X THE PLACE OF USEFUL LEARNING

University of Strathclyde Glasgow

Database Fundamentals – CS990

Database and Web Systems Development - CS952

Image source: https://www.strath.ac.uk/branding/

CS990/CS952
Database Fundamentals

Creating tables with SQL

Course Content

1. Introduction to Relational Databases *(Introduction + Relational Model)*
2. Data Modelling - *(Entity Relationship Modelling + The Enhanced Entity Relationship Model)*
3. Database Design and SQL - *(Logical modelling + Introduction to SQL)*
4. Further SQL - *(Advanced SQL queries + Creating tables with SQL)*
5. Normalisation - *(Normalisation to second normal form + Third normal form)*

Contents

- Create a table (relation)
- Specify keys and relations
- Empty and drop tables

Introduction

- SQL is a declarative language for manipulating a relational database
- Use it to issue commands to the database for tasks such as:
  - Creating and managing tables
  - Inserting data into tables
  - Deleting data and tables

Creating a Table

- The simplest form of SQL **CREATE TABLE** looks like:

```
CREATE TABLE tablename
     (colname datatype,
      ... )

CREATE TABLE Staff
     (Sno   NUMBER,
     Sname VARCHAR2(20),
     Dept  VARCHAR2(20),
     Grade VARCHAR2(7));
```

Data Types

- Data types include:
  - **VARCHAR2** variable length text strings
  - **NUMBER** numbers
  - **FLOAT** numeric values, including floating-point numbers
  - **DATE** full date (yyyy-mm-dd) TO_DATE('2022-03-09', 'YYYY-MM-DD')
    - Dates have a default form of DD-MON-YY
    - EG: '09-MAR-17' The apostrophes are required (').

https://docs.oracle.com/database/121/SQLRF/sql_elements001.htm#SQLRF30020

Table Constraints

- The specification of a column can include some extras:
  - ***default value*** (used if an insertion doesn't supply a value)

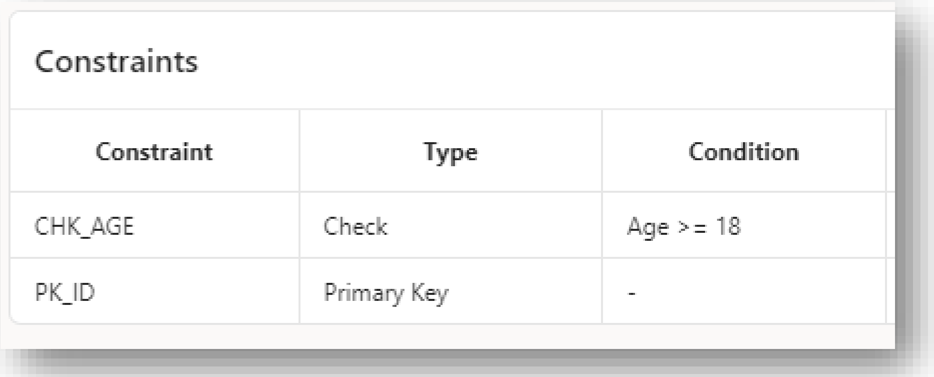
    -
      ```
      CREATE TABLE Employees (
              EmployeeID Number PRIMARY KEY,
              Name VARCHAR2(50),
              Department VARCHAR2(50) DEFAULT 'Unknown’);
      ```

    -
      ```
      INSERT INTO Employees (EmployeeID, Name) VALUES (1, 'John
      Will');
      ```
  - and a ***column constraint*** (next slide)

    -
      ```
      CREATE TABLE Example (
              ID NUMBER PRIMARY KEY,
              Name VARCHAR(50) NOT NULL);
      ```
- We can add ***table constraint(s)*** before the closing bracket

Table Constraints Cont…

```
CREATE TABLE Example (
    ID NUMBER,
    Name VARCHAR2(50),
    Age NUMBER,
    CONSTRAINT PK_ID PRIMARY KEY (ID),
    CONSTRAINT CHK_Age CHECK (Age >= 18));
```



Constraints

Constraint	Type	Condition
CHK_AGE	Check	Age >= 18
PK_ID	Primary Key	-

The **CHECK** constraint ensures that all values in a column satisfy a specific condition. If a value violates the condition, the database will reject the insert or update operation.

Data Integrity

- Column constraints
  - enforcing entity and referential integrity
    - **NOT NULL | NULL]**
    - **DEFAULT default_value]**
    - **AUTO_INCREMENT]**
    - **UNIQUE [KEY] | [PRIMARY] KEY]**
    - **COMMENT 'string’]**
    - **PRIMARY KEY (only one per table)**
    - **FOREIGN KEY REFERENCES *table (column)***

Example

```
CREATE TABLE Staff (
        Sno NUMBER (5) PRIMARY KEY,    -- primary key
        Sname VARCHAR2(20) NOT NULL );

CREATE TABLE Staff (
    Sno NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,-- autogenerated PK
    Sname VARCHAR2(20) NOT NULL );
CREATE TABLE Staff (
    Sno NUMBER(5),
    Sname VARCHAR2(20) NOT NULL,
    CONSTRAINT PK_Staff_Sno PRIMARY KEY (Sno)); -- Naming the PK constraint

CREATE TABLE Staff (
    Sno NUMBER(5) GENERATED ALWAYS AS IDENTITY, -- Auto-generated PK
    Sname VARCHAR2(20) NOT NULL,
    CONSTRAINT PK_Staff_Sno PRIMARY KEY (Sno)); -- Named PK constraint
```

Data Integrity: Foreign Keys

```
CREATE TABLE Staff
  ( ... other columns go here
  Dept VARCHAR2(20) FOREIGN KEY REFERENCES Depts(Dname),
  Grade VARCHAR2(7) FOREIGN KEY REFERENCES Paytable);


CREATE TABLE Staff (
    -- Other columns go here
    Dept VARCHAR2(20),
    Grade VARCHAR2(7),
    CONSTRAINT FK_Staff_Depts FOREIGN KEY (Dept) REFERENCES
  Depts(Dname), -- FOREIGN KEY
    CONSTRAINT FK_Staff_Paytable FOREIGN KEY (Grade)
  REFERENCES Paytable(Grade) -- FOREIGN KEY
);
```

It enforces **referential integrity**, ensuring that the value in the foreign key column must exist in the referenced table.

Data Integrity: Cascaded Deletion

- We can add `ON DELETE CASCADE` to `REFERENCES`
- This means that if a row in the other table is deleted, all matching rows in this table should be deleted too.

- For example, a `Dependant` table (for the dependants of employees) might declare the column:

```
Enum REFERENCES Employee ON DELETE CASCADE
```

- So, if we delete employee 123 from the `Employee` table, then all their dependants are deleted from the `Dependant` table, thus protecting referential integrity.

- We should only do this for **weak entities**!

Data Integrity: Composite PK

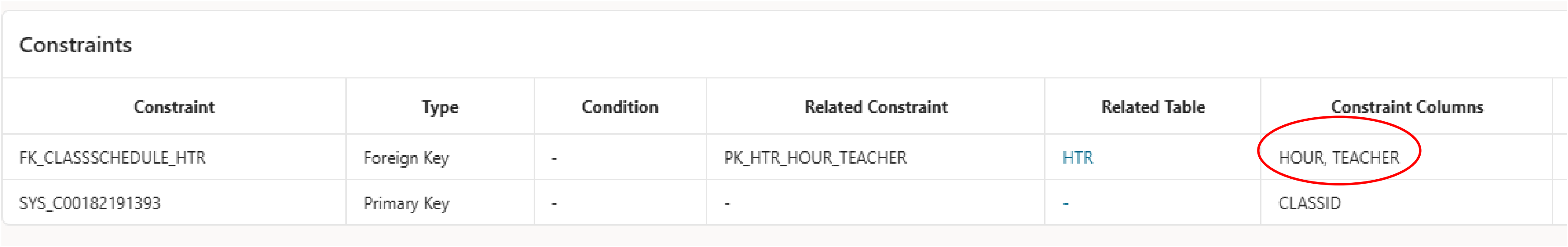
- After the last field, we can add *table-constraints*
  - these look like column-constraints, but they can reference more than one column

```
CREATE TABLE HTR (
    Hour VARCHAR2(6),
    Teacher VARCHAR2(3),
    Room VARCHAR2(4),
    CONSTRAINT PK_HTR_Hour_Teacher PRIMARY KEY
  (Hour, Teacher) -- composite primary
);
```

  - this is how to declare **composite primary keys**

Foreign Key (Composite PK)

```
CREATE TABLE ClassSchedule (
    ClassID NUMBER PRIMARY KEY,     -- PK for the new table
    Hour VARCHAR2(6), -- Part of the foreign key
    Teacher VARCHAR2(3), -- Part of the foreign key
    Room VARCHAR2(4),
    CONSTRAINT FK_ClassSchedule_HTR FOREIGN KEY (Hour, 
Teacher) REFERENCES HTR (Hour, Teacher) -- References the 
composite primary key of HTR
);
```



Constraints

Constraint	Type	Condition	Related Constraint	Related Table	Constraint Columns
FK_CLASSSCHEDULE_HTR	Foreign Key	-	PK_HTR_HOUR_TEACHER	HTR	HOUR, TEACHER
SYS_C00182191393	Primary Key	-	-	-	CLASSID

Table Constraints (continued)

```
CREATE TABLE Staff (
    Sno   NUMBER PRIMARY KEY,
    Sname VARCHAR2(20) NOT NULL,
    Dept  VARCHAR2(20),
    Grade VARCHAR2(7),
    CONSTRAINT fk_Dept FOREIGN KEY (Dept) REFERENCES Depts (Dname),
    CONSTRAINT fk_Paytable FOREIGN KEY (Grade) REFERENCES Paytable
   (Grade));
```

This could be written as:

```
CREATE TABLE Staff (
    Sno   NUMBER,
    Sname VARCHAR2(20) NOT NULL,
    Dept  VARCHAR2(20),
    Grade VARCHAR2(7),
    PRIMARY KEY(Sno), -- or CONSTRAINT pk_Staff PRIMARY KEY (Sno),
    CONSTRAINT fk_Dept FOREIGN KEY (Dept) REFERENCES Depts (Dname),
    CONSTRAINT fk_Paytable FOREIGN KEY (Grade) REFERENCES Paytable
   (Grade));
```

Altering an Existing Table

- We can change tables, using **ALTER TABLE**, even after they contain data
- Amongst other possibilities, we can add or modify columns

```
ALTER TABLE Staff ADD -- Add new columns to the 'Staff' table
    (StreetAddress VARCHAR2(20),
    TownAddress    VARCHAR2(20));


ALTER TABLE Staff MODIFY
    (TownAddress DEFAULT Glasgow');
-- Modify the 'TownAddress' column to set a default value
```

Dropping and Deleting

- We can completely remove a table: both its data (if any) and its definition

```
DROP TABLE tablename;
DROP TABLE tablename CASCADE CONSTRAINTS ;
```

  – the second form removes Foreign Key constraints in associated tables (which otherwise could not be updated)

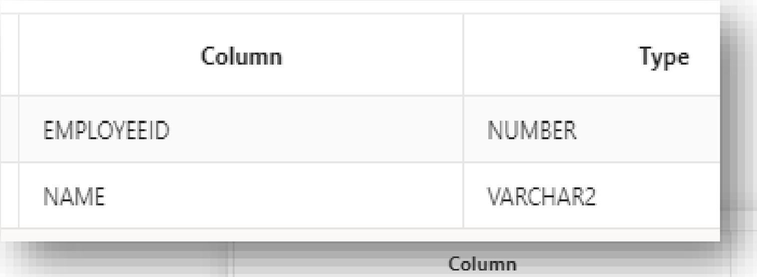
- Removing the data alone (not the definition):

```
DELETE FROM tablename;
DELETE FROM tablename WHERE condition ;
```

Examples next slide 👇

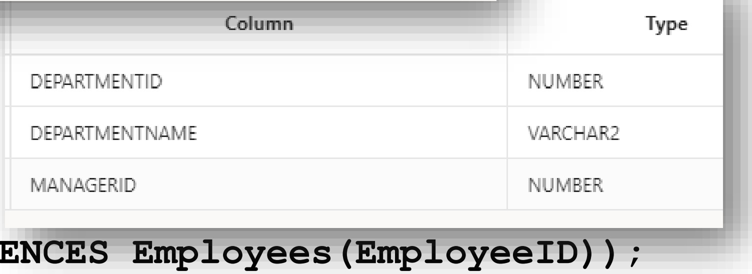
Example: CASCADE CONSTRAINTS

```
CREATE TABLE Employees (
    EmployeeID NUMBER PRIMARY KEY,
    Name VARCHAR2(50));
```



Column	Type
EMPLOYEEID	NUMBER
NAME	VARCHAR2

```
CREATE TABLE Departments (
    DepartmentID NUMBER PRIMARY KEY,
    DepartmentName VARCHAR2(50),
    ManagerID NUMBER,
    CONSTRAINT FK_Manager FOREIGN KEY (ManagerID) REFERENCES Employees(EmployeeID));
```



Column	Type
DEPARTMENTID	NUMBER
DEPARTMENTNAME	VARCHAR2
MANAGERID	NUMBER

In this case, to drop the ‘Employees’ table, you have to use DROP TABLE ... CASCADE CONSTRAINTS as below:

```
DROP TABLE Employees
        CASCADE CONSTRAINTS; -- Drop the 'Employees' table and remove all related constraints
```

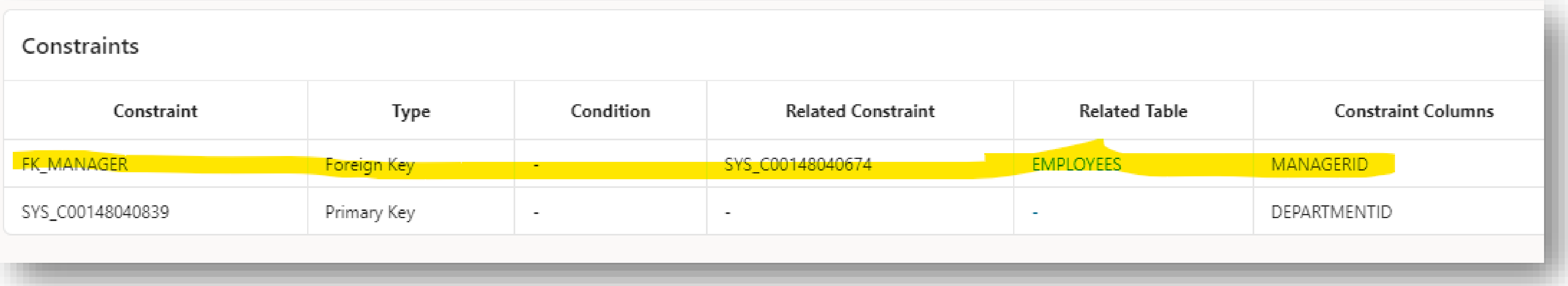
Otherwise, if you don’t use …CASCADE CONSTRAINTS; the below error will be prompted:

ORA-02449: unique/primary keys in table referenced by foreign keys

This statement with (CASCADE CONSTRAINTS) will remove the Employees table from the database and also remove any foreign key constraints in other tables that reference the Employees table. Specifically, it will delete the FK_Manager constraint from the Departments table.

Example: CASCADE CONSTRAINTS

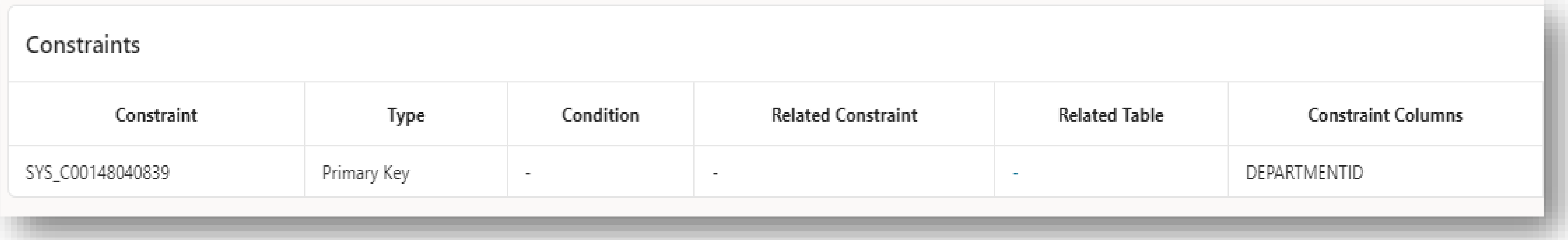
Below is the list of constraints from ‘**DEPARTMENTS’ table** before dropping the Table ‘**Employees’**



Constraints

Constraint	Type	Condition	Related Constraint	Related Table	Constraint Columns
FK_MANAGER	Foreign Key	-	SYS_C00148040674	EMPLOYEES	MANAGERID
SYS_C00148040839	Primary Key	-	-	-	DEPARTMENTID

The FK_Manager constraint has been deleted from the **DEPARTMENTS’** table after dropping the Table ‘**Employees’.**



Constraints

Constraint	Type	Condition	Related Constraint	Related Table	Constraint Columns
SYS_C00148040839	Primary Key	-	-	-	DEPARTMENTID

Getting Data into Tables

- There are two ways of using SQL to get data into tables
- Firstly, with the values in the SQL statement

```
    INSERT INTO Staff VALUES
      (123, 'Lee', 'CompSci', 'II.7');
```

  - if we are not loading all the columns, use this form:

```
    INSERT INTO Staff (Sno, Sname) VALUES
        (456, 'Waldenstein');
```

- Secondly, by extracting the data from existing tables

```
    INSERT INTO Loan
        SELECT DISTINCT
               Sno,
               Bno,
               Date_out
        FROM Staff_Borrower;
```

Getting Even More Data In…

```
INSERT INTO Books (BName ,BNumber)
VALUES ('book1', '1'), ('book2', '2'), ('book3', ‘3’);
-- will this work in Oracle?
```

While using INSERT ALL you need a select statement to retrieve records from another table.

NO!

```
INSERT ALL
      INTO Books(BName, BNumber) VALUES ('book1', '1’)
      INTO Books(BName, BNumber) VALUES ('book2', '2’)
SELECT * FROM dual;
```

dual is a special Oracle table with one row and one column, which is often used for operations like this when you don't need to query any actual data but just need a dummy SELECT for executing an INSERT ALL statement.

More examples on this link:

https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/INSERT.html#GUID-903F8043-0254-4EE9-ACC1-CB8AC0AF3423

Triggers

Integrity constraints can be used to control the way that tables respond to inserted data. However, it is sometimes necessary to write more complex commands than can easily be represented as integrity constraints. This can be achieved using embedded code in the form of triggers.

These are special type of stored procedure in a database that is automatically executed (or "triggered") when a specific event occurs.

Example 1 Triggers

```
-- Creating or replacing a trigger named 'update_order_date'
CREATE OR REPLACE TRIGGER update_order_date
-- This trigger fires BEFORE any UPDATE operation on the 'orders' table
BEFORE UPDATE ON orders
-- The trigger executes once for each row that is updated
FOR EACH ROW
BEGIN
-- Sets 'order_date' to the current date on update
    :NEW.order_date := SYSDATE; -- Automatically updates order_date
END;
/
```

Example 2: Triggers

```
CREATE TABLE CUSTOMERV5 (
    CUST_NUM  CHAR(10)
        CONSTRAINT PK_CUSTV5 PRIMARY KEY,
    CUST_NAME CHAR(30)
        CONSTRAINT CHECK_NV5 CHECK ( LENGTH(REPLACE(
            TRANSLATE(                       --Replaces each letter with 'X'
                UPPER(CUST_NAME),
                'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
                'XXXXXXXXXXXXXXXXXXXXXXXXXX'
            ),
            'X',
            ''
        )) = 0 ),
    ADDRESS   CHAR(30),
    CR_LIMIT  NUMBER
);
```

Example 3: Triggers

```
CREATE TABLE ORDERV5 (
    ORDER_NO   CHAR(10)
        CONSTRAINT PK_NUMV5 PRIMARY KEY,
    CUST_NUM   CHAR(10)
        CONSTRAINT CUSTREFV5
            REFERENCES CUSTOMERV5 ( CUST_NUM ),
    ORDER_DATE DATE,
    VALUE      NUMBER(9, 2)
        CONSTRAINT NO_NL_VALV5 NOT NULL
        CONSTRAINT CHECK_VLV5 CHECK ( VALUE > 0 ));
```

Insert or update triggers

The code in the trigger is executed before data is inserted or updated and it may result in the new data being rejected.

```
CREATE TRIGGER ORDER_VAL_LIMITSV5
BEFORE INSERT OR UPDATE OF VALUE
ON ORDERV5
FOR EACH ROW
DECLARE
       LIMIT NUMBER;
       CREDITLIMIT NUMBER
BEGIN
    SELECT CR_LIMIT
    INTO CREDITLIMIT
    FROM CUSTOMERV5
    WHERE CUSTOMERV5.CUST_NUM = :new.CUST_NUM;

    IF (:new.VALUE > CREDITLIMIT )THEN raise_application_error (-20601,:new.VALUE
    ||'is over credit limit’);
    END IF;
END;
```

Inserting: violates the credit limit

```
INSERT INTO ORDERV5 (
    ORDER_NO,
    CUST_NUM,
    ORDER_DATE,
    VALUE)
VALUES ( '1234567890',
           '1234567890',
           SYSDATE,
           1500 );
```

Changing Data in a Table

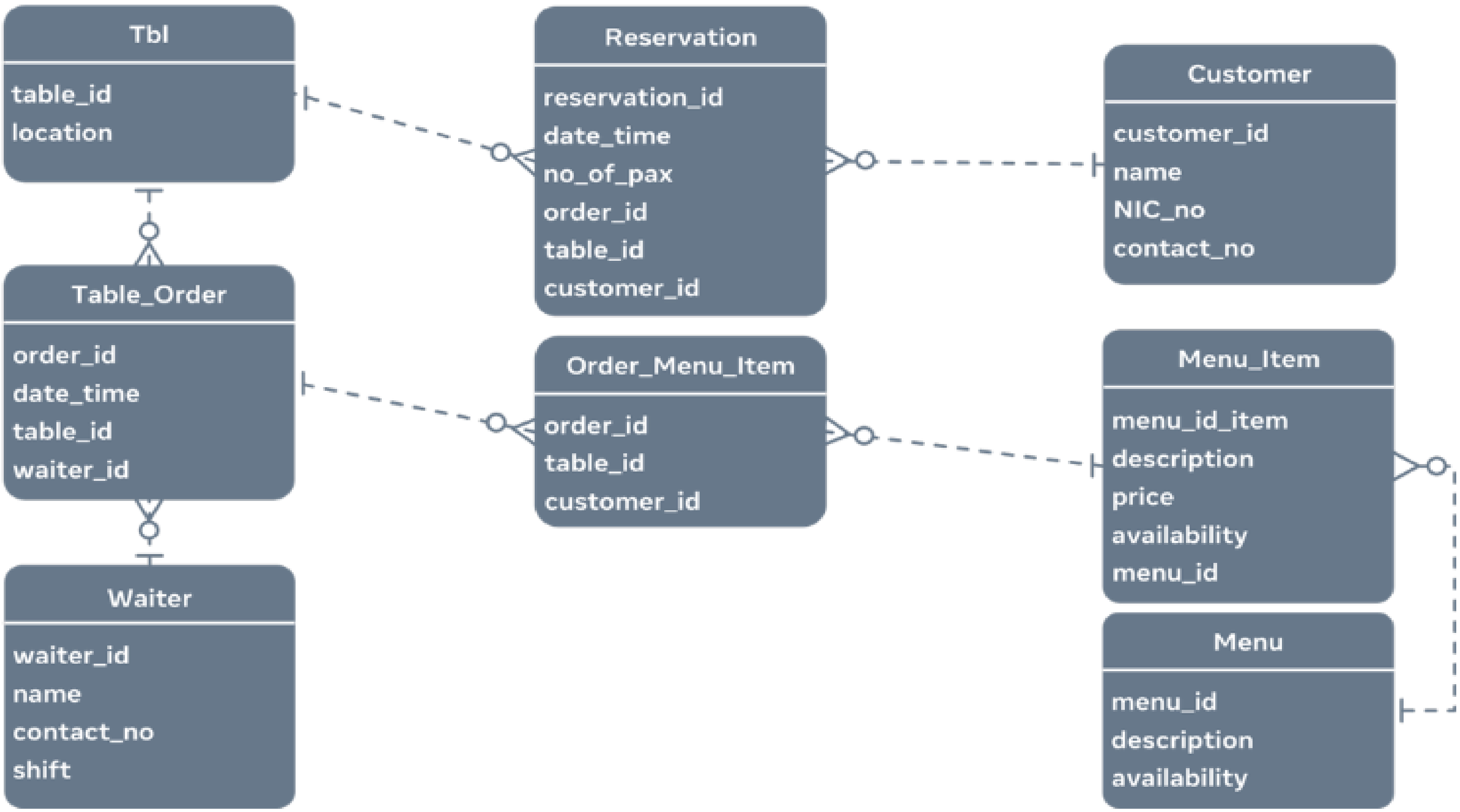
```
UPDATE table
      SET field=value, field=value WHERE
      condition;
```

```
UPDATE ORDER
      SET O_DATE = ‘09-MAR-17’
      WHERE ORDER_NUMBER = 12211;
```

Database schema

- A blueprint of how data in a database will look and be stored.
- First step in database design
- Building a simple database schema
- Schema objects:
  - *tables, columns and relationships, Data types, views, stored procedures, primary keys and foreign keys etc*

Restaurant booking scenario
ERD
Logical Schema



Tbl

table_id
location

Reservation

reservation_id
date_time
no_of_pax
order_id
table_id
customer_id

Customer

customer_id
name
NIC_no
contact_no

Table_Order

order_id
date_time
table_id
waiter_id

Order_Menu_Item

order_id
table_id
customer_id

Menu_Item

menu_id_item
description
price
availability
menu_id

Waiter

waiter_id
name
contact_no
shift

Menu

menu_id
description
availability

Physical database schema

```
CREATE TABLE tbl (
    table_id NUMBER(10),
    location VARCHAR2(255),
    CONSTRAINT pk_tbl_table_id PRIMARY KEY (table_id)
);


CREATE TABLE waiter (
    waiter_id NUMBER(10),
    name VARCHAR2(150),
    contact_no VARCHAR2(10),
    shift VARCHAR2(10),
    CONSTRAINT pk_waiter_waiter_id PRIMARY KEY (waiter_id)
);
```

Table order

```
CREATE TABLE table_order (
    order_id NUMBER(10),
    date_time TIMESTAMP,
    table_id NUMBER(10),
    waiter_id NUMBER(10),
    CONSTRAINT pk_table_order_order_id PRIMARY KEY (order_id),
    CONSTRAINT fk_table_order_table_id FOREIGN KEY (table_id)
REFERENCES tbl (table_id),
    CONSTRAINT fk_table_order_waiter_id FOREIGN KEY
(waiter_id) REFERENCES waiter (waiter_id)
);
```

Table: customer, reservation

```
CREATE TABLE customer (
    customer_id NUMBER(10),
    name VARCHAR2(100),
    NIC_no VARCHAR2(12),
    contact_no VARCHAR2(10),
    CONSTRAINT pk_customer_customer_id PRIMARY KEY (customer_id));
```

```
CREATE TABLE reservation(
    reservation_id NUMBER,
    date_time TIMESTAMP,
    no_of_pax NUMBER,
    order_id NUMBER,
    table_id NUMBER,
    customer_id NUMBER,
    PRIMARY KEY (reservation_id),
    FOREIGN KEY (order_id) REFERENCES table_order(order_id),
    FOREIGN KEY (table_id) REFERENCES tbl(table_id),
    FOREIGN KEY (customer_id) REFERENCES customer(customer_id)
);
```

Table: menu_item and menu

```
 CREATE TABLE menu (
     menu_id NUMBER,
     description VARCHAR2(255),
     availability NUMBER,
     CONSTRAINT pk_menu_menu_id PRIMARY KEY (menu_id)
 );
```

```
CREATE TABLE menu_item (
    menu_item_id NUMBER,
    description VARCHAR2(255),
    price NUMBER,
    availability NUMBER,
    menu_id NUMBER,
    CONSTRAINT pk_menu_item_menu_item_id PRIMARY KEY (menu_item_id),
    CONSTRAINT fk_menu_item_menu_id FOREIGN KEY (menu_id) REFERENCES
    menu(menu_id)
);
```

Table: order_menu_item

```
CREATE TABLE order_menu_item (
    order_id NUMBER,
    menu_item_id NUMBER,
    quantity NUMBER,
   CONSTRAINT pk_order_menu_item PRIMARY KEY (order_id, menu_item_id),
   CONSTRAINT fk_order_menu_item_order_id FOREIGN KEY (order_id) REFERENCES
      table_order(order_id),
   CONSTRAINT fk_order_menu_item_menu_item_id FOREIGN KEY (menu_item_id)
      REFERENCES menu_item(menu_item_id)
);
```

Course Content

1. Introduction to Relational Databases *(Introduction + Relational Model)*
2. Data Modelling - *(Entity Relationship Modelling + The Enhanced Entity Relationship Model)*
3. Database Design and SQL - *(Logical modelling + Introduction to SQL)*
4. Further SQL - *(Advanced SQL queries + Creating tables with SQL)*
5. Normalisation - *(Normalisation to second normal form + Third normal form)*

Thank you

X THE PLACE OF USEFUL LEARNING

University of Strathclyde Glasgow

Database Fundamentals – CS990

Database and Web Systems Development - CS952

Image source: https://www.strath.ac.uk/branding/