



## *Lecture 12*

# Chair of Mobile Business & Multilateral Security

## Business Informatics 2 (PWIN) SS 2025

### Database Management II Structured Query Language

**Prof. Dr. Kai Rannenber**

Chair of Mobile Business & Multilateral Security  
Johann Wolfgang Goethe University Frankfurt a. M.

- Introduction
- Basic SQL Language Elements
- Advanced SQL Language Elements
- SQL-driven Database Features
- Issues of SQL
- SQL Tool: W3Schools

- **SQL - Structured Query Language**
  - Developed in the 1970s
  - Current standard for management of relational databases:
    - ANSI (American National Standards Institute)
    - ISO (International Standardization Organization),
    - Current version of framework: **ISO/IEC 9075:2023**  
Information technology – Database languages – SQL – Part 1:  
Framework (SQL/Framework)
  - Non-procedural, descriptive and declarative language for the use of databases
    - With a SQL query, a user only expresses a desired result (and not the way how this result has to be generated).

- **Data Definition Language (DDL)**
  - Definition of data structures (e.g. tables, databases)
- **Data Manipulation Language (DML)**
  - Viewing, inserting, deleting and updating data in a database
- **Data Control Language (DCL)**
  - Access control for data in a database
- **Transaction Control Language (TCL)**
  - Control of transactional processing in a database
  - A transaction is a logical unit of multiple SQL statements.

- Introduction
- Basic SQL Language Elements
- Advanced SQL Language Elements
- SQL-driven Database Features
- Issues of SQL
- SQL Tool: W3Schools



# Data Definition Language: CREATE

```
CREATE TABLE table_name  
(  
    column_name1 data_type,  
    column_name2 data_type,  
    .....  
)
```

## Data types

| Data Type     |  |
|---------------|--|
| integer(size) | Integer, “size” defines the maximum number of digits.  |
| decimal(p,s)  | p specifies the maximum total number of decimal digits that can be stored, both to the left and to the right of the decimal point. s specifies the number of decimal digits to the right of the decimal point. |
| char(size)    | Fixed-length character data (length of “size”)   |
| varchar(size) | Variable-length character data (maximum length of “size”)  |
| date(yyymmdd) | Date [and time] with all four digits of the year, month, day, [hour (in 24-hour format), minute, and second], e.g. 20070115  |
| ...           |  |

```
CREATE TABLE Product_Info  
(  
  Article_no integer(10),  
  Weight decimal(2,2),  
  Resolution varchar(9),  
  Power_consumption integer(3)  
)
```

NB: “decimal(2,2)” may not make much sense in the application case, but is syntax wise correct.

### Table “Product\_Info”

| <u>ID</u> | Article_no | Weight | Resolution | Power_consumption |
|-----------|------------|--------|------------|-------------------|
|           |            |        |            |                   |
|           |            |        |            |                   |
|           |            |        |            |                   |
|           |            |        |            |                   |
|           |            |        |            |                   |

**DROP Table** Table\_name

Deletes a specified table from the database.

**DROP Database** Database\_name

Deletes all tables including the database itself on the database server.

# Data Manipulation Language: Structure of Basic Elements

- Structure of the basic elements (clauses)
  - SELECT attribute(s)
  - FROM relation(s)
  - [ WHERE condition(s) ]
  - [ GROUP BY attribute(s) ]
  - [ ORDER BY attribute(s) ]
  
- Date Format, Strings and Numbers
  - Date Format: 'YYYY-MM-DD', e.g. '1974-12-31'
  - Strings: 'String', e.g. 'I like SQL'
  - Numbers: Number, e.g. 41 or 34.12

Note:           (1) Dates and Strings have to be enclosed by two apostrophes.  
                  (2) The clauses in [ ... ] are optional.

**SELECT**            \*   ←   *All columns*  
**FROM**             Products   ←   *table "Products"*  
**ORDER BY**        ID   ←   *order result by column "ID"*

| ID  | Product_name | Colour     | Article_no | Sale_price | Purchase_price | Stock | Items_sold | City      |
|-----|--------------|------------|------------|------------|----------------|-------|------------|-----------|
| 1   | Monitor 17"  | White      | 1297812542 | 399.00     | 249.99         | 50    | 134        | Frankfurt |
| 2   | Monitor 19"  | black      | 2457897145 | 499.00     | 379.00         | 12    | 289        | Berlin    |
| 3   | Monitor 17"  | black      | 1297467815 | 405.00     | 249.99         | 25    | 124        | Frankfurt |
| 4   | Monitor 19"  | white      | 2459871327 | 509.00     | 389.99         | 150   | 12         | Frankfurt |
| 5   | Monitor 20"  | black      | 2789441512 | 799.00     | 599.00         | 520   | 1052       | Berlin    |
| 6   | Monitor 20"  | white      | 2799151424 | 829.00     | 549.99         | 100   | 26         | Berlin    |
| 7   | Monitor 20"  | anthracite | 2764657527 | 819.00     | 589.99         | 50    | 127        | Nürnberg  |
| 8   | Monitor 21"  | anthracite | 2845161215 | 999.00     | 799.99         | 100   | 279        | Hamburg   |
| 9   | Monitor 24"  | white      | 2945712415 | 1299.00    | 945.00         | 25    | 124        | Berlin    |
| 10  | Monitor 24"  | black      | 2955745742 | 1350.00    | 956.00         | 450   | 1024       | Hamburg   |
| ... |              |            |            |            |                |       |            |           |

# DML: SELECT using WHERE

```

SELECT      *
FROM        Products
WHERE       Purchase_price > 500 AND City = 'Berlin'
    
```

| <u>ID</u> | Product_name | Colour | Article_no | Sale_price | Purchase_price | Stock | Sold_items | City   |
|-----------|--------------|--------|------------|------------|----------------|-------|------------|--------|
| 5         | Monitor 20"  | black  | 2789441512 | 799.00     | 599.00         | 520   | 1052       | Berlin |
| 6         | Monitor 20"  | white  | 2799151424 | 829.00     | 549.99         | 100   | 26         | Berlin |
| 9         | Monitor 24"  | white  | 2945712415 | 1299.00    | 945.00         | 25    | 124        | Berlin |

# DML: SELECT using ORDER BY

```
SELECT      ID, City, Stock  
FROM        Products  
ORDER BY   ID
```

| <u>ID</u> | City      | Stock |
|-----------|-----------|-------|
| 1         | Frankfurt | 50    |
| 2         | Berlin    | 12    |
| 3         | Frankfurt | 25    |
| 4         | Frankfurt | 150   |
| 5         | Berlin    | 520   |
| 6         | Berlin    | 100   |
| 7         | Nürnberg  | 50    |
| 8         | Hamburg   | 100   |
| 9         | Berlin    | 25    |
| 10        | Hamburg   | 450   |
| ...       | ...       | ...   |

# SELECT using SUM and GROUP BY

```
SELECT      City, SUM(Stock)
FROM        Products
GROUP BY    City
```

| City      | SUM(Stock) |
|-----------|------------|
| Frankfurt | 225        |
| Berlin    | 657        |
| Nürnberg  | 50         |
| Hamburg   | 550        |
| ...       |            |

## DML: SQL Functions: AVG, SUM, MIN, MAX, COUNT

- Further Aggregation Functions
  - AVG(x) returns the average value of x
  - SUM(x) returns the sum of x
  - MIN(x) returns the minimum value of x
  - MAX(x) returns the maximum value of x
  - COUNT(x) returns the number of values for x
  - STDDEV(x) returns the standard deviation of x.
- x denotes an array of values (e.g. as the result of a SELECT query).

**INSERT INTO**

Product\_Info (Article\_no, Weight,  
Resolution, Power\_consumption)

**VALUES**

(2689875627,6,'1280X1024',55)

Table “Product\_Info”

| <u>ID</u> | Article_no | Weight | Resolution | Power_consumption |
|-----------|------------|--------|------------|-------------------|
| 1         | 1297812542 | 4      | 1280X1024  | 26                |
| 2         | 2457897145 | 5      | 1280X1024  | 29                |
| 3         | 1297467815 | 4      | 1280X1024  | 27                |
| 4         | 2459871327 | 5.5    | 1280X1024  | 34                |
| 5         | 2789441512 | 8      | 1600X1280  | 53                |
| 6         | 2689875627 | 6      | 1280X1024  | 55                |
| ..        |            |        |            |                   |

Note: The “ID” column is a primary key and is automatically inserted with the new record.

**INSERT INTO  
VALUES**

**Product\_Info  
(2689875627,6,'1280X1024',55)**

**Table “Product\_Info”**

| <u>ID</u> | Article_No | Weight | Resolution | Power_Consumption |
|-----------|------------|--------|------------|-------------------|
| 1         | 1297812542 | 4      | 1280X1024  | 26                |
| 2         | 2457897145 | 5      | 1280X1024  | 29                |
| 3         | 1297467815 | 4      | 1280X1024  | 27                |
| 4         | 2459871327 | 5.5    | 1280X1024  | 34                |
| 5         | 2789441512 | 8      | 1600X1280  | 53                |
| 6         | 2689875627 | 6      | 1280X1024  | 55                |
| ..        |            |        |            |                   |

Note: The “ID” column is a primary key and is automatically inserted with the new record.

```

UPDATE      Product_Info
SET         Weight = 12
WHERE      Article_no = 2689875627
    
```

Table “Product\_Info”

| <u>ID</u> | Article_no | Weight | Resolution | Power_consumption |
|-----------|------------|--------|------------|-------------------|
| 1         | 1297812542 | 4      | 1280X1024  | 26                |
| 2         | 2457897145 | 5      | 1280X1024  | 29                |
| 3         | 1297467815 | 4      | 1280X1024  | 27                |
| 4         | 2459871327 | 5.5    | 1280X1024  | 34                |
| 5         | 2789441512 | 8      | 1600X1280  | 53                |
| 6         | 2689875627 | 12     | 1280X1024  | 55                |
| ..        |            |        |            |                   |

```

UPDATE      Product_Info
SET         Weight = 12, Resolution = '1800X1400'
WHERE      Article_no = 2689875627
    
```

Table “Product\_Info”

| <u>ID</u> | Article_no | Weight | Resolution | Power_consumption |
|-----------|------------|--------|------------|-------------------|
| 1         | 1297812542 | 4      | 1280X1024  | 26                |
| 2         | 2457897145 | 5      | 1280X1024  | 29                |
| 3         | 1297467815 | 4      | 1280X1024  | 27                |
| 4         | 2459871327 | 5.5    | 1280X1024  | 34                |
| 5         | 2789441512 | 8      | 1600X1280  | 53                |
| 6         | 2689875627 | 12     | 1800X1400  | 55                |
| ..        |            |        |            |                   |

```
DELETE FROM Product_Info
WHERE Article_no = 2689875627
```

## Table “Product\_Info”

| ID | Article_no | Weight | Resolution | Power_consumption |
|----|------------|--------|------------|-------------------|
| 1  | 1297812542 | 4      | 1280X1024  | 26                |
| 2  | 2457897145 | 5      | 1280X1024  | 29                |
| 3  | 1297467815 | 4      | 1280X1024  | 27                |
| 4  | 2459871327 | 5.5    | 1280X1024  | 34                |
| 5  | 2789441512 | 8      | 1600X1280  | 53                |
| .. |            |        |            |                   |

deleted:

|   |            |    |           |    |
|---|------------|----|-----------|----|
| 6 | 2689875627 | 12 | 1280X1024 | 55 |
|---|------------|----|-----------|----|

- Introduction
- Basic SQL Language Elements
- Advanced SQL Language Elements
- SQL-driven Database Features
- Issues of SQL
- SQL Tool: W3Schools

- JOINS are used to create links between two or more tables.
- Tables are associated with each other by using unique keys.
- A primary key (or unique key) is a column containing a unique value for each row in a table.
- Some types of JOINS are:
  - **INNER JOIN**
  - **OUTER JOIN**
  - **LEFT JOIN**
  - **RIGHT JOIN**

# Example Tables for JOIN and UNION

## Table “Products”

| <u>ID</u> | Product_name | Colour | Article_no | Sale_price | Purchase_price | Stock | Items_sold | City      |
|-----------|--------------|--------|------------|------------|----------------|-------|------------|-----------|
| 1         | Monitor 17“  | white  | 1297812542 | 399.00     | 249.99         | 50    | 134        | Frankfurt |
| 2         | Monitor 19“  | black  | 2457897145 | 499.00     | 379.00         | 12    | 289        | Berlin    |
| 3         | Monitor 17“  | black  | 1297467815 | 405.00     | 249.99         | 25    | 124        | Frankfurt |
| 4         | Monitor 19“  | white  | 2459871327 | 509.00     | 389.99         | 150   | 12         | Frankfurt |
| 5         | Monitor 20“  | black  | 2789441512 | 799.00     | 599.00         | 520   | 1052       | Berlin    |
| ..        |              |        |            |            |                |       |            |           |

## Table “Product\_Info”

| <u>ID</u> | Article_no | Weight | Resolution | Power_consumption |
|-----------|------------|--------|------------|-------------------|
| 1         | 1297812542 | 4      | 1280X1024  | 26                |
| 2         | 2457897145 | 5      | 1280X1024  | 29                |
| 3         | 2459871327 | 5.5    | 1280X1024  | 34                |
| 4         | 2789441512 | 8      | 1600X1280  | 53                |
| ..        |            |        |            |                   |

## SELECT statement using INNER JOIN

```
SELECT      Products.Product_Name, Product_Info.Weight  
FROM      Products INNER JOIN Product_Info ON  
            Products.Article_No = Product_Info.Article_No
```

| Product_name | Weight |
|--------------|--------|
| Monitor 17"  | 4      |
| Monitor 19"  | 5      |
| Monitor 19"  | 5.5    |
| Monitor 20"  | 8      |

**INNER JOIN** combines selected columns of two or more tables by linking them together using a unique key from each table (here **Article\_no**).

# UNION combines the results from two SELECT statements

```
SELECT ID, Article_no FROM Products  
UNION  
SELECT ID, Article_no FROM Product_Info
```

| <u>ID</u> | Article_no |
|-----------|------------|
| 1         | 1297812542 |
| 2         | 2457897145 |
| 3         | 1297467815 |
| 4         | 2459871327 |
| 5         | 2789441512 |

```
SELECT ID, Article_no FROM Products  
UNION ALL  
SELECT ID, Article_no FROM Product_Info
```

| <u>ID</u> | Article_no |
|-----------|------------|
| 1         | 1297812542 |
| 2         | 2457897145 |
| 3         | 1297467815 |
| 4         | 2459871327 |
| 5         | 2789441512 |
| 1         | 1297812542 |
| 2         | 2457897145 |
| 3         | 2459871327 |
| 4         | 2789441512 |

- **UNION** combines the results of two SQL SELECT queries to a single result set. For this, the same number of columns and compatible data types are required in each SELECT statement. Duplicate records are automatically removed unless **UNION ALL** is used.

- Introduction
- Basic SQL Language Elements
- Advanced SQL Language Elements
- SQL-driven Database Features
- Issues of SQL
- SQL Tool: W3Schools

- “Stored procedure”
  - constitutes a logical unit of multiple SQL statements saved in a database,
  - allows more complex statements and offers several programming language aspects (e.g. use of variables).

```
CREATE PROCEDURE product_weight  
AS  
SELECT          Products.Product_name, Product_Info.Weight  
FROM           Products INNER JOIN Product_Info ON  
                Products.Article_no = Product_Info.Article_no
```

| Product name | Weight |
|--------------|--------|
| Monitor 17"  | 4      |
| Monitor 19"  | 5      |
| Monitor 19"  | 5.5    |
| Monitor 20"  | 8      |

# Stored Procedures using Variables

Table “Product\_Info”

| <u>ID</u> | Article_no | Weight | Resolution | Power_consumption |
|-----------|------------|--------|------------|-------------------|
| 1         | 1297812542 | 4      | 1280X1024  | 26                |
| 2         | 2457897145 | 5      | 1280X1024  | 29                |
| 4         | 2459871327 | 5.5    | 1280X1024  | 34                |
| 5         | 2789441512 | 8      | 1600X1280  | 53                |
| 6         | 2689875627 | 6      | 1280X1024  | 55                |
| ..        |            |        |            |                   |

```
CREATE PROCEDURE  
add_product_info
```

```
@Article_no integer(10),  
@Weight decimal(2,1)  
@Resolution varchar(12),  
@Power_consumption integer(4)
```

```
AS
```

```
INSERT INTO Product_info (Article_no, Weight, Resolution,  
Power_consumption)  
VALUES (@Article_no, @Weight, @Resolution,  
@Power_consumption)
```

- Execution of Stored Procedures
  - The product\_weight procedure does neither accept nor require any input parameters:

```
exec product_weight
```

- The add\_detail procedure requires input parameters:
  - Variables are used to pass data to the procedure:

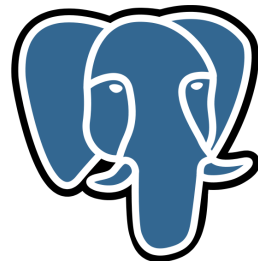
```
exec add_product_info 2689875627, 6, '1280X1024', 55
```

- A database *trigger* is a stored procedure which is **automatically executed** in case predefined events occur within in a database.
- Typical Trigger events are the insertion, update or deletion of data sets.

- Triggers can be used to:
  - enforce business rules (e.g. verify that every invoice has at least one item);
  - replicate data (e.g. create a history record for every data modification, which can be transferred to a data warehouse later);
  - enhance database performance (e.g. update account balance after every transaction for faster queries);
  - maintain the integrity of information in the database;
  - log data modifications (e.g. add time-stamp from server clock);
  - ...

- Introduction
- Basic SQL Language Elements
- Advanced SQL Language Elements
- SQL-driven Database Features
- Issues of SQL
- SQL Tool: W3Schools

- Compatibility issues between different manufacturer implementations of the SQL standard



- Introduction
- Basic SQL Language Elements
- Advanced SQL Language Elements
- SQL-driven Database Features
- Issues of SQL
- SQL Tool: W3Schools

- Created in 1998 as a school for web developers, offering free courses and exercises on web development.
- No download required, available under <https://www.w3schools.com/sql/>.
- Will be used in this course.

**SQL Statement:**

```
SELECT * FROM Customers;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL >

**Result:**

Number of Records: 91

| CustomerID | CustomerName                       | ContactName    | Address                       | City        | PostalCode | Country |
|------------|------------------------------------|----------------|-------------------------------|-------------|------------|---------|
| 1          | Alfreds Futterkiste                | Maria Anders   | Obere Str. 57                 | Berlin      | 12209      | Germany |
| 2          | Ana Trujillo Emparedados y helados | Ana Trujillo   | Avda. de la Constitución 2222 | México D.F. | 05021      | Mexico  |
| 3          | Antonio Moreno Taquería            | Antonio Moreno | Mataderos 2312                | México D.F. | 05023      | Mexico  |
| 4          | Around the Horn                    | Thomas Hardy   | 120 Hanover Sq.               | London      | WA1 1DP    | UK      |

**Your Database:**

| Tablenames   | Records |
|--------------|---------|
| Customers    | 91      |
| Categories   | 8       |
| Employees    | 10      |
| OrderDetails | 518     |
| Orders       | 196     |
| Products     | 77      |
| Shippers     | 3       |
| Suppliers    | 29      |

- Alan Beaulieu (2009) Einführung in SQL, O'Reilly.

