Section 3

Ch3. Introduction to SQL

Slide contents follow

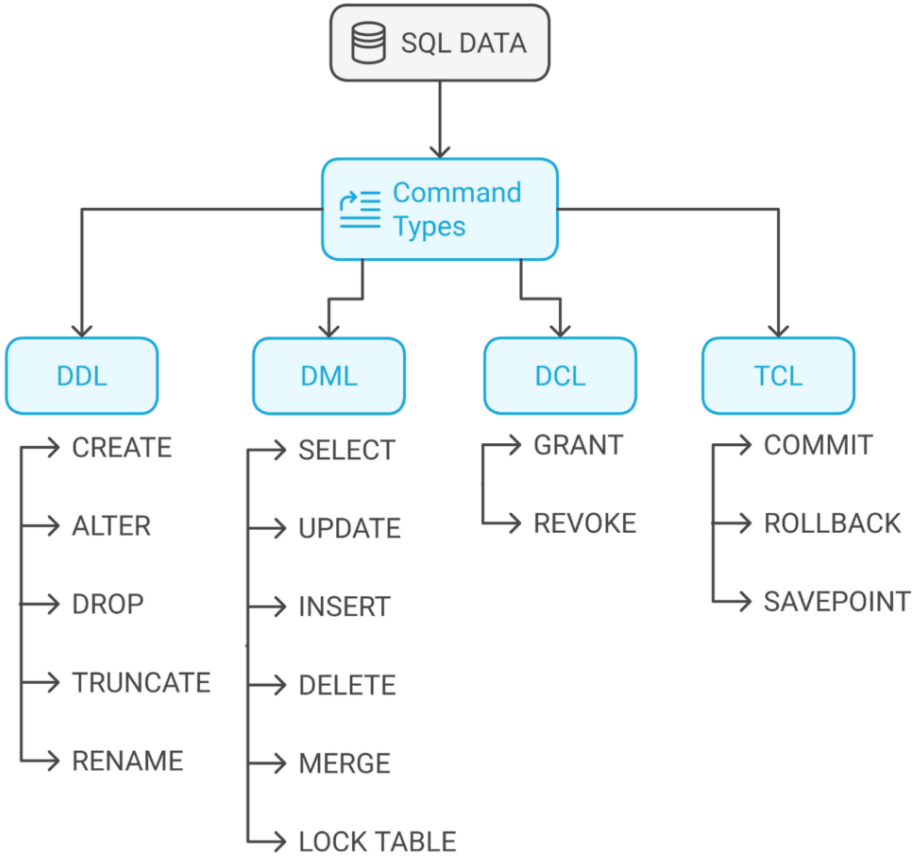
Database System Concepts, 7th edition.

Prepared by:

AbdElrahman Rabea, Eng.

SQL

- SQL refers to Structured Query Language.
- SQL serves as the interface between users or applications and the database management system (DBMS).
- It is referred as query language but it can do much more than query database
- SQL enables users to perform essential database operations, including:
    1. Data Definition: Creating, altering, and deleting database structures such as tables.
    2. Data Manipulation: Inserting, updating, retrieving, and deleting data.
    3. Data Control: Managing database access permissions and security.
    4. Transaction Control: Ensuring database integrity during operations.
- You can use it interactive (GUI), prompt or embedded in programs
- Declarative and based on relational algebra



SQL History Timeline

- **Early 1970s:** IBM developed the original version of SQL, initially called Sequel, as part of the System R project.
- **Late 1970s:** The language evolved, and its name changed to SQL (Structured Query Language).
- **1986:** The American National Standards Institute (ANSI) and International Organization for Standardization (ISO) published the first SQL standard, SQL-86.
- **1989:** ANSI released an extended standard, SQL-89.
- **1992:** The SQL-92 standard introduced significant improvements.
- **1999:** SQL:1999 introduced features like triggers, recursive queries, and procedural extensions (e.g., procedural SQL).
- **2003:** SQL:2003 added XML-related features and window functions.
- **2006:** SQL:2006 improved XML integration.
- **2008:** SQL:2008 introduced new data types, MERGE statements, and better error handling.
- **2011:** SQL:2011 focused on temporal databases, adding support for tracking historical data.
- **2016:** SQL:2016 introduced JSON support, enhanced analytics, and new security features.
- **2023:** The most recent standard, SQL:2023

Most commercial **RDBMS (Relational Database Management Systems)** fully implement **SQL-92**, along with features from later SQL standards. However, **each DBMS adds its own proprietary features** beyond the standard.
All SQL statements covered in this course are compatible with Microsoft SQL Server.

Data Definition Language (DDL)

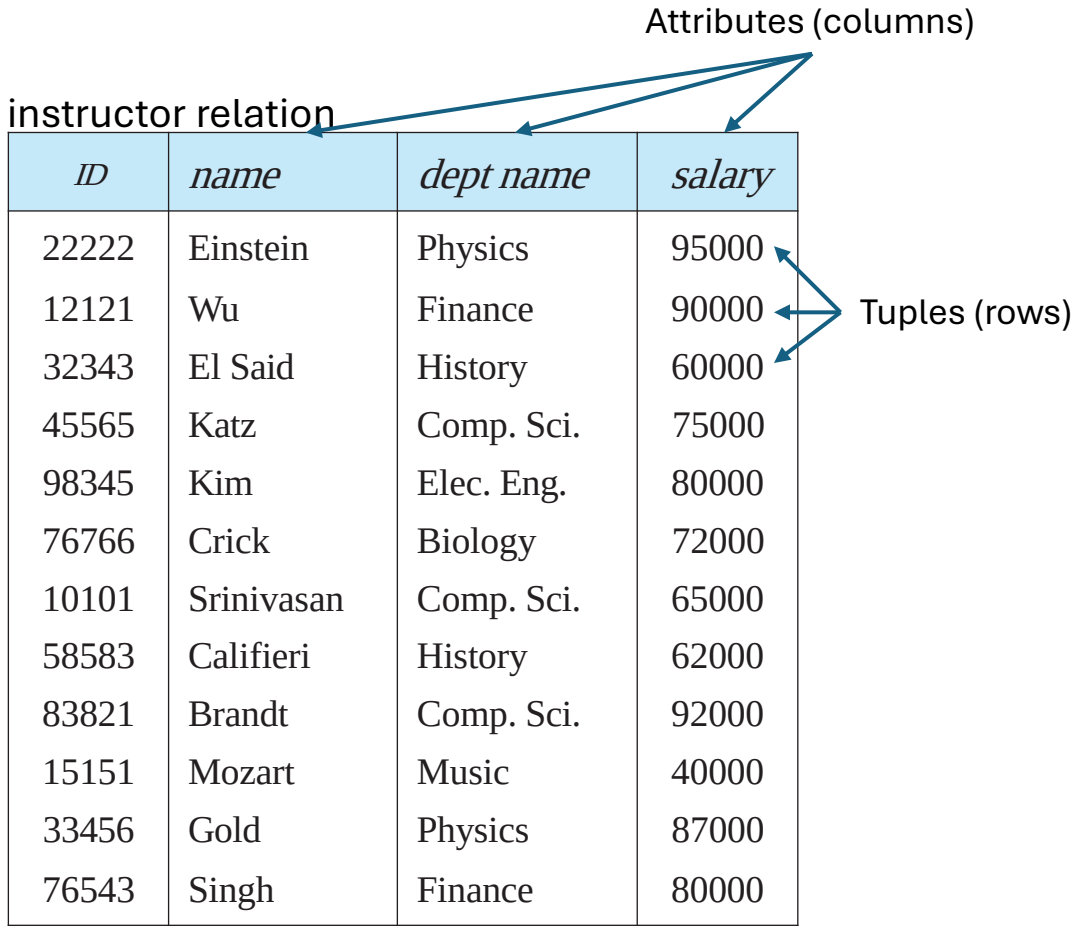
What DDL can do?

- Defines the schema for each relation.
- Specifies the types of values for each attribute.
- Enforces integrity constraints (PRIMARY KEY, FOREIGN KEY, UNIQUE, NOT NULL, CHECK).
- Defines indexes to improve query performance.
- Manages security and authorization (GRANT, REVOKE).
- Specifies the physical storage structure of relations on disk.
- Allows view creation (CREATE VIEW).
- Supports triggers for automated actions.
- Defines sequences and auto-increment for unique identifiers.

```sql
-- Create tables with constraints
CREATE TABLE Students (
    student_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(255) UNIQUE,
    birth_date DATE CHECK (birth_date < '2010-01-01')
);

CREATE TABLE Courses (
    course_id INT PRIMARY KEY,
    course_name VARCHAR(100) NOT NULL,
    credits INT CHECK (credits BETWEEN 1 AND 5)
);

CREATE TABLE Enrollments (
    enrollment_id INT PRIMARY KEY AUTO_INCREMENT,
    student_id INT,
    course_id INT,
    grade CHAR(1) CHECK (grade IN ('A', 'B', 'C', 'D', 'F')),
    FOREIGN KEY (student_id) REFERENCES Students(student_id) ON DELETE CASCADE,
    FOREIGN KEY (course_id) REFERENCES Courses(course_id) ON DELETE CASCADE
);

-- Indexing for faster searches
CREATE INDEX idx_student_name ON Students(name);

-- View for quick access
CREATE VIEW StudentEnrollments AS
SELECT s.student_id, s.name, c.course_id, c.course_name, e.grade
FROM Students s
JOIN Enrollments e ON s.student_id = e.student_id
JOIN Courses c ON e.course_id = c.course_id;

-- Granting privileges
GRANT SELECT, INSERT ON Students TO user1;
```

Recall Relational model structure

- A relational database consists of set of named relations (tables)
- Each relation has a set of named attributes (columns)
- Each tuple (row) has a value for each attribute
- Each attribute has a domain
- Domain is the set of allowed values for each attribute. Ex. dept-name
- The special value **null** is a member of every domain. Indicated that the value is “unknown”.
- The null value causes complications in the definition of many operations.



ID	*name*	*dept name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

Recall Keys

Key: attribute whose value is unique in each tuple
Or set of attributes whose combined values are unique

Primary key: key that is unique and can't be null

Foreign-key constrainst:

- **Foreign-key constraints** enforce consistency between relations in a database.
- The **dept_name** attribute in the instructor relation must correspond to an existing department in the department relation, that is No tuple in instructor should have a dept_name value that does not exist in department.
- Given a tuple **ta** from instructor, there must be a corresponding tuple **tb** in department with the same dept_name value.
- A **foreign-key constraint** from attribute(s) **A** in relation $\mathbf{r_1}$ to the primary key **B** in relation $\mathbf{r_2}$ ensures:
- The value of **A** in each tuple of $\mathbf{r_1}$ must match a value of **B** in some tuple of $\mathbf{r_2}$.
- The **foreign key (A)** belongs to $\mathbf{r_1}$ and references $\mathbf{r_2}$.
- $\mathbf{r_1}$ is called the **referencing relation**, while $\mathbf{r_2}$ is the **referenced relation**.

Classroom (*building*, *room number*, *capacity*)
department(*dept_name*, *building*, *budget*)
course(*course_id*, *title*, *dept name*, *credits*)
Instructor(*ID*, *name*, *dept name*, *salary*)
section(*course id*, *sec id*, *semester*, *year*, *building*, *room number*, *time slot id*)
teaches(*ID*, *course id*, *sec id*, *semester*, *year*)
student(*ID*, *name*, *dept name*, *tot cred*)
takes(*ID*, *course id*, *sec id*, *semester*, *year*, *grade*)
advisor(*s_ID*, *i_ID*)
time slot(*time slot id*, *day*, *start time*, *end time*)
prereq(*course id*, *prereq id*)

Schema of the university database.

Basic Data types

Category	Data Type	Description	Example
Character Types	CHAR(n)	Fixed-length character string with length n. Full form: CHARACTER.	'Hello'
	VARCHAR(n)	Variable-length character string with max length n. Full form: CHARACTER VARYING. the n defines the string length in byte-pairs	'SQL Server'
	NCHAR(5)	Fixed-length Unicode character string.	N'مرحبا'
	NVARCHAR(10)	Variable-length Unicode character string.	N'こんにちは'
Integer Types	INT	Standard integer (machine-dependent range). Full form: INTEGER.	100000
	SMALLINT	Small integer (machine-dependent range).	32000
Decimal & Numeric	NUMERIC(p, d)	Fixed-point number with p total digits, d of which are decimal places. Example: NUMERIC(3,1) stores 44.5, but not 444.5.	123.45
	DECIMAL(p, d)	Same as NUMERIC.	99.9
Floating-Point	REAL	Floating-point number with machine-dependent precision. (4bytes)	3.14
	FLOAT(n)	n is the number of bits that are used to store the mantissa of the float number in scientific notation and, therefore, dictates the precision and storage size.	2.7182818284
Date & Time	DATE	Stores only date (YYYY-MM-DD).	'2025-03-03'
	TIME	Stores only time (hh:mm:ss).	'14:30:00'
	DATETIME	Stores both date and time (YYYY-MM-DD hh:mm:ss).	'2025-03-03

Lets create our first table

- An SQL relation is defined using the **create table** command:

  create table *r*

  (A_1 D_1, A_2 D_2, …, A_n D_n,
  (integrity-constraint$_1$),
  …,
  (integrity-constraint$_k$))

  - *r* is the name of the relation
  - each A_i is an attribute name in the schema of relation *r*
  - D_i is the data type of values in the domain of attribute A_i

- Example:

```
create table instructor
  (ID     varchar(5),
   name     varchar(20) not null,
   dept_name    varchar(20),
   salary     numeric(8,2),
   primary key (ID),
   foreign key (dept_name) references department (dept_name)
  );
```

Integrity constrains

- Primary key
- Foreign key
- Unique
- Check
- Not null

```sql
CREATE TABLE instructor (
    ID          VARCHAR(5),
    name        VARCHAR(20),
    dept_name   VARCHAR(20),
    salary      NUMERIC(8,2),
    PRIMARY KEY (ID),
    CHECK (name IS NOT NULL),
    FOREIGN KEY (dept_name) REFERENCES department(dept_name),
    CHECK (salary > 29000)
);
```

For simple constraints you can do this:

```sql
CREATE TABLE instructor (
    ID          VARCHAR(5) PRIMARY KEY,
    name        VARCHAR(20) NOT NULL,
    dept_name   VARCHAR(20) REFERENCES department(dept_name),
    salary      NUMERIC(8,2) CHECK (salary > 29000)
);
```

Alter table

- Add a New Column

```
ALTER TABLE instructor
ADD hire_date DATE;
```

- Add a New NOT NULL Constraint (salary)

```
ALTER TABLE instructor
Alter column salary numeric(8,2) not null;
```

- Modify an Existing Column (Increase name Size)

```
ALTER TABLE instructor
Alter Column name VARCHAR(50);
```

- Drop an Existing Column (hire_date)

```
ALTER TABLE instructor
DROP COLUMN hire_date;
```

- And much more …..
- Mainly you are modifying the structure (rename, datatypes, domains, ….)

University Schema diagram (practice DDL)