

GlideAggregate – Guia

O que é?

A Classe `GlideAggregate` é uma extensão do `GlideRecord`, serve para efetuar buscas no banco de dados e executar cálculos de forma rápida e eficiente. Mas quais são as diferenças entre os dois?

`GlideAggregate`:

não retorna os registros, apenas “olha” para alguns dados e, com estes, realiza cálculos. O retorno dessa busca é o resultado deste cálculo.

`GlideRecord`:

retorna os registros, ou seja, com isso é possível manipular os dados de cada LINHA da sua tabela.

Logo o `GlideAggregate` é muito mais rápido que o `GlideRecord`, porém com algumas limitações. Ex.: não é possível deletar um registro ou coletar o nome dos campos utilizando `GlideAggregate`.

Como usar

```
var agg = new GlideAggregate("incident"); ①
agg.addAggregate("COUNT"); ②
agg.addEncodedQuery("active=true"); ③
agg.query(); ④

if(agg.next()){ ⑤
    gs.print(agg.getAggregate("COUNT")); ⑥
}

//CONSOLE -----
37 ⑦
```

1º criação de um novo objeto do tipo `GlideAggregate`, como parâmetro deve ser informado o nome da tabela em que se deseja buscar os dados;

2º o método `addAggregate` diz o cálculo que você deseja realizar. “COUNT” serve para contar quantos registros existem. Os outros parâmetros são:

COUNT	Retorna o número de registros encontrados
AVG	Retorna a média de todos os valores encontrados
SUM	Retorna a soma de todos os valores encontrados
MIN	Retorna o menor valor encontrado
MAX	Retorna o maior valor encontrado

Em todos, menos no COUNT, você deve especificar um **campo** que contenha o valor que o GlideAggregate deve calcular. Como:

```
agg.addAggregate("SUM","sys_mod_count");
```

Nesse exemplo será somado os valores do campo sys_mod_count (número de updates de um registro). O resultado final será a soma de todos os updates de todos os registros ativos na tabela de Incidentes;

3º Faça a sua query;

4º Busque;

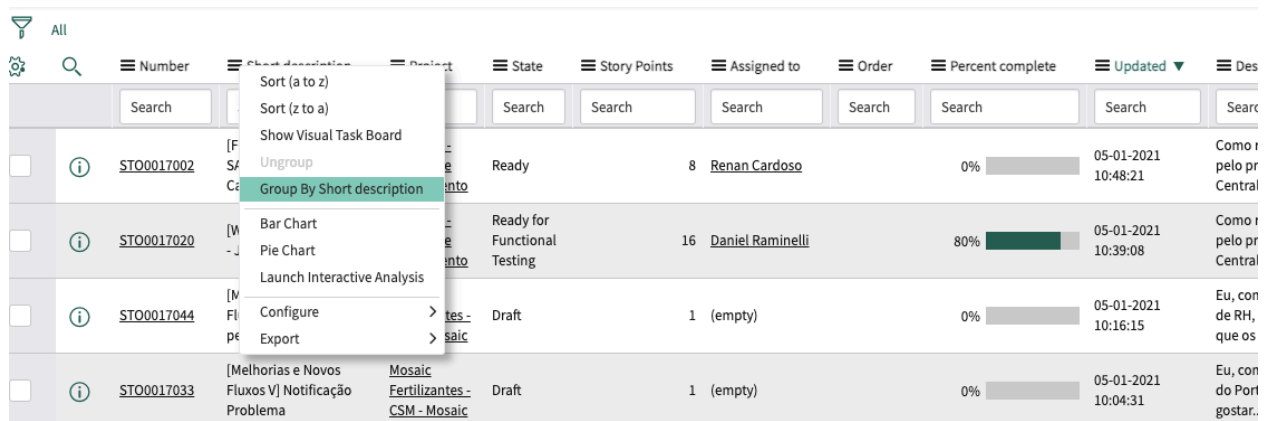
5º Repare que aqui utilizamos um **"if"**, isso porque a query irá retornar apenas um resultado e não os registros.

6º Pegue o valor do Aggregate. Apenas troque **"add"** por **"get"**;

7º O resultado no console;

groupBy

Uma das principais funcionalidades do GlideAggregate é o **"groupBy"**. Esse método agrupa os registros por um determinado campo. O seu funcionamento é semelhante ao "groupBy" de uma tabela ServiceNow:



	Number	Short description	State	Story Points	Assigned to	Order	Percent complete	Updated	Des
	STO0017002	[F...]	Ready	8	Renan Cardoso		0%	05-01-2021 10:48:21	Como r... pelo pr... Central
	STO0017020	[M...]	Ready for Functional Testing	16	Daniel Raminelli		80%	05-01-2021 10:39:08	Como r... pelo pr... Central
	STO0017044	[M...]	Draft	1 (empty)			0%	05-01-2021 10:16:15	Eu, con... de RH, que os
	STO0017033	[Melhorias e Novos Fluxos V] Notificação Problema	Draft	1 (empty)			0%	05-01-2021 10:04:31	Eu, con... do Port... gostar..

Exemplo de Código:

```
var agg = new GlideAggregate("incident");
agg.addAggregate("COUNT");
agg.addEncodedQuery("active=true");
agg.groupBy('state'); ❶
agg.query();

while(agg.next()){ ❷
    gs.print(agg.getDisplayValue('state')); ❸
    gs.print(agg.getAggregate("COUNT"));
}

//CONSOLE -----
New
11
In Progress
20
On Hold
6
//Total: 37
```

1º Chama-se o método e é passado como parâmetro o campo pelo qual queremos agrupar;

2º Perceba que dessa vez foi utilizado um “while”, isso porque o GlideAggregate irá retornar o número de incidentes por estado. Para cada laço será coletado os dados de um estado;

3º Veja que nesse código aparece o “getDisplayValue”. Isto funciona somente para o campo de “state”. Mas por quê?

Podemos utilizá-lo pois, quando definimos no groupBy o campo “state”, o GlideAggregate irá obrigatoriamente coletar esse campo para agrupar os registros. Logo, sua informação ficará salva e poderemos utilizá-la.

OBSERVAÇÃO / BÔNUS

- É permitido utilizar mais de um groupBy ao mesmo tempo. Caso isso ocorra, como exemplo utilizando um groupBy para o campo “state” e outro para o campo “assigned to”, teremos a seguinte situação:

1º A ordem afetará o resultado, vamos supor que o groupBy(‘state’) esteja na frente do groupBy(‘assigned_to’);

2º O GlideAggregate irá criar grupos (como visto no exemplo acima), contendo o valor dos estados. Após isto, cada grupo terá os seus registros agrupados pela pessoa assignada.

Ou seja, imaginando que possuamos os Incidentes:

Número	Estado	Assignado
1	New	Marcelo
2	New	Marcelo
3	New	Maria
4	In Progress	Marcelo
5	On Hold	Maria
6	On Hold	Maria

- Primeiro agruparemos pelo New: incidentes 1, 2 e 3;
- Dentro do New, será agrupado por pessoa assignada: 1 e 2 para o Marcelo e 3 para a Maria;
- Segue a lógica para o restante;

Análise dos laços:

- O primeiro laço terá a COUNT de 2 e os displayValues de Marcelo e New;
- O segundo laço terá o COUNT de 1 e os displayValues de New e Maria;
- O terceiro laço terá o COUNT de 1 e os displayValues de Marcelo e In Progress;
- O quarto laço terá o COUNT de 2 e os displayValues de On Hold e Maria;
- Fim da execução.

addTrend

Outra excelente funcionalidade do GlideAggregate é o método addTrend. Esse método é muito similar ao groupBy com a diferença de ser especificado para [datas](#).

Imagine fazer um groupBy por um campo do tipo "Date". Será impossível obter algo coerente, sendo que ele agrupará por data idênticas, ou seja, até mesmo os segundos devem ser iguais. Para isso usa-se o addTrend.

Esse método recebe alguns parâmetros, eles podem ter as iniciais em maiúscula ou não, o que muda é a formatação do resultado:

Nome	Exemplo de Resultado	Nome	Exemplo de Resultado
Hour	9/2016	hour	9
DayOfWeek	4/2016	dayofweek	Wednesday/2016
Week	33/2015	week	8/10/2015
Month	8/2016	month	Aug/2016
Quarter	3/2016	quarter	3/2016
Year	2020/2020	year	2020

Cada parâmetro é o intervalo de tempo pelo qual iremos agrupar. Caso seja escolhido “Week” os registros serão agrupados por semanas e assim por diante.

Exemplo de Código:

```
var agg = new GlideAggregate("incident");
agg.addAggregate("COUNT");
agg.addEncodedQuery("active=true");
agg.addTrend('opened_at', 'Week'); 1
agg.query();

while(agg.next()){
    gs.print(agg.getValue('timeref')); 2
    gs.print(agg.getAggregate("COUNT"));
}

//CONSOLE -----
33/2015
2
45/2015
2
...
```

1º Chame-se o método passando como parâmetro o campo “opened_at” e com o intervalo de “Week”;

2º Para pegar o valor do intervalo utilizamos o “getValue(‘timeref’)”;

INFO BÔNUS – pergunta mestre

“Quando eu devo usar o GlideAggregate ao invés do GlideRecord?”

Resp.:

você irá utilizar o GlideAggregate quando o que você precisa é somente os valores de um cálculo. Como exemplo: “Quantos incidentes eu tenho aberto hoje?”, “Qual a média de Incidentes resolvidos dentro do mês de janeiro?”, “Quanto de budget foi gasto nesse projeto?”

Perceba que em nenhum momento o registro em si interessa, o que importa é o todo. A consolidação do resultado.

INFO BÔNUS – pergunta mestre 2

“Mas eu não consigo fazer isso usando GlideRecord?”

Resp.:

SIM! Só que de uma forma muito mais lenta. Exemplificando: Suponha que você precise saber quantos incidentes eu tenho aberto hoje. Temos as duas opções que retornam o mesmo resultado para a variável “conta”:

Código 1:

```
var gr = new GlideRecord("incident");
gr.addQuery("active", "true");
gr.query();

var conta = gr.getRowCount();
```

Código 2:

```
var agg = new GlideAggregate("incident");
agg.addQuery("active", "true");
agg.addAggregate("COUNT");
agg.query();

if(agg.next())
    var conta = agg.getAggregate("COUNT");
```

Mas o que mudou? O retorno da query, no primeiro código é muito maior que o do segundo. Isso porque no código 1 você não pegou somente o número de registros. Mas sim **TODOS** os registros. Caso queira utilizar um “gr.next()”, poderá coletar todos os valores de todos os campos.

O que não ocorre no código 2 em que o retorno da query é somente o valor do “COUNT”.