

Managing the World Database

Objectives:

With this assignment, you will:

- Learn and practice working with different data types, constraints, and indexes.
- Gain hands-on experience with user management, privileges, and views.
- Understand the use of operators and query optimization techniques.
- Work with transactions to ensure data consistency.

Part 1: Data Types, Validation Restrictions & Indexes

Objective: To introduce and practice using different data types, apply validation restrictions, and understand how indexes improve performance.

Instructions:

1. Explore the "world" database:
 - List all tables in the "world" database.

```
mysql> USE world;
Database changed
mysql> SHOW TABLES;
+-----+
| Tables_in_world |
+-----+
| city             |
| country          |
| countrylanguage  |
+-----+
3 rows in set (0.00 sec)
```

- Identify the different data types used in the country, city, and countrylanguage tables.

```
mysql> describe city;
```

Field	Type	Null	Key	Default	Extra
ID	int	NO	PRI	NULL	auto_increment
Name	char(35)	NO			
CountryCode	char(3)	NO	MUL		
District	char(20)	NO			
Population	int	NO		0	

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

```
mysql> describe country;
```

Field	Type	Null	Key	Default	Extra
Code	char(3)	NO	PRI		
Name	char(52)	NO			
Continent	enum('Asia','Europe','North America','Africa','Oceania','Antarctica','South America')	NO		Asia	
Region	char(26)	NO			
SurfaceArea	decimal(10,2)	NO		0.00	
IndepYear	smallint	YES		NULL	
Population	int	NO		0	
LifeExpectancy	decimal(3,1)	YES		NULL	
GNP	decimal(10,2)	YES		NULL	
GNPOld	decimal(10,2)	YES		NULL	
LocalName	char(45)	NO			
GovernmentForm	char(45)	NO			
HeadOfState	char(60)	YES		NULL	
Capital	int	YES		NULL	
Code2	char(2)	NO			

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

```
mysql> describe countrylanguage;
```

Field	Type	Null	Key	Default	Extra
CountryCode	char(3)	NO	PRI		
Language	char(30)	NO	PRI		
IsOfficial	enum('T','F')	NO		F	
Percentage	decimal(4,1)	NO		0.0	

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

2. Modify Data Types:

- Add a new column is_population_large (BOOLEAN) to the city table.

```
mysql> alter table city add column is_population_large boolean;
Query OK, 0 rows affected (0.03 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> describe city;
```

Field	Type	Null	Key	Default	Extra
ID	int	NO	PRI	NULL	auto_increment
Name	char(35)	NO			
CountryCode	char(3)	NO	MUL		
District	char(20)	NO			
Population	int	NO		0	
is_population_large	tinyint(1)	YES		NULL	

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

- Create a new column in the country table, region_code (CHAR(3)) with the DEFAULT 'NA' value.

```
mysql> alter table country add column region_code char(3) default 'NA';
Query OK, 0 rows affected (0.05 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> describe country;
```

Field	Type	Null	Key	Default	Extra
Code	char(3)	NO	PRI		
Name	char(52)	NO			
Continent	enum('Asia','Europe','North America','Africa','Oceania','Antarctica','South America')	NO		Asia	
Region	char(26)	NO			
SurfaceArea	decimal(10,2)	NO		0.00	
IndepYear	smallint	YES		NULL	
Population	int	NO		0	
LifeExpectancy	decimal(3,1)	YES		NULL	
GNP	decimal(10,2)	YES		NULL	
GNPOld	decimal(10,2)	YES		NULL	
LocalName	char(45)	NO			
GovernmentForm	char(45)	NO			
HeadOfState	char(60)	YES		NULL	
Capital	int	YES		NULL	
Code2	char(2)	NO			
region_code	char(3)	YES		NA	

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

3. Add Validation Constraints:

- Add a CHECK constraint on the city table to ensure that the population value is never negative.

```
mysql> alter table city
    -> add constraint check_value check (population >=0 );
Query OK, 4079 rows affected (0.17 sec)
Records: 4079  Duplicates: 0  Warnings: 0
```

- Ensure that the country table's code is unique using a UNIQUE constraint.

```
mysql> alter table country
    -> add constraint unique_code unique (code);
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

4. Indexes:

- Create an index on the city table for the name column.

```
mysql> create index idx on city (name);
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> show index from city;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
city	0	PRIMARY	1	ID	A	3998	NULL	NULL	NULL	BTREE			YES	NULL
city	1	CountryCode	1	CountryCode	A	232	NULL	NULL	NULL	BTREE			YES	NULL
city	1	idx	1	Name	A	3998	NULL	NULL	NULL	BTREE			YES	NULL

3 rows in set (0.01 sec)

Part 2: Views, Users, and Privileges

Objective: To demonstrate the creation of views, manage users, and assign privileges.

Instructions:

1. Views:

- Create a view named high_population_cities that shows cities with a population over 1 million.

```
mysql> CREATE VIEW high_population_cities AS
-> select name
-> FROM city
-> WHERE population >= 1000000;
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> select * from high_population_cities;
```

name
Kabul
Alger
Luanda
Buenos Aires
La Matanza
Córdoba
Yerevan
Sydney
Melbourne
Brisbane
Perth
Bakı
Dhaka
Chittagong
São Paulo
Rio de Janeiro
Salvador
Belo Horizonte
Fortaleza
Brasília
Curitiba
Recife
Porto Alegre
Manaus
Belém
Guarulhos

- Create another view countries_with_languages that joins country and countrylanguage.

```
mysql> CREATE VIEW countries_with_languages AS
-> SELECT c.Name AS CountryName, cl.Language
-> FROM country c
-> JOIN countrylanguage cl ON c.Code = cl.CountryCode;
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> select * from countries_with_languages;
```

CountryName	Language
Aruba	Dutch
Aruba	English
Aruba	Papiamentu
Aruba	Spanish
Afghanistan	Balochi
Afghanistan	Dari
Afghanistan	Pashto
Afghanistan	Turkmenian
Afghanistan	Uzbek
Angola	Ambo
Angola	Chokwe
Angola	Kongo
Angola	Luchazi
Angola	Luimbe-nganguela
Angola	Luvale
Angola	Mbundu
Angola	Nyaneka-nkhumbi
Angola	Ovimbundu
Anguilla	English
Albania	Albaniana
Albania	Greek
Albania	Macedonian
Andorra	Catalan
Andorra	French
Andorra	Portuguese

2. Users & Privileges:

- Create a new user db_user with read and write privileges.

```
mysql> create user 'db_user' identified by 'root2';
Query OK, 0 rows affected (0.03 sec)

mysql> grant all privileges on *.* to 'db_user'@'%';
Query OK, 0 rows affected (0.01 sec)
```

- Write access (INSERT, UPDATE) on the city table.

```
mysql> grant all privileges on world to 'db_user'@'%';
Query OK, 0 rows affected (0.01 sec)
```

- Revoke the ability of db_user to modify the country table.

```
mysql> grant select on world.city to 'db_user'@'%';
Query OK, 0 rows affected (0.01 sec)

mysql> grant select on world.country to 'db_user'@'%';
Query OK, 0 rows affected (0.01 sec)
```

- Grant the db_user full access to the high_population_cities view.

```
mysql> grant insert, update on world.city to 'db_user'@'%';
Query OK, 0 rows affected (0.01 sec)
```

3. Test Privileges:

- o Log in as db_user and attempt to perform operations according to the granted/revoked privileges.

```
mysql> -- Test SELECT on city table
mysql> select * from world.city limit 5;
```

ID	Name	CountryCode	District	Population	is_population_large
1	Kabul	AFG	Kabul	1780000	NULL
2	Qandahar	AFG	Qandahar	237500	NULL
3	Herat	AFG	Herat	186800	NULL
4	Mazar-e-Sharif	AFG	Balkh	127800	NULL
5	Amsterdam	NLD	Noord-Holland	731200	NULL

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

```
mysql> INSERT INTO world.city (Name, CountryCode, District, Population) VALUES ('Greenwich', 'GBR', 'Greenwich', 250000);
Query OK, 1 row affected (0.01 sec)
```

```
mysql> -- Test SELECT on country table
mysql> select * from world.country limit 5;
```

Code	Name	Continent	Region	SurfaceArea	IndepYear	Population	LifeExpectancy	GDP	GNPOLD	LocalName	GovernmentForm	HeadOfState
ABW	Aruba	North America	Caribbean	193.00	NULL	183000	78.4	828.00	793.00	Aruba	Nonmetropolitan Territory of The Netherlands	Beatrix
AFG	Afghanistan	Asia	Southern and Central Asia	652090.00	1919	22720000	45.0	5976.00	NULL	Afghanistan/Afghanistan	Islamic Emirate	Mohammad Omar
AGO	Angola	Africa	Central Africa	1246700.00	1975	12878000	38.3	6648.00	7984.00	Angola	Republic	José Eduardo dos Santos
AI	Anguilla	North America	Caribbean	96.00	NULL	8000	76.1	63.20	NULL	Anguilla	Dependent Territory of the UK	Elisabeth II
AL	Albania	Europe	Southern Europe	28748.00	1912	3481200	71.6	3285.00	2588.00	Shqipëria	Republic	Rexhep Mejdani

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

Part 3: Operators, Comparators, and Logical Operators

Objective: To practice using operators and logical comparators to filter and manipulate data.

Instructions:

1. Comparison Operators:

- Write a query to find all countries where the population is greater than 50 million but less than 200 million.

```
mysql> select name, population
-> from country
-> where population > 50000000 and population < 200000000;
```

name	population
Bangladesh	129155000
Brazil	170115000
Congo, The Democratic Republic of the	51654000
Germany	82164700
Egypt	68470000
Ethiopia	62565000
France	59225700
United Kingdom	59623400
Iran	67702000
Italy	57680000
Japan	126714000
Mexico	98881000
Nigeria	111506000
Pakistan	156483000
Philippines	75967000
Russian Federation	146934000
Thailand	61399000
Turkey	66591000
Ukraine	50456000
Vietnam	79832000

```
20 rows in set (0.03 sec)
```

2. Logical Operators:

- Use AND, OR, and NOT operators to filter cities by population and region.


```
mysql> SELECT city.Name, city.Population, country.Region
-> FROM city
-> JOIN country ON city.CountryCode = country.Code
-> WHERE (city.Population BETWEEN 500000 AND 5000000)
-> AND (country.Region LIKE '%Europe%' OR country.Region LIKE '%Asia%')
-> AND NOT (country.Region LIKE '%Africa%');
```

Name	Population	Region
Kabul	1780000	Southern and Central Asia
Wien	1608144	Western Europe
Dhaka	3612850	Southern and Central Asia
Chittagong	1392860	Southern and Central Asia
Khulna	663340	Southern and Central Asia
Sofija	1122302	Eastern Europe
Minsk	1674000	Eastern Europe
Wuhan	4344600	Eastern Asia
Harbin	4289800	Eastern Asia
Shenyang	4265200	Eastern Asia
Kanton [Guangzhou]	4256300	Eastern Asia
Chengdu	3361500	Eastern Asia
Nanking [Nanjing]	2870300	Eastern Asia
Changchun	2812000	Eastern Asia
Xi'an	2761400	Eastern Asia
Dalian	2697000	Eastern Asia
Qingdao	2596000	Eastern Asia
Jinan	2278100	Eastern Asia
Hangzhou	2190500	Eastern Asia
Zhengzhou	2107200	Eastern Asia
Shijiazhuang	2041500	Eastern Asia
Taiyuan	1968400	Eastern Asia
Kunming	1829500	Eastern Asia

3. Complex Queries:

- Retrieve countries where the population is either over 100 million or the region is

“Europe”.

```
mysql> SELECT Name, Population, Region
      -> FROM country
      -> WHERE Population > 100000000 OR Region = 'Europe';
```

Name	Population	Region
Bangladesh	129155000	Southern and Central Asia
Brazil	170115000	South America
China	1277558000	Eastern Asia
Indonesia	212107000	Southeast Asia
India	1013662000	Southern and Central Asia
Japan	126714000	Eastern Asia
Nigeria	111506000	Western Africa
Pakistan	156483000	Southern and Central Asia
Russian Federation	146934000	Eastern Europe
United States	278357000	North America

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

Part 4: Internal and External Composition of Data

Objective: To practice subqueries, joins, and derived tables.

Instructions:

1. Internal Composition:

- Write a subquery that lists countries that have more than 5 cities with a population greater than 1 million.

```
mysql> SELECT country.Name
-> FROM country
-> WHERE country.Code IN (
->     SELECT city.CountryCode
->     FROM city
->     WHERE city.Population > 1000000
->     GROUP BY city.CountryCode
->     HAVING COUNT(city.ID) > 5
-> );
```

Name
Brazil
China
Indonesia
India
Japan
South Korea
Mexico
Pakistan
Russian Federation
United States

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

2. External Composition:

- Write a query that joins city, country, and countrylanguage to find cities with at least one

official language spoken that is not English.

```
mysql> SELECT city.Name AS City, country.Name AS Country, countrylanguage.Language
ge
-> FROM city
-> JOIN country ON city.CountryCode = country.Code
-> JOIN countrylanguage ON country.Code = countrylanguage.CountryCode
-> WHERE countrylanguage.IsOfficial = 'T'
-> AND countrylanguage.Language != 'English';
```

City	Country	Language
Oranjestad	Aruba	Dutch
Kabul	Afghanistan	Dari
Qandahar	Afghanistan	Dari
Herat	Afghanistan	Dari
Mazar-e-Sharif	Afghanistan	Dari
Kabul	Afghanistan	Pashto
Qandahar	Afghanistan	Pashto
Herat	Afghanistan	Pashto
Mazar-e-Sharif	Afghanistan	Pashto
Tirana	Albania	Albaniana
Andorra la Vella	Andorra	Catalan
Willemstad	Netherlands Antilles	Dutch

Part 5: Query Optimization

Objective: To introduce query optimization techniques for better performance.

Instructions:

1. Optimizing a Query:

- Retrieve the 10 cities with the highest populations using sorting and limiting.

```
mysql> SELECT Name, Population
-> FROM city
-> ORDER BY Population DESC
-> LIMIT 10;
```

Name	Population
Mumbai (Bombay)	10500000
Seoul	9981619
São Paulo	9968485
Shanghai	9696300
Jakarta	9604900
Karachi	9269265
Istanbul	8787958
Ciudad de México	8591309
Moscow	8389200
New York	8008278

10 rows in set (0.00 sec)

- Investigate performance using EXPLAIN.

```
mysql> EXPLAIN SELECT Name, Population
-> FROM city
-> ORDER BY Population DESC
-> LIMIT 10;
```

id	select_type	table	partitions	type	possible_keys	key	key_len
1	SIMPLE	city	NULL	ALL	NULL	NULL	NULL

1 row in set, 1 warning (0.00 sec)

- Optimize using LIMIT and indexing.

```
mysql> CREATE INDEX idx_population ON city(Population);
Query OK, 0 rows affected (0.09 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

2. Using Indexes for Optimized Search:

- Write a query that searches for all cities with a population greater than 1 million and a name starting with 'A'.

```
mysql> CREATE INDEX idx_population ON city(Population);
ERROR 1061 (42000): Duplicate key name 'idx_population'
mysql> CREATE INDEX idx_name ON city(Name);
Query OK, 0 rows affected, 1 warning (0.12 sec)
Records: 0 Duplicates: 0 Warnings: 1
```

```
mysql> SELECT Name, Population
-> FROM city
-> WHERE Population > 1000000
-> AND Name LIKE 'A%'
-> ORDER BY Population DESC;
```

Name	Population
Alexandria	3328196
Ankara	3038159
Ahmedabad	2876710
Abidjan	2500000
Addis Abeba	2495000
Alger	2168000
Aleppo	1261983
Anshan	1200000
Adana	1131198
Almaty	1129400
Accra	1070000

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

Part 6: Transactions

Objective: To practice working with transactions to ensure atomicity, consistency, isolation, and durability (ACID properties).

Instructions:

1. Basic Transaction:

- Start a transaction and insert a new city into the city table. Rollback after checking the data.

```
mysql> -- Check the data (you can run this in another query window to verify the
city)
mysql> SELECT * FROM city WHERE Name = 'New City';
+-----+-----+-----+-----+-----+-----+
--+
| ID    | Name      | CountryCode | District      | Population | is_population_large |
+-----+-----+-----+-----+-----+-----+
--+
| 4081  | New City  | USA         | New District  | 2000000    |                     |
+-----+-----+-----+-----+-----+-----+
--+
1 row in set (0.00 sec)

mysql>
mysql> -- Rollback the transaction (undo the insert)
mysql> ROLLBACK;
Query OK, 0 rows affected (0.00 sec)
```

2. Multiple Operations in a Transaction:

- Begin a transaction that inserts a new city and updates the population of a country, committing only if both succeed.

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

mysql>
mysql> -- Insert a new city
mysql> INSERT INTO city (Name, Population, CountryCode, District)
-> VALUES ('Another City', 1500000, 'USA', 'Another District');
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> -- Update the population of a country
mysql> UPDATE country
-> SET Population = Population + 1500000
-> WHERE Code = 'USA';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
mysql> -- Commit the transaction (only if both succeed)
mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)
```

3. Transaction Management:

- Use SAVEPOINT and ROLLBACK TO SAVEPOINT.

```
mysql> -- Set a SAVEPOINT after the insert
mysql> SAVEPOINT after_insert;
Query OK, 0 rows affected (0.00 sec)
```

- Simulate an error in the middle of a transaction to show how rollback works.

```
mysql> -- Simulate an error: let's say the country code doesn't exist or the update fails
mysql> UPDATE country
-> SET Population = Population + 1200000
-> WHERE Code = 'XYZ'; -- Simulating error by using a non-existing country code
Query OK, 0 rows affected (0.00 sec)
Rows matched: 0 Changed: 0 Warnings: 0

mysql>
mysql> -- Check the data (you can run SELECT to check if the city was added)
mysql> SELECT * FROM city WHERE Name = 'Temporary City';
+-----+-----+-----+-----+-----+-----+
| ID   | Name           | CountryCode | District           | Population | is_population_large |
+-----+-----+-----+-----+-----+-----+
| 4083 | Temporary City | USA         | Temporary District | 1200000   | NULL                |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
mysql> -- Rollback to SAVEPOINT (this will undo the update but keep the insert)
mysql> ROLLBACK TO SAVEPOINT after_insert;
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> -- Commit the changes (city insertion will be committed, update is rolled back)
mysql> COMMIT;
Query OK, 0 rows affected (0.01 sec)
```

Submission Instructions:

1. In your personal repository, create a subfolder named "managing-the-world-database" with the following structure:

- /queries
- /data_types_indexes
- /views_users_privileges
- /operators_comparators_logical
- /subqueries_joins_compositions
- /query_optimization
- /transactions
- /screenshots (store query result screenshots here)

Grading Criteria:

- Correctness: Ensure all SQL queries execute correctly.
- Completeness: Include all queries, explanations, and screenshots as required.
- Clarity: Maintain clear and consistent naming conventions for files and folders in the GitHub repository.