**宽表及多表数据生成文档**

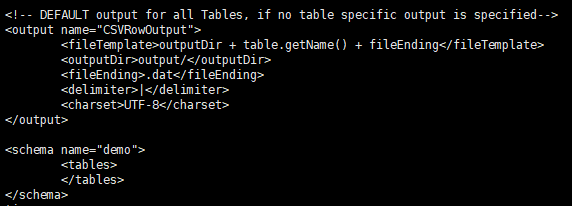
# DataHubble一张宽表数据生成说明文档

数据生成主要建立在数据基准TPCx-BB的基础上，同时参考了一些真实数据，根据系统优化以后，提供指定大小的一张宽表。

## （一）、数据生成

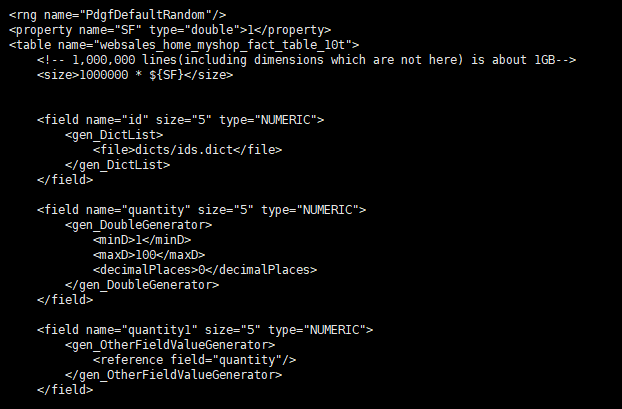
**1. 指定输出数据的格式和位置**

/Wide\_table/DataGeneration/pdgf/config/demo-generation.xml中指定了生成数据的存放位置和格式，可通过修改该文件参数自定义相关内容。



**2. 自定义数据表名称**

/Wide\_table/DataGeneration/pdgf/config/demo-schema.xml设置了表schema的具体参数，可修改生成的数据文件名称和每个字段的具体定义。

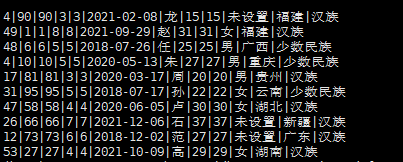


**3. 生成指定大小的数据文件，参数为要生成的数据大小（单位为GB）**

cd pdgf

./generate\_table.sh 1000

生成的数据表默认存放在/Wide\_table/DataGeneration/pdgf/output文件夹下，详细内容形如：



**4. 将生成的数据上传到HDFS并建立hive外部表fact：**

Create external table fact

(itemid INT,

quantity INT,

quantity1 INT,

discount DOUBLE,

discount1 DOUBLE,

solddate DATE,

customer STRING,

age INT,

age1 INT,

gender STRING,

province STRING,

nationality STRING)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' STORED AS TEXTFILE;

LOAD DATA LOCAL INPATH '/Wide\_table/DataGeneration/pdgf/output/websales\_home\_myshop\_fact\_table.dat' INTO TABLE fact;

关于TPCx-BB的具体内容,可参考<https://www.bankmark.de/products-and-services/pdgf/>。

而真实数据则作为dimension表（/Wide\_table/DataGeneration/table/dimension\_table.csv），上传到HDFS并建立hive外部表的方式类似，这里就不再赘述了。

**5. 在hive中将fact表和dimension表join生成最终的数据表，即为最终的宽表：**

Create external table if not exists websales\_home\_myshop\_10000

(itemname STRING,

price double,

price1 double,

quantity INT,

quantity1 INT,

discount DOUBLE,

discount1 DOUBLE,

category STRING,

solddate DATE,

customer STRING,

age INT,

age1 INT,

gender STRING,

province STRING,

nationality STRING,

itemdesc STRING);

insert into websales\_home\_myshop\_10000(itemname,price,price1,quantity,quantiry1,discount,discount1,category,solddate,customer,age,age1,gender,province,nationality,itemdesc)

select dimension.itemname,dimension.price,dimension.price,fact.quantity,fact.quantity,fact.discount,fact.discount,dimension.category,fact.solddate,fact.customer,fact.age,fact.age,fact.gender,fact.province,fact.nationality,dimension.itemdesc from

fact left outer join dimension

on fact.item\_id=dimension.item\_id;

## （二）、数据描述

1. fact表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 范围 | 描述 |
| item\_id | NUMERIC |  | 商品ID |
| quantity | NUMERIC | (1, 100) | 数量 |
| quantity1 | NUMERIC |  | 复制quantity字段 |
| discount | NUMERIC | (1, 9) | 折扣 |
| discount1 | NUMERIC |  | 复制discount字段 |
| solddate | DATE | (2017-01-01, 2022-01-01) | 销售日期 |
| customer | VARCHAR |  | 顾客姓名 |
| age | NUMERIC | (15, 40) | 顾客年龄 |
| age1 | NUMERIC |  | 复制age字段 |
| gender | CHAR |  | 顾客性别 |
| province | VARCHAR |  | 顾客省份 |
| nationality | VARCHAR |  | 顾客国籍 |

2. dimension表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 范围 | 描述 |
| item\_id | INT |  | 商品ID |
| itemname | STRING |  | 商品名称 |
| price | DOUBLE |  | 商品价格 |
| category | STRING |  | 商品类别 |
| desc | STRING |  | 商品描述 |

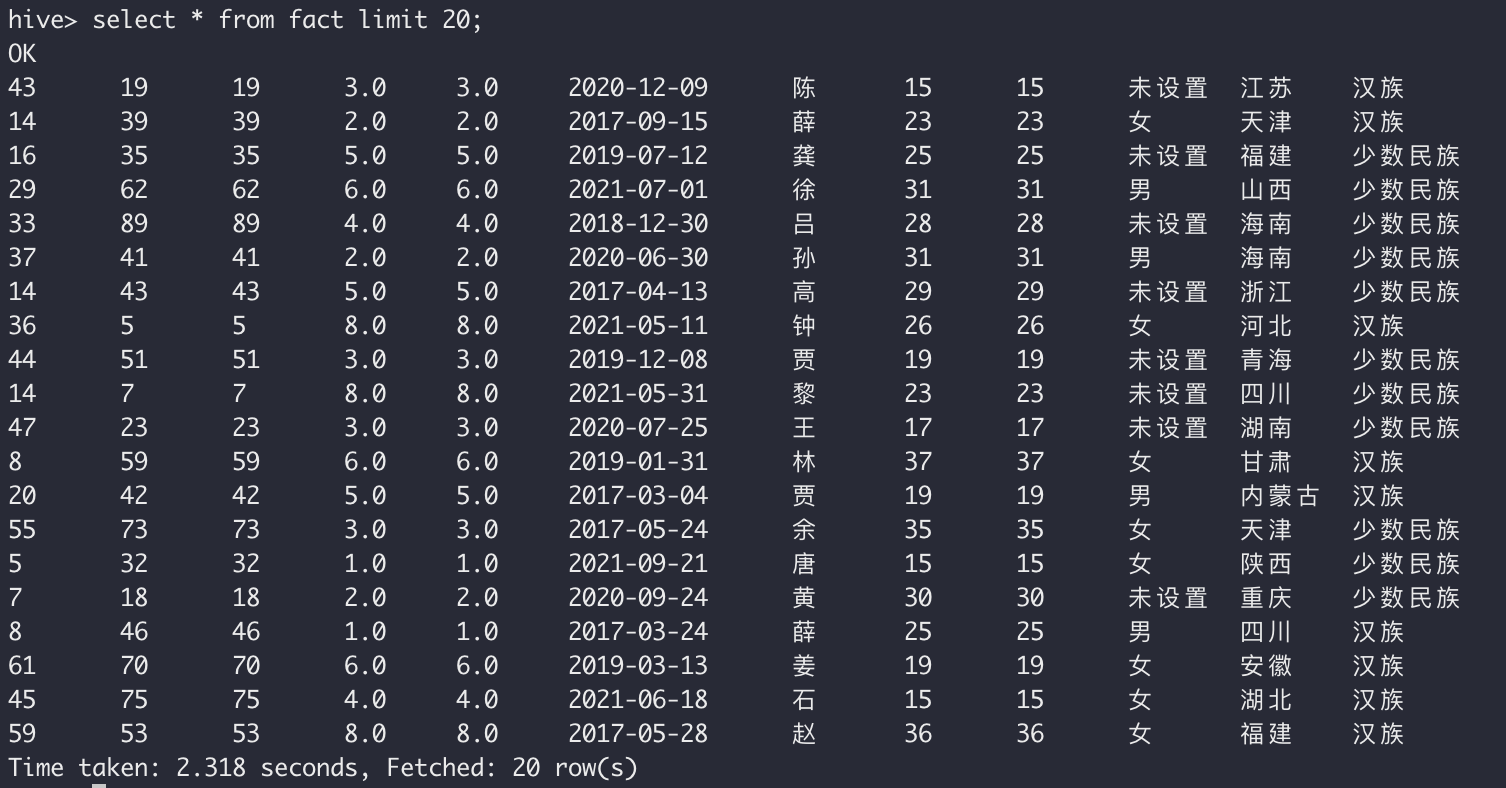
3. 最终的websales\_home\_myshop\_10000表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 范围 | 描述 |
| itemname | STRING |  | 商品名称 |
| price | DOUBLE |  | 商品价格 |
| price1 | DOUBLE |  | 商品价格 |
| quantity | NUMERIC | (1, 100) | 数量 |
| quantity1 | NUMERIC |  | 复制quantity字段 |
| discount | NUMERIC | (1, 9) | 折扣 |
| discount1 | NUMERIC |  | 复制discount字段 |
| category | STRING |  | 商品类别 |
| solddate | DATE | (2017-01-01, 2022-01-01) | 销售日期 |
| customer | VARCHAR |  | 顾客姓名 |
| age | NUMERIC | (15, 40) | 顾客年龄 |
| age1 | NUMERIC |  | 复制age字段 |
| gender | CHAR |  | 顾客性别 |
| province | VARCHAR |  | 顾客省份 |
| nationality | VARCHAR |  | 顾客国籍 |
| itemdesc | STRING |  | 商品描述 |

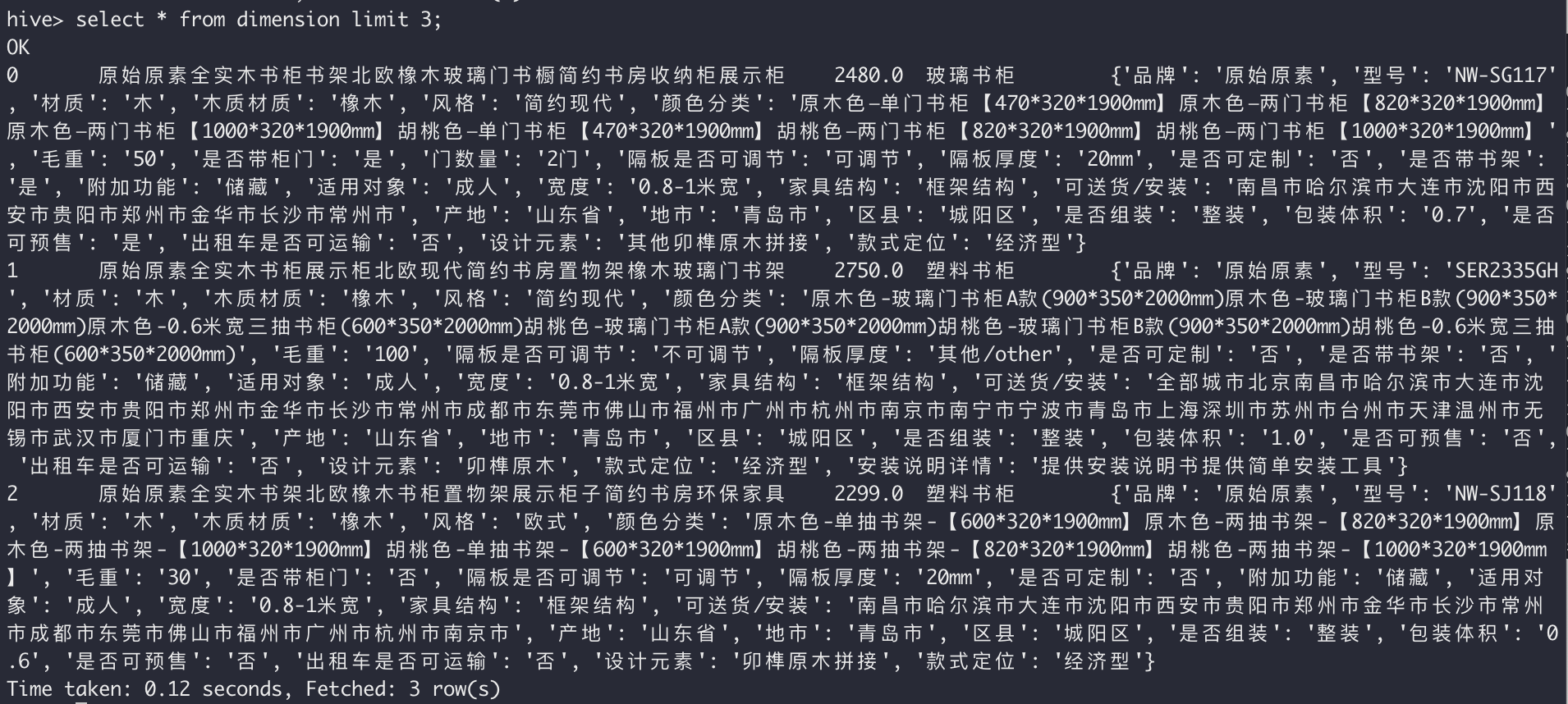
## （三）、数据样例

以下是生成的hive外部表数据，示例如下：

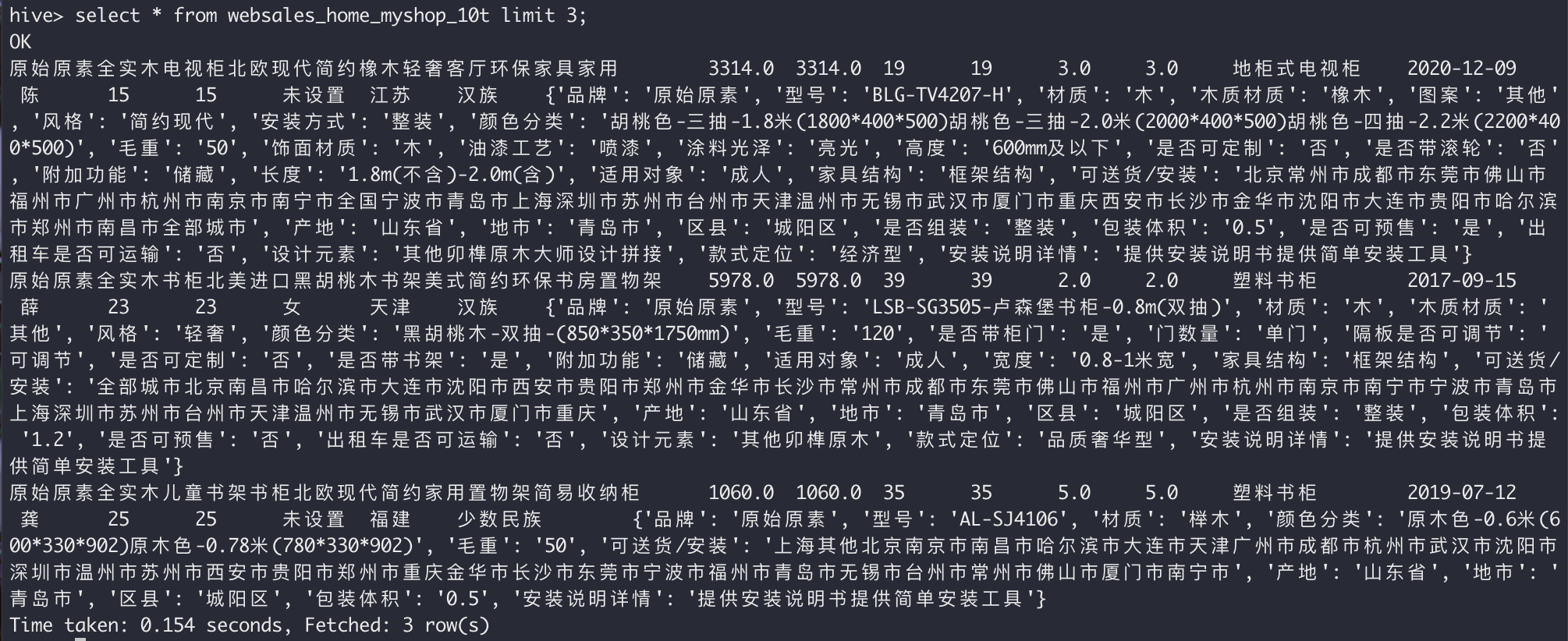
1. fact表



2. dimension表



3. websales\_home\_myshop\_10000表



# 阿里benchmark数据生成说明文档

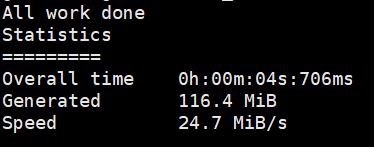
## （一）、数据生成

**1、数据生成**

数据生成使用了TPCx-BB Tools中data-generator文件夹中的pdgf.jar工具进行数据生成。生成指令如下：



该命令生成的9张表总大小约为100MB，使用默认SF=1（scale factor）



可以根据实际的需求调整sf的大小

java -jar pdgf.jar -l schema.xml -l generation.xml -c -s -sf 1

最终运行指令为：

上面的两个XML文件为config文件夹下的配置文件，其中schema.xml是对于生成数据的schema的描述，generation.xml则配置数据输出时的相应信息。

生成的数据在“/Multi\_table/pdgf/output”目录下，为“.dat“文件



**2、将数据导入hive**

首先定义表结构，以store\_returns表为例，如下所示：

CREATE TABLE store\_returns(sr\_returned\_date\_sk BIGINT, sr\_returned\_date\_sk1 BIGINT,sr\_item\_sk BIGINT, sr\_item\_sk1 BIGINT,sr\_customer\_skr BIGINT, sr\_customer\_sk1 BIGINT, sr\_ticket\_number BIGINT, sr\_ticket\_number1 BIGINT, sr\_return\_quantity INTEGER, sr\_return\_quantity1 INTEGER, sr\_return\_amt DECIMAL(7, 2), sr\_return\_amt1 DECIMAL(7, 2))

ROW FORMAT DELIMITED FIELDS TERMINATED BY '|'

STORED AS TEXTFILE;

之后将“.dat“文件中的数据导入hive中：

LOAD DATA LOCAL INPATH '/Multi\_table/pdgf/output

/store\_returns.dat' INTO TABLE ITEM\_RAW;

其余8张表也是类似的方法。

**3、融合真实数据：**

和宽表一样，需要对新建的多表中的item表进行更新，将生成的数据上传到HDFS并建立hive外部表的demension：

Create external table demension

(item\_id INT,

itemname STRING,

price DOUBLE,

category STRING,

desc STRING

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' STORED AS TEXTFILE;

LOAD DATA LOCAL INPATH '/Wide\_table/DataGeneration/table/dimension\_table.csv' INTO TABLE demension;

关于TPCx-BB的具体内容,可参考<https://www.bankmark.de/products-and-services/pdgf/>。

而真实数据则作为dimension表（/Wide\_table/DataGeneration/table/dimension\_table.csv），上传到HDFS并建立hive外部表的方式类似，这里就不再赘述了。

建立新的item表，此处为item\_2，将真实的数据从demension导入到item\_2中：

Create external table if not exists item\_2(i\_item\_sk BIGINT, i\_item\_sk1 BIGINT, i\_item\_id CHAR(16), i\_item\_desc VARCHAR(200), i\_current\_price DECIMAL(7, 2), i\_current\_price1 DECIMAL(7, 2), i\_class\_id INT, i\_class\_id1 INT, i\_category\_id INT, i\_category\_id1 INT, i\_category CHAR(50));

insert into item\_2(i\_item\_sk, i\_item\_sk1, i\_item\_id, i\_item\_desc, i\_current\_price, i\_current\_price1, i\_class\_id, i\_class\_id1, i\_category\_id, i\_category\_id1, i\_category)

select

item.i\_item\_sk, item.i\_item\_sk1, demension.itemname, demension.desc, demension.price, demension.price, item.i\_class\_id, item.i\_class\_id1, item.i\_category\_id, item.i\_category\_id1, demension.category from

item left outer join demension

on item.i\_item\_sk=demension.item\_id;

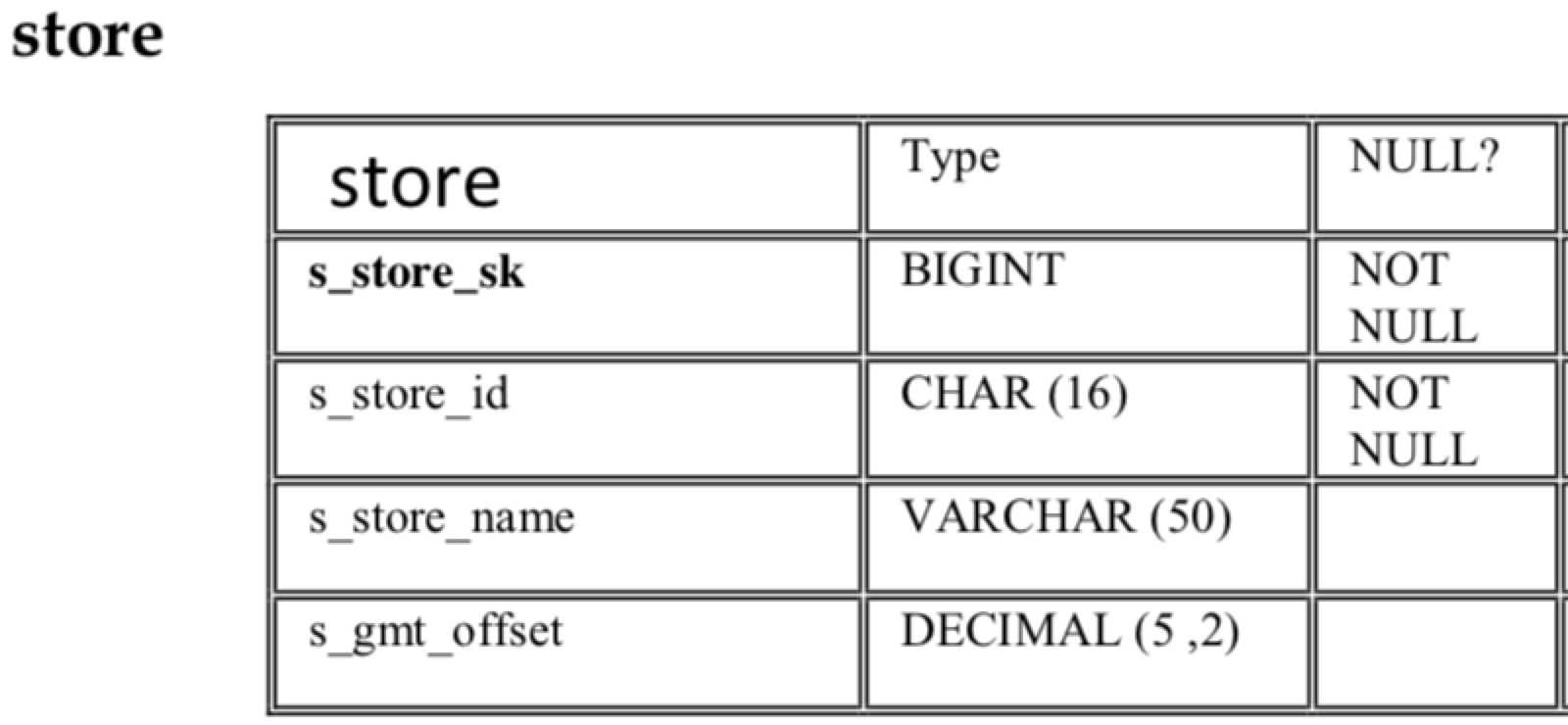
接下来对item\_2进行更名即可，这里不再赘述，新生成的item就是融合了真实数据的数据集。

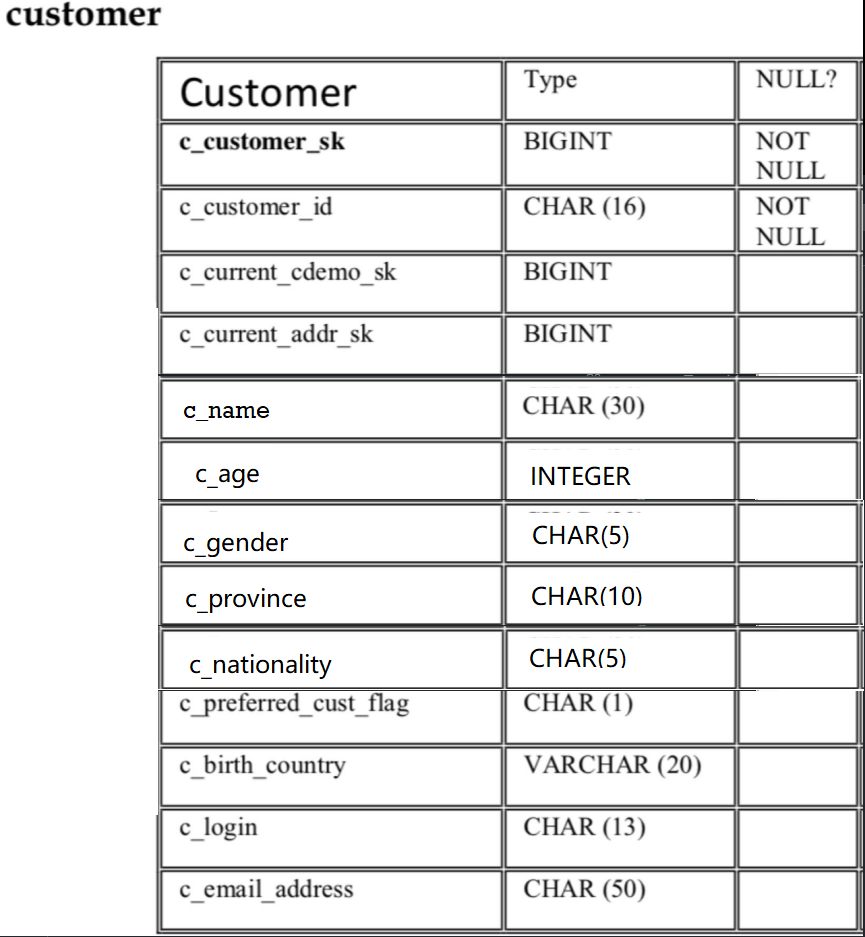
## （二）、数据描述

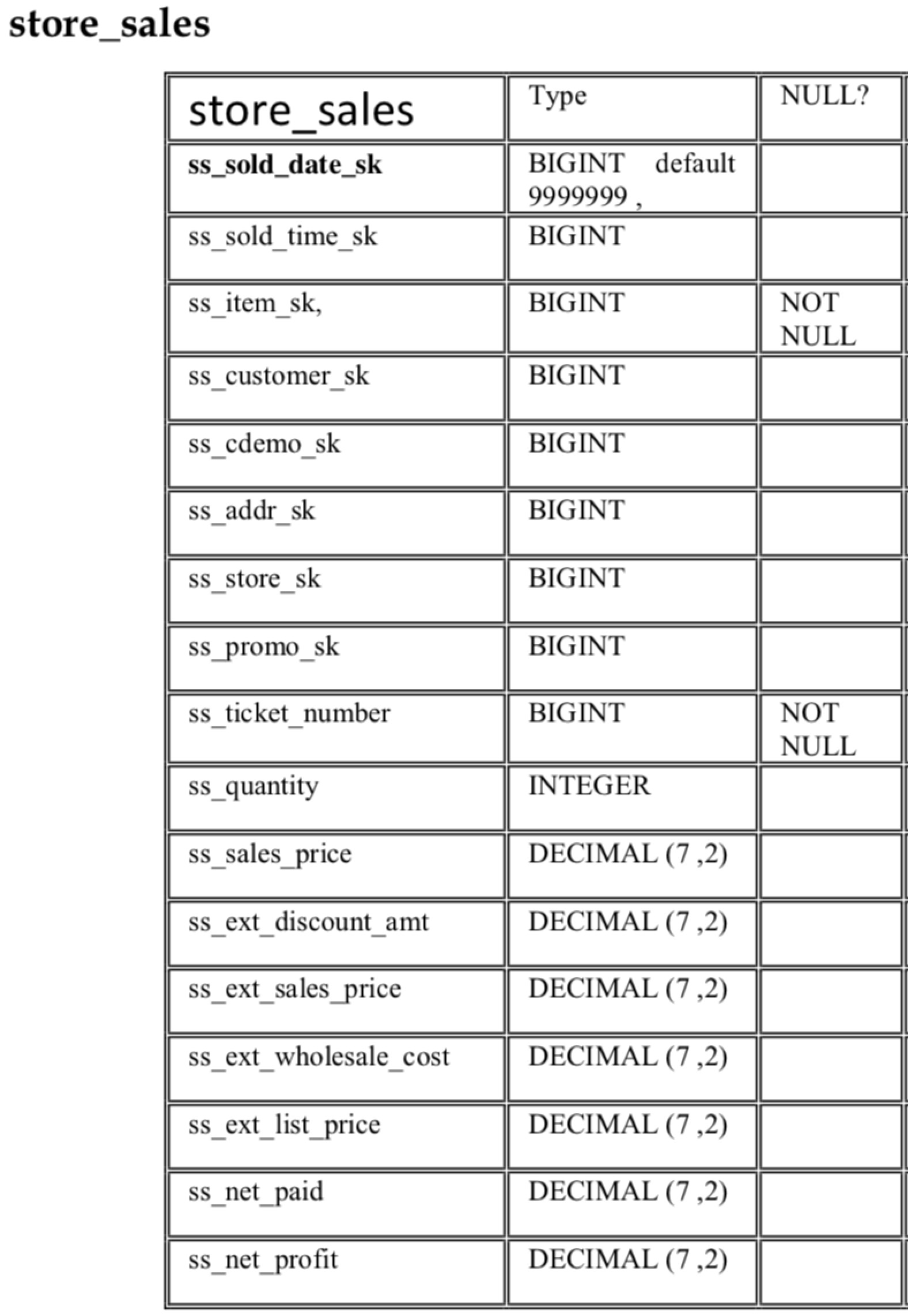
表功能描述

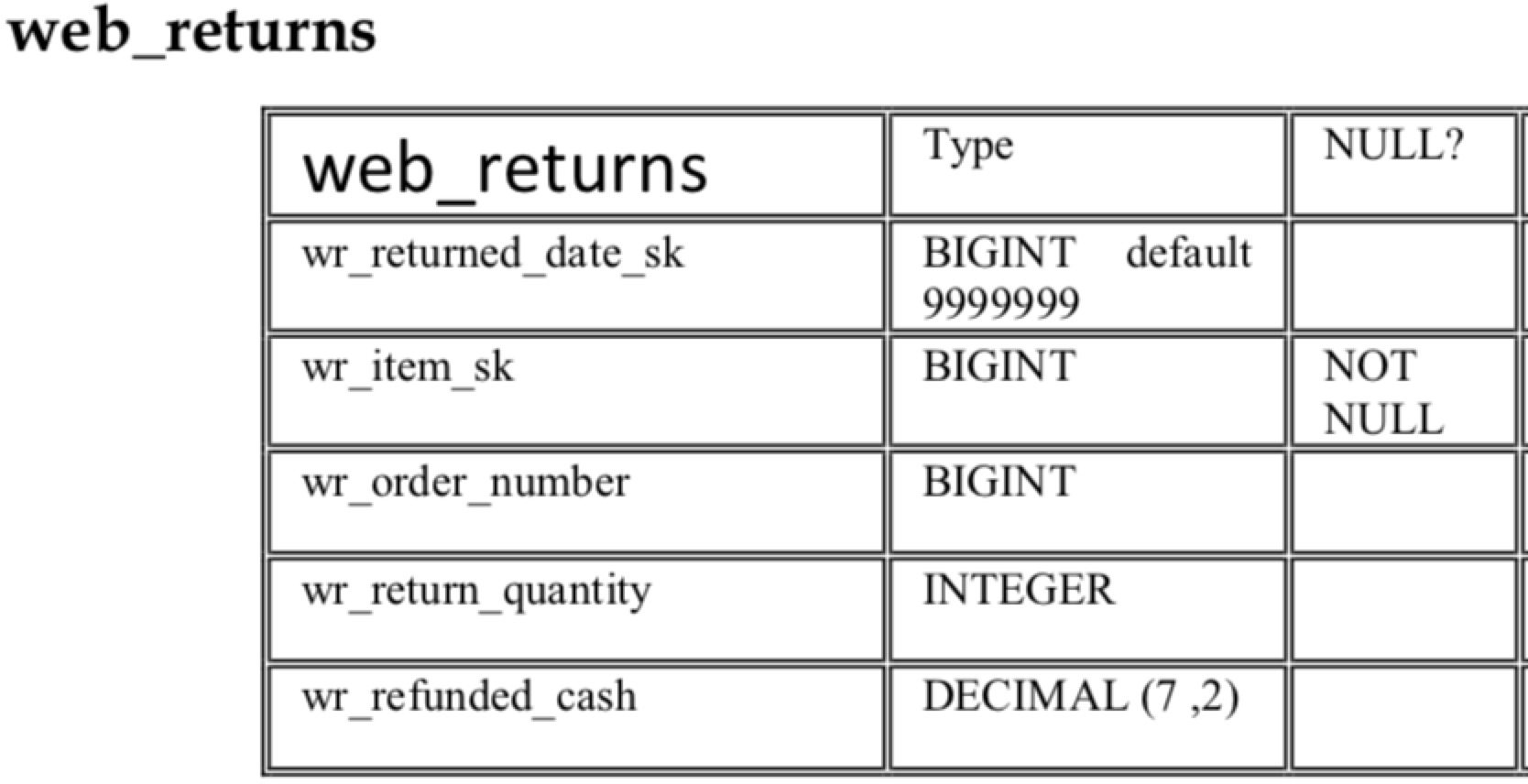
|  |  |
| --- | --- |
| 表名 | 描述 |
| web\_sales | 用来存储商品在网络上的销售情况信息 |
| web\_returns | 用来存储商品在网络上销售后发生退货的情况信息 |
| store | 用来存储商店的相关信息 |
| store\_sales | 用来存储商品在店内的销售情况信息 |
| store\_returns | 用来存储商店商品出现退货的情况信息 |
| customer | 用来存储客户的基本信息 |
| item | 用来存储商品的基本信息 |
| product\_reviews | 用来存储网络销售产品的相关评论信息 |
| date\_dim | 用来存储商品销售中与时间相关的基本信息 |

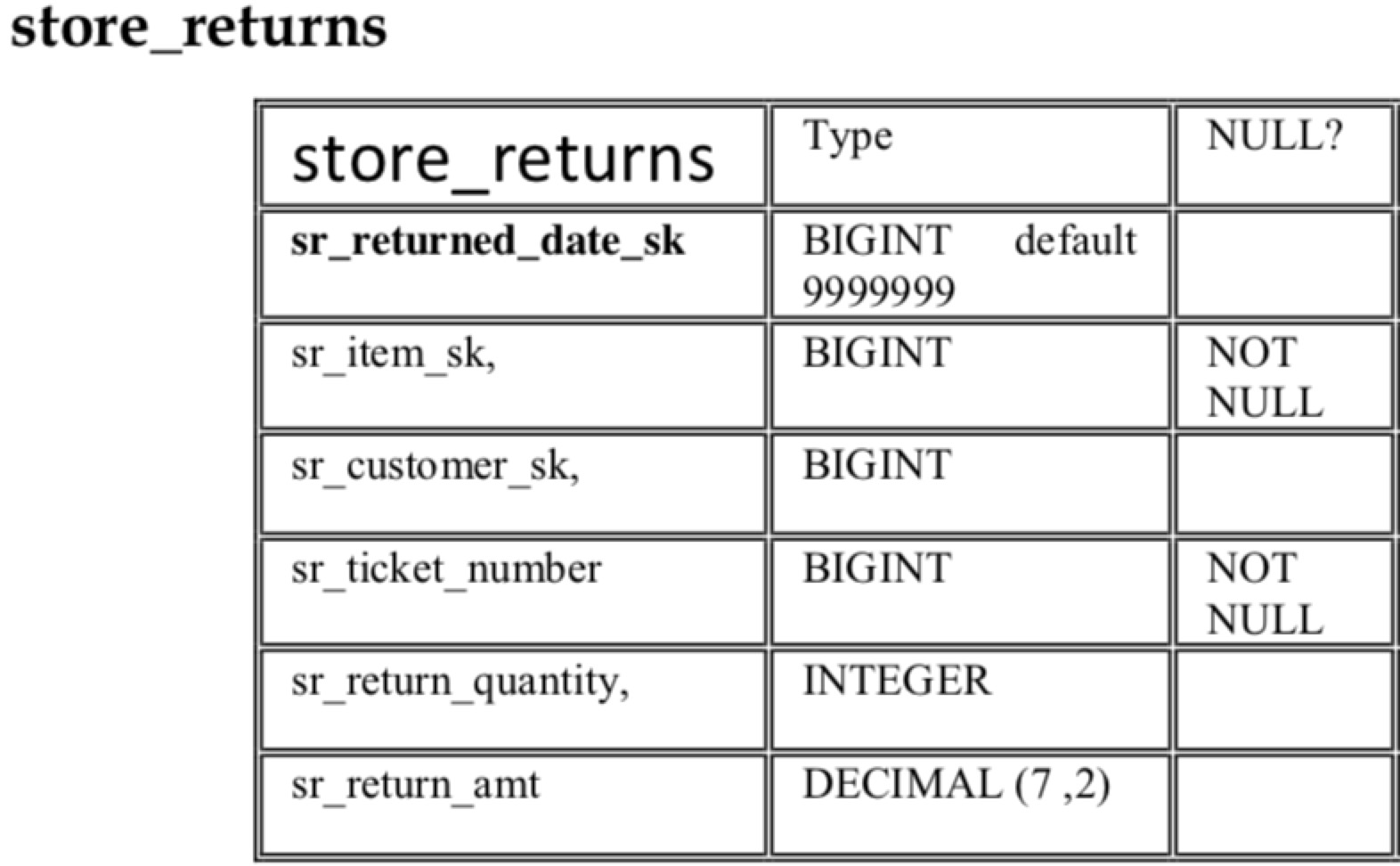
表的详细字段（省略数字型列的副本）

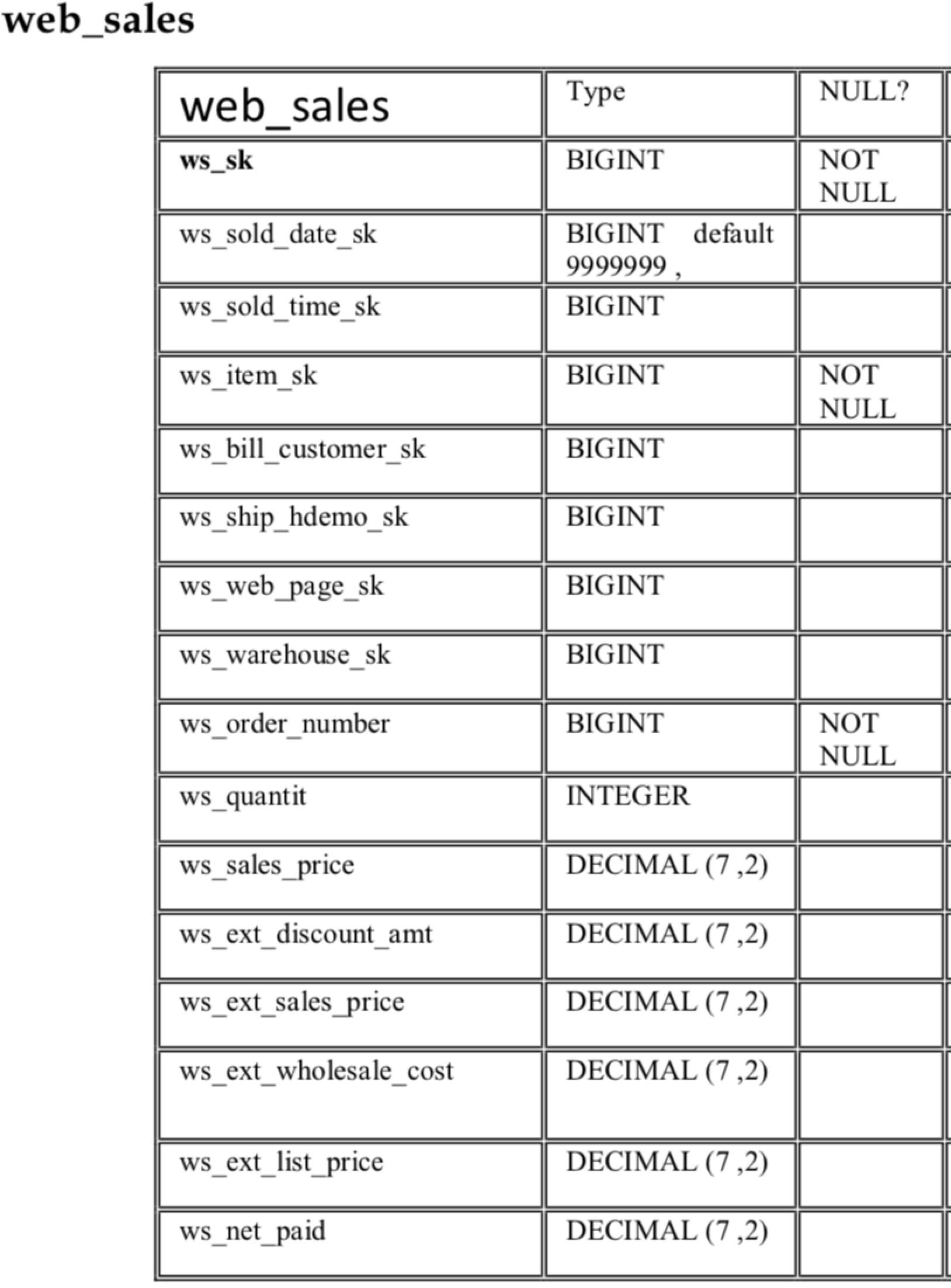


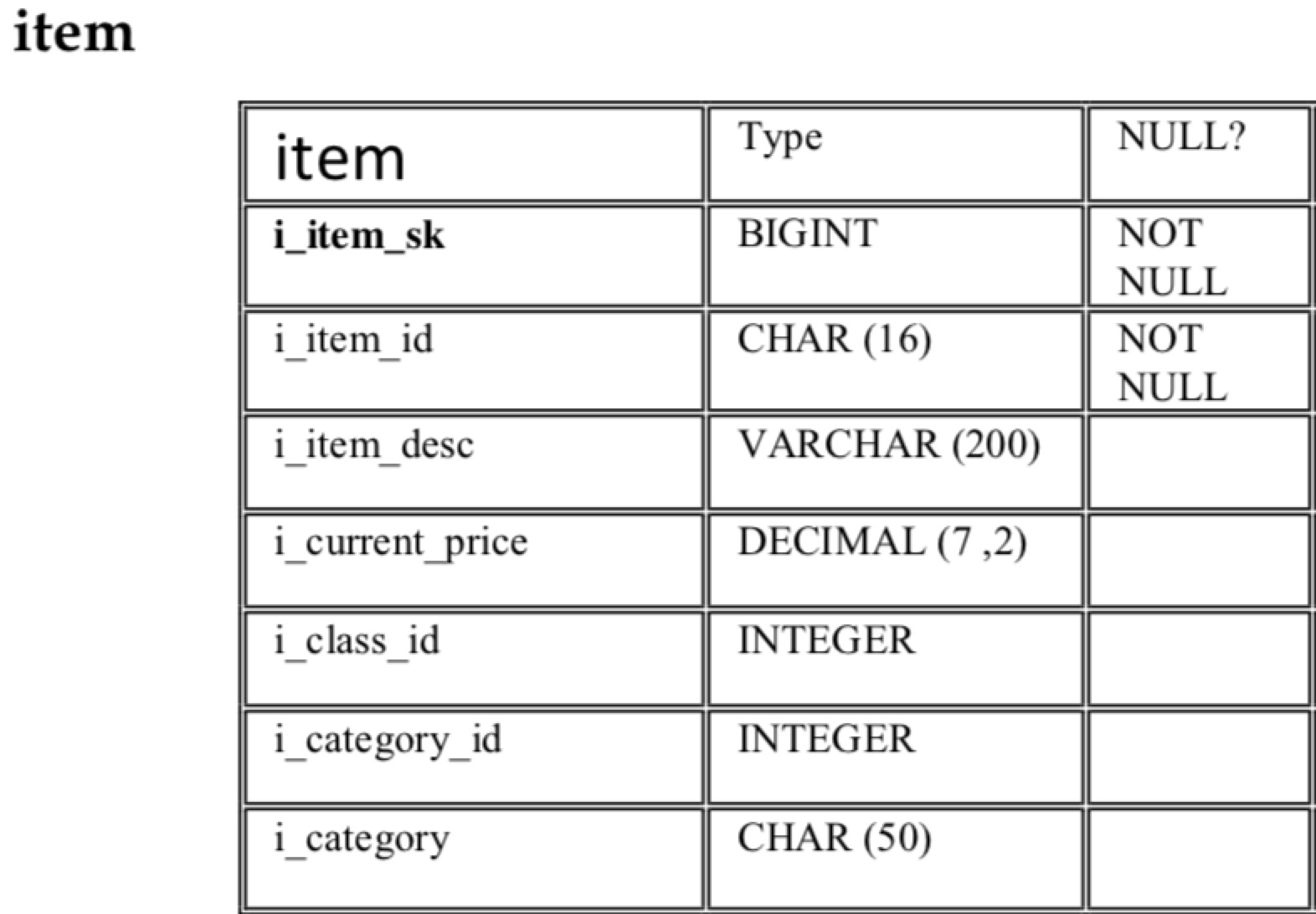


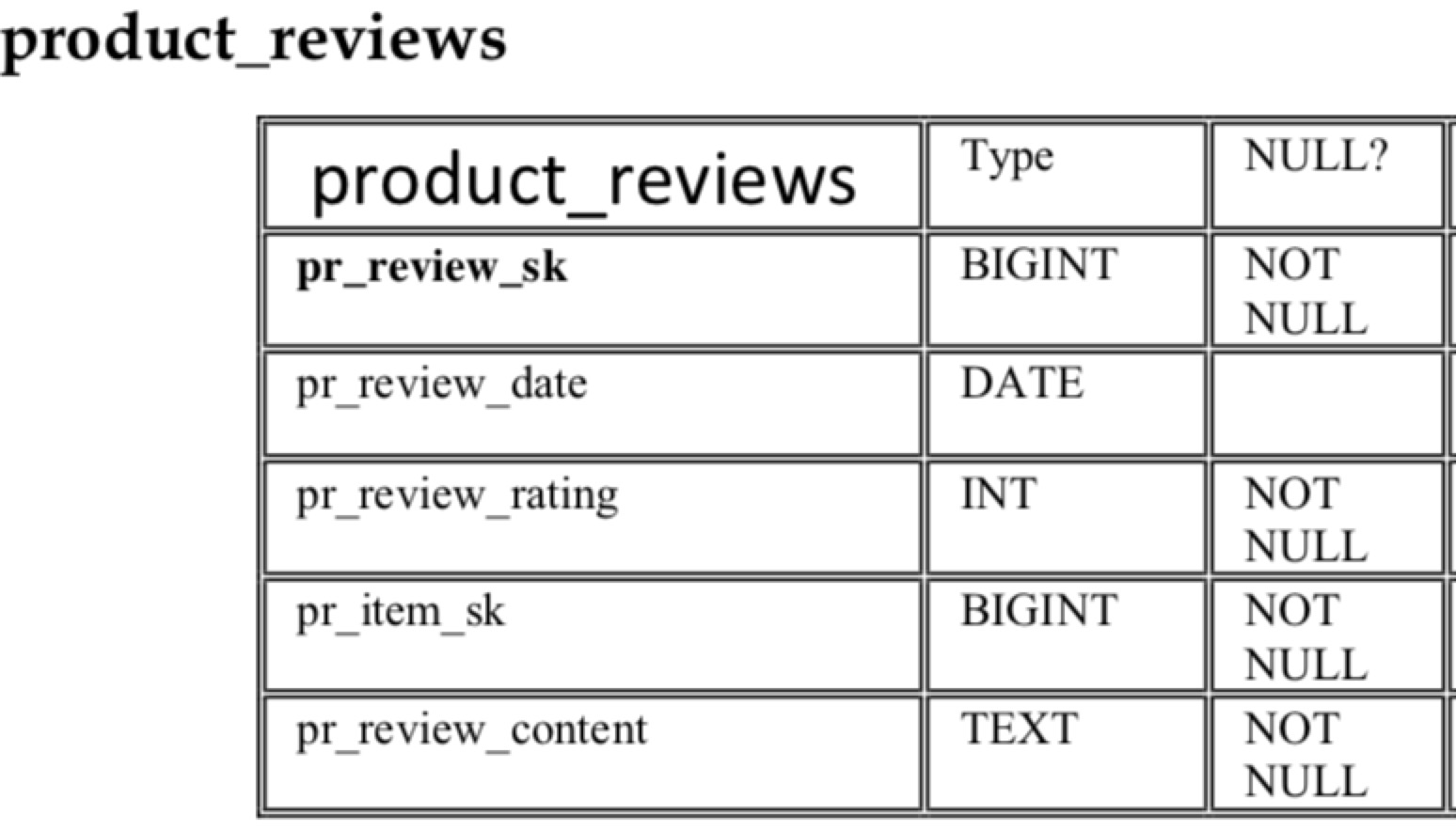


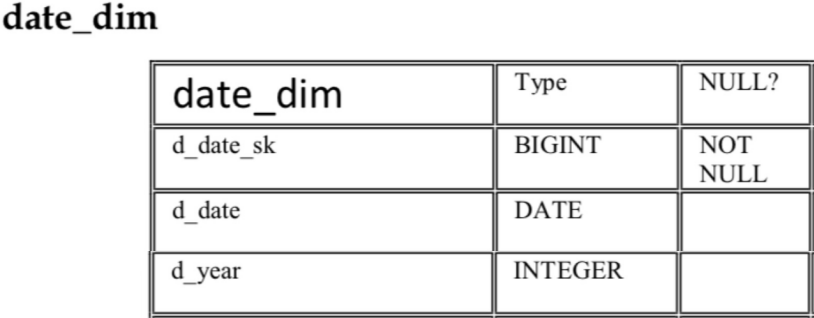












9张表的Schema如下图所示

web\_returns

wr\_item\_sk

wr\_order\_number

customer

customer\_sk

store

s\_store\_sk

web\_sale

ws\_item\_sk

ws\_customer\_sk

ws\_sold\_date\_sk

ws\_order\_number

date\_dim

d\_date\_sk

product\_reviews

pr\_item\_sk

store\_sales

ss\_item\_sk

ss\_store\_sk

ss\_customer\_sk

ss\_sold\_date\_sk

ss\_ticket\_number

store\_returns

sr\_item\_sk

sr\_customer\_sk

sr\_ticket\_number

item

i\_item\_sk