

4 Реалізація бізнес-логіки

4 Business logic implementation

4.1 Створення і використання уявлень / Creating and using views

Уявлення – віртуальна / логічна таблиця, яка базується на SQL запиті SELECT.

A database **view** is a virtual table or logical table which is defined as a SQL SELECT query.

Оскільки уявлення, також як і таблиця бази даних, містить рядки і стовпці, до нього можуть бути виконані **запити**.

Because a database view is similar to a database table, which consists of rows and columns, so you can **query** data against it.

4.1 Створення і використання уявлень / Creating and using views

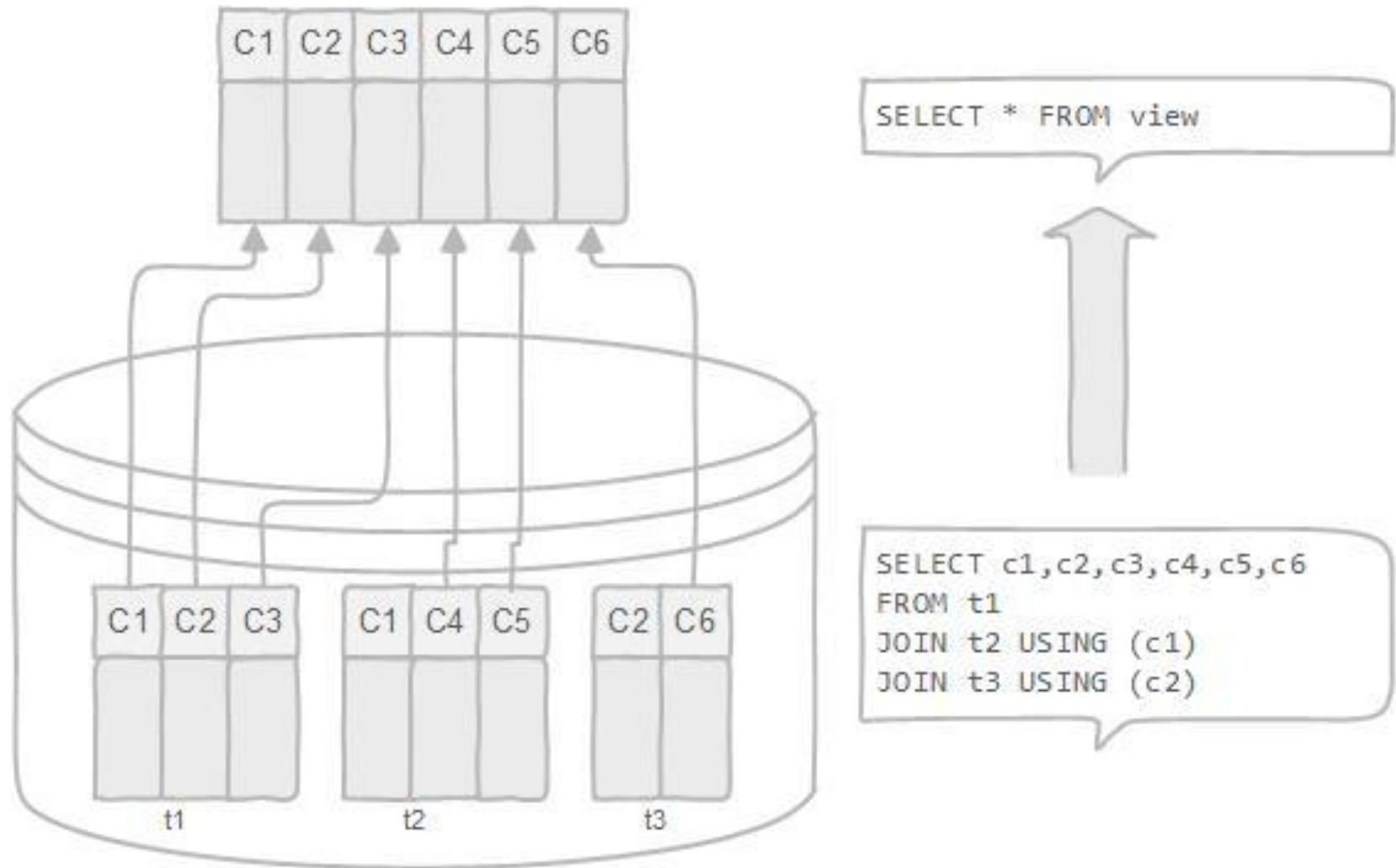
Більшість СУБД, включаючи MySQL, дозволяють **оновлювати** дані в базових таблицях через подання з деякими обмеженнями.

Most database management systems, including MySQL, allow you to **update** data in the underlying tables through the database view with some prerequisites.

Уявлення є **динамічними**, оскільки вони не пов'язані з фізичною схемою даних. СУБД зберігає уявлення як SQL SELECT запити. Коли дані таблиць змінюються, уявлення також **відображають** ці зміни.

A database view is **dynamic** because it is not related to the physical schema. The database system stores views as a SQL SELECT statement with joins. When the data of the tables changes, the view **reflects** that changes as well.

4.1 Створення і використання уявлень / Creating and using views



4.1 Створення і використання уявлень / Creating and using views

Переваги / Advantages

- Уявлення дозволяє **спростити складні запити**: уявлення визначається оператором SQL, який зв'язується з декількома базовими таблицями.
- A database view allows you to **simplify complex queries**: a database view is defined by an SQL statement that associates with many underlying tables.

4.1 Створення і використання уявлень / Creating and using views

Уявлення можна використовувати, щоб приховати складність базових таблиць для кінцевих користувачів і зовнішніх додатків. За допомогою уявлень досить використовувати тільки прості оператори SQL замість складних з безліччю об'єднань.

You can use database view to hide the complexity of underlying tables to the end-users and external applications. Through a database view, you only have to use simple SQL statements instead of complex ones with many joins.

```


SELECT supplier.supplier_id, supplier.supplier_address,
IFNULL(supplier_org.supplier_org_name,
        CONCAT(supplier_person.supplier_last_name, ' ',
        supplier_person.supplier_first_name, ' ',
        supplier_person.supplier_middle_name)) AS `Info`
FROM supplier LEFT OUTER JOIN supplier_org ON supplier.supplier_id =
supplier_org.supplier_id LEFT OUTER JOIN supplier_person ON
supplier.supplier_id = supplier_person.supplier_id;

```

```

SELECT * FROM supplier_info;

```

Result #1 (3×5)		
 supplier_id	supplier_address	Info
1	Kharkiv, Nauky av., 55, apt. 108	Petrov Pavlo Petrovych
2	Kyiv, Peremohy av., 154, apt. 3	Interfruit Ltd.
3	Kharkiv, Pushkinska str., 77	Ivanov Ilia Illych
4	Odesa, Derebasivska str., 75	Transservice LLC
5	Poltava, Soborna str., 15, apt. 43	Sydorov Serhii Stepanovych

4.1 Створення і використання уявлень / Creating and using views

- Уявлення бази даних допомагає **обмежити доступ до даних** для певних користувачів. Можливо, ви не захочете, щоб підмножина конфіденційних даних могло запитуватися усіма користувачами. Ви можете використовувати представлення бази даних, щоб надавати тільки загальнодоступні дані певної групи користувачів.
- A database view helps **limit data access** to specific users. You may not want a subset of sensitive data can be queryable by all users. You can use a database view to expose only non-sensitive data to a specific group of users.

user_ID	u_name	u_patronymic	u_surname	u_phone	u_email	u_status
1	Александр	Валерьевич	Иванов	58-98-78	ivanov@email.ru	active
2	Сергей	Иванович	Лосев	90-57-77	losev@email.ru	passive
3	Игорь	Николаевич	Симонов	95-66-61	simonov@email.ru	active
4	Максим	Петрович	Кузнецов	(NULL)	kuznetsov@email.ru	active
5	Анатолий	Юрьевич	Петров	(NULL)	(NULL)	lock
6	Александр	Александрович	Корнеев	89-78-36	korneev@email.ru	gold

```
SELECT CONCAT_WS(' ', u_name, u_patronymic, u_surname), u_status
FROM users;
```

CONCAT_WS(' ', u_name, u_patronymic, u_surname)	u_status
Александр Валерьевич Иванов	active
Сергей Иванович Лосев	passive
Игорь Николаевич Симонов	active
Максим Петрович Кузнецов	active
Анатолий Юрьевич Петров	lock
Александр Александрович Корнеев	gold

4.1 Створення і використання уявлень / Creating and using views

- Уявлення забезпечують **додатковий рівень безпеки**. Безпека є важливою частиною будь-якої СУБД. Уявлення забезпечують додатковий захист для СУБД. Можна створювати не оновлювані уявлення, щоб надавати дані тільки для читання конкретним користувачам. Користувачі можуть тільки отримувати дані в режимі read-only, але не можуть їх оновлювати.
- A database view provides **extra security layer**. Security is a vital part of any relational database management system. The database view offers additional protection for a database management system. The database view allows you to create the read-only view to expose read-only data to specific users. Users can only retrieve data in read-only view but cannot update it.

4.1 Створення і використання уявлень / Creating and using views

- Уявлення, на відміну від таблиць бази даних, можуть містити **обчислювані стовпці**.
- A database view enables computed columns. A database table should not have **calculated columns** however a database view should.

SELECT *, supplied_amount * supplied_cost FROM supplied;

supplied (5x17)				
contract_number	supplied_product	supplied_amount	supplied_cost	supplied_amount * supplied_cost
1	Audio Player	25	700.00	17,500.00
1	TV	10	1,300.00	13,000.00
1	Video Player	12	750.00	9,000.00
2	Audio Player	5	450.00	2,250.00
2	Stereo System	11	500.00	5,500.00
2	Video Player	8	450.00	3,600.00
3	Audio Player	11	550.00	6,050.00
3	Monitor	85	550.00	46,750.00
3	TV	52	900.00	46,800.00

4.1 Створення і використання уявлень / Creating and using views

- Подання бази даних забезпечує **зворотну сумісність**. Припустимо, є база даних, яку використовують програми. Виникає необхідність зміни структури бази даних, щоб виконати нові бізнес-вимоги. Деякі таблиці видаляються і створюються нові таблиці, при цьому небажано, щоб такі зміни впливали на інші додатки. В цьому випадку можна створити уявлення з тією ж структурою, що і застарілі таблиці, які були видалені.
- A database view enables **backward compatibility**. Suppose you have a central database, which many applications are using it. One day, you decide to redesign the database to adapt to the new business requirements. You remove some tables and create new tables, and you don't want the changes to affect other applications. In this scenario, you can create database views with the same schema as the legacy tables that you will remove.

4.1 Створення і використання уявлень / Creating and using views

Недоліки / Disadvantages

- **Продуктивність:** запит даних з уявлення може бути повільним, особливо якщо уявлення створюється на основі інших уявлень.
- **Performance:** querying data from a database view can be slow especially if the view is created based on other views.

4.1 Створення і використання уявлень / Creating and using views

- **Залежність таблиць:** уявлення створюється на основі базових таблиць бази даних. Всякий раз, коли структуру цих таблиць, з якими пов'язане уявлення, змінюється, також потрібно змінити уявлення.
- **Tables dependency:** you create a view based on underlying tables of the database. Whenever you change the structure of these tables that view associated with, you have to change the view as well.

4.1 Створення і використання уявлень / Creating and using views

CREATE VIEW

Використовується для створення нового уявлення у базі даних MySQL

Is used to create a new view in MySQL database

**CREATE [OR REPLACE] [ALGORITHM = {MERGE | TEMPTABLE |
UNDEFINED}]**

VIEW view_name [(column_list)]

AS select-statement

[WITH [CASCADED | LOCAL] CHECK OPTION];

4.1 Створення і використання уявлень / Creating and using views

OR REPLACE

в разі існування уявлення з таким ім'ям старе буде видалено,
а нове створено

is used to replace the existing view with the same name, the old
view will be removed and the new one will be created

Всередині бази даних **уявлення і таблиці спільно використовують один і той же простір імен**, тому уявлення і таблиця не можуть мати однакові імена. Крім того, ім'я уявлення має відповідати правилам іменування таблиці.

Within a database, **views and tables share the same namespace**, therefore, a view and a table cannot have the same name. In addition, the name of a view must follow the table's naming rules.

4.1 Створення і використання уявлень / Creating and using views

ALGORITHM

визначає алгоритм, який використовується при зверненні до подання

defines the algorithm used when a view is addressed

- **MERGE**

MySQL додає в використовуваний оператор відповідні частини з визначення уявлення і виконує результуючий оператор

MySQL adds corresponding parts from the view definition into the used operator and then executes the result operator

4.1 Створення і використання уявлень / Creating and using views

- **TEMPTABLE**

MySQL заносить вміст уявлення в тимчасову таблицю, над якою потім виконується оператор звернень до уявлення

MySQL inserts a content of the view into the temporary table and then executes the operator applied to the view

Подання, створене за допомогою оператора TEMPTABLE, не може бути оновлюваним

A view created with the TEMPTABLE operator cannot be updatable

4.1 Створення і використання уявлень / Creating and using views

- **UNDEFINED**

MySQL сам вибирає який алгоритм використовувати при зверненні до уявлення

MySQL chooses itself which algorithm should be used when a view is addressed

Використовується за замовчуванням, якщо конструкція

[ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]

відсутня

Is used by default if the **ALGORITHM** clause is not specified when view created

4.1 Створення і використання уявлень / Creating and using views

MERGE

рядки подання повинні повністю відповідати
рядкам базової таблиці

view rows must fully match the rows of the base table

CREATE VIEW v_total **AS**

SELECT supplied_product,

supplied_amount * supplied_cost **AS** total

FROM supplied

SELECT * FROM v_total;

supplied_product	total		contract_number	supplied_product	supplied_amount	supplied_cost
Audio Player	17,500.00	→	1	Audio Player	25	700.00
TV	13,000.00	→	1	TV	10	1,300.00
Video Player	9,000.00	→	1	Video Player	12	750.00
Audio Player	2,250.00	→	2	Audio Player	5	450.00
Stereo System	5,500.00	→	2	Stereo System	11	500.00
Video Player	3,600.00	→	2	Video Player	8	450.00
Audio Player	6,050.00	→	3	Audio Player	11	550.00
Monitor	46,750.00	→	3	Monitor	85	550.00

Кожен рядок відповідає рядку з таблиці supplied, використовується алгоритм MERGE

Each row corresponds to each row from the supplied table, the MERGE algorithm is used

- ім'я уявлення замінюється на ім'я таблиці
- the name of view is replaced with the name of table
- список полів замінюється на визначення полів з уявлення
- the list of fields is replaced with the fields definition from the view

4.1 Створення і використання уявлень / Creating and using views

MERGE не може бути використаний / cannot be used:

- агрегатні функції / aggregate functions

AVG, COUNT, MAX, MIN, SUM

- підзапити у SELECT або WHERE / nested queries in SELECT or WHERE clauses
- DISTINCT
- GROUP BY
- HAVING
- UNION, UNION ALL

```
CREATE VIEW v_max AS  
SELECT supplied_product, COUNT(*) AS num  
FROM supplied  
GROUP BY supplied_product;
```

supplied_product	num
Audio Player	5
Monitor	1
Phone	1
Printer	1
Stereo System	1
TV	5
Video Player	3

```
SELECT supplied_product, MAX(num) FROM v_max;
```



supplied_product	MAX(num)
Audio Player	5

```
SELECT supplied_product, MAX(COUNT(*))  
FROM supplied  
GROUP BY supplied_product;
```

SQL Error (1111): Invalid use of group function

```
SELECT supplied_product, MAX(num) FROM v_max;
```



```
DROP TABLE IF EXISTS tmp_table;
```

```
CREATE TEMPORARY TABLE tmp_table  
SELECT supplied_product, COUNT(*) AS num  
FROM supplied  
GROUP BY supplied_product;
```

```
SELECT supplied_product, MAX(num) FROM v_max;
```

```
DROP TABLE tmp_table;
```

supplied (2×1)	
supplied_product	MAX(num)
Audio Player	5

4.1 Створення і використання уявлень / Creating and using views

WITH [CASCADED | LOCAL]CHECK OPTION

все нові чи відредаговані рядки перевіряються на
відповідність визначенню уявлення

all inserted or updated records are checked to be
compatible with the view definition

перевіряється відповідність нового рядка умові WHERE у
визначенні уявлення

compatibility of a new record to the WHERE clause of the
view definition is checked

4.1 Створення і використання уявлень / Creating and using views

CASCADED, LOCAL

визначають глибину перевірки для уявлень, заснованих на інших уявленнях

define the depth of validation for views based on another views

LOCAL – перевірка умови WHERE тільки у визначенні подання / checks the WHERE clause of the view definition



CASCADED (default) – перевірка для всіх уявлень, на яких засновано дане уявлення / checks all views on which the current view is based

```
CREATE VIEW v_upd_supplied AS
SELECT * FROM supplied
WHERE supplied.supplied_amount > 10
WITH CHECK OPTION;
```



```
INSERT INTO v_upd_supplied VALUES(1, 'New Product', 5, 100);
```

SQL Error (1369): CHECK OPTION failed 'supply.v_upd_supplied'

```
INSERT INTO v_upd_supplied VALUES(1, 'New Product', 15, 100);
```

v_upd_supplied (4x13)			
 contract_number	 supplied_product	supplied_amount	supplied_cost
1	Audio Player	25	700.00
1	New Product	15	100.00
1	Video Player	12	750.00
2	Stereo System	11	500.00

```
SELECT *
FROM supplied;
```

 contract_number	 supplied_product	supplied_amount	supplied_cost
1	Audio Player	25	700.00
1	New Product	15	100.00

4.1 Створення і використання уявлень / Creating and using views

Оновлення уявлень, заснованих на декількох
таблицях

Update of views based on multiple tables

- **INSERT** працює тільки при додаванні даних в одну реальну таблицю / works if data is inserted into a single physical table
- **UPDATE** аналогічно / the same
- **DELETE** не підтримується / is not supported
- **INNER JOIN** використовується для об'єднання таблиць / is used to join the tables

4.1 Створення і використання уявлень / Creating and using views

Вертикальні уявлення – для обмеження доступу користувачів до стовпців таблиці

Горизонтальні уявлення – для обмеження доступу користувачів до рядків таблиці

Vertical views are used to restrict users' access to the table's columns

Horizontal views are used to restrict users' access to the table's rows

4.1 Створення і використання уявлень / Creating and using views

```
mysql> CREATE VIEW manager1  
-> AS SELECT * FROM books  
-> WHERE b_catID IN (SELECT catID  
-> FROM catalogs  
-> WHERE cat_name = 'Интернет' OR cat_name = 'Сети')  
-> ORDER BY b_name;  
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> SELECT b_name, b_price, b_count FROM manager1;
```

b_name	b_price	b_count
Web-конструирование	177.00	6
Анализ и диагностика компьютерных сетей	344.00	3
Безопасность сетей	462.00	5
Компьютерные сети	630.00	6
Общение в Интернете	85.00	5
Принципы маршрутизации в Internet	428.00	4
Поиск в Internet	107.00	2
Популярные интернет-браузеры	82.00	6
Локальные вычислительные сети	82.00	8
Сети. Поиск неисправностей	434.00	4
Самоучитель Интернет	121.00	4

```
11 rows in set (0.05 sec)
```

4.1 Створення і використання уявлень / Creating and using views

DROP VIEW [IF EXISTS] view_name [, view_name] ... ;

дозволяє видалити одне або кілька уявлень
used to remove one or multiple views

```
mysql> DROP VIEW cat, list_user, price;  
Query OK, 0 rows affected (0.03 sec)
```

```
mysql> SHOW TABLES;  
+-----+  
| Tables_in_book |  
+-----+  
| books          |  
| catalogs       |  
| orders         |  
| users          |  
+-----+  
4 rows in set (0.00 sec)
```

4.1 Створення і використання уявлень / Creating and using views

list_user

дозволяє відображати прізвище та ініціали покупців, приховуючи інші поля

allows to display the last name and initials of customers, whereas other fields are hidden

```
mysql> CREATE OR REPLACE VIEW list_user
-> AS SELECT CONCAT(u_surname, " ",
-> SUBSTRING(u_name,1,1), ".",
-> SUBSTRING(u_patronymic,1,1), ".") AS name
-> FROM users ORDER BY name;
Query OK, 0 rows affected (0.08 sec)
```

```
mysql> SELECT * FROM list_user;
```

name
Кузнецов М.П.
Корнеев А.А.
Петров А.Ю.
Иванов А.В.
Лосев С.И.
Симонов И.Н.

```
6 rows in set (0.02 sec)
```


4.1 Створення і використання уявлень / Creating and using views

price

дозволяє отримати загальну вартість книг в кожному каталозі
allows to retrieve the total cost of books in each catalog

```
mysql> CREATE VIEW price
-> AS SELECT b_catID, SUM(b_price*b_count) AS price
-> FROM books
-> GROUP BY b_catID
-> ORDER BY price;
Query OK, 0 rows affected (0.08 sec)
```

```
mysql> SELECT * FROM price;
```

b_catID	price
3	2908.00
2	4389.00
4	9514.00
1	12123.00
5	15177.00

```
5 rows in set (0.01 sec)
```

4.1 Створення і використання уявлень / Creating and using views

Подання можна використовувати в одному запиті SELECT
поряд з таблицею

A view might be used in the same SELECT query with a table

```
mysql> SELECT catalogs.cat_name, price.price  
-> FROM price, catalogs  
-> WHERE price.b_catID = catalogs.catID  
-> ORDER BY catalogs.catID;
```

cat_name	price
Программирование	12123.00
Интернет	4389.00
Базы данных	2908.00
Сети	9514.00
Мультимедиа	15177.00

```
5 rows in set (0.02 sec)
```

4.1 Створення і використання уявлень / Creating and using views

Запити до уявлень можуть містити функції групування,
також як і запити до звичайних таблиць

Views might be queried using aggregation functions just like the
ordinary database tables

```
mysql> SELECT MIN(price), MAX(price), SUM(price) FROM price;
```

MIN(price)	MAX(price)	SUM(price)
2908.00	15177.00	44111.00

```
1 row in set (0.02 sec)
```