

**TECHNOLOGICAL UNIVERSITY DUBLIN**  
**Grangegorman**

---

TU856 - BSc. (Honours) Degree in Computer Science  
TU858 - BSc. (Honours) Degree in Computer Science  
International

*Year 2*

---

*WINTER EXAMINATIONS 2022/23*

---

***CMPU 2007 Databases 1***

**Internal Examiner(s):**  
Dr. Mariana Rocha  
Dr. Paul Doyle

**External Examiner(s):**  
Ms. Sanita Tifentale (TU856/TU858)

**Instructions To Candidates:** Read the case study and answer ALL questions.  
There is a syntax table on the last page to assist you.

**Exam Duration:** Two hours

### **Case Study – Video games database**

The following relational schema and interpretation will be used in subsequent questions:

```
game_publisher (publisher id, publisher_name, publisher_email)
genre (genre id, genre_name)
game (game id, game_name, genre_id, publisher_id)
platform (platform id, platform_name, developer_name)
game_platform (platform id, game id, release_year)
sale (platform id, publisher id, number_of_sales)
```

Note: underlined attributes indicate how each instance of a relation is identified.

A group of game developers decided to design a database to retrieve some information about video game sales. Details about the games, their publishers, and what platforms can be used to play these games were retrieved from a dataset, besides information about how much each publisher sold on each platform per year.

Details about the publishers are stored in the `game_publisher` table, identified by a unique identifier (`publisher_ID`), and including the publisher's name (`publisher_name`) and email (`publisher_email`).

The table `genre` contains details about the game's genre, including a unique identifier (`genre_id`) and a name (`genre_name`).

Details about the games are stored in the table `game`, which includes a unique identifier (`game_id`), the name of the game (`game_name`), the game genre (`genre_id`) and the id of who published that game (`publisher_id`).

The table `platform` includes data about the type of platform that can be used to play the game, such as a unique identifier (`platform_id`), the name of the platform (`platform_name`), and the name of the company that developed that platform (`developer_name`).

The table `game_platform` includes details about when each game was released to be played on each platform, including the platform identifier (`platform_id`), the game identifier (`game_id`) and the year of release (`release_year`). Each row in this table is uniquely identified by a combination of the `platform_id` and the `game_id`.

Finally, the table `sale` includes details about how much each publisher sold according to the platform, with details like the platform identifier (`platform_id`), the publisher identifier (`publisher_id`), and the number of sales (`number_of_sales`). Each row on this table is uniquely identified by a combination of the `platform_id` and the `publisher_id`.

**1. (a)** Draw an Entity Relationship diagram (ERD) for the case study and clearly explain the following concepts, offering an example of each:

- Entity
- Attribute
- Relationship
- Primary Key
- Foreign Key

(20 marks)

**(b)** Suppose the game, platform and genre tables have been created in an Oracle database with the attributes and datatypes identified in the model given in question 1, part (a).

Write the SQL needed to add the following value constraints to the tables using the ALTER statements:

- The first character of the game\_id must be one of the following: P, L, D, A;
- Publishers' email addresses must include both the @ symbol and the . symbol;
- The last digit of a platform\_id must be between 1 and 6;
- The first character of genre\_id must be between A and Z;
- The genre\_name must be one of the following values: adventure, RPG, simulation, puzzle.

(15 marks)

**(c)** The following SQL queries were written as an attempt to insert data into the video game database. However, each query returned an error. Identify and explain the errors in the queries below:

```
INSERT INTO game_publisher VALUES ('Pewter Games', 'pewter@email.com');
```

```
INSERT INTO game VALUES ('1453', 5555, Super Mario, 1234);
```

```
INSERT INTO platform (platform_id, platform_name, developer_name) VALUES (4321, 'Xbox', 'Brain Games Studio', 'Mabel Addis');
```

(5 marks)

2. Suppose you have created and altered the tables for the case study considering the constraints added in part 1b, and data has been inserted into the tables as follows:

**genre**

genre_id	genre_name
A1235	RPG
B1237	Simulation
C1238	Puzzle
D1234	Adventure

**platform**

platform_id	platform_name	developer_name
5691	Xbox	Microsoft
9876	Steam	Valve Corporation
6782	Playstation 5	Sony
7683	Switch	Nintendo

**game\_publisher**

publisher_id	publisher_name	publisher_email
4321	Devolver Digital	devolver@email.com
5321	Digital Dragon	digital@email.com
6321	Metacore Games	metacore@email.com
8321	Amanita Design	amanita@email.com

**game**

game_id	genre_id	game_name	publisher_id
P1212	C1238	Gris	4321
L2121	B1237	Memoranda	5321
D3131	D1234	Merge Manson	6321
A1313	A1235	Machinarium	8321

**game\_platform**

platform_id	game_id	release_year
9876	P1212	2020
6782	L2121	2018
7683	D3131	2019
5691	A1313	2009

**sale**

platform_id	publisher_id	number_of_sales
9876	4321	500
6782	5321	199
7683	6321	543
5691	8321	233

(Questions 2(a), 2(b), and 2(c) are presented on the next page).

**2. (cont.)**

- (a)** Considering the relations between the tables game, game\_publisher and sale, write the SQL query to retrieve the game\_name, genre\_id, publisher\_name, and number\_of\_sales for games published by that publisher.

(12 marks)

- (b)** Rewrite the query you provided in question 2 (a) but modify the output by:

- Renaming the columns as “Game Title”, “Genre Identifier”, “Name of the publisher”, “Sales”.
- Display game\_name and publisher\_name in lowercase

(10 marks)

- (c)** The team of developers identified a new list of publishers, as seen below:

**game\_publisher**

<b>publisher_id</b>	<b>publisher_name</b>	<b>publisher_email</b>
5055	Blue Dragon	<a href="mailto:blue@email.com">blue@email.com</a>
4500	Major Galaxy	<a href="mailto:major@email.com">major@email.com</a>
3245	Fabulous Magic	<a href="mailto:fabulous@email.com">fabulous@email.com</a>
4532	Super Studio	(null)

Provide the SQL code used to populate the table game\_publisher with the new data.

(8 marks)

**(Question 3 is presented on the next page.)**

**3 (a)** Write the SQL needed to retrieve the id, name and email of each game publisher. The publisher's name should be displayed in uppercase. If an email address has no value, then "Unknown" should be output. Present the output in descending order of publisher id.

(8 marks)

**(b)** The developers who are working on this database are experts in simulation games. After adding more data to the tables, they were curious to know how many simulation games were available. Write an SQL query to calculate how many games of the genre "Simulation" are stored in the database. You must display the output following the format:

**There are [X] simulation games on the database**

[x] is a placeholder to represent the number of simulation games available.

(5 marks)

**(c)** Now, the developers wanted to check how many games of each genre Simulation, RPG, Adventure and Puzzle are available in the database. Write an SQL query to display a table that:

- Displays the name of each genre (1 per row), in a column named "Game genre"
- Displays the number of games per genre in a column named "Number of games"
- The column "Number of games" should be displayed in descending order

Hints: you will need to access **two** tables to retrieve this data. Using the SQL function **count()** will be helpful to answer this question.

(8 marks)

**(d)** To implement an efficient and consistent database, some integrity rules should be followed. Explain each of the following integrities and give examples of how these were followed in the video game database presented in question 1.

- Entity integrity
- Domain integrity
- Referential integrity

(9 marks)

# SYNTAX TABLE

```
ALTER TABLE [schema.]table column_clauses;
column_clauses:
    ADD (column datatype [DEFAULT expr] [column_constraint(s)] [, ...])
    DROP COLUMN column
        [CASCADE CONSTRAINTS] [INVALIDATE] CHECKPOINT int
    MODIFY column datatype [DEFAULT expr] [column_constraint(s)]
    RENAME COLUMN column TO new_name

ALTER TABLE [schema.]table constraint_clause [, ...];
constraint_clause:
    DROP PRIMARY KEY [CASCADE] [{KEEP|DROP} INDEX]
    DROP UNIQUE (column [, ...]) [{KEEP|DROP} INDEX]
    DROP CONSTRAINT constraint [CASCADE]
    MODIFY CONSTRAINT constraint constraint_state
    MODIFY PRIMARY KEY constraint_state
    MODIFY UNIQUE (column [, ...]) constraint_state
    RENAME CONSTRAINT constraint TO new_name

CREATE TABLE table (
    column datatype [DEFAULT expr] [column_constraint(s) [, ...]] [, column
datatype [, ...]]
    [table_constraint [, ...]])
COMMIT
DELETE FROM tablename WHERE condition
DROP [TABLE tablename|DROP VIEW viewname]
INSERT INTO tablename (column-name-list) VALUES (data-value-list)
ROLLBACK
SELECT [DISTINCT] select_list
    FROM table_list
    [WHERE conditions]
        [GROUP BY group_by_list]
        [HAVING search_conditions]
        [ORDER BY order_list [ASC | DESC] ]
```

# SYNTAX TABLE

**Conditions**: =, >, <, >=, <=, <>, BETWEEN .. AND .., IN (list), IS NULL, IS NOT NULL,  
LIKE

**Logical operators**: AND, OR, NOT

**Set operations**: UNION, MINUS, INTERSECT

**CASE [ expression ]**

```
WHEN condition_1 THEN result_1  
WHEN condition_2 THEN result_2  
WHEN condition_n THEN result_n  
ELSE result
```

**END**

**SELECT**

```
... FROM table1 LEFT JOIN table2  
      ON table1.field1 compopr table2.field2 | USING clause  
... FROM table1 RIGHT JOIN table2  
      ON table1.field1 compopr table2.field2 | USING clause  
... FROM table1 INNER JOIN table2  
      ON table1.field1 compopr table2.field2 | USING clause
```

**Key**

*table1, table2*      The tables from which records are combined.  
*field1, field2*      The fields to be joined.  
*compopr*              Any relational comparison operator: = < > <= >= or <>

**UPDATE tablename**

[SET *column-name*= <*data-value*>] [WHERE *condition*]

*Column-definition* = *column-name* [CHAR [(*n*)] | VARCHAR2(*n*) | NUMBER [*n,p*] |  
DATE | DATETIME] {[NOT NULL | UNIQUE | PRIMARY KEY]}

**Oracle Functions**

Null Handling Functions: NVL, NVL2, NULLIF, COALESCE, CASE, DECODE.

Case Conversion functions - Accepts character input and returns a character value: UPPER, LOWER and INITCAP.

Character functions - Accepts character input and returns number or character value: CONCAT, LENGTH, SUBSTR, INSTR, LPAD, RPAD, TRIM and REPLACE.

Date functions - Date arithmetic operations return date or numeric values:

MONTHS\_BETWEEN, ADD\_MONTHS, NEXT\_DAY, LAST\_DAY, ROUND and TRUNC.

Group Functions: SUM( [ALL | DISTINCT] expression ); AVG( [ALL | DISTINCT] expression ); COUNT( [ALL | DISTINCT] expression ); COUNT(\*);  
MAX(expression);MIN(expression)