Упражнение 11. Изгледи

Изгледите (views) или т.нар. виртуални таблици представляват логическо представяне на подмножество от данни, съхраняващи се в една или повече таблици. Те не съдържат собствени данни, но са като прозорци, през които данните от таблиците могат да бъдат гледани и променяни (с известни ограничения). Таблиците, върху които се изгражда един изгледът се наричат базови таблици на изгледа. Всеки изглед се съхраняват като SELECT оператор в речника на данните на съответната СУБД. В този оператор могат да участват както таблици, така и други изгледи т.е. допуска се един изглед да бъде създаден въз основа на друг. Изгледите имат една основна цел — да преобразуват начина, по който виждате една таблица, част от таблица, или група таблици, без да създавате копия на съответните данни. Те ви дават възможност да използвате данните в една по-удобна "опаковка". Например определена таблица съдържа всички валидни поръчки на всеки клиент. Вие може да създадете изглед, който да ви показва само поръчките, валидни за отделен клиент, и да извикате отново този изглед, когато показвате на своя клиент данни от вашето място в Web.

За какво се използват изгледите?
За ограничаване на достъпа до данните;
За по-лесно изпълнение на сложни заявки;
За осигуряване на по-голяма независимост по отношение на структурата на данните;
За да представят различни гледни точки върху един и същ набор от данни;
За преименуване на колони.

Изгледите ограничават достъпа до данните, защото могат да визуализират само определени колони от дадена таблица. Те могат да бъдат използвани, за да се пишат по-прости и разбираеми заявки и да се връщат резултати от по-сложни такива. Например, могат да се използват изгледи за извличане на информация от множество таблици, без потребителят да знае как се използват операторите за свързване. Изгледи могат да бъдат използвани и за предоставяне на по-голяма независимост между потребители и приложни програми. Ако дадена приложна програма използва данни представени в даден изглед, то в структурата на базовите таблици или във формата на данните им могат да настъпат изменения без това да се отрази на приложната програма, стига изгледът, от който тя черпи данни, да остане непроменен. Посредством изгледите се улеснява предоставянето на данни на групи от потребители, съобразно конкретни критери за достъп.

Най-общо изгледите могат да бъдат класификацирани в две групи: прости и сложни.
Основната разлика между тях е свързана с DML (INSERT, UPDATE, DELETE) операциите.

	Прости изглед (simple)	*Сложни изглед (complex)*
Брой таблици	Една	Една или повече
Съдържат функции	Не	Да
Съдържат групирани данни	Не	Да
Могат да изпълняват DML операции върху изгледа	Да	Не винаги

Простите изгледи извличат данните си от само една таблица, не съдържа функции или групи от данни и могат да изпълнява DML операции върху изгледа, докато сложните могат да извличат данните си от много базови таблици, да съдържа функции и групирания и върху тях не винаги могат да се изпълняват DML операции.

Създаване на изглед
Изглед се създава чрез влагане на падзаявка в оператора CREATE VIEW.

```
CREATE VIEW view
[(alias[,alias]...)]
AS subquery
[WITH CHECK OPTION],
```

където:
view - е име на изгледа.
alias - определя имената на изразите от заявката на изгледа. Броят псевдоними трябва да отговаря на броя изрази селектирани от изгледа.
subquery – е завършен SELECT оператор (може да използвате псевдоними в SELECT клаузата и да не използвате подобната възможност от синтаксиса на CREATE VIEW). Не бива да се използва ORDER BY клауза в SELECT оператора. Ако искаме да подредим резултатното множество, то трябва да приложим ORDER BY клауза в заявката към изгледа.
WITH CHECK OPTION - определя дали само редове достъпни за изгледа могат да бъдат вмъквани и променяни..
Употреба на изгледите
Навсякъде, където използвате таблица, може да използвате изглед. Може да използвате изглед в клаузата FROM на запитване на SQL и дори в командите INSERT, UPDATE или DELETE.

Примери:
Следният изглед предоставя достъп до цялата информация за класовете кораби произвеждани от САЩ:

```
CREATE VIEW v_USA_classes
AS
SELECT *
FROM classes
WHERE country ='USA';
```

Следващият изглед изкарва средния брой оръдия на произвежданите класове по старни:

```
CREATE VIEW v_Country_AvgGuns
AS
SELECT avg(numGuns) as average_Guns,country
FROM classes
GROUP BY country;
```

Алтернативно имената на колоните в резултатната виртуална таблица могат да се зададат по следния начин:

```
CREATE VIEW v_Country_AvgGuns(average_Guns,country)
AS
SELECT avg(numGuns),country
FROM classes
GROUP BY country;
```

Резултатът от изпълнението можем да видим посредством заявката:

```
SELECT *
FROM v_Country_AvgGuns
ORDER BY average_Guns DESC;
```

Информация за характеристиките на всеки кораб можем да получим посредством изгледа ships_full_info, дефиниран по следния начин:

```
CREATE VIEW v_ships_full_info
(name,type,numGuns,bore,displacement,country,launched)
AS
SELECT s.name, c.type,c.numGuns,c.bore,c.displacement,c.country,s.launched
FROM classes c,ships s
WHERE c.class=s.class
```

Модификация на изглед

Модификацията на дефиницията на даден изглед се осъществява по различен начин в различните СУБД. В MS SQL Server имаме отделна команда:

```
ALTER VIEW view
[(alias[,alias]...)]
AS subquery
[WITH CHECK OPTION]
```

С ALTER VIEW изгледът се пресъздава без това да се отрази на обектите зависещи от изгледа (като съхранени процедури, тригери и др.), запазват се и установените права за достъп до изгледа.

Модифициране на данни с използване на изгледи

Когато актуализирате данни, използвайки изглед, вие в действителност актуализирате данни от таблицата. Това условие се прилага и когато вмъквате или изтривате данни от един изглед.

Не може да използвате следните видове изгледи за модифициране на данни:

Изгледи с оператори за множества като UNION, както и другите множествени операции, ако се поддържат от съответата СУБД (INTERSECT, EXTRACT\MINUS).
Изгледи с клаузи, съдържащи GROUP BY.
Изгледи с групови функции като AVG, SUM или MAX.
Изгледи, използващи функцията DISTINCT.
Изгледи, които обединяват таблици (с някои изключения). Може да създавате изгледи, които обединяват няколко таблици и все пак позволяват актуализиране на отделна таблица. Таблицата, която се актуализира, трябва да се свързва посредством еквисъединение с уникална колона от другата таблица. Дори в този случай всички колони в изгледа не могат да се актуализират.

Правила за изпълняване на DML операции върху изгледи, базирани на една таблица:
Не можете да *изтривате* ред от изглед, ако изгледът съдържа групови функции и групирания на данни, както и ключовата дума DISTINCT.
Не можете да *модифицирате* ред от изглед, ако той съдържа групови функции, групирания на данни, ключовата дума DISTINCT или колони дефинирани чрез израз.
Не можете да *добавяте* данни чрез изглед, ако съдържа групови функции, групирания на данни, ключовата дума DISTINCT, колони дефинирани чрез израз или в базовата таблица има NOT NULL колони, които не са включени в изгледа.
Можете да си гарантирате, че DML операците изпълнени върху изгледа ще станат в областта на видимост, определена от изгледа като използвате клаузата WHIT CHECK OPTION.

Пример:
Следващият изглед изкарва информация от таблицата с работници employees, но само за тези от тях, който работят в отдел номер 20:

```
CREATE VIEW emp_example
AS
SELECT *
FROM employees
WHERE department_id = 20
WITH CHECK OPTION;
```

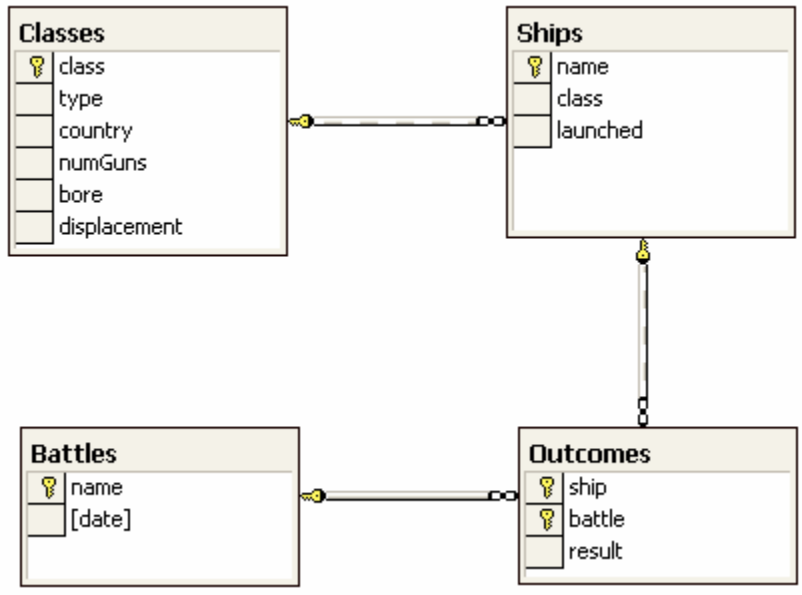
Всеки опит да променим номера на отдел в някой от редовете в изгледа ще пропадне, защото това нарушава ограничението наложено от WHIT CHECK OPTION. Тъй като отдели с номера различни от 20 не са в областта на видимост на изгледа, ако подобен UPDATE успее, то посредством изгледа няма как да видим *добре* резултата от изпълнението му - ефектът би бил изчезване на променените редове от изгледа. Целта на ограничението WHIT CHECK OPTION е недопускането на подобни ефекти.

Изтриване на изглед
DROP VIEW *view*;

Пример:
DROP VIEW emp_example;

Задачи

Върху база от данни Ships



1. Дефинирайте изглед наречен BritishShips, който дава за всеки британски кораб неговият клас, тип, брой оръдия, калибър, водоизместимост и годината, в която е пуснат на вода.
2. Напишете заявка, която използва изгледа от предната задача, за да покаже броят оръдия и водоизместимост на британските бойни кораби, пуснати на вода преди 1919.
3. Напищете съответната SQL заявка, реализираща задача 2, но без да използвате изглед.
4. Средната стойност на най-тежките класове кораби от всяка страна.
5. Създайте изглед за всички потънали кораби по битки.
6. Въведете кораба *California* като потънал в битката при *Guadalcanal* чрез изгледа от задача 5. За целта задайте подходяща стойност по премълчаване на колоната *result* от таблицата *Outcomes*.
7. Създайте изглед за всички класове с поне 9 оръдия. Използвайте WITH CHECK OPTION. Опитайте се да промените през изгледа броя оръдия на класа *Iowa* последователно на 15 и 5.
8. Променете изгледа от задача 7, така че броят оръдия да може да се променя без ограничения.
9. Създайте изглед с имената на битките, в които са участвали поне 3 кораба с под 9 оръдия и от тях поне един е бил увреден.