18 Advanced SQL

1. Transactions and ACID Properties

ACID Properties

1. **Atomicity**: Ensures that all operations within a transaction are completed successfully. If not, the transaction is aborted
2. **Consistency**: Ensures that the database transitions from one valid state to another
3. **Isolation**: Ensures that transactions are isolated from each other
4. **Durability**: Ensures that once a transaction is committed, it remains so, even in the event of a system failure

Example of Transactions

- Let's say we want to transfer a salary amount from one employee to another
- We'll use transactions to ensure the operation is atomic and consistent

```
START TRANSACTION;
```

```
UPDATE employees
SET salary = salary - 5000
WHERE name = 'Michael Scott';
```

```
UPDATE employees
SET salary = salary + 5000
WHERE name = 'Ryan Howard';
```

```
-- Check if both operations are successful
SELECT * FROM employees WHERE name IN ('Michael Scott', 'Ryan Howard');
```

```
+--------+---------------+------------+----------+
| emp_id | name          | department | salary   |
+--------+---------------+------------+----------+
|      1 | Michael Scott | Management | 65000.00 |
|      5 | Ryan Howard   | Intern     | 45000.00 |
+--------+---------------+------------+----------+
2 rows in set (0.00 sec)
```

```
-- If everything is correct, commit the transaction
COMMIT;
```

- If any error occurs during the transaction, you can roll it back:

```
ROLLBACK;
```

2. Working with Dates and Strings

Dates

```
ALTER TABLE employees ADD COLUMN date_of_joining DATE;
```

```
UPDATE employees
SET date_of_joining = '2022-01-15' WHERE name = 'Michael Scott';
```

```
UPDATE employees
SET date_of_joining = '2022-02-20' WHERE name = 'Ryan Howard';
```

```
SELECT * FROM employees WHERE name IN ('Michael Scott', 'Ryan Howard');
```

```
+--------+---------------+------------+----------+-----------------+
| emp_id | name          | department | salary   | date_of_joining |
+--------+---------------+------------+----------+-----------------+
|      1 | Michael Scott | Management | 65000.00 | 2022-01-15      |
|      5 | Ryan Howard   | Intern     | 45000.00 | 2022-02-20      |
+--------+---------------+------------+----------+-----------------+
2 rows in set (0.00 sec)
```

Example Queries:

```
UPDATE employees
SET date_of_joining = '2022-03-01'
WHERE emp_id BETWEEN 2 AND 4;
```

- **Finding employees who joined in 2022:**

```
SELECT name, date_of_joining
FROM employees
WHERE YEAR(date_of_joining) = 2022;
```

```
+---------------+-----------------+
| name          | date_of_joining |
+---------------+-----------------+
| Michael Scott | 2022-01-15      |
| Ryan Howard   | 2022-02-20      |
+---------------+-----------------+
2 rows in set (0.01 sec)
```

- **Calculating the number of days since each employee joined:**

```
SELECT name, DATEDIFF(CURDATE(), date_of_joining) AS days_since_joining
FROM employees;
```

```
+---------------+--------------------+
| name          | days_since_joining |
+---------------+--------------------+
| Michael Scott |                966 |
| Dwight Schrute |                921 |
| Jim Halpert   |                921 |
| Pam Beesly    |                921 |
| Ryan Howard   |                930 |
+---------------+--------------------+
5 rows in set (0.00 sec)
```

Strings

Example Queries:

- **Concatenating first and last names:**

```
ALTER TABLE employees ADD COLUMN first_name VARCHAR(50);
ALTER TABLE employees ADD COLUMN last_name VARCHAR(50);
```

```
UPDATE employees
SET first_name = 'Michael', last_name = 'Scott'
WHERE name = 'Michael Scott';
```

```
SELECT CONCAT(first_name, ' ', last_name) AS full_name
FROM employees;
```

3. Indexes and Performance Tuning

01 Creating Indexes

- Indexes can improve the performance of queries

- Let's create an index on the `department` column

```
CREATE INDEX idx_department ON employees(department);
```

02 Example of Performance Improvement

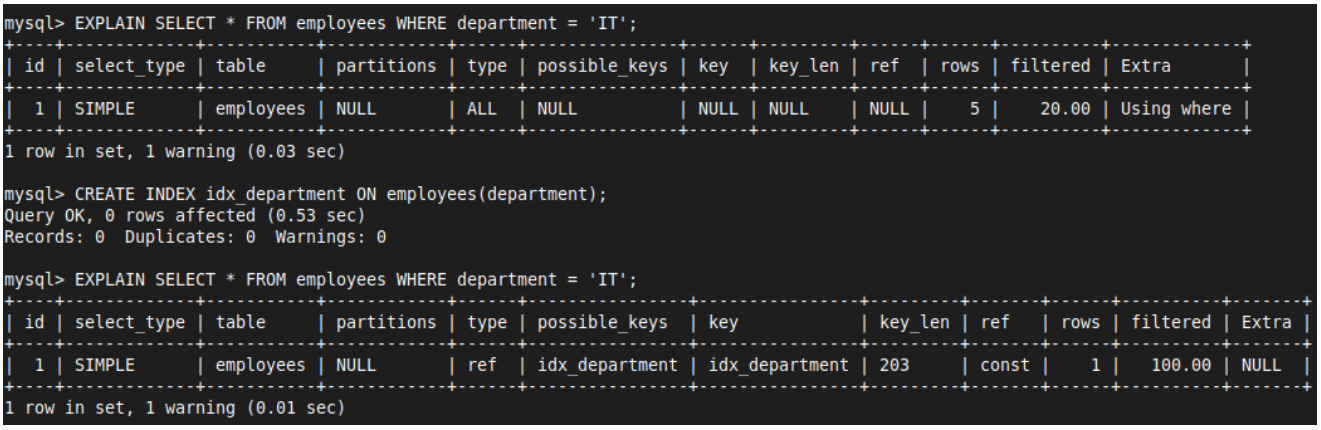
- Without index:

```
EXPLAIN SELECT * FROM employees WHERE department = 'IT';
```

- With index:

```
EXPLAIN SELECT * FROM employees WHERE department = 'IT';
```

- You should see a lower execution cost with the index



```
mysql> EXPLAIN SELECT * FROM employees WHERE department = 'IT';
+----+-------------+-----------+------------+------+---------------+------+---------+------+------+----------+-------------+
| id | select_type | table     | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra       |
+----+-------------+-----------+------------+------+---------------+------+---------+------+------+----------+-------------+
|  1 | SIMPLE      | employees | NULL       | ALL  | NULL          | NULL | NULL    | NULL |    5 |    20.00 | Using where |
+----+-------------+-----------+------------+------+---------------+------+---------+------+------+----------+-------------+
1 row in set, 1 warning (0.03 sec)

mysql> CREATE INDEX idx_department ON employees(department);
Query OK, 0 rows affected (0.53 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN SELECT * FROM employees WHERE department = 'IT';
+----+-------------+-----------+------------+------+---------------+---------------+---------+-------+------+----------+-------+
| id | select_type | table     | partitions | type | possible_keys | key           | key_len | ref   | rows | filtered | Extra |
+----+-------------+-----------+------------+------+---------------+---------------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | employees | NULL       | ref  | idx_department | idx_department | 203     | const |    1 |   100.00 | NULL  |
+----+-------------+-----------+------------+------+---------------+---------------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.01 sec)
```

03 Dropping an Index

- If an index is no longer needed, it can be dropped:

```
DROP INDEX idx_department ON employees;
```

04 Using Indexes for Composite Keys

- You can also create composite indexes on multiple columns for more complex queries:

```
CREATE INDEX idx_department_salary ON employees(department, salary);
```