Postgresql (Query) - Dicas e Funções (Versões Mais Recentes do Agilityflow)

> **IMPORTANTE**: Essa documentação é referente as versões pós 2023 do agilityflow que utilizam o Banco de Dados **Postgresql.** O agilityflow nas últimas versões está utilizando **POSTGRESQL** .

No agilityflow você pode buscar os dados utilizando SQL, através de queries. As queries devem ser compatíveis com **SQL Server.** Além disso existem alguns padrões que você deve seguir para obter um melhor resultado no agilityflow. Abaixo estão alguns detalhes importantes.

Tratamento de Registros Deletados

> **IMPORTANTE:**
> É de extrema importância o tratamento de registros deletados, usando as regras abaixo.
>
> Já está no nosso Roadmap o tratamento automático dos deletados, enquanto isso, o tratamento se torna obrigatório pelo desenvolvedor.

Para manter a integridade dos dados, o agilityflow nunca apaga um registro de uma tabela. Apenas o marca como deletado. Portanto, para qualquer query, é preciso excluir os deletados da lista.

Essa marcação é feita com campo deletado. Se o valor for **FALSE**, significa que ele não foi deletado e está visível no sistema. Caso esteja **TRUE**, significa que o registro já foi excluído e não é possível mais vê-lo na listagem principal do sistema.

Exemplo 1, sem o tratamento de deletados:

```
select usu_nome, usu_email from tbl_usuario
```

Resultado:

Exemplo 2, com o tratamento de deletados:

```
select usu_nome, usu_email from tbl_usuario
where deletado = false
```

Resultado:

Campo do tipo Date, Timestamp e Datetime (Data e Hora)

Sempre que um campo de texto com máscara do tipo data é criado, o sistema cria uma cópia com o mesmo nome seguido de "**__datetime__**" essa informação é gravada em uma coluna no banco de dados do tipo "**date**" quando for uma coluna sem hora ou "**timestamp without time zone**" quando for uma coluna com hora_, ao invés de varchar. Esses campos são criados para facilitar o uso tipado do dado nas queries

Exemplo de campo Datetime

Por exemplo, se o campo se chama "data_e_hora", haverá um outro chamado "data_e_hora__datetime__" e o conteúdo estará no formato padrão do sql server, **YYYY-MM-DD hh:mm:ss**

O campo "data_e_hora" estará sempre como varchar e o "data_e_hora__datetime__" estará como timestamp

data_e_hora	data_e_hora__datetime__
2027-01-31 23:5959	2027-01-31 23:5959
2027-01-31 23:5959	2027-01-31 23:5959
2027-01-31 23:5959	2027-01-31 23:5959

Campo do tipo Numérico (decimal, float, double, int, number)

Sempre que um campo de texto com máscara do tipo número é criado, o sistema cria uma cópia com o mesmo nome seguido de "**__number__**" essa informação é gravada em uma coluna no banco de dados do tipo **numeric(18,6),** ao invés de varchar. Esses campos são criados para facilitar o uso tipado do dado nas queries

Exemplo de campo numérico

Por exemplo, se o campo se chama "numero_exemplo", haverá um outro chamado "numero_exemplo__number__" e o conteúdo estará no formato padrão do sql server.

O campo "numero_exemplo" estará sempre como varchar e o "numero_exemplo__number__" estará como numeric(18,6)

numero_exemplo	numero_exemplo__number__
999.99	999.99
123.89	123.89

Formatação de números

Nome da Função

`format_number(number,format,idioma)`

Parâmetro: **number**

Pode ser um número inteiro ou um decimal

Parâmetro: **format**

Aqui você passa a formatação que deve ser retornada:

'0' = retorna o número sem nenhuma casa decimal
'1' = retorna o número com 1 casa decimal
'2' = retorna o número com 2 casas decimais
'3' = retorna o número com 3 casas decimais
'4' = retorna o número com 4 casas decimais
'5' = retorna o número com 5 casas decimais
'6' = retorna o número com 6 casas decimais

Parâmetro: **idioma - @sysCurrentLanguage**

Aqui você precisa passar o idioma do usuário logado, pois alguns idiomas invertem o . (ponto) e a , (virgula) do número.
O agilityflow guarda o idioma do usuário dentro da variável @sysCurrentLanguage então apenas passe no parâmetro esse variável.

Exemplo de utilização

Abaixo o número será formato para 5 casas decimais, no padrão do idioma do usuário logado.

```
select format_number(5800000.888, '5', @sysCurrentLanguage)
```

Retorno para o usuário que está logado no agilityflow em português ou espanhol

```
5.800.000,88800
```

Retorno para o usuário que está logado no agilityflow usando em inglês

```
5,800,000.88800
```

Funções de data no Postgresql

As principais funções para manipular datas são: **GETDATE**, **DATEPART**, **DATEADD** e **DATEDIFF**.

Um detalhe importante é que as funções de data trabalham referenciando unidades de data. As mais comuns são:

- year(ano);
- month(mês);
- day(dia);

CURRENT_TIMESTAMP)

A função CURRENT_TIMESTAMP retorna a data e a hora atuais do sistema.

```
SELECT CURRENT_TIMESTAMP
```

Extrair parte de uma data

A função EXTRACT retorna a parte especificada de uma data como um inteiro. Observe os exemplos:

```
SELECT EXTRACT(YEAR FROM '2024-02-01'::DATE)
```

Reposta: 2024

```
SELECT EXTRACT(MONTH FROM '2024-02-01'::DATE)
```

Reposta: 2

```
SELECT EXTRACT(DAY FROM '2024-02-01'::DATE)
```

Reposta: 1

Adicionar dias em uma data

A função **abaixo** retorna uma nova data através da soma do número de unidades especificadas pelo valor *unidade* a uma data. Observe os exemplos:

```
SELECT '2024-02-01'::DATE + INTERVAL '7 days';
```

Reposta: 2024-02-07

Calcular a diferença entre datas (date diff)

A função **abaixo** calcula a diferença entre as datas *data2 e data1*, retornando o resultado como um inteiro, cuja unidade é definida pelo valor *unidade*. Observe os exemplos:

```
SELECT EXTRACT(DAY FROM '2024-02-01'::DATE - '2023-10-10'::DATE);
```

Reposta: 114 (dias)

Cálculo de idade em anos, meses e dias

```
SELECT DATE_PART('year', AGE(NOW(), '1990-05-15')) AS idade_anos;
```

Relatório

Como fazer Filtro nos Relatórios?

Sempre que um relatório possuir um filtro, é necessário incluir esse filtro na query que gera os dados do relatório.

Não importa como é a query, ela deve incluir a cláusula where, como a chamada da função Filter. Essa função, possui 3 parâmetros:

`**Filter** (variável_filtro, tabela, campo)`

- variável_filtro: é a variável que o agilityflow criou, após a configuração do filtro.
- tabela: nome da tabela (definido nos Dados Técnicos) onde se encontra o dado a ser filtrado.
- campo: é o nome do campo (conforme informado no campo Coluna Banco de Dados SQL) onde a informação que o filtro utiliza se encontra.

No exemplo abaixo, a query lista os campos *usu_nome*, *usu_email e usu_sexo* da tabela do sistema de usuário (tbl_usuario) e se inclui a função Filter, usando a variável @sexo aplicada no campo *usu_sexo*.

```
select usu_nome, usu_email, usu_sexo from tbl_usuario
where Filter(@sexo, tbl_usuario, usu_sexo)
```

Exemplo de relatório sem filtro

Baseado na query acima, mas sem filtro configurado no relatório.

```
select usu_nome, usu_email, usu_sexo from tbl_usuario
```

Exemplo de relatório com filtro

O mesmo exemplo acima, com o filtro no campo sexo.

```
select usu_nome, usu_email, usu_sexo from tbl_usuario
where Filter(@sexo, tbl_usuario, usu_sexo)
```

relatorio_filtro.gif

Caso seja criado um novo filtro, basta colocar "*and*" depois do primeiro Filter, e colocar o segundo Filter. Não importa a quantidade de filtros, desde que sejam adicionados todos os Filter e os *and*.

```
select usu_nome, usu_email, usu_sexo from tbl_usuario
where Filter(@sexo, tbl_usuario, usu_sexo) and
    Filter(@nome, tbl_usuario, usu_nome) and
    Filter(@email, tbl_usuario, usu_email)
```

Relatório - Tabela de Dados

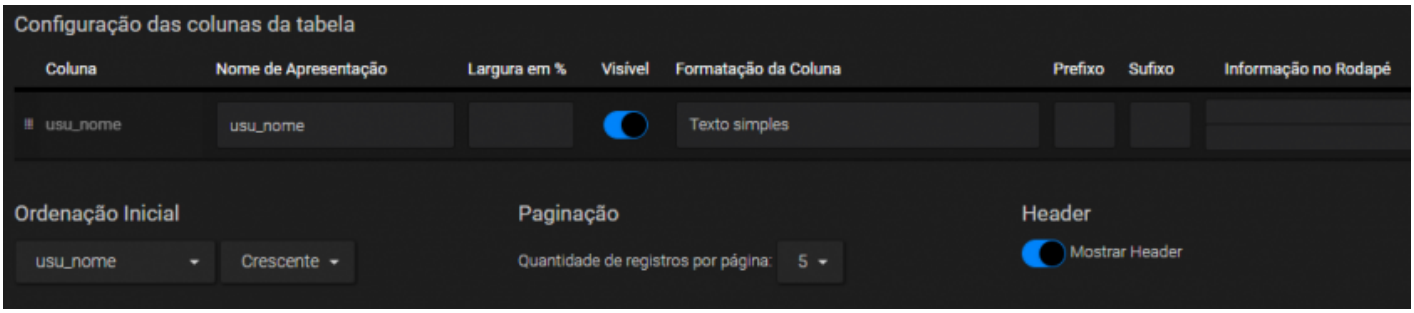
Para as tabelas de dados, existem algumas regrinhas para a customização via Query Sql

1 - Lembre-se de tratar os dados que já foram deletados, usando no where "...deletado = false.."

2 - **Não** pode conter 'Order by' no select. Essa informação será controlada automaticamente pelo agilityflow e você poderá manipular essa informação na tela de configuração, como na imagem abaixo.

3 - **Não** pode conter o controle de 'Limit' no select. Essa informação será controlada automaticamente pelo agilityflow e você poderá manipular essa informação na tela de configuração, como na imagem abaixo.

4 - **Não** pode conter regras de paginação. Essa informação será controlada automaticamente pelo agilityflow e você poderá manipular essa informação na tela de configuração, como na imagem abaixo.



Relatório - Gráfico

Para gráficos, existem algumas regrinhas para a customização via Query Sql

1 - Lembre-se de tratar os dados que já foram deletados, usando no where "...deletado = false.."

2 - A query precisa conter '**limit** ' e o máximo do limit para essa query é **100**

3 - Diferentemente da query da tabela de dados, nessa query pode sim conter regras de 'Order by'

Relatório - Label

Para as labels do relatório, existem algumas regrinhas para a customização via Query Sql

1 - Lembre-se de tratar os dados que já foram deletados, usando no where "...deletado = false.."

2 - A query precisa conter '**limit** ' e o máximo do limit para essa query é **1**

3 - Diferentemente da query da tabela de dados, nessa query pode sim conter regras de 'Order by'

Formulário

Lista Dinâmica - Regras

Os campos que são carregados com Lista dinâmica, podem ser customizados utilizando Query:

Algumas regras importantes:

1 - O resultado final da query precisa sempre ser os dados do formulário que você usou como base de dados

2 - Lembre-se de tratar os dados que já foram deletados, usando no where "...deletado = false.."

3 - A query deve retornar duas colunas com os seguintes *alias*: "Name" e "Value". O "Name" deve ser a mesma coluna definida no campo de apresentação (Nome) e o "Value", deve ser a coluna "id" do formulário definido como base de dados (Cliente). Exemplo:

4 - Na query, o campo de apresentação "Name" pode ser um campo concatenado entre outras colunas.

Exemplo simples:

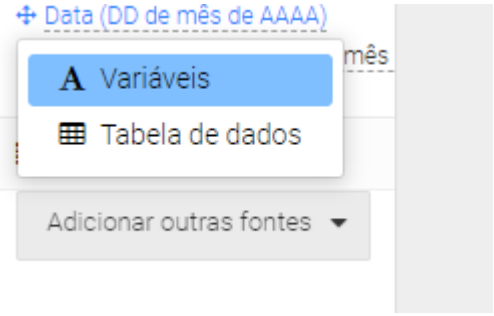
```
SELECT Id as Value, CAMPO_DE_APRESENTACAO as Name FROM [TABELA] where deletado = false
```

Report Print - Relatório de Impressão

Para as labels do relatório, existem algumas regrinhas para a customização via Query Sql

Para acessar a customização, clique no botão "Adicionar Outras Fontes de Dados" e selecione o tipo:

1. **Variáveis:** para retornar uma query com apenas 1 linha e várias colunas, cada coluna se torna um campo separado para ser utilizado no relatório
2. **Tabela de Dados:** para retornar uma tabela com diversas colunas e diversas linhas e usa-las em conjunto em uma tabela



Algumas regras importantes:

1 - Lembre-se de tratar os dados que já foram deletados, usando no where "...deletado = false.."

2 - Utilizar no "where" da query, o parâmetro **@formularioid** para filtrar pelo formulário que está sendo solicitada a impressão, caso contrário trará informações de formulários aleatórios.

3 - A query precisa conter '**limit** ' e o máximo do limit para essa query quando for uma query do tipo:

- **Tabela de Dados** é **500**,
- **Variáveis** é **1**

4 - Essa query pode sim conter regras de 'Order by'

Revision #7
Created 30 January 2025 13:57:15 by agilityflow
Updated 5 March 2025 12:33:44 by agilityflow