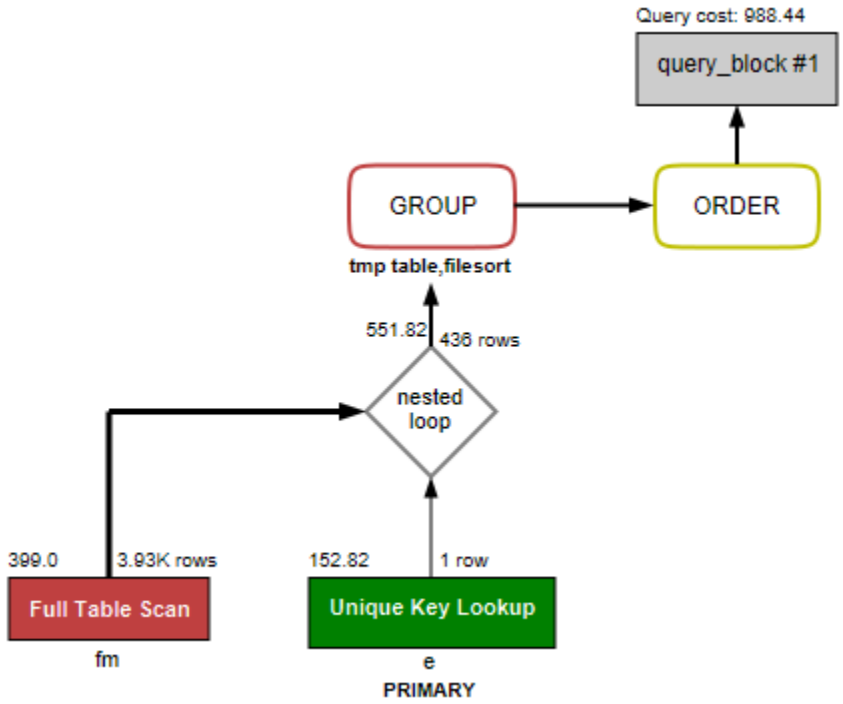
Лабораторная работа №5 Индексы, хранимые процедуры, представления	Ф.И.О.	Медведева С.А.
	Группа	ИВТ-261
	Преподаватель	Аль-Мерри Гаис
	Дата сдачи	

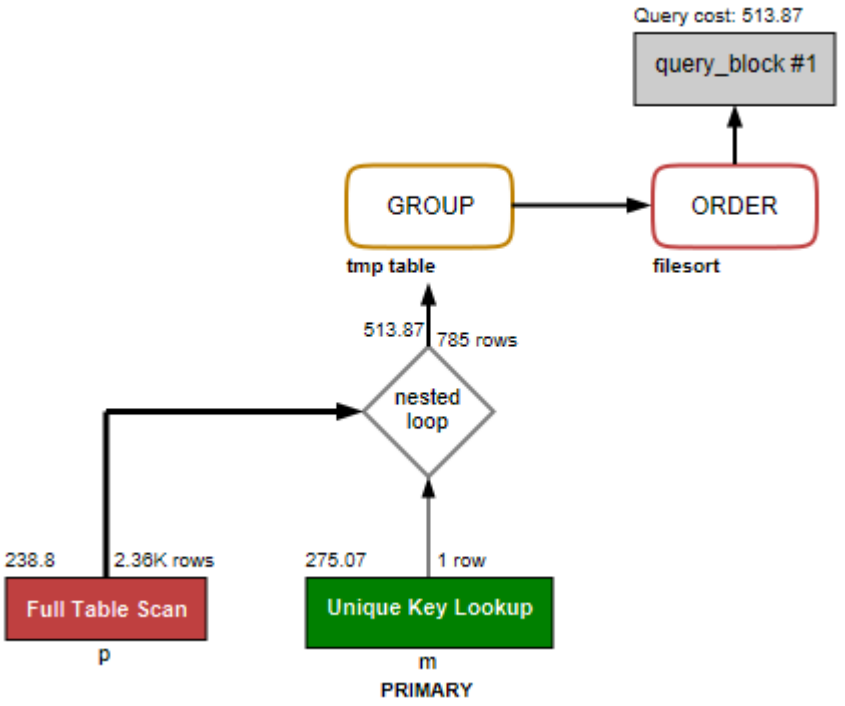
Индексы

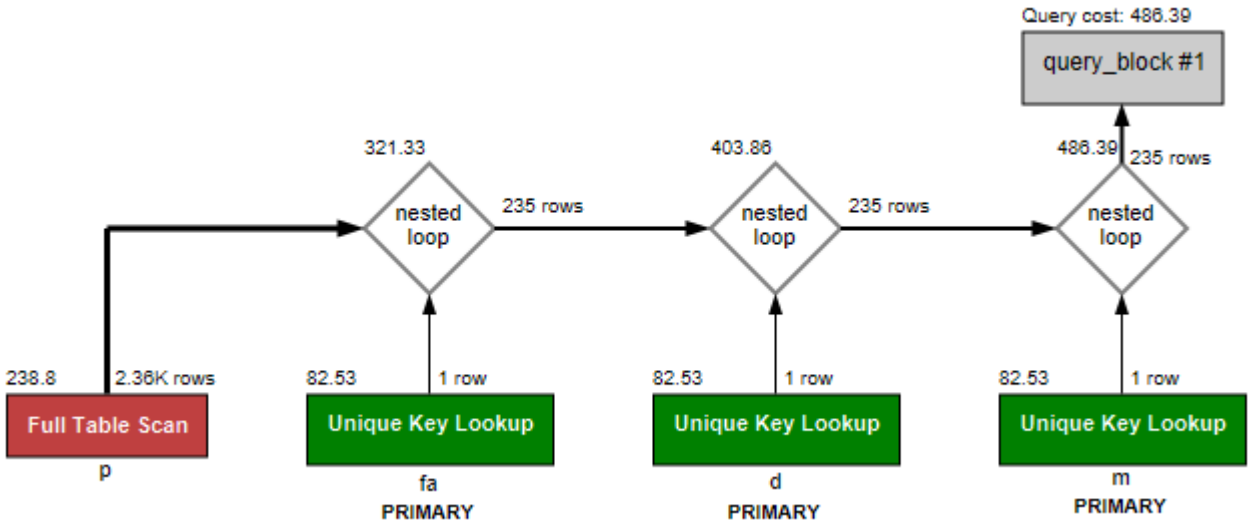
Коды самых используемых запросов или самых медленных запросов. ***Без оптимизации***

Комментарий к коду	1) Найти среднюю стоимость оборудования для фермеров с опытом от 8 до 12 лет, сгруппировать по стажу фермера
Код программы	SELECT fm.work_experience, AVG(e.cost) AS avg_equipment_cost FROM farmer fm JOIN equipment e ON fm.equipment_id = e.id WHERE fm.work_experience BETWEEN 8 AND 12 GROUP BY fm.work_experience ORDER BY fm.work_experience;
Затраченное время	**Timing (as measured at client side):** Execution time: 0:00:0.00000000 **Timing (as measured by the server):** Execution time: 0:00:0.00745390 Table lock wait time: 0:00:0.00000400

Визуальные планы выполнения запросов	
Explain	

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	fm	NULL	ALL	equipment_id	NULL	NULL	NULL	3930	11.11	Using where; Using temporary; Using filesort
1	SIMPLE	e	NULL	eq_ref	PRIMARY	PRIMARY	4	farm_db.fm.equipment_id	1	100.00	NULL

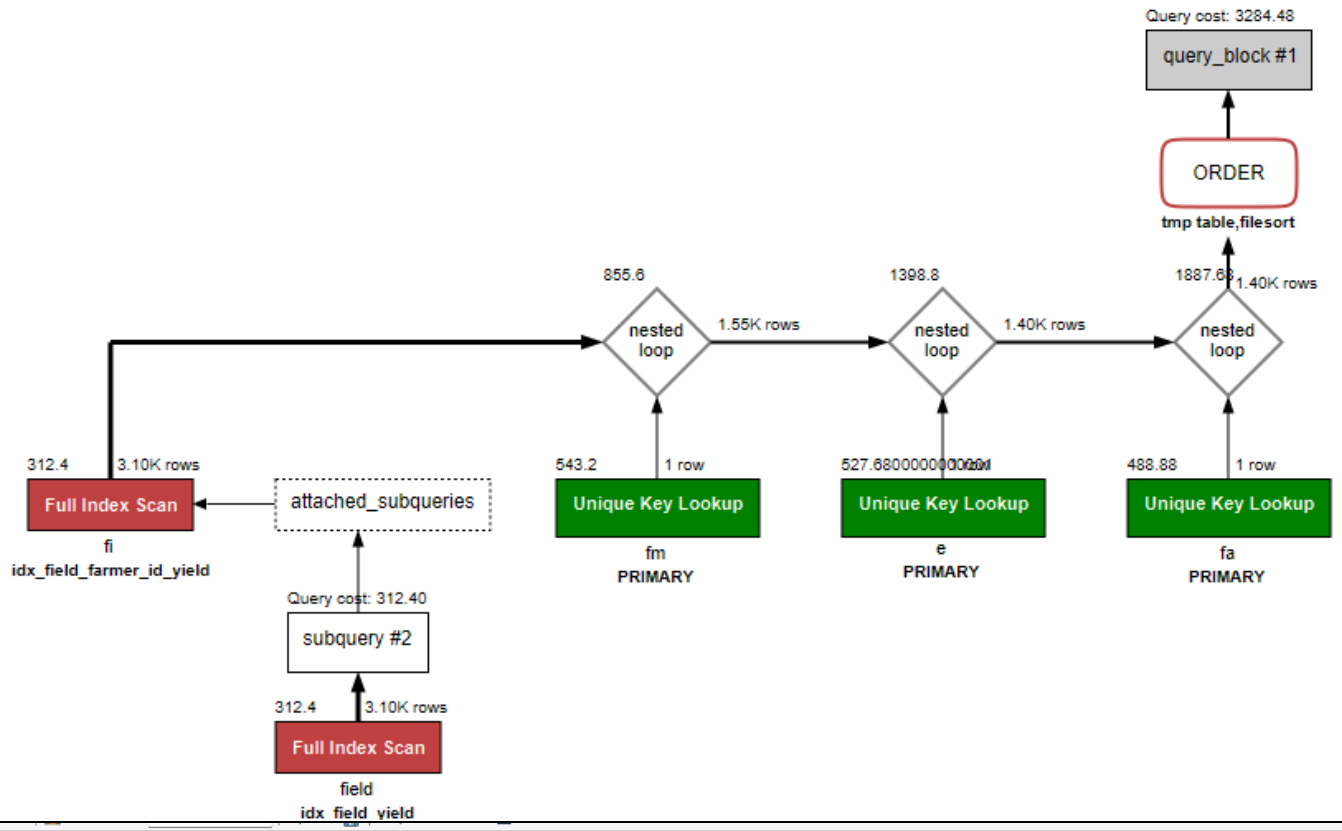
Комментарий к коду	2) Найти для каждого сегмента рынка среднюю цену продуктов, продаваемых на нем, учитывая только продукты дороже 50, и показать топ 3 сегмента по средней цене
Код программы	SELECT m.market_segment, AVG(p.price) AS avg_prod_price FROM market m JOIN product p ON m.id = p.market_id WHERE p.price > 50 GROUP BY m.market_segment ORDER BY avg_prod_price DESC LIMIT 3;
Затраченное время	**Timing (as measured at client side):** Execution time: 0:00:0.01500000 **Timing (as measured by the server):** Execution time: 0:00:0.01192880 Table lock wait time: 0:00:0.00001000
Визуальные планы выполнения запросов	
Explain	id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra 1 | SIMPLE | p | NULL | ALL | market_id | NULL | NULL | NULL | 2358 | 33.33 | Using where; Using temporary; Using filesort 1 | SIMPLE | m | NULL | eq_ref | PRIMARY | PRIMARY | 4 | farm_db.p.market_id | 1 | 100.00 | NULL

Комментарий к коду	3) Продукт -> Ферма -> Рынок -> Дистрибьютор
Код программы	EXPLAIN SELECT p.category, fa.name AS Farm, m.market_segment AS Market, d.company_name AS Distributor FROM farm_db.product p JOIN farm_db.farm fa ON p.farm_id = fa.id JOIN farm_db.market m ON fa.market_id = m.id JOIN farm_db.distributor d ON p.distributor_id = d.id WHERE p.category = 'Зерновые';
Затраченное время	**Timing (as measured at client side):** Execution time: 0:00:0.00000000 **Timing (as measured by the server):** Execution time: 0:00:0.00246040 Table lock wait time: 0:00:0.00000400
Визуальные планы выполнения запросов	

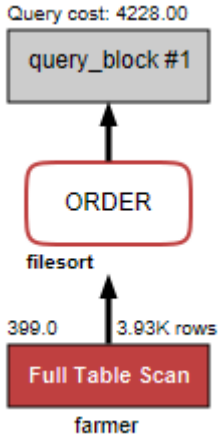
Explain	

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	p	NULL	ALL	idx_product_farm_id,idx_product_distributor_id	NULL	NULL	NULL	2358	10.00	Using where
1	SIMPLE	fa	NULL	eq_ref	PRIMARY,idx_farm_market_id	PRIMARY	4	farm_db.p.farm_id	1	100.00	Using where
1	SIMPLE	d	NULL	eq_ref	PRIMARY	PRIMARY	4	farm_db.p.distributor_id	1	100.00	NULL
1	SIMPLE	m	NULL	eq_ref	PRIMARY,idx_market_id_segment	PRIMARY	4	farm_db.fa.market_id	1	100.00	NULL

Комментарий к коду	4) Фермеры с урожайностью выше средней, использующие оборудование НЕ в "Отличном" состоянии.
Код программы	SELECT fm.full_name, fa.name AS farm_name, fi.yield, e.conditions AS equipment_condition FROM farm_db.farmer fm JOIN farm_db.field fi ON fm.id = fi.farmer_id JOIN farm_db.equipment e ON fm.equipment_id = e.id JOIN farm_db.farm fa ON fm.farm_id = fa.id WHERE fi.yield > (SELECT AVG(yield) FROM farm_db.field) AND e.conditions <> 'Отличное' ORDER BY fm.full_name;
Затраченное время	**Timing (as measured at client side):** Execution time: 0:00:0.01500000 **Timing (as measured by the server):** Execution time: 0:00:0.02420180 Table lock wait time: 0:00:0.00004700

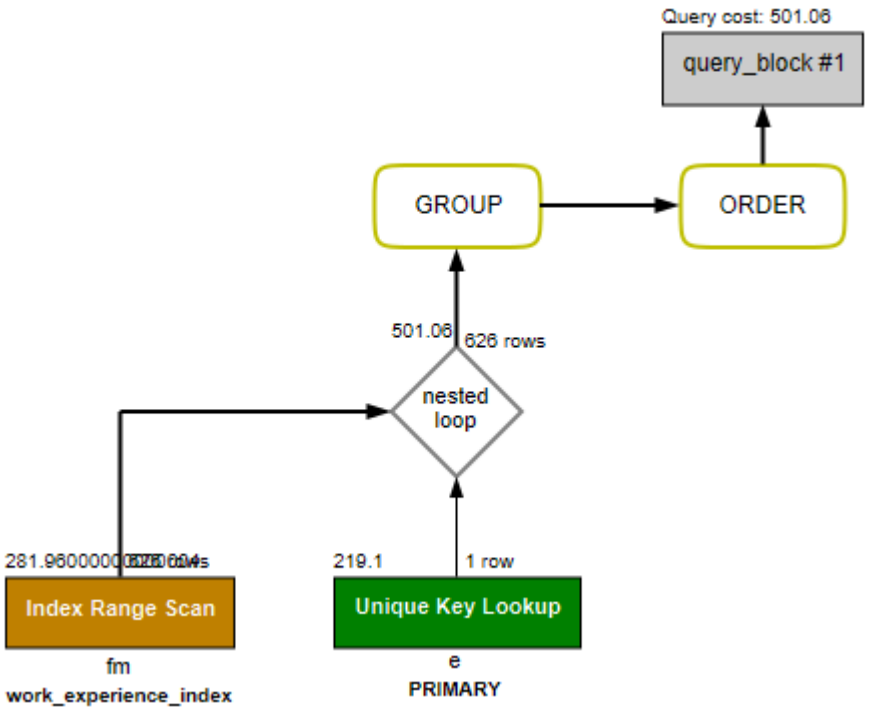
Визуальные планы выполнения запросов	
Explain	

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	fi	NULL	index	idx_field_farmer_id_yield,idx_field_yield	idx_field_farmer_id_yield	10	NULL	3099	50.08	Using where; Using index; Using temporary; Usi...
1	PRIMARY	fm	NULL	eq_ref	PRIMARY,equipment_id,idx_farmer_equipment...	PRIMARY	4	farm_db.fi.farmer_id	1	100.00	Using where
1	PRIMARY	e	NULL	eq_ref	PRIMARY	PRIMARY	4	farm_db.fm.equipment_id	1	90.00	Using where
1	PRIMARY	fa	NULL	eq_ref	PRIMARY	PRIMARY	4	farm_db.fm.farm_id	1	100.00	NULL
2	SUBQUERY	field	NULL	index	NULL	idx_field_yield	5	NULL	3099	100.00	Using index

Комментарий к коду	5) Рейтинг фермеров по соотношению зарплаты к опыту
Код программы	SELECT full_name, work_experience, labor_payment, (labor_payment / work_experience) AS salary_experience_ratio FROM farm_db.farmer WHERE work_experience > 0 ORDER BY salary_experience_ratio DESC;
Затраченное время	**Timing (as measured at client side):** Execution time: 0:00:0.00000000 **Timing (as measured by the server):** Execution time: 0:00:0.01511880 Table lock wait time: 0:00:0.00000400
Визуальные планы выполнения запросов	
Explain	id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra 1 | SIMPLE | farmer | NULL | ALL | NULL | NULL | NULL | NULL | 3930 | 33.33 | Using where; Using filesort

Коды самых используемых запросов или самых медленных запросов. *С оптимизацией*

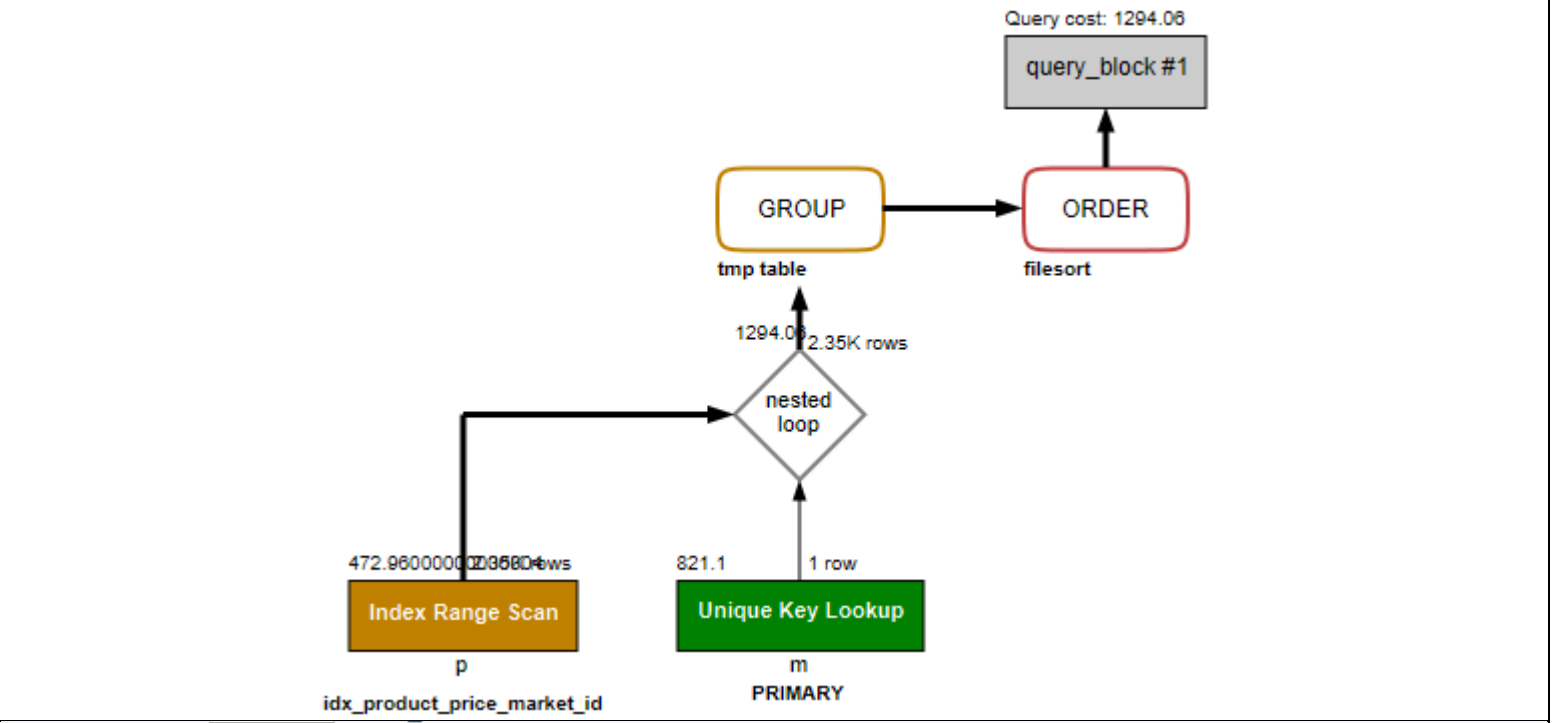
Комментарий к коду	1) Найти среднюю стоимость оборудования для фермеров с опытом от 8 до 12 лет, сгруппировать по стажу фермера
Код добавления индексов	CREATE INDEX work_experience_index ON farmer(work_experience)
Причина добавления	Индекс был добавлен на колонку work_experience, так как по ней идёт поиск
Код программы	SELECT fm.work_experience, AVG(e.cost) AS avg_equipment_cost FROM farmer fm JOIN equipment e ON fm.equipment_id = e.id WHERE fm.work_experience BETWEEN 8 AND 12 GROUP BY fm.work_experience ORDER BY fm.work_experience;
Затраченное время	**Timing (as measured at client side):** Execution time: 0:00:0.01500000 **Timing (as measured by the server):** Execution time: 0:00:0.00445320 Table lock wait time: 0:00:0.00000500

Визуальные планы выполнения запросов	
Explain	

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	fm	NULL	range	equipment_id,work_experience_index	work_experience_index	5	NULL	626	100.00	Using index condition; Using where
	1	SIMPLE	e	NULL	eq_ref	PRIMARY	PRIMARY	4	farm_db.fm.equipment_id	1	100.00	NULL

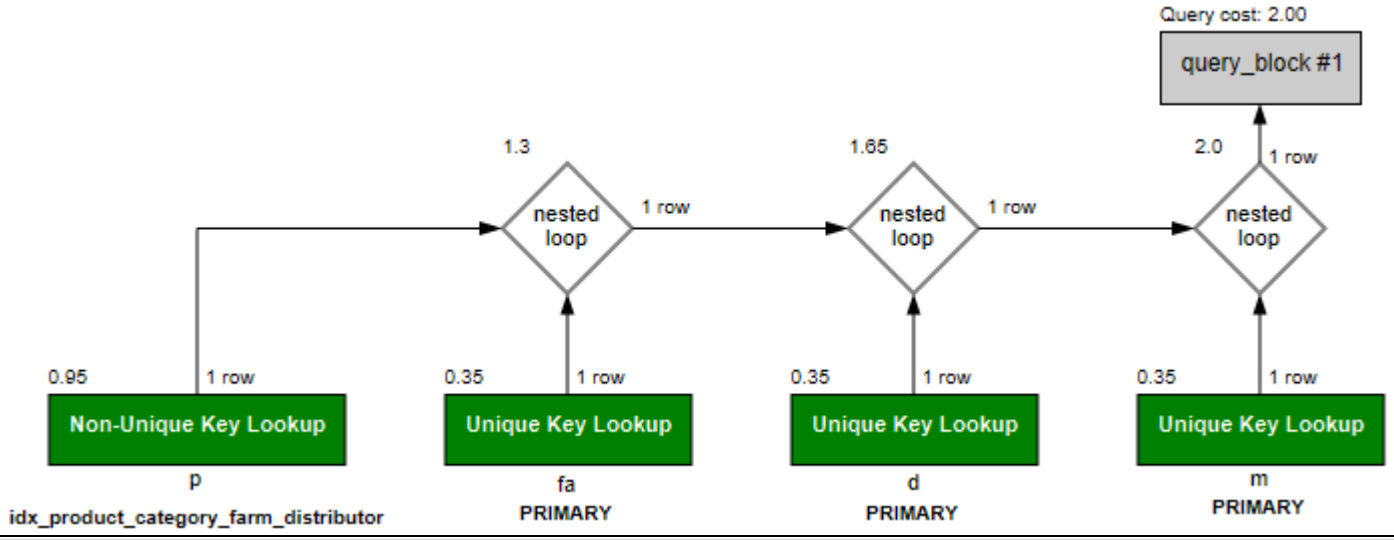
Вывод: затрачиваемое время уменьшилось на 0.0030007

Комментарий к коду	2) Найти для каждого сегмента рынка среднюю цену продуктов, продаваемых на нем, учитывая только продукты дороже 50, и показать топ 3 сегмента по средней цене
Код добавления индексов	CREATE INDEX idx_product_market_id_price ON product (market_id, price); CREATE INDEX idx_product_price_market_id ON product (price, market_id);
Причина добавления	Индексы созданы для столбцов, которые используются в условиях ON операторов JOIN. Это позволяет СУБД быстро находить связанные строки в соединяемых таблицах, избегая полного сканирования.

Код программы	SELECT m.market_segment, AVG(p.price) AS avg_prod_price FROM market m JOIN product p ON m.id = p.market_id WHERE p.price > 50 GROUP BY m.market_segment ORDER BY avg_prod_price DESC LIMIT 3;
Затраченное время	**Timing (as measured at client side):** Execution time: 0:00:0.01600000 **Timing (as measured by the server):** Execution time: 0:00:0.00766360 Table lock wait time: 0:00:0.00000400
Визуальные планы выполнения запросов	
Explain	id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra 1 | SIMPLE | p | NULL | range | idx_product_market_id_price,idx_product_price... | idx_product_price_market_id | 5 | NULL | 2346 | 100.00 | Using where; Using index; Using temporary; Usi... 1 | SIMPLE | m | NULL | eq_ref | PRIMARY,idx_market_id_segment | PRIMARY | 4 | farm_db.p.market_id | 1 | 100.00 | NULL

Вывод: затрачиваемое время уменьшилось на 0.0042652

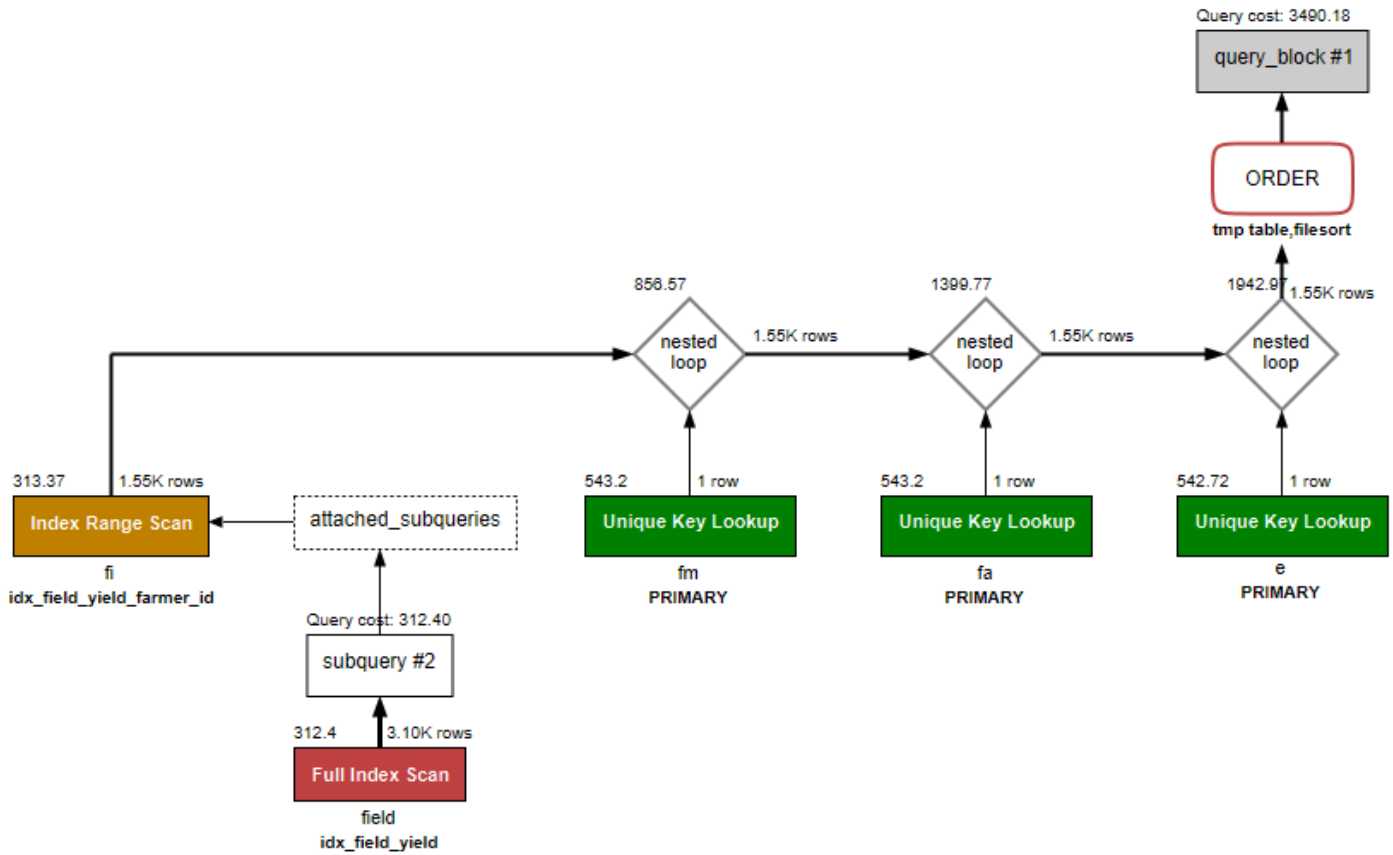
Комментарий к коду	3)Продукт -> Ферма -> Рынок -> Дистрибьютор
Код добавления индексов	CREATE INDEX idx_product_category_farm_distributor ON farm_db.product (category, farm_id, distributor_id); CREATE INDEX idx_farm_market_id ON farm_db.farm (market_id);
Причина добавления	Индексы созданы для столбцов (farm_id в product, market_id в farm, distributor_id в product), которые используются в условиях ON операторов JOIN. Это позволяет СУБД быстро находить связанные строки в соединяемых таблицах, избегая полного сканирования.
Код программы	SELECT p.category, fa.name AS Farm, m.market_segment AS Market, d.company_name AS Distributor FROM farm_db.product p JOIN farm_db.farm fa ON p.farm_id = fa.id JOIN farm_db.market m ON fa.market_id = m.id JOIN farm_db.distributor d ON p.distributor_id = d.id WHERE p.category = 'Зерновые';
Затраченное время	**Timing (as measured at client side):** Execution time: 0:00:0.00000000 **Timing (as measured by the server):** Execution time: 0:00:0.00066020 Table lock wait time: 0:00:0.00000500

Визуальные планы выполнения запросов	
Explain	

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	p	NULL	ALL	idx_product_farm_id,idx_product_distributor_id	NULL	NULL	NULL	2358	100.00	Using where
1	SIMPLE	fa	NULL	eq_ref	PRIMARY,idx_farm_market_id	PRIMARY	4	farm_db.p.farm_id	1	100.00	Using where
1	SIMPLE	m	NULL	eq_ref	PRIMARY	PRIMARY	4	farm_db.fa.market_id	1	100.00	NULL
1	SIMPLE	d	NULL	eq_ref	PRIMARY	PRIMARY	4	farm_db.p.distributor_id	1	100.00	NULL

Вывод: затрачиваемое время уменьшилось на 0.0018002

Комментарий к коду	4) Фермеры с урожайностью выше средней, использующие оборудование НЕ в "Отличном" состоянии.
Код добавления индексов	-- field CREATE INDEX idx_field_yield_for_avg ON farm_db.field (yield); -- Для подзапроса CREATE INDEX idx_field_yield_farmer_id ON farm_db.field (yield, farmer_id); -- Для основного WHERE и JOIN -- equipment CREATE INDEX idx_equipment_conditions_id ON farm_db.equipment (conditions, id); -- Для WHERE и JOIN -- farmer CREATE INDEX idx_farmer_equipment_id ON farm_db.farmer (equipment_id); -- Для JOIN CREATE INDEX idx_farmer_farm_id ON farm_db.farmer (farm_id); -- Для JOIN CREATE INDEX idx_farmer_full_name ON farm_db.farmer (full_name); -- Для ORDER BY
Причина добавления	Индексы созданы для всех столбцов, участвующих в условиях JOIN (где они не являются первичными ключами). Для таблицы field создан композитный индекс (farmer_id, yield) для ускорения JOIN и фильтрации по yield, а также отдельный индекс на yield для подзапроса и условия WHERE. Для equipment индекс на conditions ускорит фильтрацию. Индекс на farmer.full_name предназначен для ускорения ORDER BY.
Код программы	SELECT fm.full_name, fa.name AS farm_name, fi.yield, e.conditions AS equipment_condition FROM farm_db.farmer fm JOIN farm_db.field fi ON fm.id = fi.farmer_id JOIN farm_db.equipment e ON fm.equipment_id = e.id JOIN farm_db.farm fa ON fm.farm_id = fa.id WHERE fi.yield > (SELECT AVG(yield) FROM farm_db.field) AND e.conditions <> 'Отличное' ORDER BY fm.full_name;
Затраченное время	**Timing (as measured at client side):** Execution time: 0:00:0.01500000 **Timing (as measured by the server):** Execution time: 0:00:0.01837210 Table lock wait time: 0:00:0.00001900

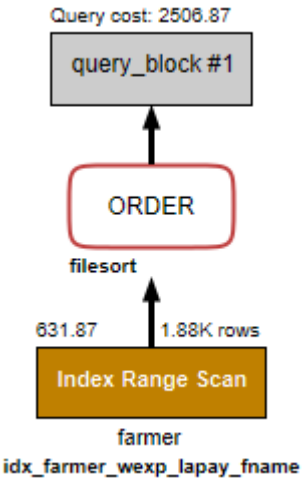
Визуальные планы выполнения запросов	
Explain	

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	fi	NULL	range	idx_field_farmer_id_yield,idx_field_yield,idx_fiel...	idx_field_yield_farmer_id	5	NULL	1552	100.00	Using where; Using index; Using temporary; Usi...
1	PRIMARY	fm	NULL	eq_ref	PRIMARY,equipment_id,idx_farmer_equipment...	PRIMARY	4	farm_db.fi.farmer_id	1	100.00	Using where
1	PRIMARY	fa	NULL	eq_ref	PRIMARY	PRIMARY	4	farm_db.fm.farm_id	1	100.00	NULL
1	PRIMARY	e	NULL	eq_ref	PRIMARY,idx_equipment_conditions_id	PRIMARY	4	farm_db.fm.equipment_id	1	99.69	Using where
2	SUBQUERY	field	NULL	index	NULL	idx_field_yield	5	NULL	3099	100.00	Using index

Вывод: затрачиваемое время уменьшилось на 0.0035133

Комментарий к коду	5) Рейтинг фермеров по соотношению зарплаты к опыту
Код добавления индексов	CREATE INDEX idx_farmer_wexp_lapay_fname ON farm_db.farmer (work_experience, labor_payment, full_name);
Причина добавления	Индекс idx_farmer_exp_payment_name создан как покрывающий. Он включает work_experience для эффективной фильтрации WHERE, а также labor_payment, необходимые для SELECT и

<table>
<tr><td></td><td>вычисления соотношения. Это позволяет MySQL получить все нужные данные из индекса после фильтрации, минимизируя обращения к таблице. Хотя сортировка по вычисляемому полю все равно потребует filesort, она будет производиться на данных, уже извлеченных из индекса</td></tr>
<tr><td>Код программы</td><td>SELECT
   full_name, work_experience, labor_payment,
   (labor_payment / work_experience) AS salary_experience_ratio
FROM farm_db.farmer FORCE INDEX FOR ORDER BY (idx_farmer_wexp_lapay_fname)
WHERE work_experience > 0
ORDER BY salary_experience_ratio DESC;</td></tr>
<tr><td>Затраченное время</td><td>Timing (as measured at client side):
Execution time: 0:00:0.00000000

Timing (as measured by the server):
Execution time: 0:00:0.01006900
Table lock wait time: 0:00:0.00000200</td></tr>
<tr><td>Визуальные планы выполнения запросов</td><td> 
</td></tr>
<tr><td>Explain</td><td>
<table>
<tr><th></th><th>id</th><th>select_type</th><th>table</th><th>partitions</th><th>type</th><th>possible_keys</th><th>key</th><th>key_len</th><th>ref</th><th>rows</th><th>filtered</th><th>Extra</th></tr>
<tr><td>▶</td><td>1</td><td>SIMPLE</td><td>farmer</td><td>NULL</td><td>range</td><td>work_experience_index,idx_farmer_exp_paym…</td><td>idx_farmer_wexp_lapay_fname</td><td>5</td><td>NULL</td><td>1875</td><td>100.00</td><td>Using where; Using index; Using filesort</td></tr>
</table>
</td></tr>
</table>

Вывод: затрачиваемое время уменьшилось на 0.0013654

ХРАНИМЫЕ ПРОЦЕДУРЫ И ФУНКЦИИ

1.ПРОЦЕДУРЫ

Процедура 1: получить информацию о продуктах определенной категории

Код программы:

```
DELIMITER //
CREATE PROCEDURE GetProductsByCategory(
   IN category_name VARCHAR(100)
)
BEGIN
   SELECT
      p.id,
      p.price,
      p.volume,
      fa.name AS farm_name,
      s.company_name AS supplier_name
   FROM
      product p
   LEFT JOIN farm fa ON p.farm_id = fa.id
   LEFT JOIN supplier s ON p.supplier_id = s.id
   WHERE
      p.category = category_name;
END //
DELIMITER ;
-- Вызов:
CALL GetProductsByCategory('Молочные продукты');
CALL GetProductsByCategory('Зерновые');
```

Результат выполнения:

id	price	volume	farm_name	supplier_name
13	378.03	2094.8	Ферма "Золотой Колос"	ООО «Панова, Сысоева и Моисеев»
21	1517.69	1005.3	Ферма "Золотой Колос"	ЗАО «Ершов, Якушева и Носов»
29	2525.14	1413.7	Ферма "Золотой Колос"	АО «Комиссаров Селиверстова»
30	3719.53	4250.7	Ферма "Золотой Колос"	ООО «Горбунова-Рогова»
36	2912.19	910	Ферма "Золотой Колос"	ГК ЕКС
37	2345.19	9805.6	Ферма "Золотой Колос"	ИП «Артемьева»
48	3619.13	1497.9	Ферма "Золотой Колос"	РАО «Карпов Кулагин»
49	1644.28	8890.7	Ферма "Золотой Колос"	X5 Retail Group
52	3596.57	2543	Ферма "Золотой Колос"	МегаФон
64	4363.59	626.6	Ферма "Золотой Колос"	РАО «Петухов, Давыдова и Константинов»
70	4058.35	9495.9	Ферма "Золотой Колос"	НПО «Самойлова Лукина»
74	4605.09	8426.2	Ферма "Золотой Колос"	ММС Рус (Mitsubishi Motors)
83	1213.25	6155	Ферма "Золотой Колос"	ИП «Евдокимова Никитина»
87	3147.65	4663.8	Ферма "Золотой Колос"	Костин Лимитед
92	4048.97	6301.7	Ферма "Золотой Колос"	НПО «Селиверстов Тарасова»
106	3524.88	6666.7	Ферма "Золотой Колос"	Российские железные дороги
108	4023.37	3141.4	Ферма "Золотой Колос"	НПО «Никифоров-Щербаков»
116	2272.82	9570.6	Ферма "Золотой Колос"	Голубева Инк

Описание: Процедура принимает на вход название категории продукта и возвращает список продуктов этой категории с указанием их цены, объема, названия фермы-производителя и компании-поставщика.

Параметры: IN category_name VARCHAR(100)

Локальные переменные: Нет.

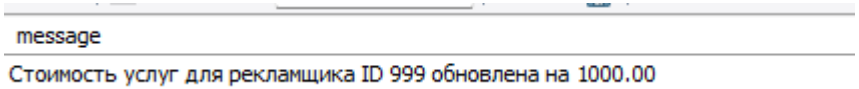
Операторы: Нет сложных (только SELECT).

Процедура 2: Обновить стоимость услуг рекламщика с проверкой
Код программы:

```
DELIMITER //
CREATE PROCEDURE UpdateAdvertiserServiceCost(
    IN advertiser_id_param INT,
    IN new_cost DECIMAL(15, 2)
)
BEGIN
    DECLARE current_effectiveness INT;
    DECLARE message VARCHAR(255);
    -- Проверяем, существует ли рекламщик
```

```
    IF NOT EXISTS (SELECT 1 FROM advertiser WHERE id = advertiser_id_param) THEN
        SET message = CONCAT('Рекламщик с ID ', advertiser_id_param, ' не найден.');
        SELECT message; -- Выводим сообщение об ошибке
    ELSE
        -- Получаем текущую эффективность
        SELECT effectiveness INTO current_effectiveness FROM advertiser WHERE id = advertiser_id_param;
        IF new_cost < 0 THEN
            SET message = 'Стоимость услуг не может быть отрицательной.';
            SELECT message;
        ELSEIF new_cost > 50000 AND current_effectiveness < 5 THEN
            SET message = 'Нельзя установить высокую стоимость для малоэффективного рекламщика.';
            SELECT message;
        ELSE
            UPDATE advertiser SET service_cost = new_cost WHERE id = advertiser_id_param;
            SET message = CONCAT('Стоимость услуг для рекламщика ID ', advertiser_id_param, ' обновлена на ', new_cost);
            SELECT message;
        END IF;
    END IF;
END //
DELIMITER ;
-- Вызов:
CALL UpdateAdvertiserServiceCost(1, 60000.00); -- Пример для рекламщика с ID 1
CALL UpdateAdvertiserServiceCost(1, 25000.00);
CALL UpdateAdvertiserServiceCost(999, 1000.00); -- Несуществующий ID
```

Результат выполнения:



message
Стоимость услуг для рекламщика ID 999 обновлена на 1000.00

Описание: Процедура обновляет стоимость услуг для указанного рекламщика. Перед обновлением проверяет, существует ли рекламщик, не является ли новая стоимость отрицательной, и не слишком ли высока стоимость для малоэффективного рекламщика.

Параметры: IN advertiser_id_param INT, IN new_cost DECIMAL(15, 2)

Локальные переменные: current_effectiveness INT, message VARCHAR(255)

Операторы: IF/ELSEIF/ELSE, CONCAT

Процедура 3: Вывести список продуктов определенной категории с ценой выше указанной.
Код программы:

```
DELIMITER //

CREATE PROCEDURE GetExpensiveProductsByCategory(
    IN category_name_param VARCHAR(100),
    IN min_price_param DECIMAL(15, 2)
)
BEGIN
    SELECT
        p.id AS product_id,
        p.category, -- Добавим категорию для наглядности
        p.price,
        p.volume,
        f.name AS farm_name -- Добавим имя фермы для контекста
    FROM
        farm_db.product p
    LEFT JOIN -- Используем LEFT JOIN, если хотим видеть продукты, даже если ферма не указана
        farm_db.farm f ON p.farm_id = f.id
    WHERE
        p.category = category_name_param
        AND p.price > min_price_param
    ORDER BY
        p.price DESC; -- Отсортируем по убыванию цены
END //

DELIMITER ;
-- Найти продукты категории "Молочные продукты" дороже 150.00
CALL GetExpensiveProductsByCategory('Молочные продукты', 150.00);
```

Результат выполнения:

product_id	category	price	volume	farm_name
9609	Молочные продукты	4999.95	6730.6	Ферма "Золотой Колос"
5275	Молочные продукты	4993.64	1054.9	Ферма "Золотой Колос"
3949	Молочные продукты	4991.06	6715.4	Ферма "Золотой Колос"
7445	Молочные продукты	4983.78	4690.6	Ферма "Золотой Колос"
947	Молочные продукты	4982.69	5203.8	Ферма "Золотой Колос"
9159	Молочные продукты	4981.5	3559.7	Ферма "Золотой Колос"
7399	Молочные продукты	4980.27	4701.6	Ферма "Золотой Колос"
7700	Молочные продукты	4979.1	9358.7	Ферма "Золотой Колос"
480	Молочные продукты	4975.53	8501.1	Ферма "Золотой Колос"
8981	Молочные продукты	4972.36	1232.8	Ферма "Золотой Колос"
9100	Молочные продукты	4967.69	3559.2	Ферма "Золотой Колос"

Описание: Процедура выводит список продуктов (ID, категория, цена, объем, название фермы) указанной категории, цена которых превышает заданное минимальное значение. Результаты сортируются по убыванию цены.

Параметры: IN category_name_param VARCHAR(100), IN min_price_param DECIMAL(15, 2)

Локальные переменные: Нет.

Операторы: SELECT, FROM, LEFT JOIN ... ON, WHERE ... AND, ORDER BY ... DESC.

2.Функции

Функция 1: Получить общую стоимость активов фермы

Код программы*:*

```
DELIMITER //
CREATE FUNCTION GetFarmTotalAssetCost(
    farm_id_param INT
)
RETURNS DECIMAL(18, 2)
DETERMINISTIC -- Если функция всегда возвращает тот же результат для тех же входных данных
READS SQL DATA -- Указывает, что функция читает данные
BEGIN
    DECLARE total_asset_cost DECIMAL(18, 2);
    SELECT asset_cost
    INTO total_asset_cost
    FROM farm_db.farm
    WHERE id = farm_id_param;
```

```
    RETURN total_asset_cost; -- Вернет NULL, если ферма не найдена
END //
DELIMITER ;
SELECT GetFarmTotalAssetCost(1) AS FarmAssets;
```

Результат выполнения:

FarmAssets
5000000.00

Описание: Функция принимает ID фермы и возвращает ее общую стоимость активов (asset_cost). Если ферма не найдена, возвращает NULL.

Параметры: farm_id_param INT

Локальные переменные: total_asset_cost DECIMAL(18, 2)

Операторы: DECLARE, SELECT ... INTO, IF ISNULL

Функция 2: Рассчитать предполагаемую выручку с поля
Код программы:

```
DELIMITER //

CREATE FUNCTION CalculateFieldRevenue(
    field_id_param INT,
    product_id_param INT
)
RETURNS DECIMAL(20, 2)
DETERMINISTIC
READS SQL DATA
BEGIN
    DECLARE field_yield_val DECIMAL(8, 2);
    DECLARE product_price_val DECIMAL(15, 2);
```

```
    DECLARE estimated_revenue DECIMAL(20, 2) DEFAULT 0.00; -- По умолчанию 0

    SELECT `yield`
    INTO field_yield_val
    FROM farm_db.field
    WHERE id = field_id_param;

    SELECT price
    INTO product_price_val
    FROM farm_db.product
    WHERE id = product_id_param;

    IF field_yield_val IS NOT NULL AND product_price_val IS NOT NULL THEN
        SET estimated_revenue = field_yield_val * product_price_val;
    END IF;

    RETURN estimated_revenue;
END //
DELIMITER ;
-- Показать все поля и их потенциальную выручку от конкретного продукта
SELECT
    fi.id AS field_id,
    fi.area,
    fi.yield,
    (SELECT p.category FROM farm_db.product p WHERE p.id = 5) AS product_category,
    CalculateFieldRevenue(fi.id, 5) AS potential_revenue_from_wheat
FROM
    farm_db.field fi;
```

Результат выполнения:

field_id	area	yield	product_category	potential_revenue_from_wheat
1	289.61	4.18	Мясо	547.41
2	11.23	1.06	Мясо	138.82
3	57.61	6.7	Мясо	877.43
4	427.76	11.41	Мясо	1494.25
5	51.87	11.87	Мясо	1554.50
6	453.22	14.32	Мясо	1875.35
7	334.35	9.64	Мясо	1262.45
8	336.14	7.82	Мясо	1024.11
9	76.31	6.95	Мясо	910.17
10	311.75	7.35	Мясо	962.56
11	197.03	6.5	Мясо	851.24
12	296.75	10.8	Мясо	1414.37
13	84.04	12.15	Мясо	1591.16
14	214.06	10.69	Мясо	1399.96

2nd_f('Rhys Glanton')
4175030

Описание: Функция принимает ID поля и ID продукта (предполагается, что с этого поля собирают этот продукт). Она рассчитывает предполагаемую выручку как произведение урожайности поля (yield) на цену продукта (price). Если поле или продукт не найдены, или цена/урожайность не заданы, возвращает 0 или NULL.

Параметры: field_id_param INT, product_id_param INT

Локальные переменные: field_yield_val DECIMAL(8, 2), product_price_val DECIMAL(15, 2), estimated_revenue DECIMAL(20, 2)

Операторы: DECLARE, SELECT ... INTO, IF ... THEN ... ELSE, ISNULL

Функция 3: Определить категорию эффективности рекламщика
Код программы:

```
DELIMITER //
CREATE FUNCTION GetAdvertiserEffectivenessCategory(
    advertiser_id_param INT
)
RETURNS VARCHAR(50)
DETERMINISTIC
READS SQL DATA
BEGIN
    DECLARE effectiveness_value INT;
    DECLARE effectiveness_category VARCHAR(50) DEFAULT 'Не определена';
    SELECT effectiveness
```

```
    INTO effectiveness_value
    FROM farm_db.advertiser
    WHERE id = advertiser_id_param;
    IF effectiveness_value IS NOT NULL THEN
        CASE
            WHEN effectiveness_value <= 3 THEN
                SET effectiveness_category = 'Низкая';
            WHEN effectiveness_value <= 7 THEN
                SET effectiveness_category = 'Средняя';
            ELSE
                SET effectiveness_category = 'Высокая';
        END CASE;
    END IF;
    RETURN effectiveness_category;
END //
DELIMITER ;
-- Вывести всех рекламщиков с их категорией эффективности
SELECT
    company_name,
    effectiveness,
    GetAdvertiserEffectivenessCategory(id) AS effectiveness_label
FROM
    farm_db.advertiser;
```

Результат выполнения:

company_name	effectiveness	effectiveness_label
ЗАО «Семенова»	6	Средняя
ООО «Поляков-Фокина»	6	Средняя
НК Дулисьма	2	Низкая
РАО «Трофимов-Кононов»	7	Средняя
ИП «Гущин-Белякова»	10	Высокая
АО «Кабанова»	8	Высокая
Сегежа	10	Высокая
НПО «Воронов»	10	Высокая
ООО «Маркова, Рожков и Русаков»	2	Низкая
Fonbet	2	Низкая
НПО «Комаров-Чернов»	6	Средняя
ОАО «Ильин»	5	Средняя
ИП «Филиппова, Русаков и Соболев»	9	Высокая
ОАО «Герасимова»	5	Средняя

Описание: Функция принимает ID рекламщика и на основе его числового показателя эффективности (effectiveness) возвращает текстовую категорию эффективности ("Низкая", "Средняя", "Высокая"). Если рекламщик не найден или эффективность не указана, возвращает "Не определена".

Параметры: advertiser_id_param INT

Локальные переменные: effectiveness_value INT, effectiveness_category VARCHAR(50)

Операторы: DECLARE, SELECT ... INTO, CASE ... WHEN ... THEN ... ELSE ... END

Представления

Представление 1: ActiveFarmersOnProductiveFields (Активные фермеры на продуктивных полях)

Код программы:

```
CREATE VIEW ActiveFarmersOnProductiveFields AS
SELECT
    fm.id AS farmer_id,
    fm.full_name AS farmer_name,
    fm.work_experience AS farmer_experience,
    f.name AS farm_name,
    fi.id AS field_id,
    fi.area AS field_area,
    fi.yield AS field_yield
FROM
    farm_db.farmer fm
JOIN
    farm_db.field fi ON fm.id = fi.farmer_id
JOIN
    farm_db.farm f ON fm.farm_id = f.id
WHERE
    fi.yield > 5.0  -- Условие продуктивности поля (можно настроить)
    AND fm.work_experience > 2; -- Условие опыта фермера (можно настроить)
SELECT farmer_name, farm_name, field_yield
FROM ActiveFarmersOnProductiveFields
WHERE farm_name = 'Ферма "Золотой Колос"';
```

Результат выполнения:

farmer_name	farm_name	field_yield
Белов Сила Дмитриевич	Ферма "Золотой Колос"	6.7
Белякова Варвара Анатольевна	Ферма "Золотой Колос"	11.41
Полина Олеговна Павлова	Ферма "Золотой Колос"	11.87
Григорьев Лукьян Данилович	Ферма "Золотой Колос"	14.32
Каллистрат Евстигнеевич Макаров	Ферма "Золотой Колос"	9.64
Лазарева Александра Тимуровна	Ферма "Золотой Колос"	7.82
Платон Харитонович Фролов	Ферма "Золотой Колос"	6.95
Казакова Ольга Леонидовна	Ферма "Золотой Колос"	7.35
Чернова Агафья Богдановна	Ферма "Золотой Колос"	6.5
Кириллов Спартак Изотович	Ферма "Золотой Колос"	10.8
Никитин Панкратий Тимурович	Ферма "Золотой Колос"	12.15
Терентьева Александра Тимофеевна	Ферма "Золотой Колос"	10.69
Бобылева Лукия Валентиновна	Ферма "Золотой Колос"	10.17

Описание: Это представление объединяет информацию о фермерах и полях, на которых они работают. Отображаются только те поля, урожайность (yield) которых выше определенного порога (например, 5 тонн/га), и фермеры с опытом работы более 2 лет. Представление показывает ФИО фермера, его опыт, название фермы, ID поля, площадь поля и его урожайность.

Представление 2: ProductSupplierDetails (Детали продуктов и их поставщиков)
Код программы:

```
CREATE VIEW ProductSupplierDetails AS
SELECT
    p.id AS product_id,
    p.category AS product_category,
    p.price AS product_price,
    p.volume AS product_volume,
    s.company_name AS supplier_company,
    s.contact_person AS supplier_contact_person,
    s.phone AS supplier_phone,
    f.name AS farm_name -- Добавим ферму, где произведен продукт
FROM
    farm_db.product p
LEFT JOIN -- Используем LEFT JOIN, если хотим видеть продукты, даже если поставщик не указан
    farm_db.supplier s ON p.supplier_id = s.id
LEFT JOIN -- Аналогично для фермы
    farm_db.farm f ON p.farm_id = f.id;
SELECT * FROM ProductSupplierDetails WHERE product_price > 200 ORDER BY product_category;
```

Результат выполнения:

product_id	product_category	product_price	product_volume	supplier_company	supplier_contact_person	supplier_phone	farm_name
12134	Зерновые	4193.78	3274.8	ООО «Гуляев-Кулаков»	Данила Григорьевич Дмитриев	+75717608683	Ферма "Золотой Колос"
12135	Зерновые	1146.04	2645.3	ИП «Наумова»	Анжелика Васильевна Фролова	+76130981880	Ферма "Золотой Колос"
12265	Зерновые	4291.49	7998.8	Мария-Ра	Юдина Надежда Петровна	8 (457) 608-7377	Ферма "Золотой Колос"
12363	Зерновые	342.34	5793.8	Фармимэкс	Ираида Кирилловна Котова	+7 827 544 9657	Ферма "Золотой Колос"
11357	Зерновые	2845.62	7709.5	Почта России	Лапин Эрнест Борисович	8 (762) 046-05-36	Ферма "Золотой Колос"
12361	Зерновые	1842.42	2743.6	ЗАО «Тарасова-Емельянов»	Сазонова Клавдия Валериевна	89159805903	Ферма "Золотой Колос"
11208	Зерновые	3002.67	4510	ООО «Тарасова Кудряшов»	Аггей Гертрудович Поляков	84797704639	Ферма "Золотой Колос"
12524	Зерновые	3446.71	695.9	ООО «Яковлева Калинина»	Игнатьева Лукия Ниловна	8 240 854 68 46	Ферма "Золотой Колос"
12047	Зерновые	3768.87	83.1	Кабанов Инкорпорэйтед	Королева Клавдия Аскольдовна	+7 044 236 8784	Ферма "Золотой Колос"
11941	Зерновые	2876	9117.8	ПК Балтика (Carlsbergfondet)	Георгий Изотович Сысоев	+7 (909) 731-51-93	Ферма "Золотой Колос"
12095	Зерновые	2177.23	293.6	НПО «Григорьев Михеев»	Никифорова Клавдия Геннадье…	80841303386	Ферма "Золотой Колос"
12301	Зерновые	965.36	8393.9	ИП «Евсеева»	Прохор Артурович Тихонов	8 633 986 5602	Ферма "Золотой Колос"
11757	Зерновые	3653.44	2940	ОАО «Журавлева, Шашков …	Михайлова Анна Юрьевна	81909083575	Ферма "Золотой Колос"
11363	Зерновые	4867.59	2621.2	НПО «Бобров Фомичев»	Якушев Творимир Игнатьевич	+7 (447) 239-93-74	Ферма "Золотой Колос"
11500	Зерновые	4231.18	6061.2	ИП «Кабанова-Анисимова»	Николаева Клавдия Евгеньевна	+76608539874	Ферма "Золотой Колос"

Описание: Это представление предоставляет объединенную информацию о продуктах и их поставщиках. Для каждого продукта показывается его категория, цена, объем, а также название компании-поставщика, контактное лицо поставщика и его телефон.

Представление 3: FarmEquipmentSummary (Сводка по оборудованию на фермах)
Код программы:

```
CREATE VIEW FarmEquipmentSummary AS
SELECT
        f.id AS farm_id,
        f.name AS farm_name,
        COUNT(e.id) AS total_equipment_units,
        SUM(e.cost) AS total_equipment_cost,
        SUM(CASE
                        WHEN e.conditions IN ('Требует обслуживания', 'Требует ремонта', 'Неисправно') THEN 1
                        ELSE 0
                END) AS units_needing_attention
FROM
        farm_db.farm f
LEFT JOIN -- Используем LEFT JOIN, чтобы включить фермы даже без оборудования
```

```
        farm_db.equipment e ON f.id = e.farm_id
GROUP BY
        f.id, f.name;
        SELECT * FROM FarmEquipmentSummary ORDER BY total_equipment_cost DESC;
```

Результат выполнения:

farm_id	farm_name	total_equipment_units	total_equipment_cost	units_needing_attention
12469	Ферма "Золотой Колос"	5	96847534	3
10539	Ферма "Золотой Колос"	4	92592570	2
18330	Ферма "Золотой Колос"	6	91248333.875	4
9226	Ферма "Золотой Колос"	6	89876225	5
15923	Ферма "Золотой Колос"	5	89458335.5	2
14750	Ферма "Золотой Колос"	4	89055482	2
4939	Ферма "Золотой Колос"	5	87708824	3
968	Ферма "Золотой Колос"	4	86973974	2
5941	Ферма "Золотой Колос"	4	85977078	3
13906	Ферма "Золотой Колос"	5	85797854	3
8287	Ферма "Золотой Колос"	4	84920776	3
18915	Ферма "Золотой Колос"	5	84758087	2

Описание: Это представление показывает сводную информацию по оборудованию для каждой фермы: название фермы, количество единиц оборудования, общую стоимость всего оборудования на ферме и количество единиц оборудования, требующих обслуживания или ремонта.

Триггеры

Триггер: LogFarmAssetChange (Логирование изменения стоимости активов фермы)

Описание: Этот триггер будет срабатывать после обновления (AFTER UPDATE) стоимости активов (asset_cost) в таблице farm. Если стоимость активов фермы изменилась, триггер будет записывать старое значение, новое значение и время изменения в отдельную таблицу логов. Это полезно для аудита и отслеживания истории изменений важных финансовых показателей.

Шаг 1: Создание таблицы для логов

Сначала нам нужна таблица, куда триггер будет записывать информацию.

```
CREATE TABLE farm_asset_log (
   log_id INT AUTO_INCREMENT PRIMARY KEY,
```

```
    farm_id INT,
    old_asset_cost DECIMAL(18, 2),
    new_asset_cost DECIMAL(18, 2),
    change_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    changed_by VARCHAR(100) DEFAULT NULL, -- Кто изменил (если есть система пользователей)
    FOREIGN KEY (farm_id) REFERENCES farm(id) ON DELETE SET NULL -- Если ферму удалят, лог останется
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

Шаг 2: Создание триггера

```
DELIMITER //
CREATE TRIGGER LogFarmAssetChange
AFTER UPDATE ON farm_db.farm -- Срабатывает ПОСЛЕ обновления таблицы farm
FOR EACH ROW -- Для каждой обновленной строки
BEGIN
    -- Проверяем, изменилась ли именно стоимость активов
    IF OLD.asset_cost <> NEW.asset_cost THEN
        INSERT INTO farm_db.farm_asset_log (
            farm_id,
            old_asset_cost,
            new_asset_cost,
            changed_by -- Добавим пользователя, который внес изменение
        )
        VALUES (
            OLD.id,             -- ID фермы, которая была обновлена
            OLD.asset_cost,     -- Старое значение asset_cost
            NEW.asset_cost,     -- Новое значение asset_cost
            USER()              -- Функция MySQL, возвращающая текущего пользователя
        );
    END IF;
END //
DELIMITER ;
```

Шаг 3: Пример работы триггера

Проверьте текущую стоимость активов фермы

```
SELECT id, name, asset_cost FROM farm_db.farm WHERE id = 1;
```

id	name	asset_cost
1	Ферма "Золотой Колос"	5000000

Обновите стоимость активов для этой фермы:

```
UPDATE farm_db.farm SET asset_cost = asset_cost + 50000 WHERE id = 1;
```

id	address	asset_cost	name	market_id	product_id
1	ул. Центральная, д. 10, с. Ивановка	5050000	Ферма "Золотой Колос"	14133	12437

Проверяем таблицу логов

```
SELECT * FROM farm_db.farm_asset_log;
```

1	1	5000000.00	5050000.00	2025-05-22 01:02:14	root@localhost