Advanced Database Systems Coursework

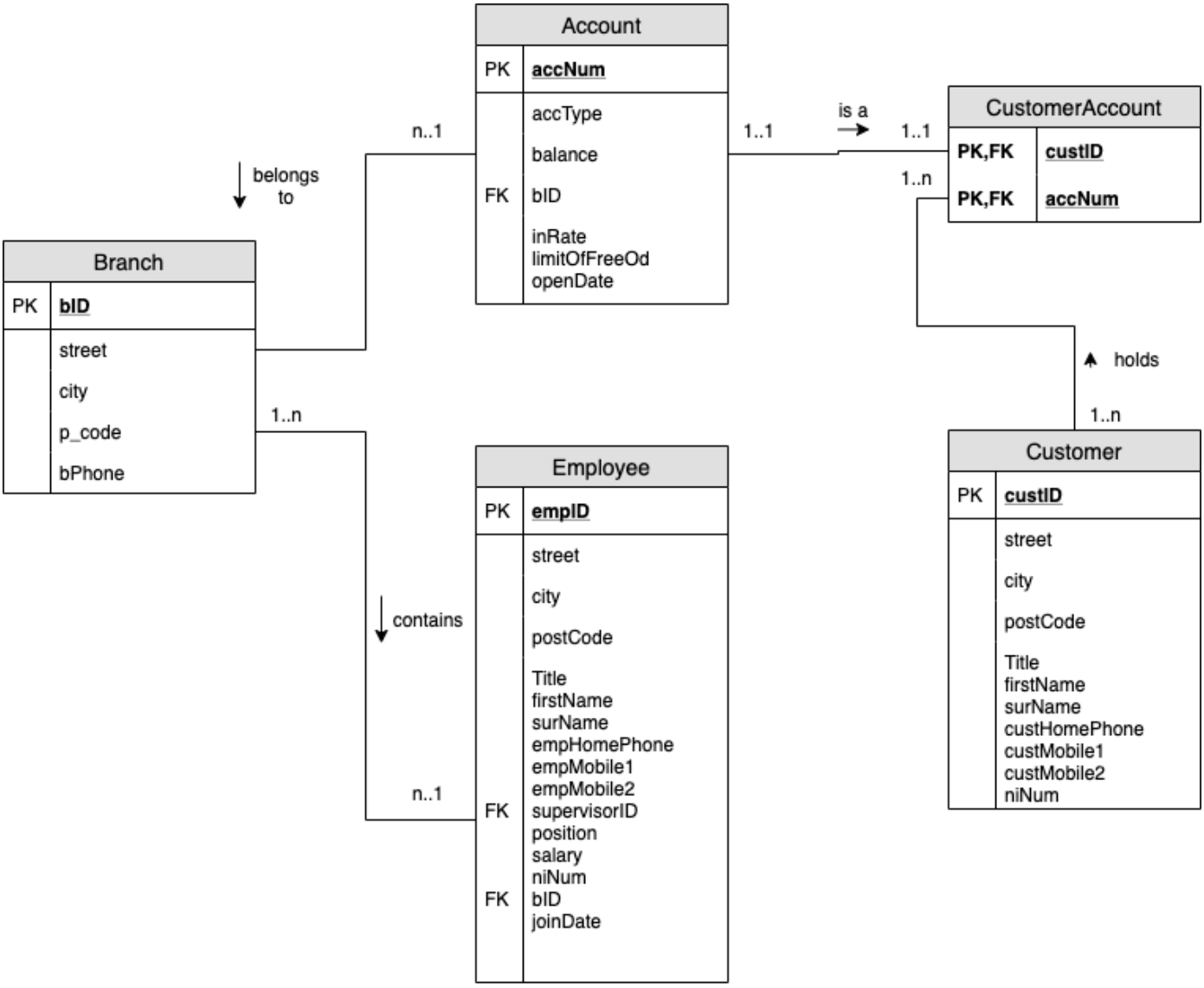
Aaron Campbell
40278819

Table of Contents

Abstract

The purpose of this coursework is to re-design a database for a bank. The database stores information on its branches, customers, employees and accounts. The initial implementation of this database has been carried out in an entity-relational manner, as will be displayed and described below. This coursework will make use of object-oriented features in the improved implementation, in order to capture more accurately capture the semantics of the database and to implement the problem in a more streamlined, efficient manner.

Entity Relational Diagram for Previous Diagram



Database Redesign

The purpose of this coursework is to re-design the database in an object-relational manner – capturing more of the semantics of the database while maintaining semantics of the previous entity-relational application. The justification for this is that an object-relational approach can be seen as an extension of the entity-relational model – capturing the semantics of objects supported in object oriented programming and introducing concepts such as encapsulation and methods to the database.

Types and Sub-Types

A structured type (or user-defined type) is a user defined composite datatype representing a data structure and functions and structures to manipulate the data [1]. The variables that form the data are called attributes, and the functions and procedures that manipulate them are called methods. For example, a baby has the attributes gender, age and weight and the actions eat, drink and sleep [2]. Object types allow you to break a large system down into logical entities. They also allow for realistic data modelling – complex real-world entities often map directly into object types [2]. A type may also be referred to as a super type, given it has a sub type.

A sub type contains all the attributes and methods of the super type [2]. They can also contain additional attributes and methods however, and can override methods from the super type. A sub type does not create a new data type, rather a derived formed from the base type [3].

Types are used extensively in this implementation to represent the data within the problem. There are 7 super types, representing the:

- Name
- Address
- Phone
- Phone Nested (the nested table, expanded on in the collections section)
- Person
- Job
- Account

and 3 sub types, representing the

- Employee
- Customer
- Branch

There are 6 tables. The use of types allows for the inheritance of attributes in the creation of the tables. As stated earlier, object types allow for the breaking down of the problem into logical entities. This can be seen in the implementation of the database, where both

Employee and Customer inherit many attributes from the Person type. This allows for more efficient code, where large chunks of code are not unnecessarily repeated.

References

Object-oriented languages allow you to create and refer to objects. This feature is replicated in the object-relational model, where references can be used to point to a tuple in another table. It has a similar function to foreign keys in the entity-relational model. References allow for the acquisition of all the data in the referenced table, while inserting it in one column.

In this implementation, references are used throughout as a way to quickly access relevant data in other tables and without the use of a JOIN function. For example, the reference to ac_job inside the ac_employee type allows for access to both the title and the salary within queries. I chose to do this as it makes referencing other tables much more efficient and optimises queries on the database.

Member Functions

Methods can be viewed as functions associated with types [4], and that can operate on the attributes of the type [5].

In this coursework, there is a function that allows for employees to be awarded at the end of the year. This is called the awardMedal function, is inside the employee type, and is based on the number of years a supervisor has been at the company, and the number of employees they supervise. There are also member functions that get the number of years the employee has been at the company, and amount of people they supervise. This is for use in the main awardMedal function and makes the code more efficient. I chose to do this as it made the awardMedal function more efficient, limiting the amount of code inside the function and making the code generally more modular – one of the main advantages of the object-relational model and object-oriented programming.

There is also a printName method, which prints the person's full title name in a formatted manner in one column. In an entity-relational implementation, these three values would have to be printed in three separate columns. Through the use of member functions, these values can be concatenated. There is a similar printAddress function.

Constraints

A check constraint allows you to specify a condition on each row in a table [6]. A primary key is used to identify the primary key for a table [7]. The unique constraint is to ensure that all column values within a table never contain a duplicate entry [7]. This allows you to sensibly restrict your data so that it cannot represent false data.

Primary key and null constraints have been used on frequently in this database. There is also a unique constraint on the national insurance number field, to ensure that there are no duplicates.

Collections

There are two collection data types supported in Oracle – Varrays and nested tables [7].

Varrays are variable-length ordered lists, where a maximum size must be specified.

Nested tables are tables within tables, and are unordered. The table stores the actual values, but cannot be used in a query. They can be thought of as one-dimensional arrays with no declared number of elements [8]. The main advantages of nested tables over varrays are the fact that there is no set number of index values and that they multiple columns into a single column.

In this database I opted to use a nested table to store the Phone type. This is because sometimes only one phone number needs to be stored, and other times multiple phone numbers need to be stored. Nested tables prevent the occurrence of null values in the scenarios where there are no phone numbers to store.

Alternative Design Considerations

There was a consideration of employing a varray rather than a nested table to store phone numbers. This was decided against because the maximum number of phone numbers needed to store is unknown. As stated earlier, nested tables have the clear advantage of being unordered. Varrays are rarely used in any database [9].

Another consideration was to not have a Job type and to keep this information in the Employee type, as this better reflected the previous entity-relational database. However, I decided that using a Job type would be more efficient as it would reduce the repetition of code – a reference to the employee's job type could be made and their salary could be accessed from there.

SQL Statements

a. Find employees whose first name includes the string “st” and live in Edinburgh, displaying their full names. (3 marks)

Query for 3a

```
SELECT
e.printName() AS "Employee Name"
FROM ac_employee_table e
WHERE INSTR(e.personName.firstName, 'on') > 0
AND e.personAddress.city = 'Edinburgh';
```

Output for 3a

	Employee Name	City
1	Mr, Don, James	Edinburgh

b. Find the number of saving accounts at each branch, displaying the number and branch’s address. (3 marks)

Query for 3b

```
SELECT
    a.branch_id.bID AS "Branch ID",
    a.branch_id.printAddress() AS "Address",
    count(a.accType) AS "Number of savings accounts"
FROM
    ac_account_table a
WHERE
    accType = 'Savings'
GROUP BY
    a.accType, a.branch_id.printAddress(), a.branch_id.bID
ORDER BY
    a.branch_id.bID ASC;
```

Output for 3b

	Branch ID	Address	Number of savings accounts
1	E001	28 Frankfurt Street, Edinburgh, EH2 2DS	3
2	G003	28 New Street, Glasgow, G9D 2P3	1
3	L001	28 Thread Street, London, EC2R 2DS	2
4	L002	28 London Street, London, EH2 9RY	1
5	L003	28 Crescent Road, London, EH2 2P3	2

d. Find employees who are supervised by a manager and have accounts in the bank, displaying the branch address that the employee works in and the branch address that the account is opened with. (3 marks)

Query for 3d

```
SELECT
    e.emp_id AS "Employee ID",
    e.printName() AS "Employee",
    e.branch_id.printAddress() AS "Work Address",
    e.personAddress.street || ', '|| e.personAddress.city || ', '|| e.personAddress.postcode AS "Home Address"
FROM
    ac_employee_table e, ac_customer_account_table c
WHERE
    c.cust_id.personName.firstName = e.personName.firstName
AND
    c.cust_id.personName.lastName = e.personName.lastName
AND
    e.supervisor_id.position.title = 'Manager'
ORDER BY
    e.emp_id ASC
```

Output for 3d

Employee ID	Employee	Work Address	Home Address
003	Mr, Dave, Doe	28 Frankfurt Street, Edinburgh, EH2 2DS	10 New Road, Edinburgh, EH10 2UE

The remaining tasks were attempted and have partial answers in the answersToTask4.sql file.

Advantages and Disadvantages of the Object Relational Model against the Entity-Relational Model

Entity-Relational Model

The entity-relational model is made up of tables and attributes [10], with relationships among the relations (tables). For example, a student enrols in a course [11]. Cardinality, the number of entities in an entity set in comparison to its related entity set, plays a large role in the entity-relation model – one to one, one to many etc. Keys also play a key role, such the Primary and Foreign Key. The main advantage of the entity-relational model is its fairly simple relation representation. This lends to its popularity among businesses, along with its simple graphical representation that may be easier to grasp for most people. However, there are some clear disadvantages. For example, it does not allow for the sufficient expression of data that does not map well to tables. It also does not allow for the creation of structured types (it only has a number of built-in types), the representation of complex entities as a single unit or the use of methods/member functions [11].

Object-Relational Model

The object-relational model extends the relational model by including object orientation and constructs to deal with added data types [11]. In introduces object-oriented concepts such as object identity, encapsulation, class hierarchies and inheritance. The object-relational model's inheritance gives it a clear advantage over the entity-relational model – types can be reused in different objects with extended functionality. This reduces the cost of maintaining the data and makes the code more efficient. Complex data types also allow for the extension of the functionality of the system. However, it can also be described as fairly complicated and therefore is seen as much more time-consuming than the entity-relational model to learn and implement.

In terms of this database, the object-relational model's advantages come through strongly. The ability to create structured types lent itself well to this implementation, in that it condensed the number of columns in each table. For example, the Name type condensed the Title, First Name and Last Name of each person into one column, without losing these pieces of information. This also applies to the Address type. If this was implemented in an entity-relational way, these fields would have had to appear in every necessary table individually and repeatedly. The concept of modularity makes the database much more efficient and logical.

The object-relational also allows for the use of methods/member functions, earlier described as functions associated with types. This feature was vital to the required implementation, and was used to capture the feature whereby employees were rewarded at the end of a year. Another example of where methods proved useful is in the two

methods that aided the main awardMedal function – which get the number of years the employee has been at the company and how many people they supervise. This would not have been possible if employing the entity-relational model. Similar examples are the printName and printAddress member functions, which allow for the concatenation of three columns and for the address to be printed in one column.

References

[1] Oracle (2005), 'PL/SQL User's Guide and Reference',
https://docs.oracle.com/cd/B19306_01/appdev.102/b14261.pdf
[2] Oracle, 'Using PL/SQL Object Types',
https://docs.oracle.com/cd/B14117_01/appdev.101/b10807/10_objs.htm
[3] Boobal Ganeson (2017), 'Advanced PL/SQL Programming: The Definitive Reference'
[4] Taoxin Peng, 'Inheritance, References & Methods'
[5] Oracle, 'Methods: Using PL/SQL'
[6] TeachOnTheNet, 'Oracle/PLSQL: Check Constraints'
https://www.techonthenet.com/oracle/check.php
[7] Taoxin Peng, 'Constraints & Collections'
[8] Oracle, 'Using PL/SQL Collections and Records'
[9] Oracle Magazine, 'Working With Collections'
[10] Thomas Connolly & Carolyn Begg (2005), 'Database Systems: A Practical Approach to Design, Implementation and Management'
[11] Tutorialspoint, 'ER Model – Basic Concepts',
https://www.tutorialspoint.com/dbms/er_model_basic_concepts.htm
[12] Taoxin Peng, 'Non-Standard DB'