

Lab 9: Views

The learning objectives of this lab are to

- Create a simple view
- Manage database constraints in views using the WITH CHECK OPTION

9.1 Views

A **view** is a virtual table based on a SELECT query. The query can contain columns, computed columns, aliases, and aggregate functions from one or more tables. The tables on which the view is based are called **base tables**. You can create a view by using the **CREATE VIEW** command:

```
CREATE VIEW viewname AS SELECT query
```

The CREATE VIEW statement is a data definition command that stores the subquery specification—the SELECT statement used to generate the virtual table—in the data dictionary. For example, to create a view of only those Theme Parks where tickets have been sold you would do so as follows:

```
CREATE VIEW TPARKSSOLD AS

SELECT      *

FROM        THEMEPARK

WHERE       EXISTS (SELECT PARK_CODE FROM SALES

WHERE       SALES.PARK_CODE = THEMEPARK.PARK_CODE);
```

To display the contents of this view you would type

```
SELECT * FROM TPARKSSOLD;
```

The created view can be seen in figure 78.

```

mysql> CREATE VIEW TPARKSSOLD AS
-> SELECT *
-> FROM THEMEPARK
-> WHERE EXISTS (SELECT PARK_CODE FROM SALES
-> WHERE SALES.PARK_CODE = THEMEPARK.PARK_CODE);
Query OK, 0 rows affected (0.00 sec)

mysql> describe tparkssold;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| PARK_CODE  | varchar(10) | NO   |     |         |       |
| PARK_NAME  | varchar(35) | NO   |     |         |       |
| PARK_CITY  | varchar(50) | NO   |     |         |       |
| PARK_COUNTRY | char(2)    | NO   |     |         |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.05 sec)

mysql> select * from tparkssold;
+-----+-----+-----+-----+
| PARK_CODE | PARK_NAME | PARK_CITY | PARK_COUNTRY |
+-----+-----+-----+-----+
| FR1001    | FairyLand | PARIS     | FR            |
| UK3452    | PleasureLand | STOKE    | UK            |
| ZA1342    | GoldTown  | JOHANNESBURG | ZA            |
+-----+-----+-----+-----+
3 rows in set (0.02 sec)

mysql>

```

Figure 78 Creating the TPARKSSOLD view.

Task 9.1 Create the TPARKSSOLD view.

As you will have learned in Chapter 8, “Introduction to Structured Query Language”, relational view has several special characteristics. These are worth repeating here:

- You can use the name of a view anywhere a table name is expected in a SQL statement

- Views are dynamically updated. That is, the view is re-created on demand each time it is invoked. Therefore, if more tickets are sold in other Theme Parks, then those new ticket sales will automatically appear (or disappear) in the TPARKSSOLD view the next time it is invoked
- Views provide a level of security in the database because the view can restrict users to only specified columns and specified rows in a table

To remove the view TPARKSSOLD you could issue the following command

```
DROP VIEW TPARKSSOLD;
```

Task 9.2 Create a view called TICKET_SALES which contains details of the min, max and average sales at each Theme Park. The name of the theme park should also be displayed. Hint 1: you will need to join three tables. Hint 2: You will need to give the columns in the query that use the functions an alias. Once you have created your view, write a query to display the contents.

Task 9.3 Add your view TICKET_SALES and the associated DROP command to your themepark.sql scrip you created in lab 2.

9.2 Views – using the WITH CHECK OPTION

It is possible to perform referential integrity constraints through the use of a view so that database constraints can be enforced. The following view DISPLAYS employees who work in Theme Park FR1001 using the WITH CHECK OPTION clause. This clause ensures that INSERTs and UPDATEs cannot be performed on any rows that the view has not selected. The results of creating this view can be seen in Figure 79.

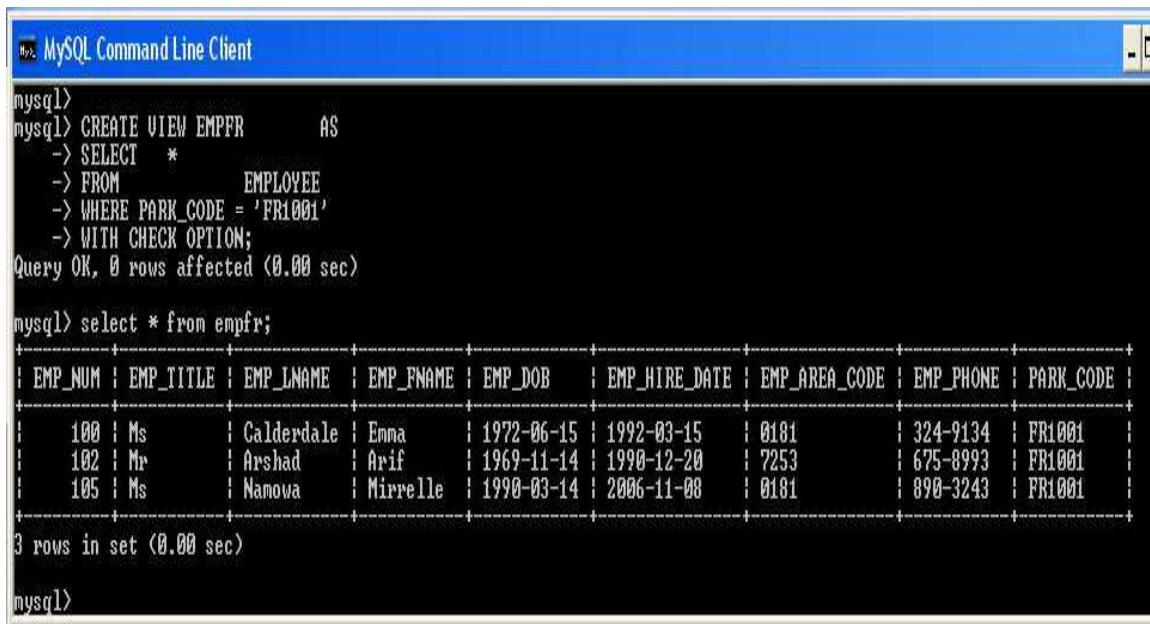
```
CREATE VIEW EMPFR      AS

SELECT      *

FROM        EMPLOYEE

WHERE PARK_CODE = 'FR1001'

WITH CHECK OPTION;
```



```
mysql>
mysql> CREATE VIEW EMPFR      AS
  -> SELECT      *
  -> FROM        EMPLOYEE
  -> WHERE PARK_CODE = 'FR1001'
  -> WITH CHECK OPTION;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from empfr;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| EMP_NUM | EMP_TITLE | EMP_LNAME | EMP_FNAME | EMP_DOB | EMP_HIRE_DATE | EMP_AREA_CODE | EMP_PHONE | PARK_CODE |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 100 | Ms | Calderdale | Emma | 1972-06-15 | 1992-03-15 | 0181 | 324-9134 | FR1001 |
| 102 | Mr | Arshad | Arif | 1969-11-14 | 1990-12-20 | 7253 | 675-8993 | FR1001 |
| 105 | Ms | Namova | Mirrelle | 1990-03-14 | 2006-11-08 | 0181 | 890-3243 | FR1001 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

Figure 79 Creating the EMPFR view

So for example if employee ‘Emma Caulderdale’ was to leave the park and move to park ‘UK3452’, we would want to update her information with the following query:

```
UPDATE EMPFR
```

```
SET PARK_CODE = 'UK3452'
```

```
WHERE EMP_NUM = 100;
```

However running this update gives the errors shown in Figure 80. This is because if the update was to occur, the view would no longer be able to see this employee.

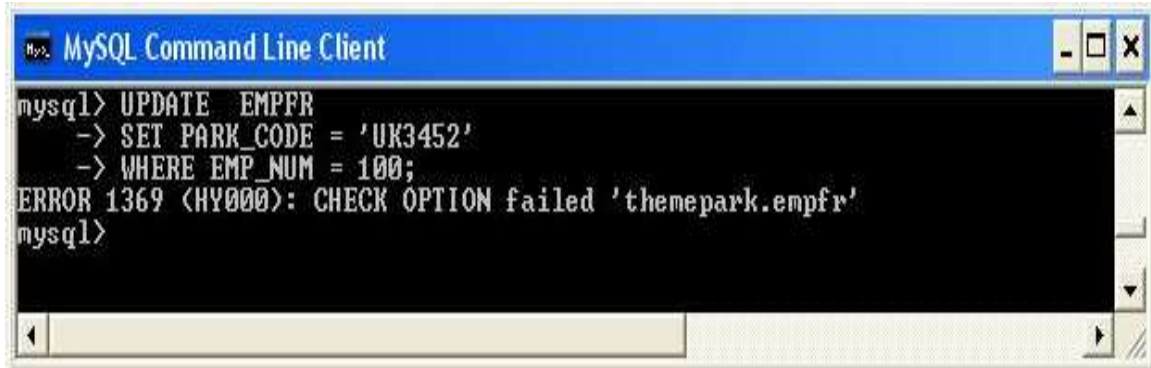


Figure 80 Creating the EMPFR view

Task 9.4 Create the view EMPFR and tray and update the Theme Park that employee number 101 works in.

Task 9.5. Employee Emma Cauderdale (EMP_NUM =100) has now changed her phone number to 324-9652. Update her information in the EMPFR view. Write a query to show her new phone number has been updated.

Task 9.6 Remove the EMPFR view.

9.4 Exercises

E9.1 The Theme Park managers want to create a view called EMP_DETAILS which contains the following information. EMP_NO, PARK_CODE, PARK_NAME, EMP_LNAME_EMP_FNAME, EMP_HIRE_DATE and EMP_DOB. The view should only be read only.

E9.2 Check that the view works, by displaying its contents.

E9.3 Using your view EMP_DETAILS, write a query that displays all employee first and last names and the park names.

E9.4 Remove the view EMPDETAILS.