padrão é 1800 (30 minu cliente ultrapassar este tempo sem enviar mensagem, a conexão é terminad                                                                                                                                                                                                                                                                                              |
| EnabledChannels       | É utilizado para habilitar o acesso a aplicaç para clientes de determinados canais, os habilitados devem estar separados por vírg espaço). O nome do canal é case sensitive. informado, apenas serão aceitos clientes especificarem o canal ao abrir sessão. Se il apenas clientes que especificarem o cana sessão presentes na lista de canais habilita aceitos. Utilize o valor “All” para permitir todos os clientes que especificarem um qualquer. |

## Tag "Options"

A tag “Options” fica dentro da tag “Application” e possui os seguintes atributos:

| Atributo            | Função                                                                                                                                                                                                                                                   |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AllowPasswordChange | Informar se é oferecida ao usuário a opção de troca de senha dentro do portal cliente. O valor zero indica que não é permitido, valores diferente de zero indicam que é permitida a troca de senha.                                                      |
| SessionTimeout      | Tempo de sessão que será mostrado no portal cliente. Se não for informado o portal cliente não mostrará o tempo restante da sessão. Deve ser configurado com um tempo menor ou igual ao tempo de sessão do servidor do portal.                           |
| TestEnvironment     | Indica qual é o ambiente que o portal estará atendendo. Pode assumir os seguintes valores:<br>0 – Produção<br>1 – Homologação<br>2 – Desenvolvimento<br>Se esse atributo não for informado, o portal assume que está executando no ambiente de produção. |
| URL                 | Indica qual página da internet será exibida no portal cliente quando nenhum sistema estiver em execução. Se não for informada, nenhuma página da internet será exibida.                                                                                  |

Tag "LDAP"

A tag “LDAP” fica dentro da tag “Application”. Ela só é necessária quando o driver de segurança for o BanestesLDAPSecurityDriver, e possui os seguintes atributos:

| Atributo         | Função                                                                                                                                                                   |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Host             | Informar qual é o servidor LDAP que será para autenticar o usuário.                                                                                                      |
| Port             | Informar qual é a porta do servidor LDAP utilizado para autenticar o usuário.                                                                                            |
| UserNameTemplate | Indica como deve ser formatado o nome de usuário informado no login para autenticação no LDAP. A palavra chave %username% é substituída pelo usuário informado no login. |

## Tag "SegurancaCorretorBanseg"

A tag “SegurancaCorretorBanseg” fica dentro da tag “Application”. Ela só é necessária quando o driver de segurança for o CorretorBansegSecurityDriver, e possui os seguintes atributos:

| Atributo         | Função                                                                                                                                                                   |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ConnectionString | Informar qual é a connection string para conexão com o banco de dados que é utilizado pelo sistema do corretor e que possui o cadastro dos usuários da sala do corretor. |

## Tag "Service"

A tag “Service” fica dentro da tag “Application”. Ela é obrigatória e possui os seguintes atributos:

| Atributo | Função |
|----------|--------|
|----------|--------|

|             |                                                                                                                                                                                     |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Name        | Especifica um nome único do serviço para o sistema. Esse nome é utilizado para iniciar e parar o serviço utilizando o comando do Windows “net start” e “net stop”, respectivamente. |
| DisplayName | Especifica o nome estendido do serviço que será mostrado na console de serviços do Windows.                                                                                         |
| Description | Especifica a descrição do serviço que será mostrada na console de serviços do Windows.                                                                                              |
| Port        | Especifica a porta TCP/IP em que o servidor de uso irá aceitar conexões de clientes, se não podem existir duas instâncias de servidor usando uma mesma porta.                       |

## Tag "DataBase"

A tag “Database” fica dentro da tag “Application”. Ela é obrigatória e possui uma ou várias tags filhas que possuem uma string válida de conexão ao banco de dados.

| Atributo               | Função                                                                            |
|------------------------|-----------------------------------------------------------------------------------|
| ActiveConnectionString | Indica qual tag filha será utilizada para efetuar a conexão com o banco de dados. |

## Tag "UseXMLEncode"

A tag “UseXMLEncode” fica dentro da tag “Application”. Ela é obrigatória e possui o seguinte atributo:

| Atributo | Função                                                                                                    |
|----------|-----------------------------------------------------------------------------------------------------------|
| Version  | Esse atributo indica qual versão de codificação XML é utilizada, atualmente apenas a versão 1.0 é aceita. |

# Tag "System"

A tag “Systems” engloba uma ou mais tags filhas “System”. Cada tag “System” é um sistema que é disponibilizado ao usuário, e possui os seguintes atributos:

| Atributo        | Função                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Code            | Especifica o código do sistema que será executado.                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Name            | Especifica o nome que aparece na drop-down de sistemas no portal cliente.                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| ServerName      | Especifica o nome do servidor onde o serviço será executado.                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| ServicePort     | Especifica a porta do servidor onde o serviço será executado.                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| ClientBPLName   | Informa o nome do arquivo BPL que implementa o sistema no cliente para o sistema.                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| EnabledChannels | É utilizado para habilitar o acesso a aplicação para clientes de determinados canais, os canais habilitados devem estar separados por vírgula e espaço). O nome do canal é case sensitive. Se não for informado, apenas serão aceitos clientes que não especificarem o canal ao abrir sessão. Se informado, apenas clientes que especificarem o canal e que estejam presentes na lista de canais habilitados serão aceitos. Utilize o valor “All” para permitir acesso a todos os clientes que especificarem um canal qualquer. |



# Manual OQL

## Introdução

A OQL (Object Query Language) é uma linguagem de pesquisa declarativa, semelhante ao SQL, mas com suporte para objetos. Em OQL, consultas (queries) são especificadas utilizando classes e seus respectivos atributos. Os relacionamentos em um modelo de objetos podem ainda ser percorridos utilizando-se o mesmo paradigma classe/membro, usado pelas linguagens de programação orientadas a objetos.

Nenhum conhecimento sobre o mapeamento de objetos para o modelo relacional é necessário ao desenvolvedor para a especificação de uma consulta OQL. Tais relações de mapeamento são inferidas a partir da camada de persistência do framework para desenvolvimento de aplicações da Conexo, em tempo de execução da consulta.

## Consultas Simples

Em OQL, os construtores básicos para a especificação de consultas são os mesmos utilizados em SQL, isto é, as cláusulas SELECT, FROM, WHERE, GROUP BY, HAVING e ORDER BY. O resultado de uma consulta é, tipicamente, uma coleção de objetos. A consulta a seguir retorna uma coleção polimórfica de elementos do tipo `NGAluno` (classe abstrata), cujo período de ingresso é maior que 2009/01. Tais elementos serão instâncias das classes concretas `NGAlunoGraduacao`, `NGAlunoMestrado`, etc. e estarão ordenados pelo número de matrícula.

```
SELECT AL
FROM NGAluno AL
WHERE AL.PeriodoIngresso >= 200901 ORDER BY AL.NumMatricula
```

Note que a classe `NGAluno` é renomeada para “AL” na cláusula FROM. A renomeação de classes é

opcional, entretanto, se uma classe for renomeada, seu alias deverá ser sempre utilizado no restante da consulta. Tanto “NGAluno as AL” quanto “AL in NGAluno” são formas alternativas e equivalentes de renomeação da classe NGAluno. Expressões de caminho são utilizadas para navegar pela estrutura dos objetos. Uma expressão de caminho (path expression) é composta de: nome de uma classe, ou seu respectivo alias, seguido por zero ou mais nomes de atributos ou de relacionamentos, separados por ponto.

## Junções Cruzadas (Cross Join)

Junções cruzadas, também chamadas de produtos cartesianos, ocorrem sempre que mais de uma classe é especificada na cláusula FROM. A consulta a seguir retorna uma coleção de pessoas que possuem nome de rua.

```
SELECT NGPessoa
FROM NGPessoa, NGEnderecoWHERE NGPessoa.Nome like NGEndereco.Rua
```

Observe que, para a consulta em OQL especificada acima, a seguinte consulta SQL é gerada:

```
SELECT P.*
FROM [Pessoa] AS P
CROSS JOIN [Endereco] AS EWHERE P.Nome like E.Rua
```

## Junções Internas (Inner Join)

Expressões de caminho explicitamente especificadas na cláusula FROM são tratadas como junções internas. A consulta a seguir retorna uma coleção de pessoas (NGPessoa) que moram na cidade de Vitória.

```
SELECT P
FROM NGPessoa P, P.Endereco, P.Endereco.CidadeWHERE P.Endereco.Cidade.Nome like 'Vitória'
```

Observe que, para a consulta em OQL especificada acima, a seguinte consulta SQL é gerada:

```
SELECT P.*  
FROM [Pessoa] AS P  
JOIN [Endereco] AS E ON (P.Endereco0ID = E.0ID)  
JOIN [Cidade] AS C ON (E.Cidade0ID = C.0ID)WHERE C.Nome like 'Vitória'
```

## Junções Externas à Esquerda (Left Outer Join)

Expressões de caminho não especificadas explicitamente na cláusula FROM são tratadas como junções externas à esquerda. A consulta a seguir retorna uma coleção de pessoas (NGPessoa) que, ou não possuem endereço, ou sejam moradores de Vitória.

Note que as expressões de caminho P.Endereco e P.Endereco.Cidade (utilizadas na cláusula WHERE) não foram especificadas na cláusula FROM.

```
SELECT P  
FROM NGPessoa P  
WHERE P.Endereco.0ID IS NULLOR P.Endereco.Cidade.Nome like 'Vitória'
```

Observe que, para a consulta em OQL especificada acima, a seguinte consulta SQL é gerada:

```
SELECT P.*  
FROM [Pessoa] AS P  
LEFT JOIN [Endereco] AS E ON (P.Endereco0ID = E.0ID)LEFT JOIN [Cidade] AS C ON (E.Cidade0ID =  
C.0ID)  
WHERE E.0ID IS NULLOR C.Nome like 'Vitória'
```

## Consultas Envolvendo Agrupamento

A cláusula GROUP BY é usada para agrupar (ou agregar) os dados retornados por uma consulta segundo critérios específicos escolhidos pelo utilizador. Quando o retorno da consulta for uma coleção de objetos, o critério de agrupamento poderá envolver apenas classes e expressões de caminho que façam parte do escopo da consulta. No caso de a consulta retornar expressões escalares, o critério de agrupamento deverá utilizar apenas atributos de objetos ou expressões escalares válidos no escopo corrente da consulta. Normalmente a cláusula GROUP BY é utilizada em

conjunto com as funções de agregação (ver tópico Funções de Agregação).

Em consultas envolvendo agrupamento, a cláusula SELECT não pode fazer referência direta a qualquer classe ou atributo que não esteja presente na cláusula GROUP BY. Entretanto, pode fazer referência a constantes e funções de agregação. A cláusula HAVING é utilizada para restringir (ou filtrar) os resultados agregados pela cláusula GROUP BY. A cláusula HAVING é aplicada a cada conjunto agrupado, de forma parecida como a cláusula WHERE é aplicada a cada item retornado pela cláusula FROM. Se não houver uma cláusula GROUP BY, a cláusula HAVING será aplicada a todo o resultado como um único grupo.

A consulta a seguir retorna uma coleção de alunos (NGAluno) que possuem média de notas maior ou igual a oito.

```
SELECT NGAluno
FROM NGAluno
GROUP BY NGAlunoHAVING AVG(NGAluno.Historico.Nota) >= 8.0
```

Já a consulta a seguir retorna um subconjunto de tuplas contendo o número de matrícula e a respectiva média das notas dos alunos que possuem média maior ou igual a oito.

```
SELECT NGAluno.NumeroMatricula, AVG(NGAluno.Historico.Nota) AS Media
FROM NGAluno
GROUP BY NGAluno.NumeroMatriculaHAVING AVG(NGAluno.Historico.Nota) >= 8.0
```

## Consultas Parametrizadas

Sempre que houver necessidade de incluir alguma parte dinâmica em uma consulta OQL, deve-se optar pelo uso de parâmetros. Parâmetros são variáveis que podem ser declaradas na cláusula WHERE e que serão substituídas por seus respectivos valores em tempo de execução da aplicação. Parâmetros são declarados em uma consulta OQL com o caractere de ponto de interrogação, seguido pelo nome da variável, seguido de dois pontos, seguido pelo seu respectivo tipo.

Por exemplo, a seguinte consulta retorna os alunos cujo período de ingresso é maior ou igual ao valor

parâmetro “NumMatricula”, do tipo “AcInt”:

```
SELECT AL
FROM NGAuno ALWHERE AL.NumeroMatricula >= ?NumMatricula:AcInt
```

## Funções Escalares

Em OQL, especialmente pelo fato de tratar-se de uma linguagem não procedimental, funções constituem um recurso importante. O cálculo da raiz quadrada de um número, por exemplo, é utilizado com certa frequência. Entretanto, seria extremamente trabalhoso para o desenvolvedor construir esse algoritmo toda vez que precisasse utilizá-lo em alguma consulta.

As funções escalares, apresentadas nesta seção, operam sobre valores simples. Uma função que calcula a raiz quadrada de um número, por exemplo, é escalar, pois é aplicada sobre um valor simples (uma expressão numérica). Atualmente, as seguintes funções escalares são suportadas:

| Função             | Descrição                                                                                  | Exemplo                             |
|--------------------|--------------------------------------------------------------------------------------------|-------------------------------------|
| CAST(expr AS tipo) | Converte o tipo de dado de uma expressão de entrada para o tipo informado.                 | CAST('256' as AcInt) :              |
| ABS(expr)          | Retorna o valor absoluto de uma expressão.                                                 | ABS(-123) = 123<br>ABS(123) = 123   |
| CEIL(expr)         | Retorna o menor inteiro que seja maior ou igual ao valor da expressão numérica de entrada. | CEIL(1.23) = 2<br>CEIL(-1.23) = -1  |
| FLOOR(expr)        | Retorna o maior inteiro que seja menor ou igual ao valor da expressão numérica de entrada. | FLOOR(1.23) = 1<br>FLOOR(-1.23) = - |

|                       |                                                                                                                                                            |                                                                                                     |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------|
| EXP(expr)             | Retorna o valor exponencial da expressão numérica de entrada. Descrita como $e^{expr}$ (onde $e$ é a constante matemática neperiana).                      | $\text{Exp}(1) = 2,718281$                                                                          |
| LN(expr)              | Retorna o logaritmo natural da expressão numérica de entrada. É, portanto, a função inversa da função exponencial.                                         | $\text{LN}(2) = 0.693147$                                                                           |
| SQRT(expr)            | Retorna a raiz quadrada da expressão numérica de entrada.                                                                                                  | $\text{SQRT}(9) = 3$                                                                                |
| MOD(expr, divisor)    | Retorna o resto da divisão da expressão numérica de entrada pelo divisor.                                                                                  | $\text{MOD}(5,2) = 1$                                                                               |
| POWER(expr, n)        | Retorna o valor da expressão de numérica de entrada elevado à $n$ ésima potência.                                                                          | $\text{POWER}(2,8) = 25$                                                                            |
| ROUND(expr, precisão) | Arredonda o valor da expressão numérica fornecida de acordo com a precisão informada. Se o segundo parâmetro for omitido, a precisão 0 (zero) é utilizada. | $\text{ROUND}(256.9989,3) = 2$<br>$\text{ROUND}(256.9994,-1) = 2$<br>$\text{ROUND}(256.9995,0) = 2$ |

|                                      |                                                                                                                                                                                                     |                                                 |
|--------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------|
| TRUNCATE(expr, precisão)             | Trunca o valor da expressão numérica fornecida de acordo com a precisão informada.                                                                                                                  | TRUNCATE(256.986,2) =<br>TRUNCATE(256.995,0) =  |
| LOWER(expr)                          | Converte a cadeia de caracteres definida pela expressão de entrada em letras minúsculas.                                                                                                            | LOWER('CONEXO') = 'c                            |
| UPPER(expr)                          | Converte a cadeia de caracteres definida pela expressão de entrada em letras maiúsculas.                                                                                                            | UPPER('Conexo') = 'CC                           |
| TRIM(expr)                           | Remove caracteres em branco das extremidades da cadeia de caracteres definida pela expressão de entrada.                                                                                            | TRIM(' Conexo ') = 'Cc                          |
| LENGTH(expr)                         | Retorna o número de caracteres presentes na cadeia de caracteres definida pela expressão de entrada.                                                                                                | LENGTH('CONEXO')                                |
| SUBSTRING(expr, origem, comprimento) | Extraí uma subcadeia de caracteres da cadeia de caracteres definida pela expressão de entrada. Outros parâmetros desta função representam a posição inicial e o comprimento da subcadeia retornada. | SUBSTRING('Conexo Pro<br>Sistemas', 1, 6) = 'Co |

|                                                               |                                                                                                                           |                                                           |
|---------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------|
| REPLACE(cadeia_original,<br>cadeia_origem,<br>cadeia_destino) | Substitui todas as ocorrências da cadeia de caracteres de origem pela cadeia de caracteres de destino na cadeia original. | REPLACE('Conexo', 'Con' = 'Complexo')                     |
| DAY(expr)                                                     | Retorna o dia de uma expressão de entrada do tipo DateTime.                                                               | DAY('10/04/2010 : 13:30:45') = 10                         |
| MONTH(expr)                                                   | Retorna o mês de uma expressão de entrada do tipo DateTime.                                                               | MONTH('10/04/2010 : 13:30:45') = 4                        |
| YEAR(expr)                                                    | Retorna o ano de uma expressão de entrada do tipo DateTime.                                                               | YEAR('10/04/2010 : 13:30:45') = 2010                      |
| HOUR(expr)                                                    | Retorna a hora de uma expressão de entrada do tipo DateTime.                                                              | HOUR('10/04/2010 : 13:30:45') = 13                        |
| MINUTE(expr)                                                  | Retorna os minutos de uma expressão de entrada do tipo DateTime.                                                          | MINUTE('10/04/2010 : 13:30:45') = 30                      |
| SECOND(expr)                                                  | Retorna os segundos de uma expressão de entrada do tipo DateTime.                                                         | SECOND('10/04/2010 : 13:30:45') = 45                      |
| TRUNCATE(expr)                                                | Remove os componentes hora, minuto e segundo de uma expressão de entrada do tipo DateTime.                                | TRUNCATE('10/04/2010 : 13:30:45') = '10/04/2010 00:00:00' |

# Expressões Condicionais

Esta seção descreve as expressões condicionais atualmente disponíveis no OQL.

## CASE

A expressão CASE do OQL é uma expressão condicional genérica, semelhante às declarações IF-THEN-ELSE das linguagens procedimentais:

```
CASE WHEN condição THEN resultado  
[WHEN ...]  
[ELSE resultado]END
```

A cláusula CASE pode ser usada em qualquer lugar onde uma expressão for válida. Entretanto, o tipo de dado de todas as expressões resultado deve ser o mesmo. A condição é uma expressão que retorna um resultado booleano. Se o resultado for verdade, então o valor da expressão CASE é o resultado que segue a condição. Se o resultado for falso, todas as cláusulas WHEN seguintes são analisadas da mesma maneira. Se o resultado de nenhuma condição WHEN for verdade, então o valor da expressão CASE é o valor do resultado na cláusula ELSE. Se a cláusula ELSE for omitida, e nenhuma condição for satisfeita, o resultado será nulo.

A expressão CASE "simplificada", mostrada abaixo, é uma variante especializada da forma geral mostrada acima:

```
CASE expressão  
WHEN valor THEN resultado  
[WHEN ...]  
[ELSE resultado]END
```

A expressão é computada e comparada com todas as especificações de valor nas cláusulas WHEN, até encontrar um que seja igual. Se não for encontrado nenhum valor igual, é retornado o resultado na cláusula ELSE (ou o valor nulo).

## COALESCE

A função COALESCE retorna o primeiro de seus argumentos que não for nulo. Só retorna nulo quando todos os seus argumentos são nulos. Assim como ocorre nas expressões CASE, o tipo de dado de todas as expressões resultado (argumentos) da função COALESCE deve ser o mesmo.

```
COALESCE(valor [, ...])
```

Geralmente o COALESCE é útil para substituir o valor padrão quando este é o valor nulo, quando os dados são usados para exibição. Por exemplo:

```
COALESCE(?Descricao:AcString, ?Descricao_curta:AcString, 'nenhuma')
```

## NULLIF

A função NULLIF retorna o valor nulo se, e somente se, os valores das duas expressões de entrada forem iguais. Caso contrário, retorna o valor da primeira expressão.

```
NULLIF(Expressao1, Expressao2)
```

Pode ser utilizada para realizar a operação inversa do exemplo para COALESCE mostrado anteriormente:

```
NULLIF(?Descricao:AcString, 'nenhuma')
```

## Consultas Envolvendo Subconsultas

Assim como em SQL, uma consulta em OQL pode fazer referência ao resultado de outras consultas nela embutidas. Esse tipo de construção permite a elaboração de consultas mais sofisticadas. No exemplo a seguir, a consulta retorna uma coleção de pessoas que não sejam as de maior idade, ou seja, cuja data de nascimento é menor que alguma das datas de

nascimento retornadas pela subconsulta.

```
SELECT P
FROM NGPessoa P
WHERE P.DataNascimento < SOME (SELECT PE.DataNascimentoFROM NGPessoa PE)
```

*OBS: Subconsultas embutidas na cláusula WHERE podem retornar apenas literais (ex: números, strings, etc.) ou coleções de literais.*

## Funções de Agregação

Enquanto as funções escalares operam sobre uma única expressão, as funções de agregação operam sobre conjuntos de valores reduzindo-os a um único valor escalar. Tipicamente, funções de agregação são utilizadas para o cálculo do valor mínimo, do valor máximo, da soma e da média de uma expressão com relação a um conjunto de valores.

Atualmente, para uma consulta OQL, as seguintes funções de agregação são definidas:

| Função           | Descrição                                                                 |
|------------------|---------------------------------------------------------------------------|
| AVG(expressão)   | Retorna a média aritmética de todos os entrada.                           |
| COUNT(*)         | Retorna o número de valores de en                                         |
| COUNT(expressão) | Retorna o número de valores de entrada p o valor da expressão não é nulo. |
| MAX(expressão)   | Retorna o valor máximo da expressão ent valores de entrada.               |
| MIN(expressão)   | Retorna o valor mínimo da expressão ent valores de entrada.               |

|                |                                                           |
|----------------|-----------------------------------------------------------|
| SUM(expressão) | Retorna o somatório da expressão para valores de entrada. |
|----------------|-----------------------------------------------------------|

Por exemplo, a consulta a seguir retorna uma coleção de alunos que obtiveram nota máxima em alguma disciplina:

```
SELECT A
FROM NGAlunoGraduacao A, A.Historico H
WHERE H.Nota = (SELECT MAX(HI.Nota)
FROM NGHistorico HI WHERE HI.Disciplina.OID = H.Disciplina.OID)
```

## Operador [NOT] IN

Subconsultas em conjunto com o operador IN definem termos lógicos que podem ser verdadeiros ou falsos. O termo lógico `expressão IN (subconsulta)` é verdadeiro quando o valor representado por expressão está contido no resultado da subconsulta. Quando o termo lógico é utilizado na forma `expressão NOT IN (subconsulta)` tem-se o inverso: é verdadeiro quando o valor de expressão não está contido no resultado da subconsulta.

Por exemplo, a consulta a seguir retorna uma coleção de alunos que não estão matriculados em nenhuma disciplina.

```
SELECT A
FROM NGAluno A
WHERE A.OID NOT IN (SELECT M.Aluno.OID FROM NGMatricula M)
```

## Operador [NOT] EXISTS

A cláusula EXISTS permite testar se o resultado de uma consulta é vazio ou não. Um resultado vazio faz com que o termo lógico `EXISTS (subconsulta)` receba o valor lógico falso. Se o resultado da subconsulta não é vazio, o valor lógico é verdadeiro.

O teste pode ser invertido quando se utiliza a sintaxe `NOT EXISTS (subconsulta)` a consulta a seguir

retorna uma coleção de alunos que possuem histórico de nota registrado para alguma disciplina.

```
SELECT A
FROM NGAluno A
WHERE EXISTS (SELECT *
FROM NGHistorico HWHERE H.Aluno.OID = A.OID)
```

## Subconsultas na Cláusula FROM

Em todos os exemplos até aqui, somente Classes e expressões de caminho foram utilizadas na cláusula FROM. Entretanto, subconsultas também podem ser utilizadas, como mostra o exemplo abaixo:

```
SELECT AL
FROM (SELECT A
FROM NGAluno A
WHERE A.Pessoa.Sexo = 'F') AS AL, AL.CursoWHERE AL.Curso.Descricao like 'Direito'
```

O resultado desta consulta contém os alunos do sexo feminino do curso de direito. A coleção de alunos do sexo feminino (sobre a qual o alias AL foi definido) é filtrada pela descrição do curso e retornada como resultado da consulta.

*OBS: Subconsultas utilizadas na cláusula FROM podem retornar apenas coleções de objetos. Tais subconsultas devem ser obrigatoriamente renomeadas.*

## Operações de Conjunto

Os resultados de duas ou mais consultas podem ser combinados utilizando-se as operações de conjunto união, interseção e diferença. A sintaxe é:

```
consulta1 UNION [ALL] consulta2
consulta1 INTERSECT [ALL] consulta2consulta1 EXCEPT [ALL] consulta2
```

As consultas 1 e 2 devem ser válidas, podendo utilizar qualquer uma das funcionalidades mostradas

até aqui, e devem retornar objetos de uma mesma classe. As operações de conjuntos também podem ser aninhadas ou encadeadas. Por exemplo:

```
consulta1 UNION consulta2 UNION consulta3
```

Efetivamente, a operação de união (UNION) anexa o resultado da consulta2 ao resultado da consulta1 (embora não haja garantia que esta seja a ordem dos objetos retornados). Além disso, são eliminados do resultado os objetos duplicados, a não ser que seja utilizado UNION ALL.

A operação de interseção (INTERSECT) retorna todas as linhas presentes tanto no resultado da consulta1 quanto no resultado da consulta2. As linhas duplicadas são eliminadas, a não ser que seja utilizado INTERSECT ALL.

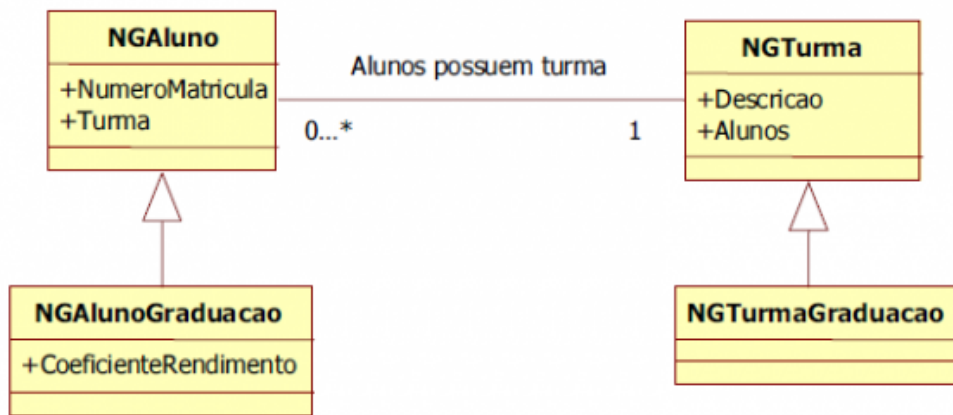
A operação de diferença (EXCEPT) retorna todas as linhas presentes no resultado da consulta1, mas que não estão presentes no resultado da consulta2. Novamente, as linhas duplicadas são eliminadas a não ser que seja utilizado EXCEPT ALL.

Para ser possível calcular a união, a interseção, ou a diferença entre duas consultas, as duas consultas devem ser "compatíveis", significando que ambas devem retornar objetos de uma mesma classe.

## Conversões Explícitas de Classe

Conversões explícitas de classe são utilizadas quando é necessário converter um objeto da classe base em um objeto da classe derivada. Isso acontece, por exemplo, quando na mesma consulta, precisamos percorrer um relacionamento definido entre classes base e acessar atributos definidos apenas para as classes derivadas.

Considere o diagrama de classes UML abaixo:



Imagine agora que desejamos consultar quais são os alunos de uma determinada turma de graduação que possuam o coeficiente de rendimento maior ou igual a sete. Poderíamos inicialmente pensar na seguinte consulta:

```
SELECT AL
FROM NGTurmaGraduacao TG, TG.Alunos AL
WHERE TG.OID = ?TurmaOID:AcIntAND AL.CoeficienteRendimento >= 7.0
```

Entretanto, na consulta acima, o caminho **AL.CoeficienteRendimento** é inválido. Observe que **TG.Alunos** retorna uma coleção de elementos do tipo **NGAluno**, pois esse relacionamento foi definido entre as classes base **NGTurma** e **NGAluno**.

Para contornar esse problema, devemos converter explicitamente os objetos da classe base **NGAluno** para objetos da classe derivada **NGAlunoGraduacao**, como mostra a consulta a seguir:

```
SELECT AL
FROM NGTurmaGraduacao TG, (NGAlunoGraduacao) TG.Alunos AL
WHERE TG.OID = ?TurmaOID:AcIntAND AL.CoeficienteRendimento >= 7.0
```

No exemplo acima, **TG.Alunos** continua retornando uma coleção de objetos do tipo **NGAluno** (note que o relacionamento continua definido entre classes base). Porém, tais objetos são convertidos para objetos da classe derivada **NGAlunoGraduacao**.

*OBS: São consideradas valias apenas conversões de classes derivadas para classes básicas.*

## Algumas Otimizações

Com o objetivo de otimizar a carga de objetos com relacionamentos definidos entre si, é permitido especificar mais uma classe na cláusula SELECT de uma consulta, como mostra o exemplo abaixo:

```
SELECT NGAaluno, NGAaluno.Pessoa, NGAaluno.Pessoa.Endereco  
FROM NGAaluno WHERE NGAaluno.TurmaAtual.OID = ?IdTurma:AcInt
```

Essa consulta parametrizada retorna uma lista de alunos de uma determinada turma. Observe que, embora a cláusula SELECT especifique as classes NGAaluno, NGPessoa (através do caminho NGAaluno.Pessoa) e NGEndereco (através do caminho NGAaluno.Pessoa.Endereco), apenas as instâncias da primeira classe especificada (NGAaluno) influenciam no retorno da consulta. Os caminhos NGAaluno.Pessoa e NGAaluno.Pessoa.Endereco são utilizados apenas para informar que estes relacionamentos devem ser pré-carregados durante o processamento da consulta.

Observe também que, no exemplo acima, as expressões de caminho NGAaluno.Pessoa, NGAaluno.Pessoa.Endereco e NGAaluno.TurmaAtual não foram especificadas explicitamente na cláusula FROM e serão, portanto, tratadas como junções externas à esquerda (ver tópico Left Outer Join).

Imagine agora que a aplicação alvo esteja interessada em carregar essa lista de alunos para exibir seus respectivos nomes e endereços. Diferentemente do lazy loading, com a pré-carga dos relacionamentos definidos entre as classes NGAaluno, NGPessoa e NGEndereco, é possível minimizar o número de acessos ao SGBD para a recuperação dessas informações, provocando um considerável ganho de performance.

*Observação: atualmente, relacionamentos com cardinalidade múltipla (N) são sempre tratados com lazy loading, não apresentando ganhos de desempenho com a pré-carga.*

