

# As Vistas

---

# Índice

---

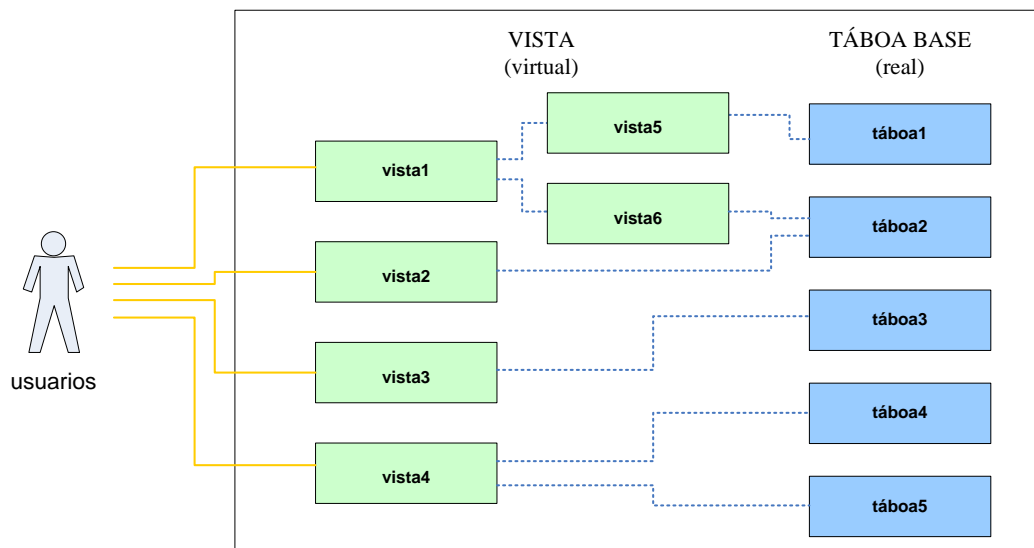
|           |  |           |
|-----------|--|-----------|
| <b>1.</b> | <b>As vistas.....</b>                                  | <b>3</b>  |
| 1.1       | Sentenza CREATE VIEW e utilización de vistas.....      | 3         |
|           | Exemplos .....   | 4         |
| 1.2       | Sentenza SHOW CREATE VIEW .....                        | 9         |
| 1.3       | Sentenza ALTER VIEW .....                              | 9         |
| 1.4       | Sentenza DROP VIEW.....                                | 10        |
| <b>2.</b> | <b>Manipulación de datos a través das vistas .....</b> | <b>10</b> |
| <b>3.</b> | <b>Avantaxes do uso de vistas .....</b>                | <b>11</b> |

# 1. As vistas

Unha vista é unha táboa virtual que non ten existencia física como unha táboa base, pero que os usuarios ven como unha táboa máis. Distintos usuarios poden ter unha percepción distinta do contido dunha mesma base de datos en función das vistas que utilizan para acceder aos datos.

As vistas teñen un nome asociado e a mesma estrutura cá unha táboa, composta de filas e columnas. A diferenza fundamental entre as vistas e as táboas base é que das vistas só se almacena a definición da súa estrutura, e os datos só se almacenan nas táboas base.

As vistas son como un filtro entre os usuarios e os datos almacenados nas táboas base. Este filtro está definido como unha selección de datos mediante unha sentenza SELECT sobre unha ou máis táboas, ou sobre outras vistas.



Os permisos que teña o usuario limitan as operacións que pode facer nas bases de datos. Se un usuario pode consultar datos (SELECT), e realizar o resto de operacións de manipulación de datos (INSERT, UPDATE, DELETE), tamén terá permiso para facer as mesmas operacións utilizando vistas. Para crear ou modificar vistas, necesita ter permisos especiais.

A maioría dos SGBDR permiten a creación e manexo de vistas; MySQL, incorpora as vistas a partir da versión 5.0.1.

## 1.1 Senteza CREATE VIEW e utilización de vistas

A sentenza CREATE VIEW permite crear novas vistas, ou substituír vistas que xa existen. Sintaxe:

```
CREATE [OR REPLACE]
[DEFINER = { usuario | CURRENT_USER }]
[SQL SECURITY { DEFINER | INVOKER }]
VIEW nome_vista [(lista_columnas)]
AS sentenza_select
[WITH [CASCADED | LOCAL] CHECK OPTION]
```

- A cláusula opcional OR REPLACE permite borrar unha vista que exista co mesmo nome e crear a nova vista. Se non se utiliza esta cláusula e xa existe unha vista co nome da que se vai a crear, prodúcese unha condición de erro e non se crea a nova vista.

- As cláusulas `DEFINER` e `SQL SECURITY` especifican o contexto de seguridade no momento da execución da vista. Con `DEFINER` pódese indicar o nome do usuario que vai ser considerado como o creador da vista. Se non se especifica nada, tómasse `CURRENT_USER` que fai referencia ao usuario actual que está creando a vista.
- A cláusula `SQL SECURITY` permite indicar se na execución da vista se utilizan privilexios do creador da vista (`DEFINER` - valor por defecto) ou do usuario que a chama (`INVOKER`).
- O nome da vista, *nome\_vista*, ten que cumprir as mesmas condicións que calquera nome de táboa, en canto a caracteres e lonxitude permitidos. A continuación do nome da vista pódense poñer, pechados entre parénteses, os nomes que van a ter as columnas da nova vista separados por comas (*lista\_columnas*). Se non se especifican os nomes, tómanse os nomes das columnas da lista de selección da consulta empregada para crear a vista. O número de nomes da *lista\_columnas* ten que coincidir co número de columnas da lista de selección da consulta.
- *sentenza\_select* representa unha sentenza `SELECT`, que selecciona os datos de definición da vista. Os datos poden ser seleccionados de táboas base ou doutras vistas. A sentenza `SELECT` utilizada para crear unha vista ten unha serie de restricións:
  - Non pode conter subsentenzas na cláusula `FROM`.
  - Non pode conter referencias a variables do sistema ou de usuario.
  - Non pode facer referencia a táboas de tipo temporal, e non se poden crear vistas temporais.
- A cláusula `WITH CHECK OPTION` utilízase para crear vistas actualizables, que poden ser utilizadas para inserir ou modificar datos das táboas. Cando se utiliza a cláusula, só permitirá inserir ou modificar filas que cumpran as condicións recollidas na cláusula `WHERE` da *sentenza\_select*. A cláusula `WITH CHECK OPTION` incorpórase na versión MySQL 5.0.2.
- As opcións `LOCAL` e `CASCADE` pódense utilizar cando na definición da vista se utilicen outras vistas. Permiten establecer a forma en que se comprobarán as condicións incluídas nas cláusulas `WHERE` das vistas. Cando se utiliza a palabra `LOCAL`, a cláusula `CHECK OPTION` só comproba as condicións da *sentenza\_select* da vista que se está creando. Cando se utiliza a palabra `CASCADE`, compróbanse as condicións de todas as vistas que interveñen na definición. Se non se especifica nada tómasse, por defecto, `CASCADE`.

O usuario que cree as vistas ten que ter o privilexio `CREATE VIEW` e os privilexios adecuados sobre as columnas ás que se fai referencia na `SELECT`. No caso de utilizar a opción `OR REPLACE`, tamén é necesario que teña o privilexio `DROP VIEW` para borrar a vista.

As vistas utilízanse como se fosen táboas base xunto con `SELECT`, `INSERT`, `DELETE` OU `UPDATE`.

## Exemplos

- Exemplo de creación dunha vista para manexar a información dos departamentos que pertencen á provincia de Lugo (*id\_provincia* = 27).

```
create view departamento_lugo
as
select codigo, nome, tipo, cidade, id_provincia
from departamento
where id_provincia = 27;
```

- Exemplo de utilización da vista anterior para que os usuarios só poidan consultar os departamentos da súa provincia, e non as doutras provincias.

```
select * from departamento_lugo;
```

| codigo | nome     | tipo | cidade   | id_provincia |
|--------|----------|------|----------|--------------|
| 1      | Central  | H    | Lugo     | 27           |
| 2      | Oficina1 | H    | Monforte | 27           |
| 6      | Oficina5 | A    | Villalba | 27           |
| 8      | Oficina7 | H    | Lugo     | 27           |

Cando se fai a consulta anterior a vista de exemplo como filtro, só se mostran os departamentos que teñan en *id\_provincia* o valor 27, e as columnas vense cos nomes que teñen na táboa *departamento*.

- Exemplo de creación dunha vista para inserir ou modificar a información dos departamentos que pertencen á provincia de Lugo.

No caso de querer utilizar a vista para actualizar datos sobre a táboa *departamento*, verificando a condición que a columna *id\_provincia* só pode tomar o valor 27, hai que engadirlle á vista a cláusula **WITH CHECK OPTION**. No código que ven a continuación incorpórase esa cláusula, e ademais, cámbiaselle o nome ás columnas poñendo a lista de columnas, pechadas entre parénteses a continuación do nome da vista, e utilízase a opción **OR REPLACE**, que fai que a vista creada antes co mesmo nome se borre, e créase unha vista coas novas especificacións.

```
create or replace
view departamento_lugo (dl_codigo, dl_nome, dl_tipo, dl_cidade, dl_provincia)
as
select codigo, nome, tipo, cidade, id_provincia
from departamento
where id_provincia = 27
with check option;
```

- Exemplo de utilización da vista anterior nunha consulta.

Mostra os mesmo datos que na consulta anterior, pero cambian os nomes das columnas.

```
select * from departamento_lugo;
```

| dl_codigo | dl_nome  | dl_tipo | dl_cidade | dl_provincia |
|-----------|----------|---------|-----------|--------------|
| 1         | Central  | H       | Lugo      | 27           |
| 2         | Oficina1 | H       | Monforte  | 27           |
| 6         | Oficina5 | A       | Villalba  | 27           |
| 8         | Oficina7 | H       | Lugo      | 27           |

- Exemplo de utilización da vista anterior para unha inserción.

Ao levar a cláusula **WITH CHECK OPTION**, a vista anterior tamén pode ser utilizada para actualizar datos da táboa *departamento*. Cando se utiliza para inserir ou modificar filas, o sistema comproba que o valor almacenado na columna *dl\_provincia* tome o valor 27; se toma un valor distinto, non permite a operación e mostra unha mensaxe de erro informando que non se cumpre as condicións **CHECK OPTION**; se toma o valor 27, realiza a inserción.

– Exemplo de inserción de datos da provincia 15.

```
insert into departamento_lugo
values (101,'proba local','H','','15');
```

- Exemplo de inserción de datos da provincia 27.

```
insert into departamento_lugo
values (101,'proba local','H','proba',27);
```

55 13:37:22 insert into departamento\_lugo values (101,'proba local','H','proba',27)

1 row(s) affected

- Exemplo de utilización dunha vista creada con LOCAL en CHECK OPTION.

Crear unha vista para manexar a información dos departamentos que pertencen á provincia de Lugo (id\_provincia = 27), e que sexan de tipo 'H'.

Créase a vista sobre a vista *departamento\_lugo* que xa selecciona os departamentos de Lugo, e engádese a condición de que o contido da columna *dl\_tipo* tome o valor 'H'.

```
create view departamentoh_lugo
as
select dl_codigo, dl_nome, dl_tipo, dl_cidade, dl_provincia
from departamento_lugo # créase a vista sobre unha vista que existe
where dl_tipo = 'H'
with local check option;
```

- Cando se fai a consulta utilizando esta vista, móstranse os departamentos que sexan da provincia de Lugo (*dl\_provincia* = 27), e teñan na columna *dl\_tipo* o valor 'H'.

```
select * from departamentoh_lugo;
```

| dl_codigo | dl_nome     | dl_tipo | dl_cidade | dl_provincia |
|-----------|-------------|---------|-----------|--------------|
| 1         | Central     | H       | Lugo      | 27           |
| 2         | Oficina1    | H       | Monforte  | 27           |
| 8         | Oficina7    | H       | Lugo      | 27           |
| 101       | proba local | H       | proba     | 27           |

- Cando se actualizan datos utilizando esta vista, o sistema só comproba que o valor que toma a columna *dl\_tipo* sexa 'H', pero non comproba a condición da vista *departamento\_lugo* (id\_provincia = 27) porque a cláusula WITH CHECK OPTION leva a opción LOCAL.

A sentenza INSERT seguinte intenta inserir unha fila na táboa departamento, utilizando como filtro a vista *departamentoh\_lugo*. O valor que toma a columna *dl\_tipo* non cumpre a condición LOCAL establecida na sentenza SELECT de creación da vista, polo que a fila non se insire e móstrase unha mensaxe de erro.

```
insert into departamentoh_lugo
values (102,'proba local','A','proba',15);
```

79 14:03:54 insert into departamentoh\_lugo values (102,'proba local','A','proba',15)

Error Code: 1369. CHECK OPTION failed 'practicas5.departamentoh\_lugo'

A sentenza INSERT seguinte execútase correctamente porque o valor da columna *dl\_tipo* cumpre a condición establecida na sentenza SELECT de creación da vista (*dl\_tipo* = 'H'), e non se ten en conta que non cumpre a condición establecida na sentenza SELECT de creación da vista *departamento\_lugo* (id\_provincia = 27), porque a cláusula WITH CHECK OPTION leva a opción LOCAL.

```
insert into departamentoh_lugo
values (102,'proba local','H','proba',15);
```

80 14:06:15 insert into departamentoh\_lugo values (102,'proba local','H','proba',15)

1 row(s) affected

- Exemplo de utilización dunha vista creada con **CASCADED** en **CHECK OPTION**.

No código que ven a continuación vaise a substituír a vista *departamentoh\_lugo*, que utiliza a opción **LOCAL** na cláusula **WITH CHECK OPTION**, por outra vista co mesmo nome pero utilizando a opción **CASCADED** na cláusula **WITH CHECK OPTION**, para ver a diferenza de comportamento entre as dúas opcións.

```
create or replace view departamentoh_lugo
as
select dl_codigo, dl_nome, dl_tipo, dl_cidade, dl_provincia
from departamento_lugo      # créase a vista sobre unha vista que existe
where dl_tipo = 'H'
with cascaded check option;
```

- Cando se fai a consulta utilizando esta vista, móstranse os departamentos que son da provincia de Lugo (*dl\_provincia* = 27), e teñen na columna *dl\_tipo* o valor 'H', igual que na vista anterior.

```
select * from departamentoh_lugo;
```

| dl_codigo | dl_nome     | dl_tipo | dl_cidade | dl_provincia |
|-----------|-------------|---------|-----------|--------------|
| 1         | Central     | H       | Lugo      | 27           |
| 2         | Oficina1    | H       | Monforte  | 27           |
| 8         | Oficina7    | H       | Lugo      | 27           |
| 101       | proba local | H       | proba     | 27           |

- Cando se utiliza a vista para actualizar datos, é cando cambia o comportamento. Ao inserir ou modificar unha fila, o sistema comproba que o valor que toma a columna *dl\_tipo* sexa 'H', e ademais, comproba a condición da vista *departamento\_lugo* (*id\_provincia* = 27) porque a cláusula **WITH CHECK OPTION** leva a opción **CASCADED**.

A sentenza **INSERT** seguinte intenta inserir unha fila na táboa *departamento*, utilizando como filtro a vista *departamentoh\_lugo*. O valor que toma a columna *dl\_tipo* cumpre a condición establecida na sentenza **SELECT** de creación da vista, pero a columna *dl\_provincia* toma o valor 15 e non cumpre a condición establecida na sentenza **SELECT** de creación da vista *departamento\_lugo*, polo que a fila non se insire e móstrase unha mensaxe de erro.

```
insert into departamentoh_lugo
values (103,'proba local','H','proba',15);
```

✖ 83 14:21:25 insert into departamentoh\_lugo values (103,'proba local','H','',15)

Error Code: 1369. CHECK OPTION failed 'practicass5.departamentoh\_lugo'

A sentenza **INSERT** seguinte execútase correctamente porque cumpre as condicións contidas nas dúas vistas.

```
insert into departamentoh_lugo
values (103,'proba local','H','proba',27);
```

✔ 85 14:22:52 insert into departamentoh\_lugo values (103,'proba local','H','proba',27)

1 row(s) affected

- Exemplo con limitación de columnas.

Crear unha vista que mostre o *dni*, *nome*, *apelidos*, e *departamento* de todos os empregados, ocultando os datos de salario bruto e xefe.

Nos exemplos feitos ata o momento faise unha limitación de filas da táboa *departamento*, introducindo unha cláusula **WHERE** na consulta de creación da vista que filtra as filas que cumpren a condición. Cando se quere facer unha limitación das columnas o

único que hai que facer é poñen na lista de selección da consulta só as columnas ou expresións que se lle permiten ver aos usuarios que manexan a vista. Cando se executa unha sentenza INSERT sobre a vista, as columnas que non figuran na lista de selección toman o valor nulo (NULL).

```
create view vistaEmpleado
as
select dni, nome, apellidos, departamento
from empleado;
```

Esta vista permite ocultar a un grupo de usuarios a información sobre os salarios brutos dos seus compañeiros. Por exemplo, supoñamos que o xerente da empresa ten acceso á táboa *empleado*, pero o resto de traballadores non teñen permisos de acceso a esa táboa, pero se poden consultar datos utilizando a vista; cando executan unha consulta utilizando a vista, verán só as columnas contidas na lista de selección da consulta asociada á vista, pero eles poden ter a percepción de que son todos os datos que hai almacenados.

```
select * from vistaEmpleado;
```

The screenshot shows a database interface with a 'Result Grid' displaying the results of a query. The grid has columns: dni, nome, apellidos, and departamento. Below the grid, there is an 'Output' section showing the query executed: 'select \* from vistaEmpleado' and the message '20 row(s) returned'.

| dni      | nome      | apellidos          | departamento |
|----------|-----------|--------------------|--------------|
| 12549563 | Fernanda  | Case Rodriguez     | 4            |
| 12852654 | Benito    | Martinez Iglesias  | 1            |
| 15245258 | Antonia   | Nuñez Bernardes    | 2            |
| 33123456 | Jose Luis | Fernandez Lopez    | 1            |
| 33147258 | Dario     | Ruiz Macias        | 10           |
| 33219853 | Valentina | Hernandez Valin    | 3            |
| 33251256 | Carlos    | Martinez Diaz      | 2            |
| 33254916 | Adolfo    | Iglesias Dominguez | NULL         |
| 33257964 | Rosario   | Villar Bernal      | 4            |
| 33322541 | Teolindo  | Villar Bernal      | 5            |

Comprobación do contido da táboa departamento despois das operacións feitas coas vistas.

```
select * from departamento;
```

The screenshot shows a database interface with a 'Result Grid' displaying the results of a query. The grid has columns: codigo, nome, tipo, cidade, and id\_provincia. The results show various departments and their locations.

| codigo | nome        | tipo | cidade   | id_provincia |
|--------|-------------|------|----------|--------------|
| 1      | Central     | H    | Lugo     | 27           |
| 2      | Oficina1    | H    | Monforte | 27           |
| 3      | Oficina2    | B    | Ferrol   | 15           |
| 4      | Oficina3    | H    | Vigo     | 36           |
| 5      | Oficina4    | A    | Ourense  | 32           |
| 6      | Oficina5    | A    | Villalba | 27           |
| 7      | Oficina6    | H    | Ourense  | 32           |
| 8      | Oficina7    | H    | Lugo     | 27           |
| 9      | Oficina8    | A    | Coruña   | 15           |
| 10     | Oficina9    | B    | Villalba | 28           |
| 101    | proba local | H    | proba    | 27           |
| 102    | proba local | H    | proba    | 15           |
| 103    | proba local | H    | proba    | 27           |

Unha vez finalizadas as probas de inserción e para non deixar os datos de proba na táboa, aconséllase borrar as filas que se inseriron de proba.

```
delete from departamento
where codigo >=100;
```



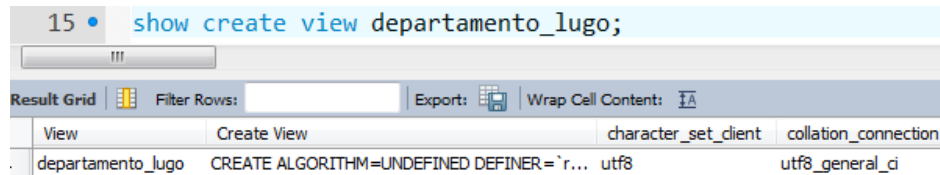
## 1.2 Sentenza SHOW CREATE VIEW

A información sobre as vistas creadas gárdase no dicionario de datos, igual có resto de obxectos das bases de datos. En MySQL, a información sobre as vistas pódese consultar en *information\_schema.views* mediante unha sentenza SELECT e dispón, ademais, da sentenza SHOW CREATE VIEW para consultar información sobre as vistas. Sintaxe:

```
SHOW CREATE VIEW nome_vista
```


Exemplo: Mostrar información sobre a vista *departamento\_lugo*.

```
show create view departamento_lugo
```



| View              | Create View                                    | character_set_client | collation_connection |
|-------------------|--|----------------------|----------------------|
| departamento_lugo | CREATE ALGORITHM=UNDEFINED DEFINER=`r...` utf8 | utf8                 | utf8_general_ci      |

O resultado da sentenza anterior móstrase na zona *Result Grid* de Workbench. Móstranse catro columnas con información sobre o nome da vista, a sentenza de creación, o xogo de caracteres asociado e o sistema de colación. A información sobre a sentenza de creación non se pode ver porque é moi larga e non entra na pantalla. Para poder vela cun formato lexible pódense seguir estes pasos:

- Seleccionar o contido da columna *Create View* onde está a sentenza de creación.
- Facer clic co botón dereito do rato.
- Elixir a opción *Copy Field (unquoted)*.
- Abrir unha nova pestana de consulta SQL e pegar nela o contido da columna.
- Pinchar na icona  *Beautify/Reformat SQL script* e aparece o código na versión "bonita" de Workbench.

**CREATE**

```
ALGORITHM = UNDEFINED  
DEFINER = `root`@`localhost`  
SQL SECURITY DEFINER
```

**VIEW** `departamento\_lugo` **AS**

```
select  
  `departamento`.`codigo` AS `dl_codigo`,  
  `departamento`.`nome` AS `dl_nome`,  
  `departamento`.`tipo` AS `dl_tipo`,  
  `departamento`.`cidade` AS `dl_cidade`,  
  `departamento`.`id_provincia` AS `dl_provincia`  
from  
  `departamento`  
where  
  (`departamento`.`id_provincia` = 27) WITH CASCADED CHECK OPTION
```

## 1.3 Sentenza ALTER VIEW

Permite cambiar a definición dunha vista que existe. A sintaxe é similar á da sentenza CREATE VIEW, e é o mesmo que utilizar a sentenza CREATE OR REPLACE VIEW. Sintaxe:

```
ALTER  
[DEFINER = { usuario | CURRENT_USER }]  
[SQL SECURITY { DEFINER | INVOKER }]  
VIEW nome_vista [(lista_columnas)]  
AS sentença_select  
[WITH [CASCADED | LOCAL] CHECK OPTION]
```

O usuario que modifique vistas ten que ter o privilexio CREATE VIEW, DROP VIEW e os privilexios adecuados sobre as columnas a que se fai referencia na SELECT.

## 1.4 Sentenza DROP VIEW

Para borrar vistas utilízase a sentenza DROP VIEW. Os usuarios que a utilicen teñen que ter o privilexio DROP VIEW. A sintaxe é a seguinte:

```
DROP VIEW [IF EXISTS] nome_vista [, nome_vista] ...
```

- A opción IF EXISTS permite evitar que se produza un erro cando a vista que se intenta borrar non existe. Ten utilidade cando a sentenza vai incluída nun script.
- Pódense borrar varias vista coa mesma sentenza poñendo os nomes separados por comas.

## 2. Manipulación de datos a través das vistas

---

As operacións de manipulación de datos autorizadas sobre as vistas son as mesmas que sobre as táboas (consultas, inserción, modificación e supresión). Como se comentou anteriormente, todas as vistas poden ser utilizadas para consultar datos, pero non todas as vistas poden ser utilizadas para actualizar datos sobre as táboas.

Chámase vista 'actualizable' (*updatable*) aquela que pode ser utilizada para manipular os datos das táboas coas que está relacionada mediante INSERT, UPDATE ou DELETE.

O manual de referencia de MySQL expón detalladamente as condicións que ten que cumprir unha vista para que sexa 'actualizable'. Resumo desas condicións:

- Debe haber unha relación 'un a un' entre as filas da vista e as filas da táboa coa que está relacionada.
- Non contén:
  - Funcións de agrupamento (AVG(), SUM(), COUNT(), ...).
  - Cláusulas: DISTINCT, GROUP BY, HAVING, UNION ou UNION ALL.
  - Subsentenzas na lista de selección.
  - Certas combinacións con JOIN.
  - Referencias a vistas non 'actualizables' na cláusula FROM.
- Para inserir filas utilizando unha vista, esta debe incorporar na lista de selección todas as columnas que non admiten valores nulos (propiedade NOT NULL), e non pode conter expresións; só nomes de columnas.

Cando se crea unha vista, o servidor engádelle un indicador que toma o valor verdadeiro (*true*) cando a vista é 'actualizable' e o valor falso (*false*) se non o é. Esta información pódese consultar na columna IS\_UPDATABLE da táboa VIEW da base de datos INFORMATION\_SCHEMA, aínda que nalgúns casos pode ter o valor verdadeiro e non pode ser utilizada para calquera operación de actualización de datos. Isto significa que cando se executa unha sentenza INSERT, UPDATE ou DELETE sobre unha vista, o servidor sabe se están permitidas esas operacións sobre a vista, e en caso de que non estean mostra a mensaxe de erro correspondente.

Exemplos dalgunhas sentenzas SELECT que non se poderían utilizar para crear unha vista 'actualizable':

```
select dni,apellidos,nome,salario_bruto *20/100
from empregado;
```

Porque contén unha expresión e non pode ser utilizada para inserir filas, aínda que podería ser utilizada para facer modificacións e borrado de filas

```
select id_provincia, cidade , count(*)
from departamento
group by id_provincia;
```

Porque utiliza a cláusula GROUP BY, e ademais a función de agrupamento COUNT(\*)

```
select dni,apellidos, nome, salario_bruto,
(select avg() from empregado) as media_salario
from empregado
where salario_bruto >= 50000;
```

Porque contén unha subconsulta na lista de selección.

```
select apellidos, nome, salario_bruto
from empregado
where salario_bruto between 50000 and 70000;
```

Non pode ser utilizada para inserir filas, porque non inclúe a columna dni que non admite valores null.

### 3. Avantaxes do uso de vistas

De forma resumida os beneficios máis importantes que aportan as vistas son os seguintes:

- Almacénanse no servidor, polo que o consumo de recursos e eficacia sempre serán óptimos. Cando se crea unha vista faise unha compilación da sentenza *SELECT* que contén, e gárdase asociada ao nome da vista. Desta maneira o tempo de execución dunha vista é menor que se se executa a sentenza *SELECT* directamente, xa que esta ten que pasar por o proceso de compilación.

- Unha vista é un camiño fácil para gardar consultas complexas na propia base de datos. Ocorre con frecuencia que os usuarios solicitan información que require realizar consultas complexas que poden levar moito tempo deseñar e probar, e que poden ser utilizadas noutro momento. Se estas consultas se fan con vistas, quedarán almacenadas no servidor e estarán accesibles cando se necesiten.

Tamén facilita o traballo aos desenvolvedores de software que teñan pouca experiencia traballando con bases de datos e dificultades para facer consultas complexas, dándolles a opción de chamar á vista almacenada na base de datos para poder obter os datos.

- Proporcionan flexibilidade na formulación de consultas moi complexas. Unha vista pode ser creada coa idea de facilitar a formulación de sentenzas complexas que non son soportadas nunha única sentenza *SELECT*.

- Son unha ferramenta importante para manter a confidencialidade dos datos.

Poden ser utilizadas como filtro entre os usuarios e as táboas base, de maneira que os usuarios non acceden directamente ás táboas e non teñen porque ver todos os datos da táboas se non que só ven os datos que foron seleccionados na consulta coa que se creou a vista.

Pódese limitar o acceso aos usuarios a determinadas filas ou columnas dunha ou máis táboas, sen ter que conceder privilexios aos usuarios sobre as táboas, sendo suficiente conceder privilexios sobre as vistas.

Aos desenvolvedores de software facilítalles o acceso ás vistas cos datos que necesitan, pero non se lles dá acceso as táboas.

- Axudan a manter a integridade referencial. Permiten controlar a integridade referencial para táboas non transaccionais nas operacións de inserción de filas utilizando a cláusula *WITH CHECK OPTION*.