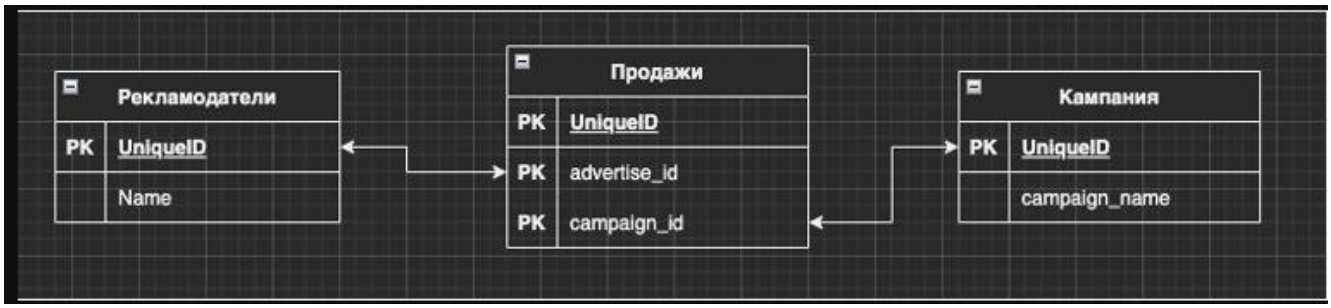
Домашнее задание №3

1. Денормализуйте таблицу так, чтобы не нужно было для каждого рекламодателя постоянно подсчитывать количество кампаний и продаж.



Выполнение:

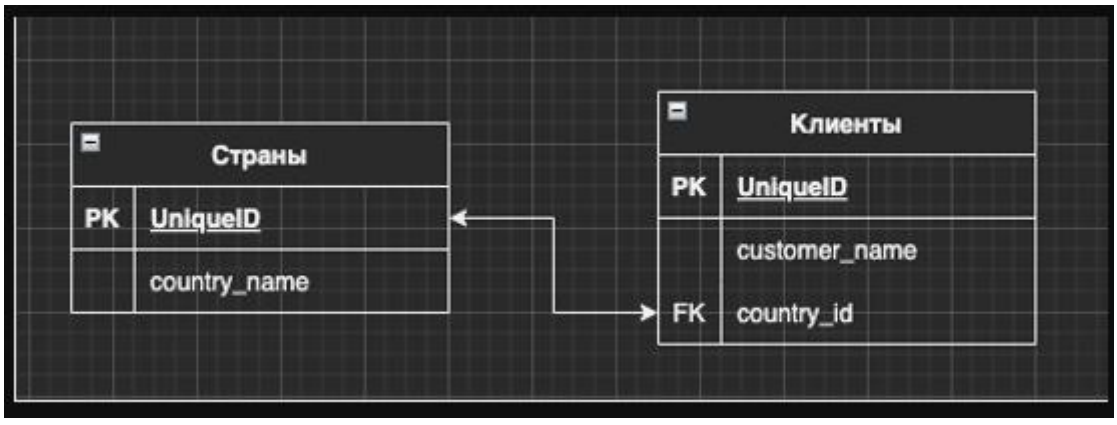
Добавим дополнительные столбцы в таблицу Рекламодатели, чтобы хранить информацию о количестве кампаний и общей сумме продаж для каждого клиента.

- num_campaigns (количество кампаний для данного клиента)
- total_sales (общая сумма продаж для данного клиента)

Эти значения можно обновлять при добавлении новых кампаний или продаж чтобы учесть изменения.

Таким образом, не нужно будет каждый раз подсчитывать количество кампаний и продаж при выполнении запросов, связанных с клиентами.

2. В базе данных есть две таблицы: страны и клиенты. Одной из потребностей компании является исследование клиентов и стран с точки зрения эффективности продаж, поэтому часто выполняются объединения между таблицами: клиенты и страны. Что нужно сделать, чтобы ограничить частое объединение этих двух таблиц?



Выполнение:

1) Использовать кэширование результатов запросов, вместо того, чтобы каждый раз выполнять объединение таблиц при запросе данных, можно сохранить результаты запроса в кэше и использовать их при последующих запросах. Таким образом, избежать лишних операций объединения таблиц и ускорить выполнение запросов.

2) Оптимизировать запросы, чтобы минимизировать необходимость объединения таблиц. Например, можно добавить дополнительные поля в таблицу клиентов, содержащие информацию о стране клиента, чтобы не приходилось каждый раз объединять таблицы для получения этой информации.

3) Использовать индексы на таблицах, чтобы ускорить выполнение запросов объединения. Создание индексов на полях, по которым происходит объединение таблиц, поможет ускорить поиск и сопоставление данных.

4) Создать представление, которое объединяет эти две таблицы. После создания представления можно выполнять запросы к таблицам вместо того, чтобы объединять таблицы каждый раз.

3. Вернемся к первому примеру. Предположим, компания хочет регулярно извлекать данные о продажах, например, о кампаниях или рекламодателях с полными именами. Как мы можем решить проблему постоянной необходимости объединения таблиц?

Выполнение:

1. Использование представлений:

Создать представление, которое объединяет необходимые таблицы. Представление будет виртуальной таблицей, которая будет отображать объединенные данные, но не будет содержать фактических данных.

2. Использование подзапросов:

Использовать подзапросы для извлечения полных имен рекламодателей в запросе о продажах.

3. Денормализация:

Денормализация включает добавление столбца с полными именами рекламодателей в таблицу Продажи. Это устранит необходимость в объединении таблиц для получения этой информации.

```
/*
chcp 65001 && spark-shell -i C:\Users\CND\Desktop\job\hw3.scala --conf
"spark.driver.extraJavaOptions=-Dfile.encoding=utf-8"
*/
import org.apache.spark.internal.Logging
import org.apache.spark.sql.functions.{col, collect_list, concat_ws}
import org.apache.spark.sql.{DataFrame, SparkSession}
val t1 = System.currentTimeMillis()
if(1==1){
var df1 = spark.read.format("com.crealytics.spark.excel")
        .option("sheetName", "Sheet1")
        .option("useHeader", "false")
        .option("treatEmptyValuesAsNulls", "false")
        .option("inferSchema", "true").option("addColorColumns", "true")
        .option("usePlainNumberFormat","true")
        .option("startColumn", 0)
        .option("endColumn", 99)
        .option("timestampFormat", "MM-dd-yyyy HH:mm:ss")
        .option("maxRowsInMemory", 20)
        .option("excerptSize", 10)
        .option("header", "true")
        .format("excel")
        .load("C:/Users/CND/Desktop/job/s3.xlsx")
        df1.show()
        df1.write.format("jdbc").option("url","jdbc:mysql://localhost:3306/
spark_?user=root&password=vfhbfv")
        .option("driver", "com.mysql.cj.jdbc.Driver").option("dbtable",
"hw3")
        .mode("overwrite").save()
val q = """
        SELECT ID_Тикета, FROM_UNIXTIME (Status_Time) Status_Time,
        (LEAD(Status_Time) OVER(PARTITION BY ID_Тикета ORDER BY
Status_Time) - Status_Time) / 3600 Длительность,
        CASE WHEN Статус IS NULL THEN @PREV1 ELSE @PREV1:=СТатус END
Статус,
        CASE WHEN Группа IS NULL THEN @PREV2 ELSE @PREV2:=Группа END
Группа,
        Назначение
        FROM (
            SELECT ID_Тикета, Status_Time, Статус, IF (ROW_NUMBER()
OVER(PARTITION BY ID_Тикета ORDER BY Status_Time) = 1 AND Назначение IS
NULL, '',  Группа) Группа, Назначение
            FROM (
                SELECT DISTINCT a.objectid ID_Тикета, a.restime
Status_Time, Статус, Группа, Назначение, (SELECT @PREV1:=''), (SELECT
@PREV2:='')
```

```
                FROM (
                    SELECT DISTINCT objectid, restime
                    FROM spark_.hw3
                    WHERE fieldname IN ('gname2', 'status')
                ) a
                LEFT JOIN (SELECT DISTINCT objectid, restime, fieldvalue
статус
                    FROM spark_.hw3
                    WHERE fieldname IN ('status')
                ) a1 ON a.objectid = a1.objectid AND a.restime = a1.restime
                LEFT JOIN (SELECT DISTINCT objectid, restime, fieldvalue
Группа, 1 Назначение
                    FROM spark_.hw3
                    WHERE fieldname IN ('gname2')
                ) a2 ON a.objectid = a2.objectid AND a.restime = a2.restime
            ) b1
        ) b2
        """
        spark.read.format("jdbc").option("url","jdbc:mysql://localhost:3306
/spark_?user=root&password=vfhbfv")
        .option("driver", "com.mysql.cj.jdbc.Driver").option("query", q)
        .load()
        .write.format("jdbc").option("url","jdbc:mysql://localhost:3306/spa
rk_?user=root&password=vfhbfv")
        .option("driver", "com.mysql.cj.jdbc.Driver").option("dbtable",
"hw3_1")
        .mode("overwrite").save()

var df2 =
spark.read.format("jdbc").option("url","jdbc:mysql://localhost:3306/spark_?
user=root&password=vfhbfv")
        .option("driver", "com.mysql.cj.jdbc.Driver").option("dbtable",
"hw3_1")
        .load()
        df2.select(col("ID_Тикета"),date_format(col("Status_Time"),"dd.MM.y
yyy hh.mm") as "Status_Time",col("Группа"),col("Статус"))
        .withColumn("Назначение", concat($"Status_Time", lit(" | "),
$"Статус", lit(" | "), $"Группа"))
        .write.format("jdbc").option("url","jdbc:mysql://localhost:3306/spa
rk_?user=root&password=vfhbfv")
        .option("driver", "com.mysql.cj.jdbc.Driver").option("dbtable",
"hw3_2")
        .mode("overwrite").save()

val qq = """
```

```
        SELECT m1.`ID_Тикета`,GROUP_CONCAT(m2.Status_Time,' |
',m2.`Статус`,' | ',m2.`Группа` ORDER BY m2.Status_Time SEPARATOR '\n') AS
`Статус`
        FROM spark_.hw3_3 m1
        JOIN (
            SELECT `ID_Тикета`,Status_Time,`Статус`,`Группа`
            FROM spark_.hw3_3
        ) m2 ON m1.`ID_Тикета` = m2.`ID_Тикета`
        GROUP BY m1.`ID_Тикета`
        """

var df3 =
spark.read.format("jdbc").option("url","jdbc:mysql://localhost:3306/spark_?
user=root&password=vfhbfv")
        .option("driver", "com.mysql.cj.jdbc.Driver").option("query", qq)
        .load()
        .write.format("jdbc").option("url","jdbc:mysql://localhost:3306/spa
rk_?user=root&password=vfhbfv")
        .option("driver", "com.mysql.cj.jdbc.Driver").option("dbtable",
"hw3_4")
        .mode("overwrite").save()


    println("home work 3")
}
val s0 = (System.currentTimeMillis() - t1)/1000
val s = s0 % 60
val m = (s0/60) % 60
val h = (s0/60/60) % 24
println("%02d:%02d:%02d".format(h, m, s))
System.exit(0)
```