DDL and views

Agenda

- Data Definition Language (DDL)
- Creating and Manipulating Tables
- Creating and Manipulating Views

Data Definition Language (DDL)

Data Definition Language

- Types of commands
  - Defining / editing objects
    - `CREATE`
    - `ALTER`
    - `DROP`
  - Managing access permissions
    - `GRANT`
    - `REVOKE`

Data Definition Language

- `CREATE` / `CREATE OR REPLACE` commands
  - `CREATE TABLE <name> (<fields definitions>)`
  - `CREATE VIEW <name> AS <select>`

```
CREATE TABLE PERSONS (
   PERSON_ID INTEGER NOT NULL,
   NAME VARCHAR(50) NOT NULL,
   CONSTRAINT PERSON_PK PRIMARY KEY(PERSON_ID)
)

CREATE OR REPLACE VIEW PERSONS_TOP_10 AS
SELECT NAME FROM EMPLOYEES LIMIT 10;
```

Data Definition Language

- ALTER command
  - ALTER TABLE <name> <command>

```
-- Add a foreign key constraint TOWN --> COUNTIRY
ALTER TABLE TOWN
  ADD CONSTRAINT TOWN_COUNTRY_FK
  FOREIGN KEY (COUNTRY_ID)
  REFERENCES COUNTRY(ID) ENABLE

-- Add column COMMENT to the table PERSON
ALTER TABLE PERSONS ADD ("COMMENT" VARCHAR2(800))

-- Remove column COMMENT from the table PERSON
ALTER TABLE PERSONS DROP COLUMN "COMMENT"
```

Data Definition Language

- DROP command
  - DROP TABLE <name>
  - DROP TRIGGER <name>
  - DROP INDEX <name> on <table>

```
DROP TABLE PERSONS;
DROP TRIGGER TRG_EMP_BEFORE;
DROP INDEX IND_NAME ON EMPLOYEES;
```

Data Definition Language

- `GRANT` command

```
GRANT <persmission> ON <object> TO <user>
```

  - Example:

```
GRANT ALL ON hrm.* TO ''@'localhost';
```

- `REVOKE` command

```
REVOKE <persmission> ON <object> FROM <user>
```

  - Example:

```
REVOKE SELECT ON HRM.* FROM ''@'localhost';
```

Creating and Manipulating Tables

Creating and Manipulating Tables

- In order to create a new table:
  - Define the table name
  - Define the columns and their types
  - Define the table constraints (including primary/foreign keys)

Creating and Manipulating Tables

- A table is created with the CREATE TABLE statement
- Constraints may be specified:
  - as part of the column definitions
  - in the CREATE TABLE statement (outside the column definitions)
  - outside the CREATE TABLE statement using ALTER TABLE statements

Creating and Manipulating Tables

- Types of constraints:
  - NOT NULL (in MySQL it is not defined as a constraint)
  - UNIQUE
  - PRIMARY KEY
  - FOREIGN KEY
  - CHECK (not enforced by MySQL)

Creating and Manipulating Tables

- Example:

```
CREATE TABLE PEOPLE(
   PERSON_ID NUMBER AUTO_INCREMENT NOT NULL,
   NAME VARCHAR(100) NOT NULL,
   CONSTRAINT PERSONS_PK PRIMARY KEY(PERSON_ID)
);
```

Creating and Manipulating Tables

- Example (constraints are part of the column definitions):

```
CREATE TABLE LOCATIONS1 (
  Id INT AUTO_INCREMENT PRIMARY KEY,
  City VARCHAR(100) NOT NULL UNIQUE,
  Country VARCHAR(100) DEFAULT 'Bulgaria',
  DateAdded DATE,
  Status VARCHAR(10) CHECK (Status in ('OPENING',
'OPENED')),
  ManagerId INT REFERENCES Employees(Id)
);
```

Creating and Manipulating Tables

- Example (constraints are outside the column definitions):

```
CREATE TABLE LOCATIONS2 (
  Id INT AUTO_INCREMENT,
  City VARCHAR(100) NOT NULL,
  Country VARCHAR(100) DEFAULT 'Bulgaria',
  DateAdded DATE,
  Status VARCHAR(10),
  ManagerId INT,
  CONSTRAINT c_locations2_PK PRIMARY KEY(Id),
  CONSTRAINT c_locations2_City_Unq UNIQUE (City),
  CONSTRAINT c_locations2_Status CHECK (Status in
('OPENING', 'OPENED')),
  CONSTRAINT loc2_fk_empl FOREIGN KEY(ManagerId)
REFERENCES Employees(Id));
```

Creating and Manipulating Tables

- Example (constraints are outside the table definitions):

```
CREATE TABLE locations3 (
  Id INT AUTO_INCREMENT,
  City VARCHAR(100),
  Country VARCHAR(100) DEFAULT 'Bulgaria',
  DateAdded DATE,
  Status VARCHAR(10),
  ManagerId INT
);
```

NO CONSTRAINTS YET

Creating and Manipulating Tables

- Example (constraints are outside the table definitions):

CONSTRAINTS ARE ADDED AFTER THE TABLE DEFINITION

```
alter table Locations3 add constraint c_locations3_PK
PRIMARY KEY(Id);
alter table Locations3 add constraint
c_locations3_City_Unq UNIQUE (City);
alter table Locations3 add constraint
c_locations3_Status CHECK (Status in ('OPENING',
'OPENED'));
alter table Locations3 add constraint c_locations3_fk
FOREIGN KEY(ManagerId) REFERENCES Employees(Id);
alter table Locations3 modify City varchar(100) NOT
NULL;
```

Creating and Manipulating Tables

- MySQL has special syntax for dropping of constraints
- Example:

```
DROP INDEX C_LOCATIONS4_CITY_UNQ ON LOCATIONS3;
ALTER TABLE LOCATIONS3 DROP PRIMARY KEY;
ALTER TABLE LOCATIONS3 DROP FOREIGN KEY
C_LOCATIONS4_FK;
```

- In MySQL a unique constraint is dropped by dropping the corresponding index

Creating and Manipulating Tables

- Columns can be added/modified using the `ALTER TABLE` command:

- Example:

```
ALTER TABLE LOCATIONS3 ADD CAPACITY INT;
ALTER TABLE LOCATIONS3 MODIFY CAPACITY
DOUBLE(5,1);
```

Creating and Manipulating Views

Creating and Manipulating Views

- Views are named SQL SELECT queries
  - Usually views join multiple tables and provide filtering
  - Views simplify queries

```
CREATE VIEW V_SEATTLE_EMPLOYEES AS
  SELECT E.EMPLOYEE_ID, E.FIRST_NAME,
    E.LAST_NAME, CITY, DEPARTMENT_NAME
  FROM EMPLOYEES E
  JOIN DEPARTMENTS D
    ON D.DEPARTMENT_ID = E.DEPARTMENT_ID
  JOIN LOCATIONS L
    ON D.LOCATION_ID = L.LOCATION_ID
  WHERE CITY='Seattle'
```

Creating and Manipulating Views

- Views:
  - are virtual tables and hence don't exist physically on disk
  - can be used in SELECT, INSERT, UPDATE and DELETE depending on the type of view
  - can be used to imply security on the data they represent
  - views are roughly divided in two types - simple and complex

Creating and Manipulating Views

- Simple views:
  - `SELECT` from one table
  - no SQL functions and GROUP BY clauses
  - DML statements can typically be used (`INSERT`, `UPDATE` and `DELETE`)

Creating and Manipulating Views

- Complex views:
  - `SELECT` from one or more tables
  - can use SQL functions and GROUP BY clauses
  - DML statements typically cannot be used (`INSERT`, `UPDATE` and `DELETE`)

Questions ?

Exercises (1)

1. What is complex view ?
2. What is the purpose of DDL commands ?
3. How can constraints be added on the table data ?
4. Create table EmployeesCopy with the same structure as the Employees table.

Exercises (2)

5. Add a new column called Mandatory to the Certificates table with default value False and check constraint for three possible values - **False** , **True** and **Only for Developers.**

6. Create a view called Emp_Qualifications that displays the name of an employee along with the number of certificates and skills he has.

Exercises (3)

7. Write a SQL statement to create a table `USERS`. Users should have username, password, full name and last login time. Choose appropriate data types for the fields of the table. Define a primary key column with a primary key constraint. Define a sequence for populating the primary key. Define a trigger to update the primary key column value before inserting a record.

8. Write a SQL statement to create a view that displays the users from the `USERS` table that have been in the system today. Test if the view works correctly.

9. Write a SQL statement to create a table `GROUPS`. Groups should have unique name (use unique constraint). Define autoincrement primary key.

Exercises (4)

10. Write a SQL statement to add a column `GROUP_ID` to the table `USERS`. Fill some data in this new column and as well in the GROUPS table. Write a SQL statement to add a foreign key constraint between tables `USERS` and `GROUPS`.
11. Write SQL statements to insert several records in the `USERS` and `GROUPS` tables.
12. Write SQL statements to insert in the `USERS` table the names of all employees from the `EMPLOYEES` table. Combine the first and last names as a full name. For user name use the email column from `EMPLOYEES`. Use blank password.
13. Run the above 10 times to generate enough testing data for the USERS table.

Exercises (5)

14. Write a SQL statement that changes the password to `NULL` for all `USERS` that have not been in the system since 10.03.2006. Select table data to see the changes. Rollback the transaction to forget all changes and return back.

15. Write a SQL statement that deletes all users without passwords (`NULL` or empty password). Select table data to see the changes. Rollback the transaction to forget all changes and return back.

16. Write a SQL query to list all users whose username starts with `'s'` and the number of groups for each of them. View its execution plan.

Exercises (6)

17. Define table WORKHOURS to store work reports for each employee (date, task, hours, comments). Don't forget to define automatically populated primary key (autoincrement primary key constraint).
18. Define foreign key between the tables WORKHOURS and EMPLOYEE. Add additional column in the employee table if needed.
19. Write several SQL statements to fill some data in the WORKHOURS table.
20. Write a SQL query to find all the average work hours per week for each country.
21. Write a SQL query to find all the departments where some employee worked overtime (over 8 hours/day) during the last week.