

5 Цілісність даних, транзакції, права користувачів

5 Data integrity, transactions, user privileges

## 5.1    Механізми контролю цілісності даних / Data integrity control mechanisms

```
FOREIGN KEY [name_key] (col1, ... ) REFERENCES tbl (tbl_col, ... )  
[ON DELETE {CASCADE | SET NULL | NO ACTION | RESTRICT |  
SET DEFAULT}]  
[ON UPDATE {CASCADE | SET NULL | NO ACTION | RESTRICT |  
SET DEFAULT}]
```

Конструкція дозволяє задати зовнішній ключ з необов'язковим ім'ям `name_key` на стовпцях, які задаються в круглих дужках (один або декілька)

The design allows you to specify a foreign key with the optional name *name\_key* on the columns, which are specified in parentheses (one or more)

## 5.1    Механізми контролю цілісності даних / Data integrity control mechanisms

Ключове слово REFERENCES вказує таблицю *tbl*, на яку посилається зовнішній ключ, в круглих дужках вказуються імена стовпців

The keyword REFERENCES indicates the *tbl* table referenced by the foreign key, column names are indicated in parentheses

Необов'язкові конструкції ON DELETE і ON UPDATE дозволяють задати поведінку СУБД при видаленні і оновленні рядків з таблиці-предка

Optional constructions ON DELETE and ON UPDATE allow you to specify the behavior of the DBMS when deleting and updating rows from the parent table

## 5.1    Механізми контролю цілісності даних / Data integrity control mechanisms

Параметри, які йдуть за цими ключовими словами, мають таке значення  
The parameters following these keywords have the following meanings

### **CASCADE**

при видаленні або оновленні запису в батьківській таблиці, що містить  
первинний ключ, записи з посиланнями на це значення в таблиці-  
нащадку видаляються або оновлюються автоматично

when deleting or updating an entry in the parent table containing the primary  
key, entries with references to this value in the child table are deleted or  
updated automatically

## 5.1 Механізми контролю цілісності даних / Data integrity control mechanisms

### **SET NULL**

при видаленні або оновленні запису в батьківській таблиці, що містить первинний ключ, в таблиці-нащадку значення зовнішнього ключа, що посилається на батьківську таблицю, встановлюються в NULL

when deleting or updating an entry in the parent table containing the primary key, in the child table, the foreign key values referring to the ancestor table are set to NULL

### **NO ACTION**

при видаленні або оновленні записів, що містять первинний ключ, з таблицею-нащадком ніяких дій не проводиться

when deleting or updating records containing the primary key, no action is taken with the child table

## 5.1      Механізми контролю цілісності даних / Data integrity control mechanisms

### **RESTRICT**

якщо в таблиці-нащадку є записи, що посилаються на первинний ключ батьківської таблиці, при видаленні або оновленні записів з таким первинним ключем повертається помилка

if there are records in the child table that refer to the primary key of the parent table, an error is returned when the records with this primary key are deleted or updated

### **SET DEFAULT**

відповідно до стандарту SQL, при видаленні або оновленні первинного ключа в таблиці-нащадку для записів, що посилаються на нього, в поле зовнішнього ключа повинно встановлюватися значення за замовчуванням (в MySQL це ключове слово не обробляється)

according to the SQL standard, when deleting or updating the primary key in the child table, the default key value must be set for the records referring to it in the foreign key field (in MySQL this keyword is reserved but not processed)

## 5.1 Механізми контролю цілісності даних / Data integrity control mechanisms

```
ALTER TABLE contract
DROP FOREIGN KEY contract_ibfk_1;

ALTER TABLE contract
ADD CONSTRAINT contract_ibfk_1 FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id) ON DELETE NO ACTION ON UPDATE NO ACTION;

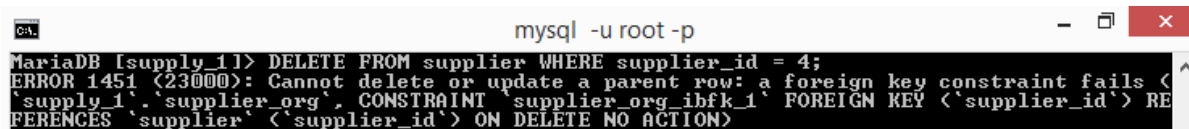
ALTER TABLE supplier_org
DROP FOREIGN KEY supplier_org_ibfk_1;

ALTER TABLE supplier_org
ADD CONSTRAINT supplier_org_ibfk_1 FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id) ON DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE supplier_person
DROP FOREIGN KEY supplier_person_ibfk_1;

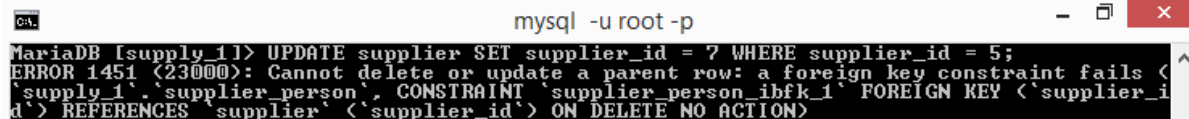
ALTER TABLE supplier_person
ADD CONSTRAINT supplier_person_ibfk_1 FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id) ON DELETE NO ACTION ON UPDATE NO ACTION;

DELETE FROM supplier WHERE supplier_id = 4;
```



A terminal window titled 'mysql -u root -p' shows the following error message: 'ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (<'supply\_1', 'supplier\_org', CONSTRAINT 'supplier\_org\_ibfk\_1' FOREIGN KEY (<'supplier\_id') REFERENCES 'supplier' (<'supplier\_id') ON DELETE NO ACTION)'. The error message is displayed in a black box with white text.

```
UPDATE supplier SET supplier_id = 7 WHERE supplier_id = 5;
```



A terminal window titled 'mysql -u root -p' shows the following error message: 'ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (<'supply\_1', 'supplier\_person', CONSTRAINT 'supplier\_person\_ibfk\_1' FOREIGN KEY (<'supplier\_id') REFERENCES 'supplier' (<'supplier\_id') ON DELETE NO ACTION)'. The error message is displayed in a black box with white text.

# 5.1 Механізми контролю цілісності даних / Data integrity control mechanisms

```
ALTER TABLE contract
DROP FOREIGN KEY contract_ibfk_1;

ALTER TABLE contract
ADD CONSTRAINT contract_ibfk_1 FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id) ON DELETE CASCADE ON UPDATE CASCADE;

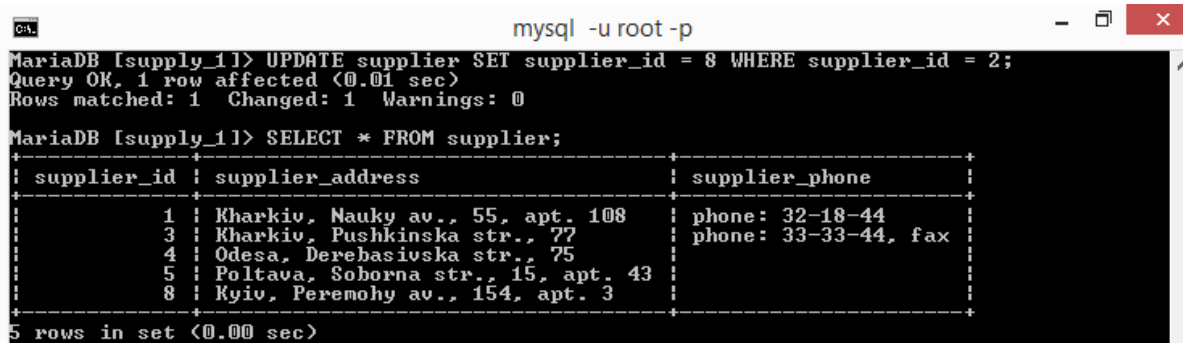
ALTER TABLE supplier_org
DROP FOREIGN KEY supplier_org_ibfk_1;

ALTER TABLE supplier_org
ADD CONSTRAINT supplier_org_ibfk_1 FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE supplier_person
DROP FOREIGN KEY supplier_person_ibfk_1;

ALTER TABLE supplier_person
ADD CONSTRAINT supplier_person_ibfk_1 FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id) ON DELETE CASCADE ON UPDATE CASCADE;
```

```
UPDATE supplier SET supplier_id = 8 WHERE supplier_id = 2;
```

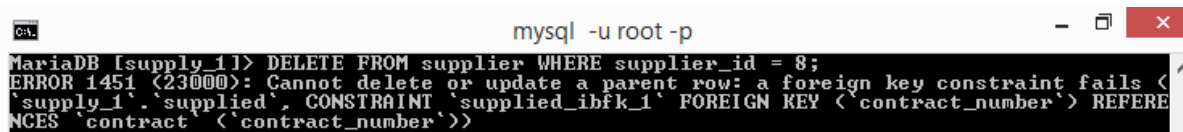


The screenshot shows a MySQL terminal window with the title 'mysql -u root -p'. The user is in the 'supply\_1' database. The first command executed is 'UPDATE supplier SET supplier\_id = 8 WHERE supplier\_id = 2;', which returns 'Query OK, 1 row affected (0.01 sec)' and 'Rows matched: 1 Changed: 1 Warnings: 0'. The second command is 'SELECT \* FROM supplier;', which returns a table with 5 rows. The table has columns 'supplier\_id', 'supplier\_address', and 'supplier\_phone'. The data is as follows:

supplier_id	supplier_address	supplier_phone
1	Kharkiv, Nauky av., 55, apt. 108	phone: 32-18-44
3	Kharkiv, Pushkinska str., 77	phone: 33-33-44, fax
4	Odesa, Derebasivska str., 75	
5	Poltava, Soborna str., 15, apt. 43	
8	Kyiv, Peremohy av., 154, apt. 3	

The terminal also shows '5 rows in set (0.00 sec)'.

```
DELETE FROM supplier WHERE supplier_id = 8;
```



The screenshot shows a MySQL terminal window with the title 'mysql -u root -p'. The user is in the 'supply\_1' database. The command executed is 'DELETE FROM supplier WHERE supplier\_id = 8;'. The terminal returns an error message: 'ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (<'supply\_1'.>'supplier', CONSTRAINT '<'supplier\_ibfk\_1'>' FOREIGN KEY (<'contract\_number'>' REFERENCES <'contract'>'contract' (<'contract\_number'>))'.



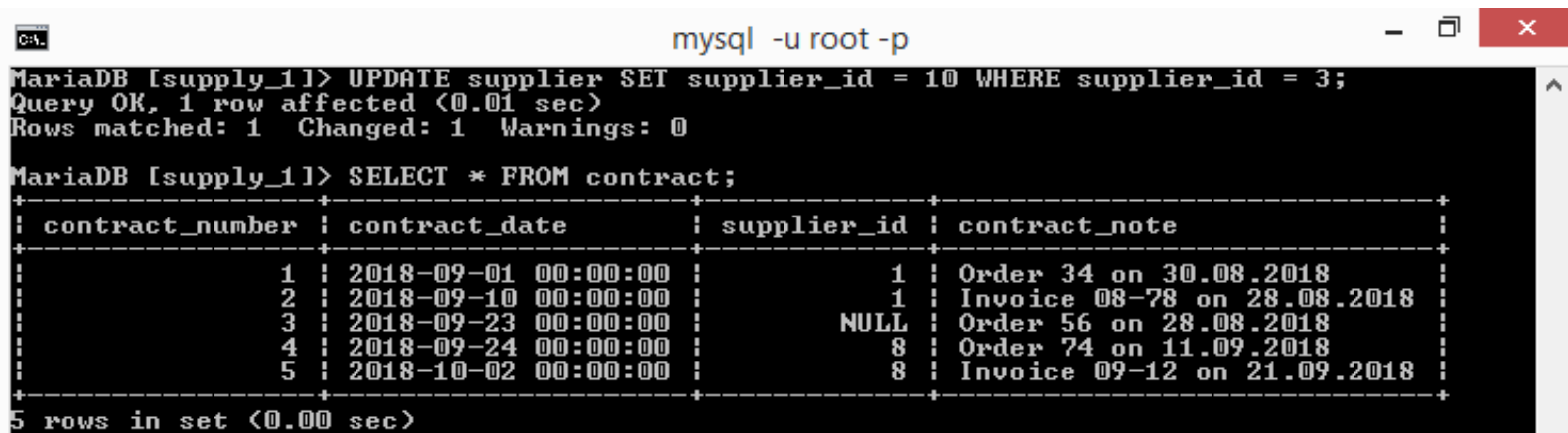
## 5.1 Механізми контролю цілісності даних / Data integrity control mechanisms

```
ALTER TABLE contract  
DROP FOREIGN KEY contract_ibfk_1;
```

```
ALTER TABLE contract  
MODIFY supplier_id INT NULL;
```

```
ALTER TABLE contract  
ADD CONSTRAINT contract_ibfk_1 FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id) ON DELETE SET NULL ON UPDATE SET NULL;
```

```
UPDATE supplier SET supplier_id = 10 WHERE supplier_id = 3;
```



The screenshot shows a MySQL terminal window with the title bar "mysql -u root -p". The terminal displays the following commands and results:

```
MariaDB [supply_11] > UPDATE supplier SET supplier_id = 10 WHERE supplier_id = 3;  
Query OK, 1 row affected (0.01 sec)  
Rows matched: 1  Changed: 1  Warnings: 0
```

```
MariaDB [supply_11] > SELECT * FROM contract;
```

contract_number	contract_date	supplier_id	contract_note
1	2018-09-01 00:00:00	1	Order 34 on 30.08.2018
2	2018-09-10 00:00:00	1	Invoice 08-78 on 28.08.2018
3	2018-09-23 00:00:00	NULL	Order 56 on 28.08.2018
4	2018-09-24 00:00:00	8	Order 74 on 11.09.2018
5	2018-10-02 00:00:00	8	Invoice 09-12 on 21.09.2018

```
5 rows in set (0.00 sec)
```

## 5.2 Механізм транзакцій / Transactional mechanism

Зміни в БД часто вимагають виконання кількох запитів, наприклад при покупці в електронному магазині потрібно додати запис в таблицю замовлень і зменшити число товарних позицій на складі

Changes in the database often require several requests, for example, when buying from an electronic store, you need to add an entry to the order table and reduce the number of items in the warehouse

У промислових БД одна подія може зачіпати більше число таблиць і вимагати численних запитів

In enterprise databases, one event may affect a lot of tables and require multiple queries

## 5.2 Механізм транзакцій / Transactional mechanism

Якщо на етапі виконання одного із запитів відбувається збій, це може порушити цілісність БД (товар може бути проданий, а число товарних позицій на складі не оновлено)

If at the stage of execution of one of the requests a failure occurs, it can break the integrity of the database (the goods can be sold and the number of items in the warehouse is not updated)

Щоб зберегти цілісність БД, всі зміни повинні виконуватися як єдине ціле

To preserve the integrity of the database, all changes must be made as a whole

## 5.2 Механізм транзакцій / Transactional mechanism

Або всі зміни успішно виконуються, або, в разі збою, БД приймає стан, який був до початку змін

Either all changes are successfully executed, or, in the case of a failure, the database returns to a state that was before the start of changes

Це забезпечується засобами обробки транзакцій

This is provided by transaction processing mechanism

Транзакція – послідовність операторів SQL, що виконуються як єдина операція, яка не переривається іншими клієнтами

Transaction is a sequence of SQL statements executed as a single operation that is not interrupted by other clients

## 5.2 Механізм транзакцій / Transactional mechanism

Поки відбувається робота з записами таблиці (оновлення або видалення), ніхто інший не може отримати доступ до цих записів, т. к. MySQL автоматично блокує доступ до них

While working with table entries (update or delete), no one else can access these records, since MySQL automatically blocks access to them

Таблиці ISAM, MyISAM і HEAP не підтримують транзакції, зараз їх підтримка здійснюється тільки в таблицях BDB і InnoDB

ISAM, MyISAM and HEAP tables do not support transactions, currently they are only supported in BDB and InnoDB tables

## 5.2 Механізм транзакцій / Transactional mechanism

Транзакції дозволяють об'єднувати оператори в групу і гарантувати, що всі оператори групи будуть виконані успішно

Transactions allow you to combine statements into a group and ensure that all statements of the group are executed successfully

Якщо частина транзакції виконується зі збоєм, результати виконання всіх операторів транзакції до місця збою скасовуються, приводячи БД до виду, в якому вона була до виконання транзакції

If a part of the transaction fails, the results of the execution of all transaction statements before the point of failure are canceled, leading the database to the form in which it was before the execution of the transaction

## 5.2 Механізм транзакцій / Transactional mechanism

За замовчуванням MySQL працює в режимі автоматичного завершення транзакцій, тобто як тільки виконується оператор поновлення даних, який модифікує таблицю, зміни тут же зберігаються на диску

By default, MySQL operates in the mode of automatic completion of transactions, i.e., as soon as the data update statement that modifies the table is executed, the changes are immediately saved on disk

Щоб об'єднати оператори в транзакцію, слід відключити цей режим: `SET AUTOCOMMIT = 0;`

To combine operators into a transaction, you should disable this mode: `SET AUTOCOMMIT = 0;`

## 5.2 Механізм транзакцій / Transactional mechanism

Після відключення режиму для завершення транзакції необхідно ввести оператор COMMIT, для відкату – ROLLBACK

After disabling the mode, you must enter the COMMIT statement to complete the transaction, and ROLLBACK for a rollback

Включити режим автоматичного завершення транзакцій для окремої послідовності операторів можна за допомогою оператора START TRANSACTION

You can enable the automatic completion of transactions for a separate sequence of statements using the START TRANSACTION operator



## 5.2 Механізм транзакцій / Transactional mechanism

Для таблиць InnoDB є оператори SAVEPOINT і ROLLBACK TO SAVEPOINT, які дозволяють працювати з іменованими точками початку транзакції

For InnoDB tables, there are SAVEPOINT and ROLLBACK TO SAVEPOINT statements that allow you to work with named transaction start points

```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO catalogs VALUES(NULL,'Периферия');
Query OK, 1 row affected (0.00 sec)

mysql> SAVEPOINT point1;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO catalogs VALUES(NULL,'Разное');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM catalogs;
+-----+-----+
| cat_ID | cat_name |
+-----+-----+
|      1 | Программирование |
|      2 | Интернет |
|      3 | Базы данных |
|      4 | Сети |
|      5 | Мультимедиа |
|     12 | Периферия |
+-----+-----+
6 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM catalogs;
+-----+-----+
| cat_ID | cat_name |
+-----+-----+
|      1 | Программирование |
|      2 | Интернет |
|      3 | Базы данных |
|      4 | Сети |
|      5 | Мультимедиа |
|     12 | Периферия |
|     13 | Разное |
+-----+-----+
7 rows in set (0.00 sec)

mysql> ROLLBACK TO SAVEPOINT point1;
Query OK, 0 rows affected (0.02 sec)
```

## 5.2 Механізм транзакцій / Transactional mechanism

### **Атомарність / Atomicity**

транзакція є атомарної одиницею обробки даних, отже вона або виконується повністю, або не виконується зовсім

a transaction is an atomic unit of processing, that is, either it is performed in its entirety or not performed at all

### **Узгодженість / Consistency**

транзакція повинна перевести базу даних з одного узгодженого стану в інший узгоджений стан

a transaction should take the database from one consistent state to another consistent state

## 5.2 Механізм транзакцій / Transactional mechanism

### **Ізоляція / Isolation**

транзакція повинна бути виконана так, як якщо б вона була єдиною в системі, не повинно бути ніяких перешкод від інших одночасних транзакцій, які одночасно виконуються

a transaction should be executed as if it is the only one in the system, there should not be any interference from the other concurrent transactions that are simultaneously running

### **Довговічність / Durability**

якщо зафіксована транзакція призводить до зміни, це зміна має бути довготривалою в базі даних і не губитися в разі будь-якого збою

If a committed transaction brings about a change, that change should be durable in the database and not lost in case of any failure

## 5.2 Механізм транзакцій / Transactional mechanism

При паралельному виконанні транзакцій можливі наступні проблеми  
The following problems are possible when executing transactions in parallel

### загублене оновлення / lost update

при одночасній зміні одного блоку даних різними транзакціями  
втрачаються всі зміни, крім останньої

when simultaneously changing one data block with different transactions, all changes are lost except the last

Transaction 1	Transaction 2
UPDATE table SET a = a + 20 WHERE a = 1	UPDATE table SET a = a + 25 WHERE a = 1

## 5.2 Механізм транзакцій / Transactional mechanism

### брудне читання / dirty read

читання даних, доданих або змінених транзакцією, яка згодом не підтвердиться (відкотиться)

reading data added or modified by a transaction that is not subsequently confirmed (rolled back)

Transaction 1	Transaction 2
UPDATE table SET a = a + 1 WHERE a = 1	
	SELECT a FROM table WHERE a = 1
ROLLBACK	

## 5.2 Механізм транзакцій / Transactional mechanism

### неповторюване читання / non-repeatable read

при повторному читанні в рамках однієї транзакції раніше прочитані дані виявляються зміненими

when re-reading in the same transaction, the previously read data is changed

Transaction 1	Transaction 2
	SELECT a FROM table WHERE a = 1
UPDATE table SET a = a + 1 WHERE a = 1	
COMMIT	
	SELECT a FROM table WHERE a = 1

## 5.2 Механізм транзакцій / Transactional mechanism

### фантомне читання / phantom read

Ситуація, коли при повторному читанні в рамках однієї транзакції одна і та ж вибірка дає різні множини рядків

The situation when, when re-reading in the same transaction, the same sample gives different sets of rows

Transaction 1	Transaction 2
	SELECT SUM(b) FROM table
INSERT INTO table (a, b) VALUES (15,20)	
COMMIT	
	SELECT SUM(b) FROM table

## 5.2 Механізм транзакцій / Transactional mechanism

### **Рівень ізоляції транзакцій / Transaction isolation level**

ступінь, що забезпечується внутрішніми механізмами СУБД, захисту від усіх або деяких видів перерахованих вище неузгодженостей даних, що виникають при паралельному виконанні транзакцій

the degree of protection provided by the internal mechanisms of the DBMS against all or some of the above listed inconsistencies of data arising during the parallel execution of transactions

### **READ UNCOMMITTED**

Якщо кілька паралельних транзакцій намагаються змінювати один і той же рядок таблиці, то в остаточному варіанті рядок буде мати значення, визначене усім набором успішно виконаних транзакцій

If several parallel transactions attempt to change the same row of the table, then in the final version the row will have the value defined by the entire set of successfully completed transactions



## 5.2 Механізм транзакцій / Transactional mechanism

### **READ COMMITED**

На цьому рівні забезпечується захист від «брудного» читання, тим не менш, в процесі роботи однієї транзакції інша може бути успішно завершена і зроблені нею зміни зафіксовані

At this level, protection against a “dirty” reading is provided, however, during the execution of one transaction, the other one can be successfully completed and the changes made by it are fixed

### **REPEATABLE READ**

Рівень, при якому читаюча транзакція «не бачить» зміни даних, які були нею раніше прочитані. При цьому ніяка інша транзакція не може змінювати дані, що читаються поточної транзакцією, поки та не закінчена

The level at which the reading transaction "does not see" the changes in the data that it had previously read. However, no other transaction can change the data read by the current transaction until it is completed

## 5.2 Механізм транзакцій / Transactional mechanism

### **SERIALIZABLE**

Найвищий рівень ізолюваності; транзакції повністю ізолюються одна від одної, кожна виконується послідовно, як ніби паралельних транзакцій не існує. Тільки на цьому рівні паралельні транзакції не схильні до ефекту «фантомного читання»

The highest level of isolation; transactions are completely isolated from each other, each performed sequentially, as if parallel transactions do not exist. Only at this level parallel transactions are not affected by "phantom reading"

## 5.2 Механізм транзакцій / Transactional mechanism

Isolation level	Phantom reads	Non-repeatable read	Dirty read	Lost update
SERIALIZABLE	+	+	+	+
REPEATABLE READ	-	+	+	+
READ COMMITTED	-	-	+	+
READ UNCOMMITTED	-	-	-	+
NULL	-	-	-	-

## 5.2 Механізм транзакцій / Transactional mechanism

**SET [ GLOBAL | SESSION ] TRANSACTION ISOLATION LEVEL**

**{ READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ |  
SERIALIZABLE }**

За замовчуванням рівень ізоляції встановлюється для подальшої (не початкової) транзакції

By default, the isolation level is set for a subsequent (non-initial) transaction

При використанні ключового слова GLOBAL дана команда встановлює рівень ізоляції за замовчуванням глобально для всіх нових з'єднань, створених від цього моменту

When using the GLOBAL keyword, this command sets the default isolation level globally for all new connections created from this moment

При використанні ключового слова SESSION встановлюється рівень ізоляції за замовчуванням для всіх майбутніх транзакцій, які виконуються в поточному з'єднанні

Using the SESSION keyword sets the default isolation level for all future transactions performed on the current connection

## 5.2 Механізм транзакцій / Transactional mechanism

```
SELECT supplied.contract_number, supplied.supplied_product, supplied.supplied_cost, supplied.supplied_amount,  
       supplier.supplier_address, contract.contract_date  
FROM supplied, contract, supplier  
WHERE contract.contract_number = supplied.contract_number AND supplier.supplier_id = contract.supplier_id  
AND contract.contract_number = 1;
```

```
SET AUTOCOMMIT = 0;  
START TRANSACTION;  
INSERT INTO supplied VALUES (1, 'Vacuum cleaner', 22, 390);
```

```
SELECT supplied.contract_number, supplied.supplied_product, supplied.supplied_cost, supplied.supplied_amount,  
       supplier.supplier_address, contract.contract_date  
FROM supplied, contract, supplier  
WHERE contract.contract_number = supplied.contract_number AND supplier.supplier_id = contract.supplier_id  
AND contract.contract_number = 1;
```

```
ROLLBACK;
```

```
SELECT supplied.contract_number, supplied.supplied_product, supplied.supplied_cost, supplied.supplied_amount,  
       supplier.supplier_address, contract.contract_date  
FROM supplied, contract, supplier  
WHERE contract.contract_number = supplied.contract_number AND supplier.supplier_id = contract.supplier_id  
AND contract.contract_number = 1;
```

## 5.2 Механізм транзакцій / Transactional mechanism

```
mysql -u root -p
MariaDB [supply_11] > SELECT supplied.contract_number, supplied.supplied_product, supplied.supplied_cost, supplied.supplied_amount,
-> supplier.supplier_address, contract.contract_date
-> FROM supplied, contract, supplier
-> WHERE contract.contract_number = supplied.contract_number AND supplier.supplier_id = contract.supplier_id
-> AND contract.contract_number = 1;
```

contract_number	supplied_product	supplied_cost	supplied_amount	supplier_address	contract_date
1	Audio Player	700.00	25	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00
1	TU	1300.00	10	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00
1	Video Player	750.00	12	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00

3 rows in set (0.00 sec)

```
mysql -u root -p
MariaDB [supply_11] > SELECT supplied.contract_number, supplied.supplied_product, supplied.supplied_cost, supplied.supplied_amount,
-> supplier.supplier_address, contract.contract_date
-> FROM supplied, contract, supplier
-> WHERE contract.contract_number = supplied.contract_number AND supplier.supplier_id = contract.supplier_id
-> AND contract.contract_number = 1;
```

contract_number	supplied_product	supplied_cost	supplied_amount	supplier_address	contract_date
1	Audio Player	700.00	25	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00
1	TU	1300.00	10	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00
1	Vacuum cleaner	390.00	22	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00
1	Video Player	750.00	12	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00

4 rows in set (0.00 sec)

```
mysql -u root -p
MariaDB [supply_11] > ROLLBACK;
Query OK, 0 rows affected (0.00 sec)

MariaDB [supply_11] >
MariaDB [supply_11] > SELECT supplied.contract_number, supplied.supplied_product, supplied.supplied_cost, supplied.supplied_amount,
-> supplier.supplier_address, contract.contract_date
-> FROM supplied, contract, supplier
-> WHERE contract.contract_number = supplied.contract_number AND supplier.supplier_id = contract.supplier_id
-> AND contract.contract_number = 1;
```

contract_number	supplied_product	supplied_cost	supplied_amount	supplier_address	contract_date
1	Audio Player	700.00	25	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00
1	TU	1300.00	10	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00
1	Video Player	750.00	12	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00

3 rows in set (0.00 sec)

## 5.2 Механізм транзакцій / Transactional mechanism

```
create table m2_order (  
    order_id int not null,  
    product_id int not null,  
    product_amount int not null,  
    primary key (order_id, product_id),  
    foreign key (product_id) references m2_products(product_id)  
);
```

```
set AUTOCOMMIT = 0;
```

```
start transaction;
```



```
insert into m2_order (order_id, product_id, product_amount) values (1, 1, 1);  
insert into m2_order (order_id, product_id, product_amount) values (1, 3, 3);  
insert into m2_order (order_id, product_id, product_amount) values (1, 4, 2);
```

```
select * from m2_order;
```

```
rollback;
```

```
select * from m2_order;
```

```
set AUTOCOMMIT = 1;
```

m2_order (3×3)			m2_order (3×0)		
 order_id	 product_id	product_amount			
1	1	1			
1	3	3			
1	4	2			