<u>Índice</u>

1. Introducción	2
2. Creación de vistas	3
3. Vistas con columnas virtuales	5
4. Combinaciones para definir vistas	6
5. Manipulación de datos y vistas	7
6. Borrado de vistas	10
7. Resumen	11
8. Ejercicios propuestos	12

1. INTRODUCCIÓN

Una vista es una ventana a través de la cual vemos subconjuntos de datos de tablas ya existentes.

Los "datos" de las vistas no se almacenan físicamente, se obtienen de las tablas que se están especificando.

Las vistas se definen por medio de sentencias SQL, que se almacenan en la vista del diccionario de datos VIEWS.

La información se recupera a través de las vistas con la misma sintaxis que hemos utilizado para las tablas poniendo en vez del nombre de la tabla el nombre de la vista.

Una ventana puede especificar un subconjunto de datos vertical (subconjunto de filas) y/o horizontal (subconjunto de columnas) de una tabla.

Los datos se pueden obtener a partir de múltiples tablas: unión de tablas y combinación de tablas.

Razones para la utilización de vistas

1.- Simplificar consultas

Algunas aplicaciones requieren consultas muy complejas, almacenando consultas intermedias se permite a los usuarios simplificar sus operaciones.

2.- Seguridad

Utilizando los mecanismos de seguridad sobre tablas y vistas (lenguaje de control de datos DCL) y las vistas se tiene un sistema potente y flexible de control de acceso a los datos.

3.- Almacenamiento

Las vistas son proposiciones SQL que se almacenan en el diccionario de datos de forma compilada, por lo que el costo de análisis y compilación sólo se produce al crearlas.

Las vistas contienen sólo datos virtuales, los datos reales se almacenan en los bloques de datos de las tablas, por lo tanto sólo se almacena la definición compilada, por lo que el gasto de almacenamiento es mínimo.

4.- Independencia lógica

Conforme va creciendo la base de datos las tablas se pueden alterar, unir y combinar con otras, etc ...

Creando vistas de esas tablas originales se garantiza la independencia lógica de los datos, los usuarios y las aplicaciones podrán seguir refiriéndose a los datos como si estos tuviesen la misma estructura que antes.

2. CREACION DE VISTAS

Para crear vistas se utiliza el comando del lenguaje de definición de datos CREATE VIEW, que sirve para darle un nombre a un comando SQL válido.

```
CREATE VIEW <nombre_de_la_vista> [<lista_de_columnas>]
AS <sentencia_SQL_válida>;
```

Se utiliza la lista_de_columnas para cambiarle el nombre a las columnas que se obtienen con la sentencia SQL.

Se quiere crear una vista que contenga información sobre los jefes y el presidente.

```
CREATE VIEW jefes_pres(id_emp, id_dept, nombre, sueldo)

SELECT employee_id, department_id, first_name, salary

FROM employee

WHERE job_id IN (select job_id
from job
where function in('MANAGER','PRESIDENT'));
```

view JEFES_PRES creado.

Como comentamos anteriormente las vistas se tratan como las tablas para recuperar datos. Seleccione los jefes y el presidente que ganen más de 2000.

SELECT *
FROM jefes_pres
where sueldo >2000;;

	£	ID_EMP	A	ID_DEPT	A	NOMBRE	A	SUELDO
1		7569		12	СН	RIS		3000
2		7782		10	CA	ROL		2450
3		7566		20	ΤE	RRY		2975
4		7507		14	LE	SLIE		2200
5		7506		23	LY	NN		2750
6		7505		13	JΕ	AN		2850
7		7698		30	MA	RION		2850
8		7839		10	FR	ANCIS		5000

El departamento financiero necesita información sobre los sueldos y números de empleado, su comisión si existe y su número de departamento, para realizar la nómina. Se omitirán todas las otras columnas que no se utilicen para confeccionar la nómina por motivos de seguridad.

CREATE VIEW nomina (id_emp, id_dept, sueldo, comisión) **AS** SELECT employee_id, department_id, salary, commission FROM employee;

view NOMINA creado.

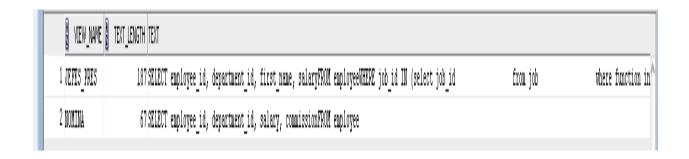
SELECT * FROM NOMINA;

	🚱 🌉 SQL	Todas las F	ilas Recupera	adas: 32 en 0,01
	ID_EMP	ID_DEPT	SUELDO	2 COMISIÓN
1	7369	20	800	(null)
2	7499	30	1600	300
3	7505	13	2850	(null)
4	7506	23	2750	(null)
5	7507	14	2200	(null)
6	7521	30	1250	500
7	7555	13	1250	300
8	7557	13	1250	1200
9	7560	23	1250	(null)
10	7564	23	1250	300
11	7566	20	2975	(null)
12	7569	12	3000	(null)
13	7600	13	1250	900
14	7609	24	1800	(null)
15	7654	30	1250	1400
16	7676	34	1850	(null)
17	7698	30	2850	(null)
18	7782	10	2450	(null)
19	7788	20	3000	(null)
20	7789	23	1500	1000
21	7799	12	3000	(null)
22	7820	43	1300	800
23	7839	10	5000	(null)
24	7844	30	1500	0
25	7876	20	1100	(null)
26	7900	30	950	(null)
27	7902	20	3000	(null)
28	7916	12	2875	(null)

Cada vez que se crea una vista se actualizan las tablas del diccionario de datos para incluir el nombre y la definición de este nuevo objeto.

Para comprobarlo los usuarios pueden consultar la vista del diccionario de datos **VIEWS.**

SELECT *
FROM USER_VIEWS
WHERE VIEW_NAME = 'NOMINA' OR VIEW_NAME = 'JEFES_PRES';



3. VISTAS CON COLUMNAS VIRTUALES

Las columnas que se crean con una vista no tienen porqué ser columnas de la tabla de la que se obtiene la vista, pueden ser también columnas virtuales en las que se agrupa la información de las columnas de la tabla.

El departamento financiero necesita una vista que le proporcione información estadística sobre los sueldos de los empleados agrupados por departamento y tipo de empleo.

CREATE VIEW estadistica (departamento,empleo,salario_mínimo,salario_máximo, salario_medio, suma_de_salarios)

SELECT department_id, job_id, min(salary), max(salary), sum(salary), sum(salary)

FROM employee

GROUP BY department_id, job_id;

view ESTADISTICA creado.

SELECT * FROM ESTADISTICA;

A	DEPARTAMENTO	2 EMPLEO	SALARIO_MÍNIMO	SALARIO_MÁXIMO	SALARIO_MEDIO	SUMA_DE_SALARIOS
1	30	670	1250	1600	1400	5600
2	10	671	2450	2450	2450	2450
3	12	667	800	800	800	800
4	13	670	1250	1250	1250	3750
5	23	670	1250	1500	1333,3333333	4000
6	20	669	3000	3000	3000	6000
7	12	669	2875	3000	2937,5	5875
8	10	672	5000	5000	5000	5000
9	34	668	1850	1850	1850	1850
10	10	667	1300	1300	1300	1300
11	14	671	2200	2200	2200	2200
12	24	668	1800	1800	1800	1800
13	13	667	750	750	750	750
14	20	667	800	1100	950	1900
15	13	671	2850	2850	2850	2850
16	23	667	750	750	750	750
17	23	671	2750	2750	2750	2750
18	20	671	2975	2975	2975	2975
19	12	671	3000	3000	3000	3000
20	30	667	950	950	950	950
21	30	671	2850	2850	2850	2850
22	43	670	1300	1300	1300	1300

4. COMBINACIONES PARA DEFINIR VISTAS

Se pueden definir vistas que obtengan los datos de la combinación de varias tablas. Queremos que en la vista anterior se recupere el nombre del departamento, en vez de su número.

CREATE VIEW estadistica_1 (departamento, empleo, salario_mínimo, salario_máximo, salario_medio, suma_de_salarios) AS

SELECT name, job_id, min(salary), max(salary), avg(salary), sum(salary)

FROM employee, department

WHERE department.department_id = employee.department_id

GROUP BY name, job_id;

view ESTADISTICA_1 creado

SELECT *
FROM ESTADISTICA_1;

	B	l			le.	
	DEPARTAMENTO		SALARIO_MÍNIMO	SALARIO_MÁXIMO	SALARIO_MEDIO	SUMA_DE_SALARIOS
1	ACCOUNTING	667	1300	1300	1300	1300
2	RESEARCH	671	2975	3000	2987,5	5975
3	ACCOUNTING	672	5000	5000	5000	5000
4	SALES	667	750	950	816,66666666	2450
5	SALES	671	2750	2850	2816,6666666	8450
6	OPERATIONS	668	1800	1850	1825	3650
7	OPERATIONS	671	2200	2200	2200	2200
8	ACCOUNTING	671	2450	2450	2450	2450
9	RESEARCH	669	2875	3000	2968,75	11875
10	SALES	670	1250	1600	1331,8181818	14650
11	RESEARCH	667	800	1100	900	2700

5. MANIPULACION DE DATOS A TRAVES DE LAS VISTAS

Actualización de las tablas

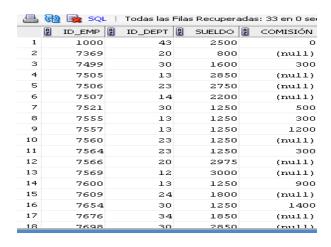
Como las vistas son ventanas a través de las que se ven los datos de las tablas, cualquier actualización (inserción, borrado, modificación) de una fila quedará automáticamente reflejada en la vista.

Si insertamos un nuevo empleado en la tabla employee se reflejará en las vistas nómina, estadistica y estadistica_1.

insert into employee values(1000, 'CANO', 'LUISA', 'Q', 669, 7505, SYSDATE, 2500, 0, 43);

1 filas insertadas.

SELECT * FROM NOMINA;



Actualización de las vistas

También se puede utilizar el lenguaje de manipulación de datos con las vistas, y los cambios quedarán automáticamente reflejados en los bloques de datos de las tablas.

Al igual que con las tablas se pueden conceder privilegios (comando GRANT) sobre las vistas a otros usuarios.

Al insertar filas a través de una vista hay que tener en cuenta que las columnas no especificadas en la definición de la vista tomarán valores nulos, por lo que si una está definida como no nula, no se podrán insertar filas a través de la vista.

La manipulación de datos a través de las vistas se podrá realizar sólo si:

- la vista está basada en una única tabla
- su definición no contiene ni funciones ni expresiones
- su definición no contiene cláusulas GROUP BY

Ejemplo:

INSERT INTO NOMINA VALUES (5000,30,4000,100);

1 filas insertadas.

SELECT * FROM NOMINA;

	A	ID_EMP	A	ID_DEPT	£	SUELDO	A	COMISIÓN
1		1000		43		2500		0
2		5000		30		4000		100
3		7369		20		800		(null)
4		7499		30		1600		300
5		7505		13		2850		(null)
6		7506		23		2750		(null)
7		7507		14		2200		(null)
_								

SELECT * FROM EMPLOYEE;

A	EMPLOYEE_ID	LAST_NAMI	FIRST_NAME	MIDDLE_INITIAL	JOB_ID	MANAGER_ID	HIRE_DATE	SALARY 2	COMMISSION	DEPARTMENT_ID
1	1000	CANO	LUISA	Q	669	7505	07/03/17	2500	0	43
2	5000	(null)	(null)	(null)	(null)	(null)	(null)	4000	100	30
3	7369	SMITH	JOHN	Q	667	7902	17/12/84	800	(null)	20
4	7499	ALLEN	KEVIN	J	670	7698	20/02/85	1600	300	30
5	7505	DOYLE	JEAN	K	671	7839	04/04/85	2850	(null)	13
6	7506	DENNIS	LYNN	S	671	7839	15/05/85	2750	(null)	23

En el tema anterior creamos la siguiente tabla:

CREATE TABLE libros

(titulo VARCHAR2(40) NOT NULL,

fecha_de_publicación DATE, precio NUMBER, resumen LONG, autor VARCHAR2(40));

table LIBROS creado.

Clausula "with check option"

Esta opción fuerza a que las inserciones y modificaciones hechas a través de la vista afecten sólo a filas que cumplan las condiciones de creación de la vista.

CREATE VIEW <nombre_de_la_vista> [lista_de_columnas>]
AS <sentencia_SQL_válida>
WITH CHECK OPTION;

Ejemplo:

CREATE VIEW libros_caros AS SELECT autor, titulo, precio FROM libros WHERE precio >= 4000 WITH CHECK OPTION;

view LIBROS_CAROS creado.

INSERT INTO libros_caros

VALUES ('CERVANTES', 'DON QUIJOTE DE LA MANCHA', 3000);

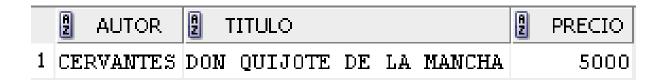
informe de error:

Error SQL: ORA-01402: view WITH CHECK OPTION where-clause violation 01402. 00000 - "view WITH CHECK OPTION where-clause violation"

INSERT INTO libros_caros VALUES ('CERVANTES', 'DON QUIJOTE DE LA MANCHA', 5000);

1 filas insertadas.

SELECT *
FROM libros_caros;



6. BORRADO DE VISTAS

El comando DROP VIEW borra la definición de las vistas del diccionario de datos, por lo que no se podrán volver a utilizar.

Las tablas referenciadas en la vista borrada no se verán afectadas.

DROP VIEW <nombre_de_la_vista>;

No existe un comando para modificar la definición de una vista, si se desea hacerlo habrá que borrarla y volver a crear la vista.

Si se borra una tabla o vista que estuviese contenida en la definición de una vista, se borrará esta última vista también.

7. RESUMEN

Los nombres por omisión de las columnas de las vistas coinciden con los nombres en las tablas de las que se han obtenido.

Cuando se seleccionen columnas virtuales para crear vistas se les deberá dar un nuevo nombre.

Cuando se seleccionen columnas de varias tablas habrá que cambiar el nombre a las columnas que resulten ambiguas.

Para definir vistas se pueden utilizar otras vistas o tablas.

Se pueden utilizar funciones, expresiones, la cláusula "group by" para definir nuevas vistas.

No se puede utilizar la cláusula "order by" para definir nuevas vistas, se podrá utilizar cuando se recuperen los datos de las vistas.

8. EJERCICIOS PROPUESTOS

1°. Cree una vista llamada **MEJORES** que contenga a los empleados que ganen más de 3000. Debe contener sólo el número del empleado, el salario, la comisión y el trabajo.

- 2°. Seleccione todos los datos utilizando la vista anterior, para su comprobación.
- 3°. Crear una vista, que obtenga una estadística de los salarios medios, máximos y la suma de salarios por cada tipo de empleo, y que figure el nombre de su función.
- 4°. Presentar la vista del ejercicio 1 MEJORES, ordenada por sueldo descendente.
- 5°. Crear una vista llamada ventas que contenga a los CLIENTES(cod_cliente, fecha_pedido cod pediod y cod productos) que hicieron el pedido en febrero del 1989
- 6°. Presentar la vista del ejercicio anterior
- 7°. Crea una vista llamada trabajadores_dep que contenga cuantos trabajadores tiene cada departamento de cada localidad.
- 8°. Presentar la vista del ejercicio anterior
- 9°. Crear una vista llamada dep23 que contenga a los trabajadores del departamento 23 que cobren comison. De ellos solo queremos saber el cod_empleado, salario, comision y departamento al que pertenecen.
- 10°. Presentar la vista del ejercicio anterior
- 11°. Insertar los siguientes valores en la vista dep23 cod_empleado=1000, salario=785, comision =50, departamento=10.
- 12°.- comprobar que se ha actualizado la vista y la tabla employee

13°.Los empleados del departamento 23 con comision 1000 pasan a tener una comision 500. Actualizar la comision a traves de la vista.

- 14°. Comprobar que se han actualizado dichos empleados en la tabla employee y en la vista.
- 15°. Borrar a los empleados del departamento 23 con comision 300 a traves de la vista.
- 16°. Consultar los datos de las vistas en el diccionario de datos
- 17° borrar la vista MEJORES