

Practice №1. Distributed table in ClickHouse. Accessing files in S3 via ClickHouse.

Ex. 1. Creating a distributed table in ClickHouse cluster.

For this, first we need to connect to the cluster. After connecting, we can see all the tables in its database, as seen on the figure 1.

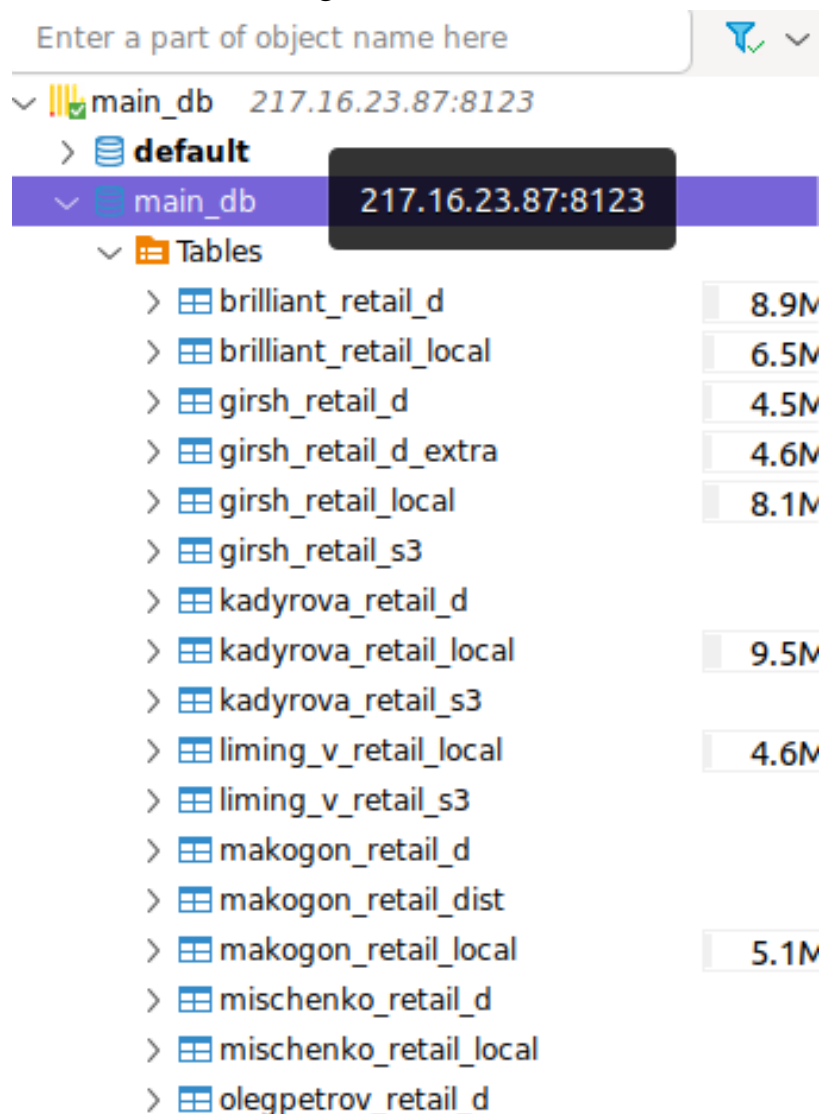


Figure 1. Database Navigator in DBeaver after a successful connection to the cluster

After that, we use VK Cloud to store our data. This is done by creating a bucket and populating it with the tables provided for this task, as seen on figures 2 and 3.

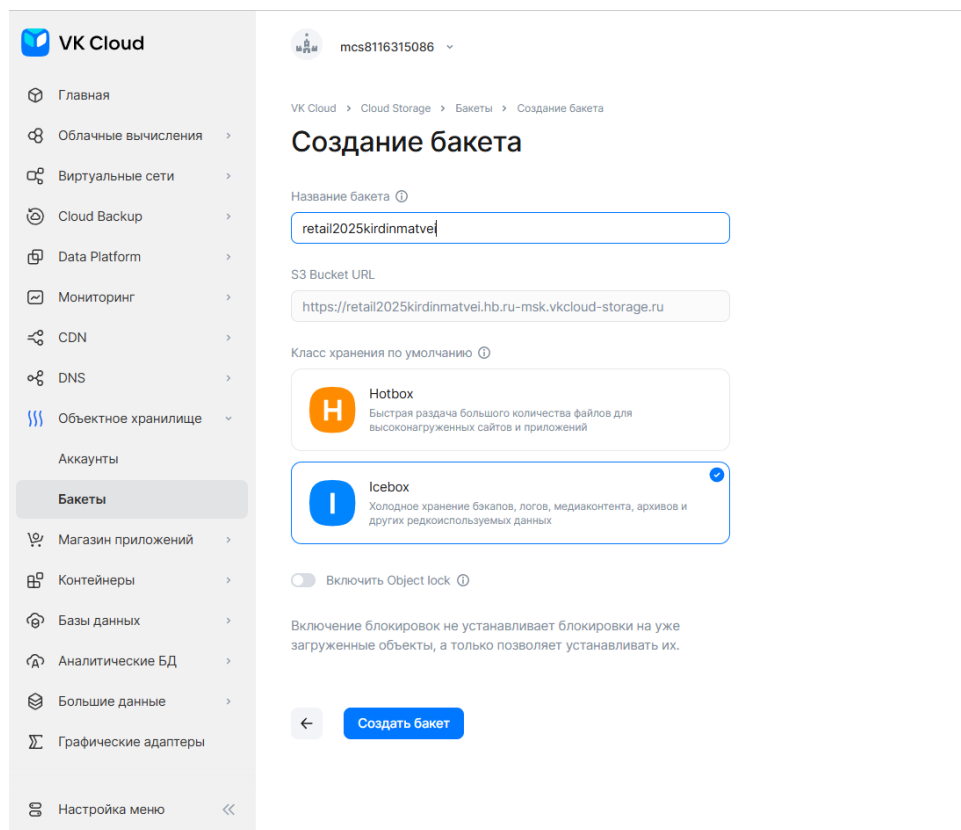


Figure 2. Bucket creation interface in VK Cloud



Figure 3. Bucket with the gzip archives of data uploaded

Now we can start creating distributed tables on the ClickHouse cluster. First, we construct two tables with local and distributed and merge tree engines to serve as distributed and local tables respectively. For local table we run the SQL script given on figure 4.

```
CREATE TABLE IF NOT EXISTS kirdin_retail_local ON CLUSTER 'cluster' (  
    invoiceno String,  
    stockcode String,  
    description String,  
    quantity Float32,  
    invoicedate DateTime,  
    unitprice Float32,  
    customerid UInt32,  
    country String  
)  
ENGINE = MergeTree  
PRIMARY KEY (invoiceno, stockcode)  
ORDER BY (invoiceno, stockcode, customerid)
```

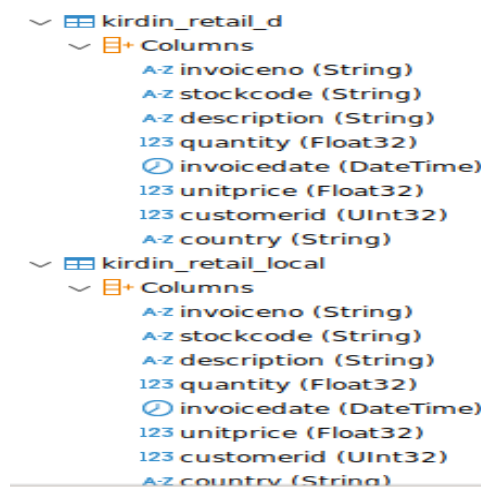
Figure 4. Script that creates the local table with certain columns

After running this script, we also run the script from figure 5.

```
CREATE TABLE IF NOT EXISTS kirdin_retail_d ON CLUSTER 'cluster' AS kirdin_retail_local  
ENGINE = Distributed('cluster', main_db, kirdin_retail_local,  
    murmurHash2_32(invoiceno)%2);
```

Figure 5. Script that creates distributed table, which will house the data

Results can be seen on figure 6. They indicate that both of those table were correctly created.



```
└─ kirdin_retail_d  
  └─ Columns  
    └─ invoiceno (String)  
    └─ stockcode (String)  
    └─ description (String)  
    └─ quantity (Float32)  
    └─ invoicedate (DateTime)  
    └─ unitprice (Float32)  
    └─ customerid (UInt32)  
    └─ country (String)  
└─ kirdin_retail_local  
  └─ Columns  
    └─ invoiceno (String)  
    └─ stockcode (String)  
    └─ description (String)  
    └─ quantity (Float32)  
    └─ invoicedate (DateTime)  
    └─ unitprice (Float32)  
    └─ customerid (UInt32)  
    └─ country (String)
```

Figure 6. Database Navigator in DBeaver after successful creation of tables

With tables in place, we can write the data from cluster to the distributed table by running a script from figure 7.

```
INSERT INTO kirdin_retail_d SELECT * FROM
s3Cluster('cluster',
  'https://hb.ru-msk.vkcs.cloud/retail2025kirdinmatvei/online_retail_*.csv.gz',
  '[REDACTED]',
  '[REDACTED]',
  'CSVWithNames',
  'InvoiceNo String, StockCode String, Description String, Quantity Float32, InvoiceDate DateTime, UnitPrice Float32',
  'gzip')
settings max_threads=4, max_insert_threads=4, input_format_parallel_parsing=0;
```

Figure 7. This query inserts all the data from VK Cloud bucket to our table

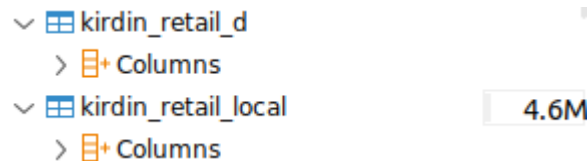


Figure 8. Changes to the tables we created

As indicated on figure 8, the data indeed did get copied to the local table.

Ex. 2. Accessing S3 files via ClickHouse without copying data (Data Lake-style manipulations).

First, we create a table with S3 engine which will not populate a table in DB. Instead, it will provide access to the files in tabular style. This can be done by running a script from figure 9. Its' results can be seen on the figure 10.

```
CREATE TABLE kirdin_retail_s3 (  
    InvoiceNo String,  
    StockCode String,  
    Description String,  
    Quantity Float32,  
    InvoiceDate DateTime,  
    UnitPrice Float32,  
    CustomerID UInt32,  
    Country String  
)  
ENGINE = S3('https://hb.ru-msk.vkcs.cloud/retail2025kirdinmatvei/online_retail_*.csv.gz',  
    '[REDACTED]',  
    '[REDACTED]',  
    'CSVWithNames');
```

Figure 9. This script creates a table with a specific set of rows and an S3 engine

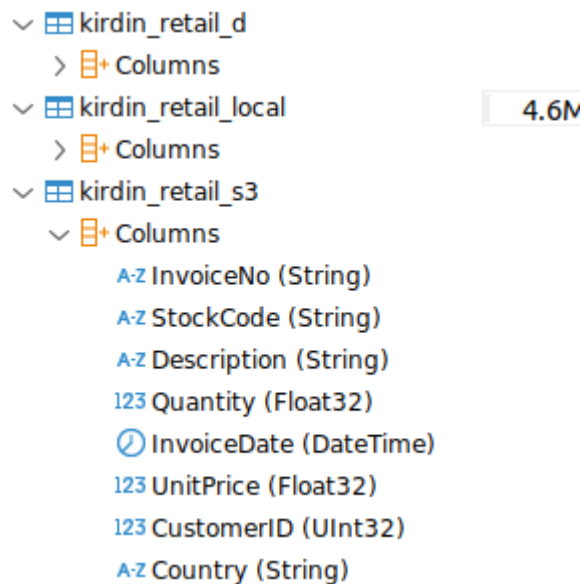


Figure 10. Database Navigator after a successful creation of S3 table

Now we can write queries to this table. For example, let us see the set of all customers who purchased “CHRISTMAS STAR WISH LIST CHALKBOARD” stocks and the number of them (figures 11 through 13). Let us also write a query for an average transaction value for each unique customer (figures 14 and 15).

```

select * from kirdin_retail_s3
where Description = 'CHRISTMAS STAR WISH LIST CHALKBOARD';

```

Figure 11. A query that returns all the rows with “CHRISTMAS STAR WISH LIST CHALKBOARD” description

	A-Z InvoiceNo	A-Z StockCode	A-Z Description	123 Quantity	InvoiceDate	123 UnitPrice	123 CustomerID
1	C542346	22596	CHRISTMAS STAR WISH LIST CHA	-1	2011-01-27 12:00:00	0.72	12,
2	557858	22596	CHRISTMAS STAR WISH LIST CHA	5	2011-06-23 11:51:00	1.25	14,
3	558763	22596	CHRISTMAS STAR WISH LIST CHA	12	2011-07-03 13:08:00	1.25	12,
4	559052	22596	CHRISTMAS STAR WISH LIST CHA	1	2011-07-05 16:53:00	2.46	15,
5	559545	22596	CHRISTMAS STAR WISH LIST CHA	12	2011-07-10 14:42:00	1.25	15,
6	559551	22596	CHRISTMAS STAR WISH LIST CHA	1	2011-07-10 16:18:00	1.25	16,
7	559693	22596	CHRISTMAS STAR WISH LIST CHA	3	2011-07-11 16:21:00	2.46	15,
8	559979	22596	CHRISTMAS STAR WISH LIST CHA	12	2011-07-14 11:20:00	1.25	14,
9	560125	22596	CHRISTMAS STAR WISH LIST CHA	1	2011-07-15 10:44:00	1.25	18,
10	560360	22596	CHRISTMAS STAR WISH LIST CHA	48	2011-07-18 11:55:00	1.25	12,
11	560394	22596	CHRISTMAS STAR WISH LIST CHA	60	2011-07-18 13:35:00	1.25	14,
12	560434	22596	CHRISTMAS STAR WISH LIST CHA	3	2011-07-18 15:56:00	2.46	15,
13	560444	22596	CHRISTMAS STAR WISH LIST CHA	4	2011-07-18 17:15:00	1.25	13,

Figure 12. The result of query from figure 11

```

select count(distinct CustomerID) as cnt
from kirdin_retail_s3
where Description = 'CHRISTMAS STAR WISH LIST CHALKBOARD';

```

Figure 12. A query that returns the number of distinct customers that bought stocks with “CHRISTMAS STAR WISH LIST CHALKBOARD” description

	123 cnt
1	198

Figure 13. The result of query from figure 13

```

SELECT CustomerID, AVG(Quantity * UnitPrice) AS AvgValue
FROM main_db.kirdin_retail_s3
GROUP BY CustomerID
ORDER BY AvgValue DESC;

```

Figure 14. A query that returns the average transaction value for each customer from most to least.

	123 CustomerID	123 AvgValue
106	17,941	152.2799978256
107	13,689	151.6666634878
108	17,404	150.7503467786
109	14,609	150.3899963032
110	18,246	149.0249963999
111	15,171	147.7857112885
112	15,653	146.2400057316
113	16,429	142.2750029564
114	12,590	140.8768557583
115	17,389	139.7324982977
116	12,643	139.3599996865
117	17,929	139.083331426
118	14,866	137.839319665
119	14,616	135.3500010000

Figure 15. The result of query from the figure 14