

Views

Thomas Schwarz, SJ

Virtual Views

- Relations can be real
 - CREATE TABLE ...
- or virtual
 - CREATE VIEW
 - Do not exist physically
 - Defined through a query like expression
 - Can be queried as if they are real tables

Virtual Views

- SQL Programming Language:
 - Table: Relation that exists
 - View: Relation that is virtual
 - Temporary: Created while a query is executed and afterwards discarded

Virtual Views

- Another perspective:
 - Frequent queries
 - Can be typed in
 - Can be made into an sql script
 - Can be made into a Java / Python / ... script
 - Or can be made into a view
 - Views are frozen queries?!?

Creating Views

- Views are defined via CREATE VIEW

```
CREATE VIEW MGMMovies AS
  SELECT title, year
  FROM Movies
  WHERE studioName = 'MGM';
```

Creating Views

```
movies(title, year, length, genre, studioName, producerC#)  
    movieExec(name, address, cert#, netWorth)
```

```
CREATE VIEW MovieProd AS  
    SELECT title, name  
    FROM movies, movieExec  
    WHERE producerC# = cert#;
```

Creating Views

```
SELECT * FROM classicmodels.employees;
```

```
CREATE VIEW managers AS
```

```
    SELECT
```

```
        employeeNumber, firstName, lastName, jobTitle
```

```
FROM
```

```
    employees
```

```
WHERE
```

```
    jobTitle = 'President'
```

```
    OR jobtitle LIKE '%VP %'
```

```
    OR jobtitle LIKE '% Manager %';
```

Creating Views

- We can now access the view as a normal table

```
SELECT * FROM classicmodels.managers;
```

employeeNumb...	firstName	lastName	jobTitle
▶ 1002	Diane	Murphy	President
1056	Mary	Patterson	VP Sales
1076	Jeff	Firrelli	VP Marketing
1088	William	Patterson	Sales Manager (APAC)
1102	Gerard	Bondur	Sale Manager (EMEA)
1143	Anthony	Bow	Sales Manager (NA)

Creating Views

- We can now access the view as a normal table:
- People reporting to someone with a last name in manager

```
SELECT
    e.firstName,
    e.lastName,
    e.jobTitle,
    e2.firstName,
    e2.lastName,
    e2.jobTitle
FROM
    employees e,
    employees e2,
    managers m
WHERE
    e.reportsTo = e2.employeeNumber
    AND e2.lastName = m.lastName;
```

Creating Views

- We can now access the view as a normal table:
 - People reporting to someone with a last name in manager

Result Grid						
Filter Rows:			Search	Export:		
firstName	lastName	jobTitle	firstName	lastName	jobTitle	
Gerard	Bondur	Sale Manager (EMEA)	Mary	Patterson	VP Sales	
William	Patterson	Sales Manager (APAC)	Mary	Patterson	VP Sales	
Mami	Nishi	Sales Rep	Mary	Patterson	VP Sales	
Tom	King	Sales Rep	William	Patterson	Sales Manager (APAC)	
Peter	Marsh	Sales Rep	William	Patterson	Sales Manager (APAC)	
Andy	Fixter	Sales Rep	William	Patterson	Sales Manager (APAC)	
Anthony	Bow	Sales Manager (NA)	Mary	Patterson	VP Sales	
Gerard	Bondur	Sale Manager (EMEA)	Mary	Patterson	VP Sales	
William	Patterson	Sales Manager (APAC)	Mary	Patterson	VP Sales	
Martin	Gerard	Sales Rep	Gerard	Bondur	Sale Manager (EMEA)	
Barry	Jones	Sales Rep	Gerard	Bondur	Sale Manager (EMEA)	
Larry	Bott	Sales Rep	Gerard	Bondur	Sale Manager (EMEA)	
Pamela	Castillo	Sales Rep	Gerard	Bondur	Sale Manager (EMEA)	
Gerard	Hernandez	Sales Rep	Gerard	Bondur	Sale Manager (EMEA)	
Loui	Bondur	Sales Rep	Gerard	Bondur	Sale Manager (EMEA)	
George	Vanauf	Sales Rep	Anthony	Bow	Sales Manager (NA)	
Foon Yue	Tseng	Sales Rep	Anthony	Bow	Sales Manager (NA)	
Steve	Patterson	Sales Rep	Anthony	Bow	Sales Manager (NA)	
Julie	Firrelli	Sales Rep	Anthony	Bow	Sales Manager (NA)	
Leslie	Thompson	Sales Rep	Anthony	Bow	Sales Manager (NA)	
Leslie	Jennings	Sales Rep	Anthony	Bow	Sales Manager (NA)	

Creating Views

- You can create views that do not depend on tables

```
CREATE VIEW daysofweek (day) AS
  SELECT 'Mon'
  UNION
  SELECT 'Tue'
  UNION
  SELECT 'Web'
  UNION
  SELECT 'Thu'
  UNION
  SELECT 'Fri'
  UNION
  SELECT 'Sat'
  UNION
  SELECT 'Sun';
```

Creating Views

- You get rid of views by using a drop statement




```
DROP VIEW managers;
```

Creating Views

- Or you can alter a view:
 - Managers have someone that reports to them

```
ALTER VIEW managers AS
SELECT DISTINCT (e.employeeNumber), e.firstName,
e.lastName, e.jobTitle
FROM employees e, employees e2
WHERE e2.reportsTo = e.employeeNumber;
```

Creating Views

Result Grid   Filter Rows: <input type="text" value="Search"/> Export: 				
	employeeNumb...	firstName	lastName	jobTitle
▶	1002	Diane	Murphy	President
	1056	Mary	Patterson	VP Sales
	1088	William	Patterson	Sales Manager (APAC)
	1102	Gerard	Bondur	Sale Manager (EMEA)
	1143	Anthony	Bow	Sales Manager (NA)
	1621	Mami	Nishi	Sales Rep

Creating Views

- You can name columns when you create a view

```
CREATE OR REPLACE VIEW customerOrders AS
SELECT
    orderNumber,
    customerName,
    SUM(quantityOrdered * priceEach) total
FROM
    orderDetails
INNER JOIN orders o USING (orderNumber)
INNER JOIN customers USING (customerNumber)
GROUP BY orderNumber;
```

Creating Views

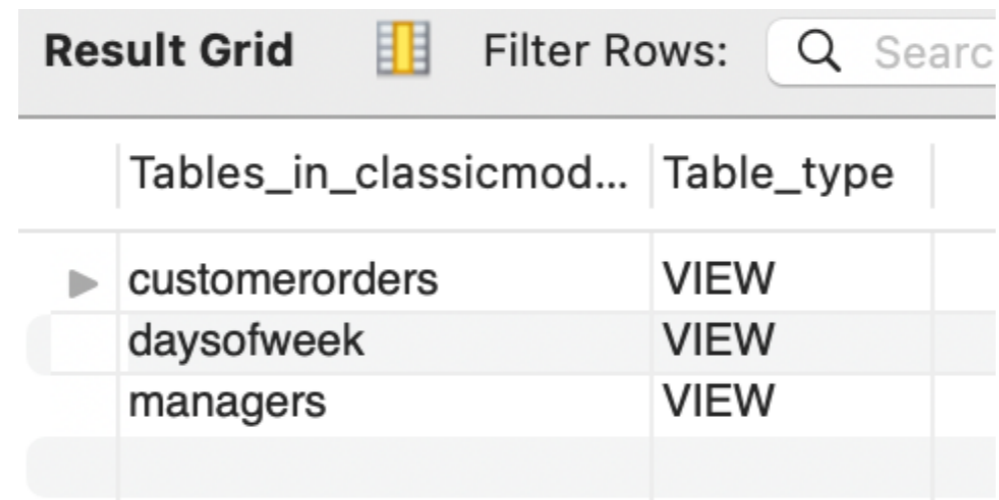
```
SELECT customerName, count(orderNumber) AS nrOrders
FROM customerorders
GROUP BY customerNumber
ORDER BY nrOrders DESC;
```


Result Grid		Filter Rows:
customerName	nrOrders	
▶ Euro+ Shopping Channel	26	
Mini Gifts Distributors Ltd.	17	
Reims Collectables	5	
Down Under Souvenirs, Inc	5	
Australian Collectors, Co.	5	
Dragon Souvenirs, Ltd.	5	
Danish Wholesale Imports	5	
Kelly's Gift Shop	4	
Land of Toys Inc.	4	
Blauer See Auto, Co.	4	
Volvo Model Replicas, Co	4	
Baane Mini Imports	4	
The Sharp Gifts Warehouse	4	
La Rochelle Gifts	4	
Muscle Machine Inc	4	

Creating Views

- You can check on your defined views by

```
SHOW FULL TABLES  
WHERE table_type = 'VIEW';
```



Result Grid  Filter Rows:

Tables_in_classicmod...	Table_type
▶ customerorders	VIEW
daysofweek	VIEW
managers	VIEW

Creating Views

```
SELECT
    table_name view_name
FROM
    information_schema.tables
WHERE
    table_type = 'VIEW' AND
    table_schema = 'classicmodels';
```

Result Grid	
view_name	
▶ customerorders	
daysofweek	
managers	

Interacting with Views

- Interacting with Views
 - A view, once defined, can be queried just like a real table

```
SELECT title  
FROM MGMMovies  
WHERE year = 1979;
```

Interacting with Views

```
starName(title, year, name)
```

```
SELECT DISTINCT starName  
FROM MGMMovies, starsIn  
WHERE title = movieTitle AND year = movieYear
```

Interacting with Views

- We can rename the attributes in a VIEW

```
CREATE VIEW movieProd(movieTitle, prodName) AS
    SELECT title, name
    FROM movies, movieExec
    WHERE producerC# = cert#;
```

- attribute names in the view are now movieTitle and prodName

Exercises

```
movieStar(name, address, gender, birthday)
movieExec(name, address, cert#, netWorth)
studio(name, address, presC#)
```

- A view RichExec with name address, certificate number, and net-worth of all executives with more than 10 million net-worth

Exercises

```
movieStar(name, address, gender, birthday)
movieExec(name, address, cert#, netWorth)
studio(name, address, presC#)
```

- A view RichExec with name, address, certificate number, and net-worth of all executives with more than 10 million net-worth

```
CREATE VIEW RichExec(execName, execAddress, cert#,
netWorth) AS
    SELECT name, address, cert#, netWorth
    WHERE netWorth > 10000000;
```

Exercises

```
movieStar(name, address, gender, birthday)
movieExec(name, address, cert#, netWorth)
studio(name, address, presC#)
```

- A view StudioPres with name, address, netWorth of studio presidents

Exercises

```
movieStar(name, address, gender, birthday)
movieExec(name, address, cert#, netWorth)
studio(name, address, presC#)
```

- A view StudioPres with name, address, netWorth of studio presidents

```
CREATE VIEW StudioPres AS
  SELECT name, address, netWorth
  FROM movieExec
  WHERE cert# IN (
    SELECT presC#
    FROM studio );
```

Exercises

```
movieStar(name, address, gender, birthday)
movieExec(name, address, cert#, netWorth)
studio(name, address, presC#)
```

- A view ExecutiveStar giving the name, address, gender, birth date and certificate number of movie stars that are also movie executives
 - Assume that executives with the same name and address as a movie star are the movie star
 - Even though there is no reason to assume this

Exercises

```
movieStar(name, address, gender, birthday)
movieExec(name, address, cert#, netWorth)
studio(name, address, presC#)
```

- A view ExecutiveStar giving the name, address, gender, birth date and certificate number of executives that are also movie executives

```
CREATE VIEW ExecutiveStar AS
SELECT ms.name, ms.address, ms.gender,
       ms.birthdate, me.cert#
FROM movieStar ms, movieExec me
WHERE ms.name = me.name AND ms.address = me.address
```

View Algorithms in MySQL

- Views are defined in the SQL standard but DBMS are free to add to them
- MySQL has an optional algorithm field
 - Determines how views are integrated into queries

```
CREATE [OR REPLACE] [ALGORITHM = {MERGE | TEMPTABLE |  
UNDEFINED}] VIEW  
    view_name[(column_list)]  
AS  
    select-statement;
```

View Algorithms in MySQL

- Merge:
 - Merge the input query with the SELECT statement in the view
 - Execute the combined query

View Algorithms in MySQL

- Example:

```
CREATE OR REPLACE
  ALGORITHM = MERGE
VIEW hr_contacts AS
  SELECT
    employeeNumber,
    CONCAT(firstName, ' ', lastName) AS emp_name,
    email,
    CONCAT(phone, ' ', extension) AS emp_phone
  FROM
    employees
    INNER JOIN
    offices USING (officeCode);
```

View Algorithms in MySQL

- Issue a query

```
SELECT DISTINCT
    hrc.emp_name
FROM
    hr_contacts hrc,
    customers cus
WHERE
    cus.country = 'USA'
    AND cus.salesRepEmployeeNumber =
        hrc.employeeNumber
ORDER BY hrc.emp_name ASC;
```

View Algorithms in MySQL

- This query and the view query are then merged

```
SELECT DISTINCT
    CONCAT(emp.firstName, ' ', emp.lastName) AS ename
FROM
    employees emp,
    customers cus
WHERE
    cus.country = 'USA'
    AND cus.salesRepEmployeeNumber =
        emp.employeeNumber
ORDER BY ename ASC;
```


View Algorithms in MySQL

- If you use `TEMPTABLE` instead, then
 - MySQL creates a temporary table to store the view
 - Execute the query using the temporary table
- Temporary table will be created every time anew

Modifying Views

- Some views can be used to update the underlying tables
- View Removal
 - `DROP VIEW MGMMovies`
- Just like Table removal
 - `DROP TABLE movies`
 - which would also make the view MGMMovies unusable

Modifying Views

- Updatable views
 - SQL has clear, but complicated definitions when a view can be updated (and an underlying table changed)
 - View must be defined by SELECT
 - There is only one relation R in the definition
 - No subquery involving R in the WHERE clause
 - Enough attributes of R are involved in the view

Modifying Views

- MGMMovies fulfills the requirements
- If we insert via the view:
 - ```
INSERT INTO MGMMovies
VALUES ('Get Shorty', 1995)
```
  - movies will get a new tuple
    - title: 'Get Shorty', year: 1995
    - Everything else: Null
- Interestingly, because of the latter, the view itself would not be updated

```
movies(title, year, length, genre, studioName, producerC#)
```

# Modifying Views

- The view insertion

```
INSERT INTO MGMMovies
VALUES ('Get Shorty', 1995)
```

- has the same effect as inserting into the underlying table

```
INSERT INTO movies
VALUES ('Get Shorty', 1995)
```

# Modifying Views

- To address this anomaly, need to add to the view

```
CREATE OR REPLACE VIEW MGMMovies(name, title, studio) AS
 SELECT name, title, studioName
 FROM movies
 WHERE studio = 'MGM';
```

# Modifying Views

- Now it works

```
INSERT INTO MGMMovies
VALUES ('Find Shorty', 1995, 'MGM')
```

- which is equivalent to

```
INSERT INTO movies(name, year, studioName)
VALUES ('Find Shorty', 1995, 'MGM')
```

- and assumes that we do not have any triggers or constraints against NULL values for the other attributes
- but now the view also changes

# Modifying Views

- Deletions are also passed through the underlying table

- ```
DELETE FROM MGMMovies
WHERE title LIKE '%Shorty%';
```

- gets translated into

```
DELETE FROM movies
WHERE title LIKE '%Shorty%' AND studioName = 'MGM';
```


Modifying Views

```
UPDATE MGMMovies  
SET year = 1968  
WHERE title = 'Get Shorty';
```

- becomes

```
UPDATE movies  
SET year = 1968  
WHERE title = 'Get Shorty' AND  
      studioName = 'MGM';
```

Modifying Views

- Including all properties in a view is a kludge
 - Can use a trigger instead
 - Use the INSTEAD OF syntax

```
CREATE TRIGGER mgmInserts
INSTEAD OF INSERT ON mgmInserts
REFERENCING NEW ROW as newRow
FOR EACH ROW
INSERT INTO movies(title, year, studioName)
VALUES(newRow.title, newRow.year, 'MGM');
```

Modifying Views in MySQL

- MySQL only started to support views in Version 5 (2008)
- Supports updatable views
 - But not the INSTEAD trigger

Updating Views in MySQL

- Create a view

```
CREATE VIEW officeInfo AS
  SELECT
    officeCode, phone, city
  FROM
    offices;
```

Updating Views in MySQL

- We can query the view

```
SELECT
    *
FROM
    officeinfo;
```

Result Grid				Filter Rows:	Search	Export
	officeCode	phone	city			
▶	1	+1 650 219 4782	San Francisco			
▶	2	+1 215 837 0825	Boston			
▶	3	+1 212 555 3000	NYC			
▶	4	+33 14 723 4404	Paris			
▶	5	+81 33 224 5000	Tokyo			
▶	6	+61 2 9264 2451	Sydney			
▶	7	+44 20 7877 2041	London			

Updating Views in MySQL


- We can update the view

```
UPDATE officeinfo
SET
    phone = '+01 408 7241044'
WHERE
    city LIKE '%San Francisco%';
```


Updating Views in MySQL

- Insertion does **not** work because there are no default values for other columns

```
INSERT INTO officeinfo (  
    officeCode,  
    phone,  
    city  
)  
VALUES (  
    13,  
    '01 408 724 1999',  
    'San Jose'  
);
```



Error Code: 1423. Field of view
'classicmodels.officeinfo' underlying table
doesn't have a default value

Updating Views in MySQL

- Create a view for VPs

```
CREATE OR REPLACE VIEW vps AS
SELECT
    employeeNumber,
    lastName,
    firstName,
    jobTitle,
    extension,
    email,
    officeCode,
    reportsTo
FROM
    employees
WHERE
    jobTitle LIKE '%VP%';
```

employeeNumb...	lastName	firstName	jobTitle	extension	email
▶ 1056	Patterson	Mary	VP Sales	x4611	mpatterso@classicmodelcars.c
1076	Firrelli	Jeff	VP Marketing	x9273	jfirrelli@classicmodelcars.com

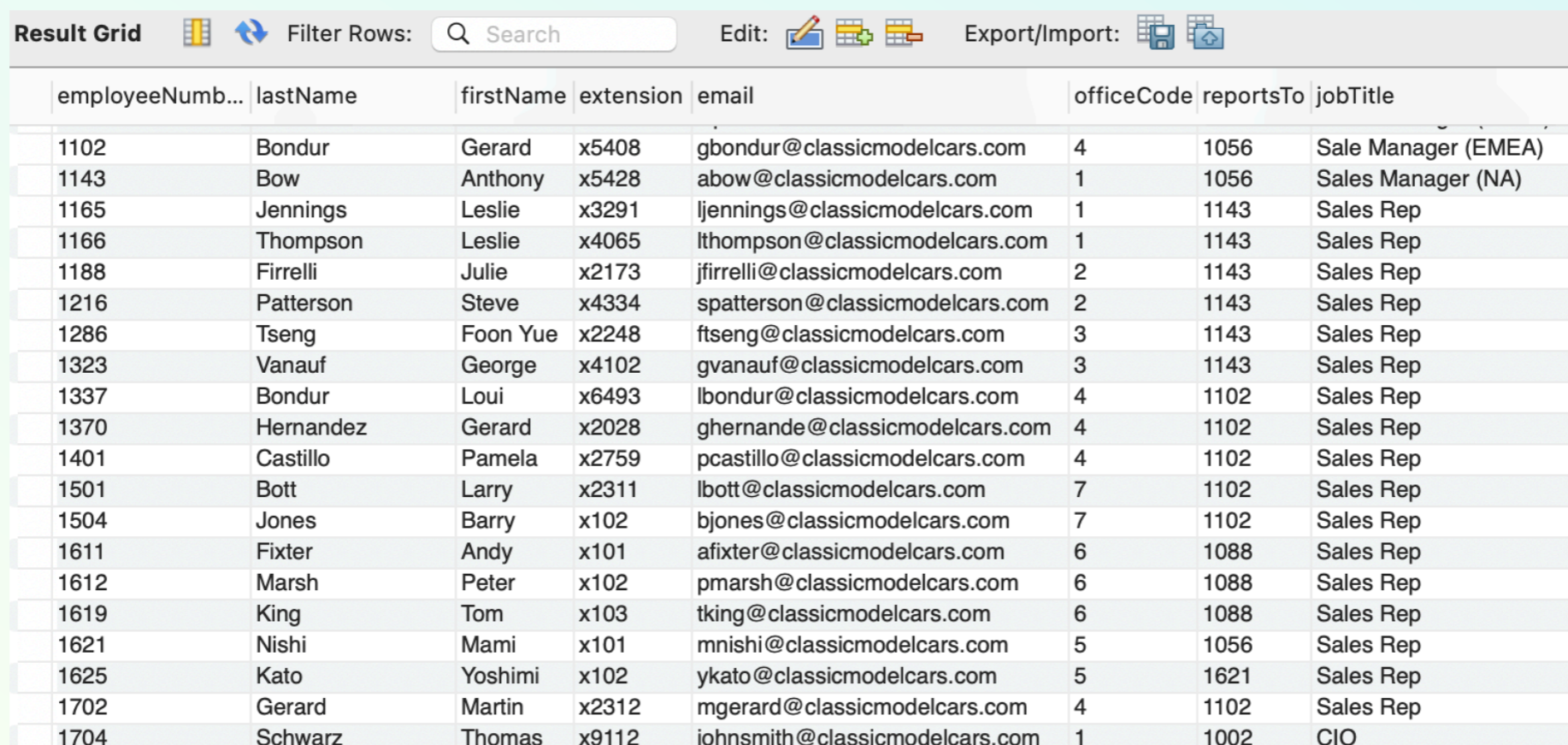
Updating Views in MySQL

- Now we have a view that we can update

```
INSERT INTO
vps (employeeNumber, firstname, lastname,
jobtitle, extension, email, officeCode, rep
ortsTo)
VALUES (
1704, 'Thomas', 'Schwarz', 'CIO', 'x9112', '
tschwarz@classicmodelcars.com', 1, 1002);
```

Updating Views in MySQL

- But the new “employee” is not visible in the vps view
 - Because the title does not have VP in it
- But it is in the employees table



The screenshot shows a MySQL database interface with a result grid. The grid contains 20 rows of employee data. The columns are: employeeNumb..., lastName, firstName, extension, email, officeCode, reportsTo, and jobTitle. The data is as follows:

employeeNumb...	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
1102	Bondur	Gerard	x5408	gbondur@classicmodelcars.com	4	1056	Sale Manager (EMEA)
1143	Bow	Anthony	x5428	abow@classicmodelcars.com	1	1056	Sales Manager (NA)
1165	Jennings	Leslie	x3291	ljennings@classicmodelcars.com	1	1143	Sales Rep
1166	Thompson	Leslie	x4065	lthompson@classicmodelcars.com	1	1143	Sales Rep
1188	Firrelli	Julie	x2173	jfirrelli@classicmodelcars.com	2	1143	Sales Rep
1216	Patterson	Steve	x4334	spatterson@classicmodelcars.com	2	1143	Sales Rep
1286	Tseng	Foon Yue	x2248	ftseng@classicmodelcars.com	3	1143	Sales Rep
1323	Vanauf	George	x4102	gvanauf@classicmodelcars.com	3	1143	Sales Rep
1337	Bondur	Loui	x6493	lbondur@classicmodelcars.com	4	1102	Sales Rep
1370	Hernandez	Gerard	x2028	ghernande@classicmodelcars.com	4	1102	Sales Rep
1401	Castillo	Pamela	x2759	pcastillo@classicmodelcars.com	4	1102	Sales Rep
1501	Bott	Larry	x2311	lbott@classicmodelcars.com	7	1102	Sales Rep
1504	Jones	Barry	x102	bjones@classicmodelcars.com	7	1102	Sales Rep
1611	Fixter	Andy	x101	afixter@classicmodelcars.com	6	1088	Sales Rep
1612	Marsh	Peter	x102	pmarsh@classicmodelcars.com	6	1088	Sales Rep
1619	King	Tom	x103	tking@classicmodelcars.com	6	1088	Sales Rep
1621	Nishi	Mami	x101	mnishi@classicmodelcars.com	5	1056	Sales Rep
1625	Kato	Yoshimi	x102	ykato@classicmodelcars.com	5	1621	Sales Rep
1702	Gerard	Martin	x2312	mgerard@classicmodelcars.com	4	1102	Sales Rep
1704	Schwarz	Thomas	x9112	johnsmith@classicmodelcars.com	1	1002	CIO

Updating Views in MySQL

- To prevent this, we redefine VPs with the check option

```
CREATE OR REPLACE VIEW vps AS
  SELECT
    employeeNumber,
    lastName,
    firstName,
    jobTitle,
    extension,
    email,
    officeCode,
    reportsTo
  FROM
    employees
  WHERE
    jobTitle LIKE '%VP%'
WITH CHECK OPTION;
```

Updating Views in MySQL

- Now the same query is rejected

```
INSERT INTO
vps (employeeNumber, firstname, lastname,
jobtitle, extension, email, officeCode, rep
ortsTo)
VALUES (
1704, 'Thomas', 'Schwarz', 'CIO', 'x9112', '
tschwarz@classicmodelcars.com', 1, 1002);
```



**Error Code: 1369. CHECK OPTION failed
'classicmodels.vps'**

Materialized Views

- Views are virtual
 - Created whenever they are accessed
 - But views can be heavily used
 - Views are used to:
 - Easier query logic because the definition of the view encompasses the difficulties
 - E.g. a view that uses a join of many tables
 - Security: Restrict access to tables, but give access to views
 - Enforce business rules: What is "active", what is "popular"

Materialized Views

- Virtual views that are heavily used means
 - running a query against a view
 - running a query to recreate the view
- Materialized views store the view in a derived table
 - Not all DBMS support materialized views
 - Some give it a different name

- Typical command:

```
CREATE MATERIALIZED VIEW movieProd AS
  SELECT title, year, name
  FROM movies, movieExec
  WHERE procuderC# = cert#
```

Materialized Views

- Materialized views need to be maintained
 - Some updates / inserts / deletes to movieExec and movies need to be intercepted
 - The changes to the materialized view are incremental

Materialized Views in MySQL

- They do not exist as materialized views
- But we can work around it:
 - Materialized views are tables that are modified by modifications to the base tables
 - Can use triggers to intercept modifications of the base tables in order to update the materialized view

Try It Out

- Use the employees database in MySQL
 - You might want to turn off automatic commits, then do a commit and at the end of the session a rollback
 - Task 1: Convince yourself that there are no emp_no larger than 500000

Try It Out

```
USE employees;
```

```
SELECT *  
FROM dept_emp  
WHERE emp_no >=500000;
```

Try It Out

- Task 2: Insert three persons into the employees table with employee numbers 600000, 600001, 600002. You can invent the missing dates.
- The hire date should be the day of today
 - In MySQL that is CURDATE()

Try It Out

```
INSERT INTO employees(emp_no, birth_date, first_name,
last_name, gender, hire_date)
VALUES
    (600000, '1980-01-01', 'Hector', 'Garcia Molinas',
'M', CURDATE()),
    (600001, '1981-01-01', 'Ursula', 'Leyendorf', 'F',
CURDATE()),
    (600002, '1982-01-01', 'Bob', 'Karragher', 'M',
CURDATE());
```

Try It Out

- Create a view of dept_emp that only contains entries with to_date unlimited
 - i.e. '9999-01-01' which is used to indicate an open contract.
- Call the view v_current_dept_emp
 - Include all attributes so that we can update

Try It Out

```
CREATE OR REPLACE VIEW v_current_dept_emp AS
  SELECT emp_no, dept_no, from_date, to_date
  FROM dept_emp
  WHERE to_date = '9999-01-01';
```

Try It Out

- Now insert the three new employees into the view
 - from_date is today
 - Department is 'd004'

Try It Out

```
INSERT INTO v_current_dept_emp (emp_no, dept_no,  
from_date, to_date)  
VALUES  
    (600000, 'd004', CURDATE(), '9999-01-01'),  
    (600001, 'd004', CURDATE(), '9999-01-01'),  
    (600002, 'd004', CURDATE(), '9999-01-01');
```

Try It Out

- Check that these updates made it to the dept_emp table as well as the view

Try It Out

```
SELECT *  
FROM v_current_dept_emp  
WHERE emp_no >=500000;
```

```
SELECT *  
FROM dept_emp  
WHERE emp_no >=500000;
```

Try It Out

- Change the view `v_current_dept_emp` to have only three columns: `emp_no`, `dept_no`, `from_date` by recreating it

Try It Out

```
CREATE OR REPLACE VIEW v_current_dept_emp AS
  SELECT emp_no, dept_no, from_date
  FROM dept_emp
  WHERE to_date = '9999-01-01';
```

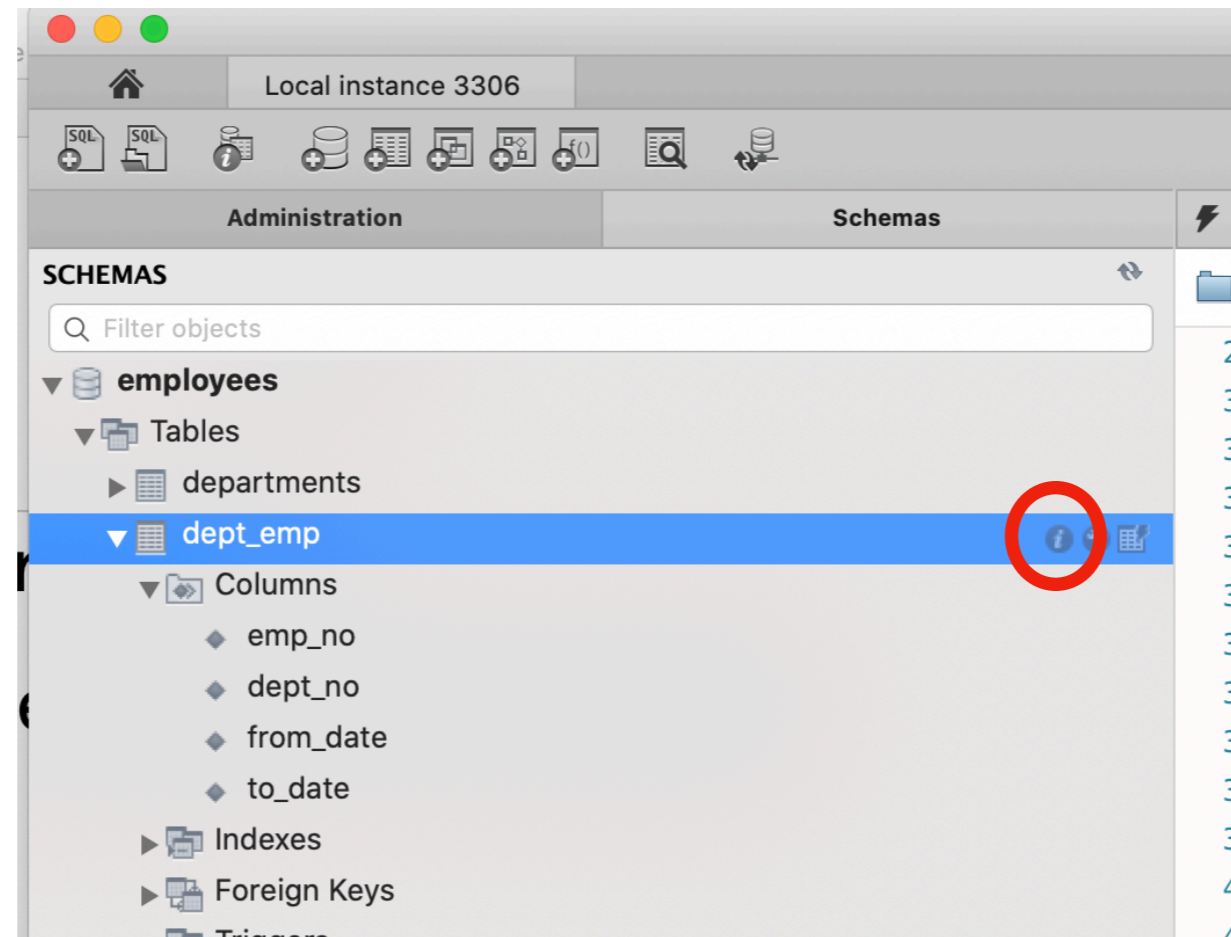
- The CREATE OR REPLACE clause makes it easy.
- You could also say DROP VIEW and then do a CREATE VIEW

Try It Out

- Check the table `dept_emp` for its definition

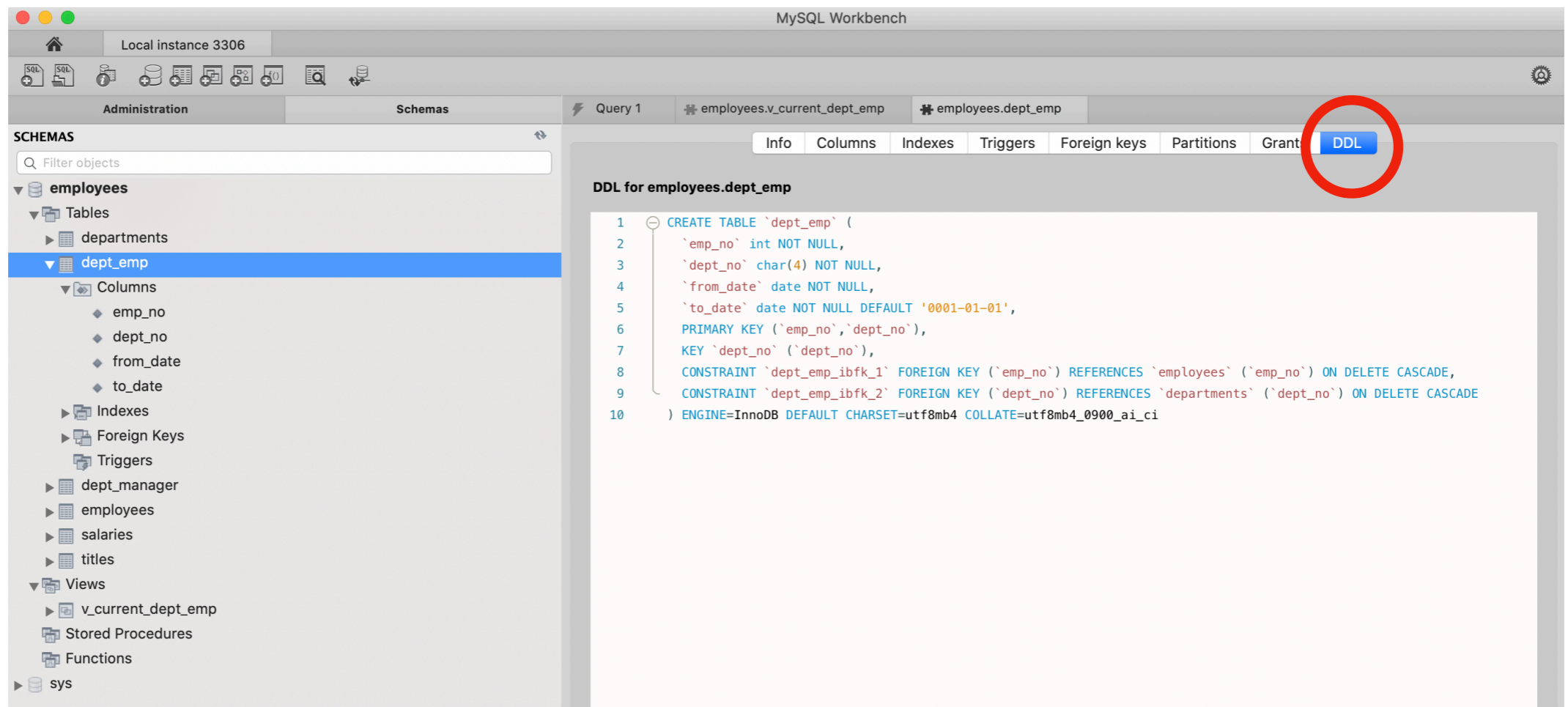
Try It Out

- In MySQLWorkbench:
 - Click on the table and the info tab



Try It Out

- In the view, select DDL, which gives you the definition of the table



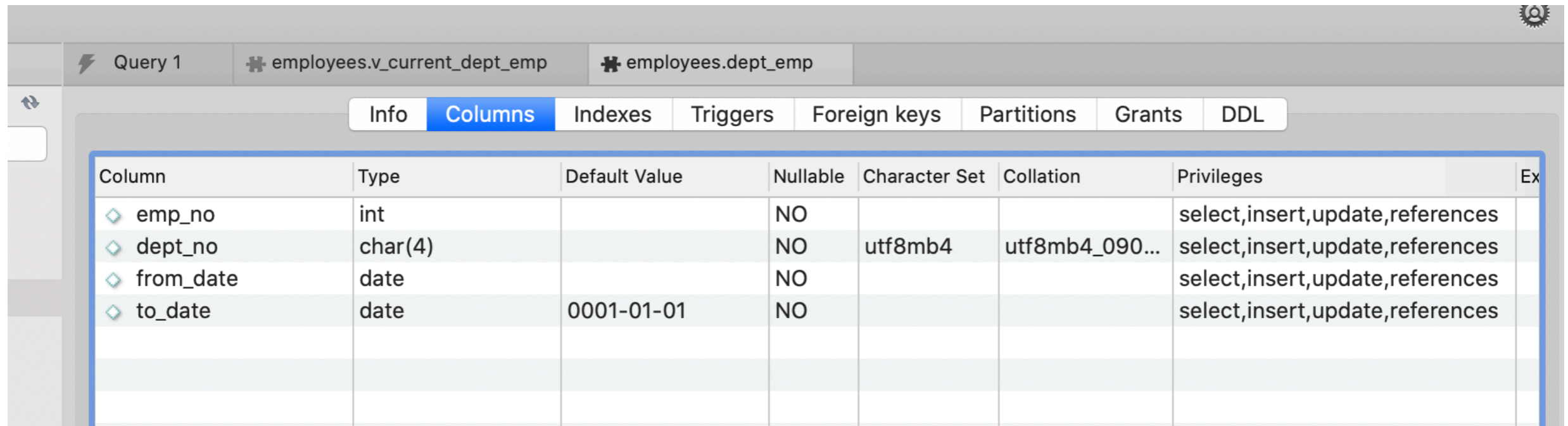
The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' tree is expanded to show the 'employees' database, with the 'dept_emp' table selected. The 'Columns' section for 'dept_emp' is visible, listing 'emp_no', 'dept_no', 'from_date', and 'to_date'. On the right, the 'DDL for employees.dept_emp' tab is active, displaying the following SQL code:

```
1 CREATE TABLE `dept_emp` (  
2   `emp_no` int NOT NULL,  
3   `dept_no` char(4) NOT NULL,  
4   `from_date` date NOT NULL,  
5   `to_date` date NOT NULL DEFAULT '0001-01-01',  
6   PRIMARY KEY (`emp_no`,`dept_no`),  
7   KEY `dept_no` (`dept_no`),  
8   CONSTRAINT `dept_emp_ibfk_1` FOREIGN KEY (`emp_no`) REFERENCES `employees` (`emp_no`) ON DELETE CASCADE,  
9   CONSTRAINT `dept_emp_ibfk_2` FOREIGN KEY (`dept_no`) REFERENCES `departments` (`dept_no`) ON DELETE CASCADE  
10 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

The 'DDL' tab is highlighted with a red circle.

Try It Out

- Alternatively, you can select columns



Column	Type	Default Value	Nullable	Character Set	Collation	Privileges	Ex
emp_no	int		NO			select,insert,update,references	
dept_no	char(4)		NO	utf8mb4	utf8mb4_090...	select,insert,update,references	
from_date	date		NO			select,insert,update,references	
to_date	date	0001-01-01	NO			select,insert,update,references	

- Both methods show that we have a NOT NULL constraint on to_date

Try It Out

- Alter the table dept_emp to have a default value of '9999-01-01' in the to_date.
- We could also remove the NOT NULL restriction

Try It Out

```
ALTER TABLE dept_emp  
MODIFY COLUMN to_date date NOT NULL DEFAULT '1-01-01';
```

Try It Out

- If we try to add directly to the table with new values, we violate a foreign key constraint.

```
INSERT INTO v_current_dept_emp(emp_no, dept_no, from_date)
VALUES
    (600003, 'd004', CURDATE()),
    (600004, 'd004', CURDATE()),
    (600005, 'd004', CURDATE());
```