UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO

Computer Science Department

Master Degree in Computer Science

Practical Part of **Database Systems**

Brightway: Technical Documentation

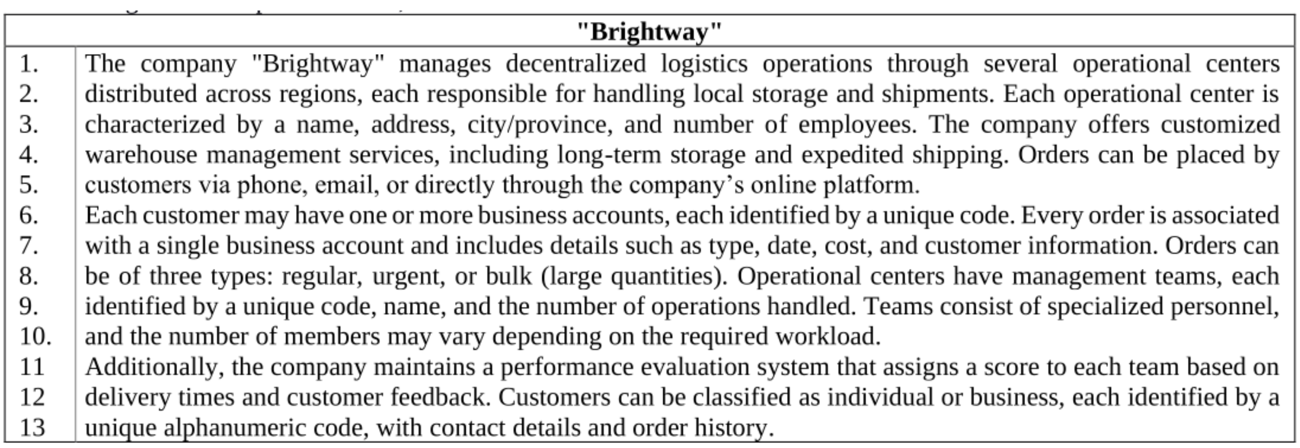
Student: Student ID:

Mattia Curri **832437**

ACADEMIC YEAR **2024-2025**

Contents

1 | Requirements



"Brightway"

The company "Brightway" manages decentralized logistics operations through several operational centers distributed across regions, each responsible for handling local storage and shipments. Each operational center is characterized by a name, address, city/province, and number of employees. The company offers customized warehouse management services, including long-term storage and expedited shipping. Orders can be placed by customers via phone, email, or directly through the company's online platform.

Each customer may have one or more business accounts, each identified by a unique code. Every order is associated with a single business account and includes details such as type, date, cost, and customer information. Orders can be of three types: regular, urgent, or bulk (large quantities). Operational centers have management teams, each identified by a unique code, name, and the number of operations handled. Teams consist of specialized personnel, and the number of members may vary depending on the required workload.

Additionally, the company maintains a performance evaluation system that assigns a score to each team based on delivery times and customer feedback. Customers can be classified as individual or business, each identified by a unique alphanumeric code, with contact details and order history.

2 | Conceptual Design

2.1 Requirements Analysis

2.2 Reorganize sentences for specific concepts

General Phrases

We want to create a database that manages decentralized logistics operations through several operational centers, storing information about orders, teams, and customers. The company offers customized warehouse management services, including long-term storage and expedited shipping.

Phrases related to Operational Centers

Operational centers are distributed across regions, each responsible for handling local storage and shipments.
For each operational center, we will hold name, address, city/province, and number of employees.
Operational centers have management teams.

Phrases related to Orders

Orders can be placed by customers via phone, email, or online.
For each order, we will hold type, date, cost, and customer information.
Every order is associated with a single business account.
Orders can be of three types: regular, urgent, or bulk (large quantities).

Phrases related to Business Accounts

Each business account will be identified by a unique code.
Each customer may have one or more business accounts.
Every order is associated with a single business account.

Phrases related to Teams

For each team, we will identify them via a unique code, and we will hold name and number of orders handled.
Teams consist of employees, and the number of employees may vary depending on the required workload.
The company maintains a performance evaluation system that assigns a score to each team based on delivery times and customer feedback.

Phrases related to Customers

Each customer may have one or more business accounts.
Customers can be classified as individual or business, each identified by a unique alphanumeric code, with contact details and order history.

Phrases related to Employees

Teams consist of employees, and the number of employees may vary depending on the required workload.

2.3 Level of abstraction

For **customer information**, we consider *name*, *surname*, and *date of birth* for individual customers, and *company name* and *address* for business customers, where the address consists of *street*, *civic number*, *city*, *province*, *region*, and *state*.

For **contact details**, we consider *phone number* and *email*.

Specialized personnel is replaced by *employees*.

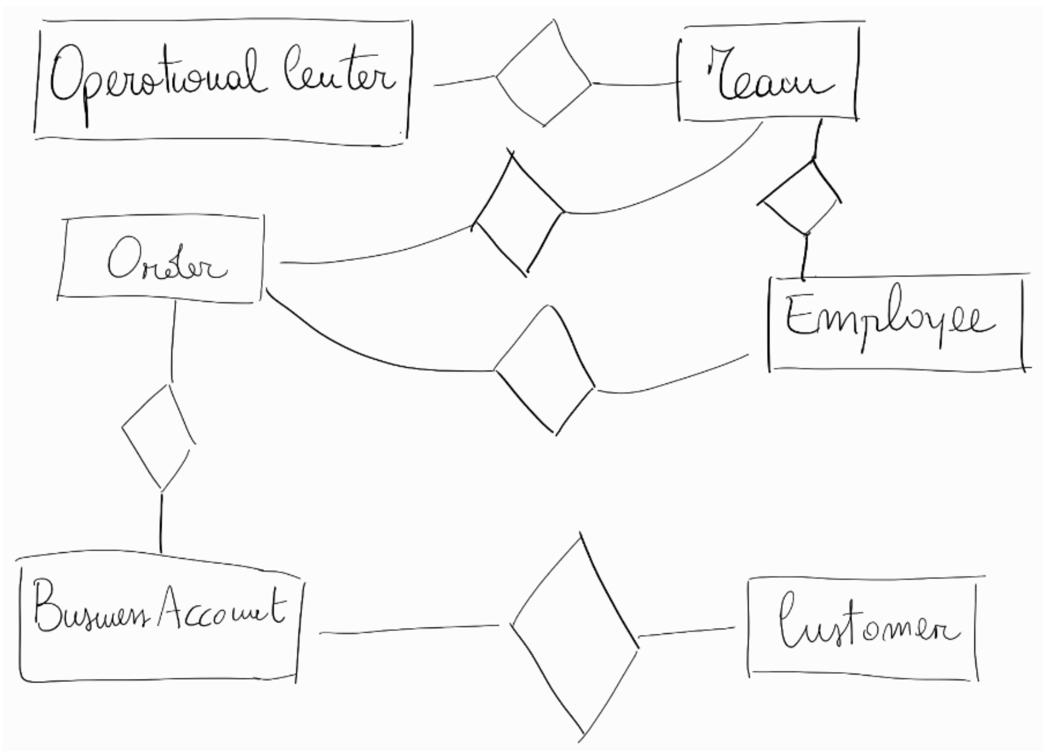
Number of members is replaced by *number of employees*.

Number of operations is replaced by *number of orders*.

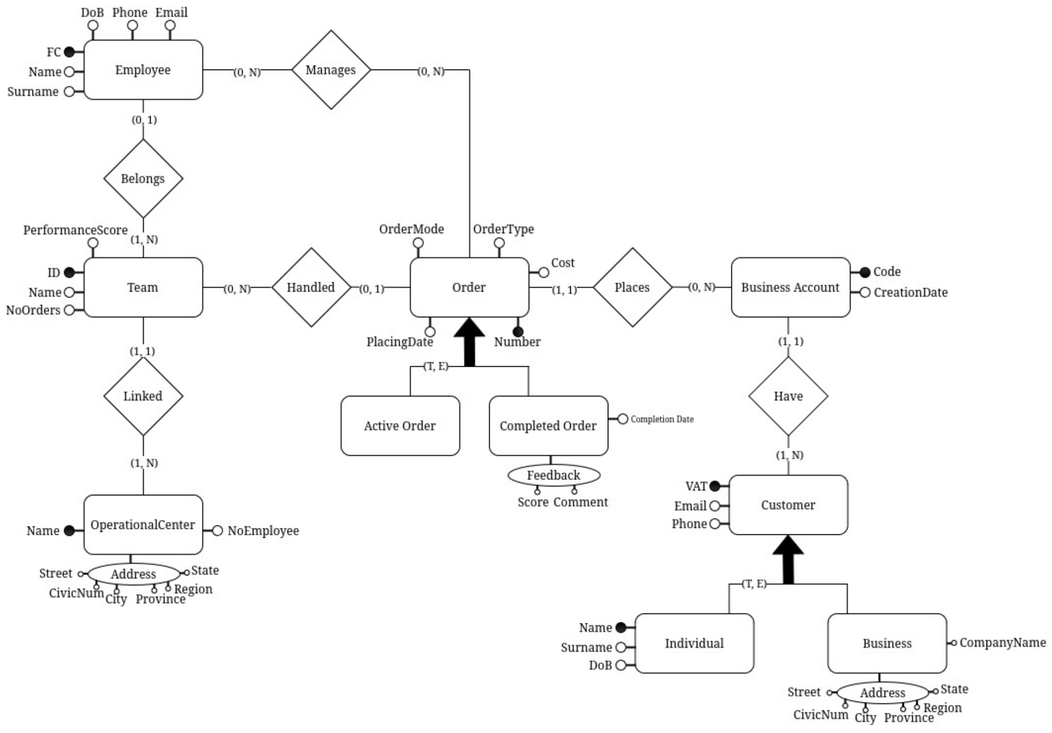
Term	Description	Synonyms	Connections
Operational Center	Decentralized locations handling local storage and shipments. Characterized by name, address, city/province, and number of employees.	/	Team
Order	Requests for services or goods from customers, identified by type (regular, urgent, or bulk). Includes type, date, cost, and customer details.	Operation	Customer, Business Account, Team
Business Account	Accounts tied to customers, containing unique codes and details like orders and customer type (individual or business).	Account	Customer, Order
Team	Groups of employees linked to an operational center, evaluated on delivery times and customer feedback.	Management Team	Operational Center, Order, Employee
Customer	Individuals or businesses placing orders. Identified by unique alphanumeric codes, contact details, and order history.	Individual, Business	Order, Business Account
Employee	Specialized personnel working in teams and handling orders.	Specialized Personnel	Team

2.4 Glossary of terms

2.4.1 Skeleton ER Schema



2.4.2 Final ER Schema



- There are three redundancies: `NoOrders` and `PerformanceScore` in `Team` and `NoEmployee` in `Operational Center`, that will be evaluated later.

- There are two generalizations for `Order` and `Customer`, that will be evaluated later.

2.5 Business Rules

- Employees linked to an order must be in the same team that handle the order.
- Number of employees is the sum of all employees in a team.
- Number of orders is the sum of all orders handled by a team.
- Completion date cannot be before placing date.
- PerformanceScore is computed as the mean of all feedbacks.
- A team has a maximum of 8 members.
- OrderType must be of three types: regular, urgent, or bulk.
- OrderMode must be of three types: phone, email, or online.
- Feedback of CompletedOrder must be between 1 and 5.
- A feedback cannot be given before the order is completed.
- A team must be assigned before an order is completed.
- A team cannot be changed after an order is completed.
- Employees of a completed order cannot be changed.
- An order can be deleted only if there are not information for the business account or for the feedback computation.

3 | Logical Design

3.1 Volumes Table

We will consider a span of a month to evaluate the volumes of the entities.

Concept	Type	Computation	Final Volume
Operational Center	E	15 (assuming 10 team per `Operational Center`)	15
Linked To	R	150 (same as `Team`)	150
Team	E	150 (*given*)	150
Handled By	R	45000 (same as (assigned) `Order`)	45000
Order	E	$300 \text{ op/m} \cdot 150 \cdot 12 + 100$ not assigned	45100
Belongs To	R	150 `Team` · 6.4 current `Employee`	960
Employee	E	80% of $(150 \cdot 8) + 140$ past `Employee`	1100
Manages	R	$45000 \cdot 3$ (assuming 3 `Employee` working on an `Order`)	135000
Places	R	45100 (same as `Order`)	45100
Business Account	E	$\frac{45100 \texttt{ Order}}{1.5 \texttt{ Order}/\text{month}}$	30000
Have	R	30000 (same as `Business Account`)	30000
Customer	E	$\frac{30000 \texttt{ Business Account}}{1.5 \texttt{ Business Account/Customer}}$	20000
Active Order	E	20% of $45000 + 100$ not assigned	9100
Completed Order	E	80% of 45000	36000
Individual	E	90% of `Customer`	18000
Business	E	10% of `Customer`	2000

3.2 Operations Analysis

Table 3.1: Operations frequency analysis

Operation	Type	Frequency
Operation 1	I	10/day
Operation 2	I	1000/day
Operation 3	I	500/day
Operation 4	I	200/day
Operation 5	I	20/day

In the next section, we will double count the cost of writing operations.

Operation 1: Register a new customer

Operation cost: $6 \text{ accesses} \cdot 10 \text{ days} = 60 \text{ accesses/day}$

Operation 2: Add a new order

Operation cost: $4 \text{ accesses} \cdot 1000 \text{ days} = 4000 \text{ accesses/day}$

Table 3.2: Access analysis for registering a new customer

Concept	Type	No. Access	Access Type
Customer	E	1	W
Have	R	1	W
Business Account	E	1	W

Table 3.3: Access analysis for adding a new order

Concept	Type	No. Access	Access Type
Order	E	1	W
Places	R	1	W

Operation 3: Assign an order to a management team

Access *without* redundancy `Team.NoOperations`:

Table 3.4: Access analysis for assigning order to team (without redundancy)

Concept	Type	No. Access	Access Type
Handled By	R	1	W
Manages	R	3	W

Access *with* redundancy `Team.NoOperations`:

Table 3.5: Access analysis for assigning order to team (with redundancy)

Concept	Type	No. Access	Access Type
Handled By	R	1	W
Team	E	1	R
Team	E	1	W
Manages	R	3	W

Operation cost (*without redundancy*): 8 accesses $\cdot$ 500 days = 4000 accesses/day

Operation cost (*with redundancy*): 11 accesses $\cdot$ 500 days = 5500 accesses/day

Operation 4A: View the total number of operations handled by a specific team

Access *with* redundancy `Team.NoOperations`:

Table 3.6: Access analysis for viewing team operations (with redundancy)

Concept	Type	No. Access	Access Type
Team	E	1	R

Access *without* redundancy `Team.NoOperations`:

Operation cost (*with redundancy*): 1 access $\cdot$ 200 days = 200 accesses/day

Operation cost (*without redundancy*): 300 accesses $\cdot$ 200 days = 60000 accesses/day

Table 3.7: Access analysis for viewing team operations (without redundancy)

Concept	Type	No. Access	Access Type
Handled By	R	300	R

Operation 4B: Show the total cost of the orders handled by a specific team

TODO

Operation 5: Print a list of teams sorted by their performance score

Access *with* redundancy `Team.PerformanceScore`:

Table 3.8: Access analysis for sorting teams by performance (with redundancy)

Concept	Type	No. Access	Access Type
Team	E	150	R

Access *without* redundancy `Team.PerformanceScore`:

Table 3.9: Access analysis for sorting teams by performance (without redundancy)

Concept	Type	No. Access	Access Type
Team	E	150	R
Handled By	R	45000	R
Order	E	45000	R

Operation cost (*with redundancy*): 150 accesses $\cdot$ 20 days = 3000 accesses/day

Operation cost (*without redundancy*): 90150 accesses $\cdot$ 20 days = 1803000 accesses/day

3.3 Redundancy Analyisis

- `OperationalCenter.NoEmployees`: given that there are no operations that involve this attribute, so we decide to **eliminate** the redundancy.
- `Team.NoOperations`: with the redundancy, we have 4200 accesses combining both Op. (3) and (4); on the other hand, without the redundancy, we have 65700 accesses. In the end, we decide to **maintain** the redundancy.
- `Team.PerformanceScore`: with the redundancy, we have 3000 accesses on Op. (5); on the other hand, without the redundancy, we have 1803000 accesses. In the end, we decide to **maintain** the redundancy.

3.4 Partitioning and Merging

- Merging `Actual Order` and `Completed Order` into `Order`: given the fact that there are not operations that require to distinguish between the two entities, we can **merge** them into a single entity.

- Merging `Individual` and `Business` into `Customer`: given the fact that there are not operations that require to distinguish between the two entities, we can **merge** them into a single entity.

3.4.1 Restructured Schema

INSERIRE SCHEMA UML

4 | Implementation

4.1 Types definition

```
CREATE OR REPLACE TYPE AddressTY AS OBJECT (
    street VARCHAR2(50),
    civicNum NUMBER,
    city VARCHAR2(50),
    province VARCHAR2(50),
    region VARCHAR2(50),
    state VARCHAR2(50)
);
```

```
CREATE OR REPLACE TYPE OperationalCenterTY AS OBJECT (
    name VARCHAR2(50),
    address AddressTY
);
```

```
CREATE OR REPLACE TYPE TeamTY AS OBJECT (
    ID VARCHAR2(50), -- Format: TXXXXXX
    name VARCHAR2(20),
    numOrder NUMBER,
    performanceScore NUMBER(4, 2),
    operationalCenter ref OperationalCenterTY
);
```

```
CREATE OR REPLACE TYPE EmployeeTY AS OBJECT (
    FC VARCHAR2(16),
    name VARCHAR2(20),
    surname VARCHAR2(20),
    dob DATE,
    phone VARCHAR2(14),
    email VARCHAR2(50),
    team ref TeamTY
);
```

```
CREATE OR REPLACE TYPE FeedbackTY AS OBJECT (
    score NUMBER(1), -- check (score between 1 and 5),
    commentF VARCHAR2(1000)
);
```

```
CREATE OR REPLACE TYPE CustomerTY AS OBJECT (
    VAT VARCHAR2(11),
    phone VARCHAR2(14),
    email VARCHAR2(50),
    type VARCHAR2(10), -- CHECK (type IN ('individual', 'business')
    name VARCHAR2(20),
    surname VARCHAR2(20),
    dob DATE,
    companyName VARCHAR2(50),
    address AddressTY
) NOT FINAL;
```

```
CREATE OR REPLACE TYPE BusinessAccountTY AS OBJECT (
    CODE VARCHAR2(10), -- Format: BXXXXXXXXX
    creationDate DATE,
    customer ref CustomerTY
);
```

```
CREATE OR REPLACE TYPE EmployeeVA AS VARRAY(8) OF REF EmployeeTY;
```

```
CREATE OR REPLACE TYPE OrderTY AS OBJECT (
    ID VARCHAR2(10), -- Format: OXXXXXXXXX
    placingDate DATE,
    orderMode VARCHAR2(6), -- CHECK (orderMode IN ('online', 'phone',
        'email')),
    orderType VARCHAR2(7), -- CHECK (orderType IN ('regular', 'urgent
        ', 'bulk')),
    cost NUMBER(10, 2),
    businessAccount ref BusinessAccountTY,
    team ref TeamTY,
    employees EmployeeVA,
    completionDate DATE,
    feedback FeedbackTY
```

4.2 Table definition

```
CREATE TABLE OperationalCenterTB OF OperationalCenterTY (
    name PRIMARY KEY,
    address NOT NULL
);
```

```
CREATE TABLE TeamTB OF TeamTY (
    ID PRIMARY KEY,
    name NOT NULL,
    numOrder check (numOrder >= 0),
    performanceScore check (performanceScore between 1 and 5),
    operationalCenter NOT NULL
);
```

```
CREATE TABLE EmployeeTB OF EmployeeTY (
    FC PRIMARY KEY CHECK (LENGTH(FC) = 16),
    name NOT NULL,
    surname NOT NULL,
    dob NOT NULL,
    phone NOT NULL,
    email NOT NULL
);
```

```
CREATE TABLE CustomerTB OF CustomerTY (
    VAT PRIMARY KEY CHECK (LENGTH(VAT) = 11),
    phone NOT NULL,
    email NOT NULL,
    type NOT NULL CHECK (type IN ('individual', 'business'))
);
```

```
CREATE TABLE BusinessAccountTB OF BusinessAccountTY (
    CODE PRIMARY KEY,
    creationDate NOT NULL,
    customer NOT NULL
);
```

```
CREATE TABLE OrderTB OF OrderTY (
    ID PRIMARY KEY,
    placingDate NOT NULL,
    orderMode NOT NULL CHECK (orderMode IN ('online', 'phone', 'email'
       )),
    orderType NOT NULL CHECK (orderType IN ('regular', 'urgent', 'bulk
       ')),
    cost NOT NULL CHECK (cost > 0),
    businessAccount NOT NULL,

    check (placingDate <= completionDate),
    check (feedback.score between 1 and 5)
);
```

5 | Trigger Implementation

5.1 Triggers

CheckOrderInsertOrUpdate

This trigger enforces logical constraints on orders:

```
CREATE OR REPLACE TRIGGER CheckOrderInsertOrUpdate
    % ...existing code...
END;
/
```

CheckCustomerType

Enforces that "individual" customers cannot have business data and vice versa:

```
CREATE OR REPLACE TRIGGER CheckCustomerType
    % ...existing code...
END;
/
```

UpdateNumOrders

Keeps each team’s numOrder up-to-date when orders are inserted, updated, or deleted:

```
CREATE OR REPLACE TRIGGER UpdateNumOrders
    % ...existing code...
END;
/
```

CheckTeamInsertInitialization

Initializes numOrder and performanceScore when inserting a new team:

```
CREATE OR REPLACE TRIGGER CheckTeamInsertInitialization
    % ...existing code...
END;
/
```

CheckNumEmployeeInTeam

Ensures a team cannot exceed 8 employees:

```
CREATE OR REPLACE TRIGGER CheckNumEmployeeInTeam
    % ...existing code...
END;
/
```

ComputePerformanceScore

Recalculates a team's average feedback score whenever a new order or feedback is added or updated:

```
CREATE OR REPLACE TRIGGER ComputePerformanceScore
    % ...existing code...
END;
/
```

AddAccount

Automatically creates a business account for every new customer:

```
CREATE OR REPLACE TRIGGER AddAccount
    % ...existing code...
END;
/
```

DeleteTeamAfterOperationalCenter

Deletes teams that lose their operational center reference upon an operational center's deletion:

```
CREATE OR REPLACE TRIGGER DeleteTeamAfterOperationalCenter
    % ...existing code...
END;
/
```

UpdateEmployeeAfterTeam

Sets an employee's team reference to NULL if the team is deleted, and unassigned orders if they're not completed:

```
CREATE OR REPLACE TRIGGER UpdateEmployeeAfterTeam
    % ...existing code...
END;
/
```

DeleteAccountAfterCustomer

Deletes business accounts that become orphaned when their customer is removed:

```
CREATE OR REPLACE TRIGGER DeleteAccountAfterCustomer
    % ...existing code...
END;
/
```

DeleteOrdersAfterTeam

Removes orders that have lost their team and business account references:

```
CREATE OR REPLACE TRIGGER DeleteOrdersAfterTeam
    % ...existing code...
END;
/
```

DeleteOrdersAfterAccount

Deletes orders that have lost their related business account, under certain conditions:

```
CREATE OR REPLACE TRIGGER DeleteOrdersAfterAccount
    % ...existing code...
END;
/
```

PreventOrderDeletion

Prevents order deletion unless it has lost all references or is uncompleted with no account:

```
CREATE OR REPLACE TRIGGER PreventOrderDeletion
    % ...existing code...
END;
/
```

EmptyEmployeeListAfterTeamUpdate

Clears the employee list if a team reference changes:

```
CREATE OR REPLACE TRIGGER EmptyEmployeeListAfterTeamUpdate
    % ...existing code...
END;
/
```

6 | Functions Implementation

6.1 Operations Implementation

6.1.1 Operation 1: registerCustomer

Registers a new customer with optional business or individual data.

```
CREATE OR REPLACE PROCEDURE registerCustomer(
    VAT IN VARCHAR2,
    phone IN VARCHAR2,
    email IN VARCHAR2,
    type IN VARCHAR2,
    name IN VARCHAR2,
    surname IN VARCHAR2,
    dob IN DATE,
    companyName IN VARCHAR2,
    address IN AddressTY
) AS
    customer CustomerTY;
BEGIN
    % ...existing code...
END;
/
```

6.1.2 Operation 2: addOrder

Adds a new order by referencing a business account and optional employees.

```
CREATE OR REPLACE PROCEDURE addOrder(
    ID IN VARCHAR2,
    placingDate IN DATE,
    orderMode IN VARCHAR2,
    orderType IN VARCHAR2,
    cost IN NUMBER,
    businessAccountName IN VARCHAR2,
    employees IN EmployeeVA DEFAULT NULL
) AS
```

```
    % ...existing code...
END;
/
```

6.1.3 Operation 3: assignOrderToTeam

Assigns an order to a specific team.

```
CREATE OR REPLACE PROCEDURE assignOrderToTeam(
    orderID IN VARCHAR2,
    teamID IN VARCHAR2
) AS
    % ...existing code...
END;
/
```

6.1.4 Operation 4A: totalNumOrder

Returns the total number of orders handled by a given team.

```
CREATE OR REPLACE FUNCTION totalNumOrder(
    teamID IN VARCHAR2
) RETURN NUMBER AS
    % ...existing code...
END;
/
```

6.1.5 Operation 4B: totalOrderCost

Calculates the total cost of all orders for a team.

```
CREATE OR REPLACE FUNCTION totalOrderCost(
    teamID IN VARCHAR2
) RETURN NUMBER AS
    % ...existing code...
END;
/
```

6.1.6 Operation 5: printTeamsByPerformanceScore

Prints teams sorted by performance score in descending order.

```
CREATE OR REPLACE PROCEDURE printTeamsByPerformanceScore AS
BEGIN
    % ...existing code...
END;
/
```