# SQL:
# A COMMERCIAL DATABASE LANGUAGE

Data Definition,
Basic Integrity Constraints,
Schema Changes

# Outline of Chapters 8, 9

1. Introduction
2. Data Definition, Basic Constraints, and Schema Changes
3. Basic Queries
4. More complex Queries
5. Aggregate Functions and Grouping
6. Summary of SQL queries
7. Data Change statements
8. Views
9. Complex Constraints
10. Database Programming

# 1. Introduction (1)

- The name **SQL** is derived from Structured Query Language. Originally SQL was called SEQUEL (for Structured English QUEry Language).
- SQL is a *comprehensive database language*. It has statements for *data definition, query* and *update* (in this sense it is both a *DDL* and a *DML*). In addition, it has facilities for *defining views*, for specifying *security and authorization*, for defining *integrity constraints*, and for specifying *transaction controls*. It also has rules for *embedding SQL statements* into a general-purpose programming language.
- SQL became a *standard for relational databases* in 1986 (called SQL-86 or SQL1). A revised and more extended version became a standard in 1992 (called SQL-92 or SQL2). A new version of the standard (called SQL:1999) extends SQL with object-oriented and other recent database concepts.

# 1. Introduction (2)

- The SQL language provides a *high-level declarative language* interface: the user specifies what the result is to be and leaves the decisions on how to execute the query to the DBMS.

- SQL includes features from *Relational Algebra* but it is based to a greater extend on *Tuple Relational Calculus*. However, Relational Algebra is very important for *query processing and optimization* of SQL queries in relational DBMS.

- In class we will overview the most important features of SQL. Details can be found in SQL documents.

# 1. Introduction (3)

- The terms table, row and column are used in SQL documents for relation, tuple and attribute respectively.

- *SQL is case insensitive*. It treats upper- and lower-case letters as the same letter. Only inside quotes does SQL make a distinction between upper- and lower-case letters.

# 2. Data Definition, Constraints and Schema Changes in SQL

- In this section we discuss DDL commands in SQL.

# 2.1.Schema and Catalog concepts (1)

- The concept of a **schema** is used in SQL to group together constructs that belong to the same database application

- A schema includes **schema elements** such as *tables, constraints, views, users* and other constructs.

- An SQL schema is identified by a **schema name** and includes an **authorization identifier** to indicate the user who owns the schema. In addition, it can include **descriptors** for every element in the schema.

- The following statement creates a **schema/database** called COMPANY

**CREATE DATABASE** COMPANY;

# 2.1.Schema and Catalog concepts (2)

- The concept of a **catalog** is used in SQL to denote a named collection of schemas.

- A catalog always contains a special schema called INFORMATION_SCHEMA. This schema provides information on all the schemas in the catalog and all the element descriptors in these schemas.

- Some elements (e.g. domain definitions) *can be shared by schemas in the same catalog*.

  Integrity constraints can be defined between relations *only if they exist in schemas of the same catalog*.

# 2.2. Data Types (1)

- The data types available for attribute values include numeric, character-string, bit-string, date and time.
- **Numeric**:

  INTEGER or INT: signed integer

  Also TINYINT, SMALLINT, MEDIUMINT, BIGINT

  FLOAT: floating point number

  Also REAL, DOUBLE (DOUBLE PRECISION)

  DEC(i,j) , DECIMAL(i,j) or NUMERIC(i,j): decimal number
- **Character string**:

  CHARACTER(n) or CHAR(n): fixed-length character string

  VARCHAR(n): varying length character string
- **Bit string**:

  BIT(n): fixed-length bit string
- More types at https://mariadb.com/kb/en/library/data-types

# 2.2. Data Types (2)

- **Date and time:**

  DATE: Made up of year-month-day in the format yyyy-mm-dd

  TIME: Made up of hour:minute:second in the format hh:mm:ss

  TIME(i): Made up of hour:minute:second plus i additional digits specifying fractions of a second.
  Format is hh:mm:ss:ii…i

  TIMESTAMP: Has both DATE and TIME components

# 2.4. CREATE TABLE Statement (1)

- The **CREATE TABLE** command is used to specify a new relation, its attributes, their data type and constraints.
- The key, entity integrity and referential integrity constraints can be specified within a CREATE TABLE statement or later using the ALTER TABLE command.

```
CREATE TABLE   EMPLOYEE
  (FNAME        VARCHAR(15)        NOT NULL,
   MINIT        CHAR,
   LNAME        VARCHAR(15)        NOT NULL,
   SSN          CHAR(9)            NOT NULL,
   BDATE        DATE,
   ADDRESS      VARCHAR(30),
   SEX          CHAR,
   SALARY       DECIMAL(10,2),
   SUPERSSN     CHAR(9),
   DNO          INTEGER            NOT NULL,
   PRIMARY KEY (SSN),
   FOREIGN KEY (DNO) REFERENCES DEPARTMENT(DNUMBER),
   FOREIGN KEY (SUPERSSN) REFERENCES   EMPLOYEE(SSN) );
```

# 2.4. CREATE TABLE Statement (2)

- An example with attribute default values and secondary key specification.

```
CREATE TABLE   DEPARTMENT
  (DNAME    VARCHAR(10)   NOT NULL,
  DNUMBER    INT   NOT NULL,
  MGRSSN  CHAR(9)  NOT NULL  DEFAULT
  '888665555',
  MGRSTARTDATE DATE,
  PRIMARY KEY (DNUMBER),
  UNIQUE (DNAME),
  FOREIGN KEY (MGRSSN) REFERENCES EMPLOYEE(SSN) );
```

# 2.4. CREATE TABLE Statement (3)

- A constraint can be given a name following the keyword **CONSTRAINT**. Giving names to constraints is optional but is a good practice to follow.
  The names of all constraints within a particular schema must be unique.
  A constraint name can be used to drop a constraint (and replace it later by another constraint).
- SQL has a default policy that *any modifications violating the referential integrity constraints is rejected by the system*. However, in SQL the schema designer can specify the action to be taken upon *deletion of a referenced tuple* or *upon modification of a referenced primary key value*, by attaching a referential triggered action clause to a foreign key constraint.
  The options include SET NULL, CASCADE, SET DEFAULT.
  They can be qualified on either ON DELETE or ON UPDATE.

# 2.4. CREATE TABLE Statement (4)

- Example.

```
CREATE TABLE EMPLOYEE

   (FNAME          VARCHAR(15)          NOT NULL,
    …

    DNO            INTEGER     NOT NULL    DEFAULT  4,

   CONSTRAINT EMPPK
   PRIMARY KEY (SSN),

   CONSTRAINT EMPSUPERFK
   FOREIGN KEY (SUPERSSN) REFERENCES EMPLOYEE
   ON DELETE SET NULL ON UPDATE CASCADE

   CONSTRAINT EMPDEPTFK
   FOREIGN KEY (DNO) REFERENCES DEPARTMENT
   ON DELETE SET DEFAULT ON UPDATE CASCADE );
```

# 2.4. CREATE TABLE Statement (5)

- Example.

```
CREATE TABLE   DEPARTMENT
  (DNAME    VARCHAR(10)   NOT NULL,
  DNUMBER       INT     NOT NULL,
  MGRSSN CHAR(9)  NOT NULL DEFAULT '888665555',
  MGRSTARTDATE       DATE,

  CONSTRAINT DEPTPK
  PRIMARY KEY (DNUMBER),

  CONSTRAINT DEPTSK
  UNIQUE (DNAME),

  CONSTRAINT DEPTMGRFK
  FOREIGN KEY (MGRSSN) REFERENCES EMPLOYEE
  ON DELETE SET DEFAULT ON UPDATE CASCADE);
```

- The name of a schema can be explicitly specified in the definition of a table:

```
CREATE TABLE   COMPANY.DEPARTMENT …
```

# 2.5. DROP DATABASE Command

- A schema can be deleted using the **DROP DATABASE** command.
- IF EXISTS can be used to prevent an error if the database doesn't exist.
- **USE THIS COMMAND VERY CAREFULLY**

**DROP SCHEMA IF EXISTS** COMPANY **;**

# 2.6. DROP TABLE Command

- A table can be deleted using the **DROP TABLE** command.
- IF EXISTS can be used to prevent errors
- If a foreign key references this table, the table cannot be dropped. In this case, it is necessary to drop the foreign key first.
- **USE THIS COMMAND WITH EXTREME CAUTION**

**DROP TABLE IF EXISTS** DEPENDENT**;**

# 2.7. ALTER TABLE Command (1)

- Adding a column.

  **ALTER TABLE** COMPANY.EMPLOYEE **ADD JOB VARCHAR(12);**

  **ALTER TABLE** COMPANY.EMPLOYEE **ADD JOB VARCHAR(12) NOT NULL;**

  A DEFAULT clause can also be specified. If no such clause is specified, the new attribute will have NULLs in all the tuples of the relation after the command is executed.

- Dropping a column.

  **ALTER TABLE** COMPANY.EMPLOYEE **DROP** ADDRESS **CASCADE;**

  CASCADE option: all constraints and views that reference the column are dropped from the schema.
  RESTRICT option: the command is successful only if no views or constraints reference the column.

# 2.7. ALTER TABLE Command (2)

- Dropping a default clause in a column.

  **ALTER TABLE** COMPANY.DEPARTMENT **ALTER** MGRSSN **DROP DEFAULT;**

- Adding a default clause in a column.

  **ALTER TABLE** COMPANY.DEPARTMENT **ALTER** MGRSSN **SET DEFAULT '33344555';**

# 2.7. ALTER TABLE Command (3)

- Dropping a constraint.

  To be dropped a constraint must have a name.

  **ALTER TABLE** COMPANY.EMPLOYEE **DROP CONSTRAINT** EMPSUPERFK **CASCADE;**

  *CASCADE and RESTRICT have meaning only if the constraint is a candidate key definition*. In this case, if CASCADE is specified, all the foreign key definitions that reference the candidate key will be dropped too.

- IF NOT EXISTS can be used to prevent errors.

- Adding a constraint.

  **ALTER TABLE** COMPANY.EMPLOYEE **ADD CONSTRAINT** EMPSUPERFK **FOREIGN KEY** EMPLOYEE(SUPERSSN) **REFERENCES** EMPLOYEE(SSN)**;**

- The constraint to be created can be named or unnamed.

# 2.8. CIRCULAR FOREIGN KEYS (1)
# the chicken-and-egg problem

- Defining foreign keys in tables raises some other issues:
- Consider the table definitions:

```
CREATE TABLE   EMPLOYEE
…
FOREIGN KEY (DNO) REFERENCES  DEPARTMENT(DNUMBER)
```

and

```
CREATE TABLE   DEPARTMENT
…
FOREIGN KEY (MGRSSN) REFERENCES EMPLOYEE(SSN)
```

- The problem is that either EMPLOYEE or DEPARTMENT table has to be defined first.
  In both cases the system will issue an error message.
  Why?

# 2.8. CIRCULAR FOREIGN KEYS (2)
# the chicken-and-egg problem

- One solution is to postpone the introduction of the foreign-key constraint in the first table.

- For instance, if CREATE TABLE EMPLOYEE
  is executed first, we should not have the foreign key clause
  FOREIGN KEY (DNO) … in the statement.

- After CREATE TABLE DEPARTMENT has
  been processed, we can add the desired constraint to employee
  using the ALTER TABLE command:

  **ALTER TABLE** EMPLOYEE
  **ADD CONSTRAINT** EMPDEPTFK
  **FOREIGN KEY (**DNO**) REFERENCES**
  DEPARTMENT(DNUMBER)

# 2.8. CIRCULAR FOREIGN KEYS (3)
# the chicken-and-egg problem

- If we want to populate our COMPANY database, we are in front of another problem.

- If we try to insert a tuple to relation EMPLOYEE with any non-NULL value for attribute DNO, *our insertion will be rejected as it violates the foreign key constraint*
  DNO references DEPARTMENT(DNUMBER).

  If we try to insert a tuple to relation DEPARTMENT with any non-NULL value for attribute MGRSSN, *our insertion will be rejected as it violates the foreign key constraint*
  MGRSSN references EMPLOYEE(SSN).

- We are again facing the chicken-and-egg problem!

# 2.8. CIRCULAR FOREIGN KEYS (4)
# the chicken-and-egg problem

- One solution is to initially replace DNO component in all tuples to be inserted in the EMPLOYEE relation with NULL. Then, when DEPARTMENT is populated with appropriate tuples, we can scan relation EMPLOYEE and replace NULLs with valid department numbers.

- This solution is *awkward* and *error-prone*.

- A better solution is to use *transactions* and *deferred checking of integrity constraints*. We will discuss this issue later (if we have time).