Flexible search

Lets understand some basic concepts about flexible search in Hybris with few examples

It's a hybris **built-in query language** based on **SQL syntax**.

It enables us to **search** the **records**from the **database** using **item types**.

In flexible search queries, we **never** use database **table names**.

We always use **item types** which will be **mapped** to a corresponding **tables** by Hybris.

Flexible search query execution has 2 phases

1) Pre-parsing phase
In this phase, Hybris **converts** the **flexible search** query into **SQL** query.

2) Executing the SQL converted query
In this phase,converted **SQL query** will be executed by Hybris.

Examples :


```
Select * from {Product}
```

Copy this code

This query **retrieves** all the **columns** and **rows** of a **Product item type**.

The **item type** specified in the **curly braces** is the **exact item type code** defined in the **items.xml**.

This should have been **mapped** to specific **table** using **deployment tag** in **items.xml**

for more details on how **tables** and **item types** are **related**, click **here** (http://javainsimpleway.com/defining-item-types-in-hybris/) and **here** (http://javainsimpleway.com/deployment-and-typecodes-in-items-xml/)

```
SELECT {name[de]}, {name[en]} FROM {Product}
```

Copy this code

This query **retrieves name of a Product** in **German** and **English** locale

```
SELECT {p.code} FROM {Product AS p} ORDER BY {p.code}
```

Copy this code

This query **retrieves Product code** from **Product type** using **alias** with the name **"p"** for **Product type**.

We can **specify** the **join** between **2 types** as below

```
SELECT * FROM {Product JOIN Category}
```

Copy this code

This query makes the **Join** between **Product** and **Category** item type.

Note:

We can make different types of Joins like Left outer join,Right outer join and Inner join.

Sub Types

When we search **any type** in flexible search, by default its **subtypes** will also be **retrieved** in the result.

Example:

```
Select * from {Product}
```

Copy this code

This will **retrieve** the **instances** of **Product** and **VariantProduct**.

Since **VariantProduct** is a **sub type** of **Product**, it **retrieves** its **sub types** as well.

If we want to **exclude** the **subtypes** in the result, we must use **exclamation mark (!)** while specifying type as below

```
Select * from {Product!}
```

Copy this code

In this example, we have used **!** for **product type** and hence it retrieves **only Product instances** but **not** its **variants**.

Using conditions

```
SELECT * FROM {Product} WHERE {code} LIKE '%001%'
```

Copy this code

Retrieves **all the products** whose **code** has **001** in it.

```
SELECT * FROM {Product} WHERE {code} LIKE '%001%' AND {code} LIKE '%m%'
```

Copy this code

Retrieves **all the products** whose **code** has **001** and **m** in it.

```
SELECT * FROM {Product} WHERE {code} NOT LIKE '%001%'
```

Copy this code

Retrieves **all the products** whose **code does no**t contain **001** in it.

```
SELECT {code},{pk} FROM  {Product} ORDER BY {code} DESC
```

Copy this code

Retrieves the **result** and **Sorts** the **search results** based on the **code**column in the database in **descending order**

Runtime Parameters

We can **specify** run time parameters in **flexible search** query using **placeholder(?) prefix** to a specific **column**.

```
SELECT {p:pk} FROM {Product AS p} WHERE {p:code} LIKE ?code
```

Copy this code

Here we have specified **?code** which means **run time parameter** is **code** whose **value** will be added at**run time** using **query params HashMap**.

Executing flexible search queries using API