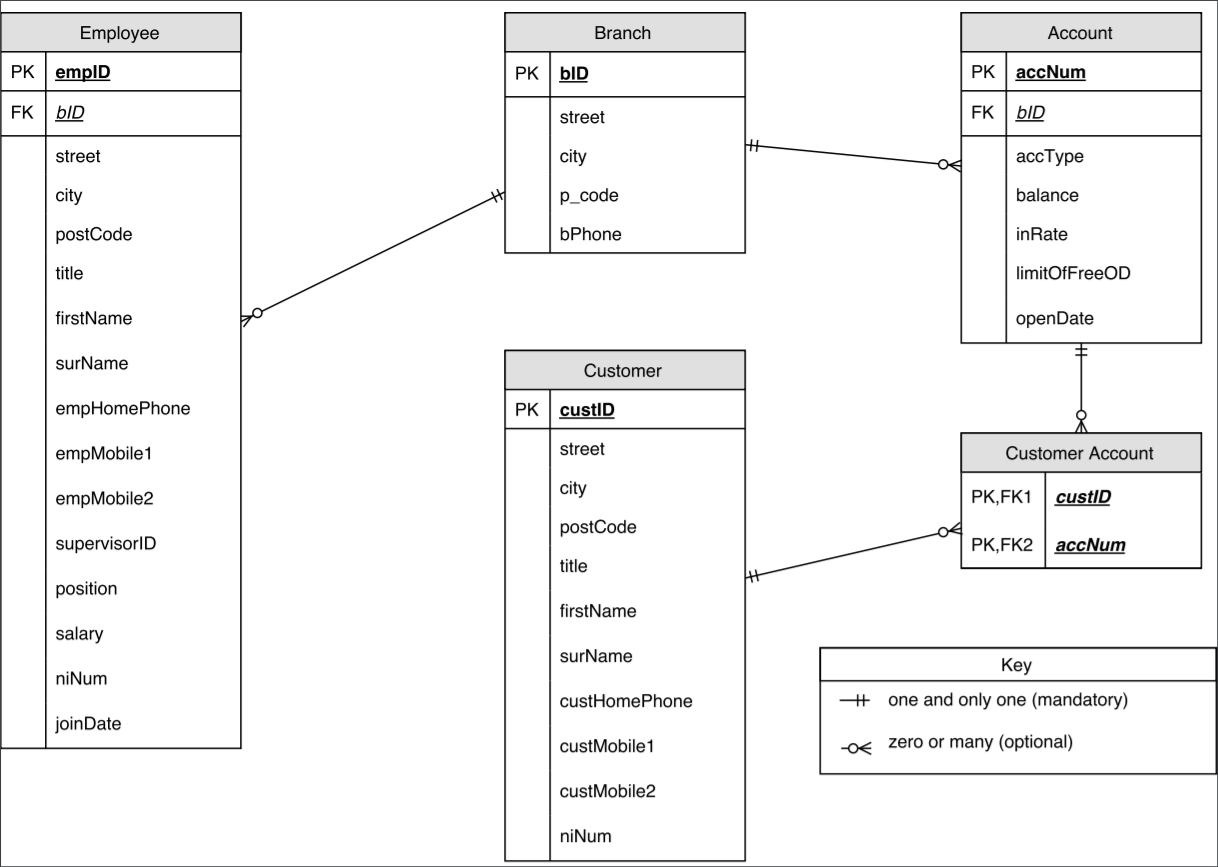
**Task 1** - ER Diagram



**Task 2** - Proposed Object-Relational Database Redesign

**Structured Types**

Type (**attribute** DATA\_TYPE [CONSTRAINT])

Name (**title** VARCHAR2(8) [IS NOT NULL], **firstName** VARCHAR2(20) [IS NOT NULL], **surName** VARCHAR2(20) [IS NOT NULL])

Address (**street** VARCHAR2(20) [IS NOT NULL], **city** VARCHAR2(20) [IS NOT NULL], **p\_code** VARCHAR2(8) [IS NOT NULL])

Phone (**homePhone** VARCHAR2(15) [IS NOT NULL], **mobilePhone** mobilePhones\_nested)

Branch (**bID** NUMBER [PRIMARY KEY], **bAddress** Address, **bPhone** VARCHAR2(15) [IS NOT NULL])

Job (**position** VARCHAR2(20) [CHECK IN ("Head", "Manager", "Accountant", "Leader", "Cashier")], **salary** NUMBER [IS NOT NULL], ***bID***ref Branch, **joinDate** DATE [IS NOT NULL])

Person (**pName** Name, **pAddress** Address, **pPhone** Phone, **niNum** VARCHAR2(15) [UNIQUE])

Customer UNDER Person (**custID** NUMBER [PRIMARY KEY])

Employee UNDER Person (**empID** NUMBER [PRIMARY KEY], **supervisorID** NUMBER [IS NOT NULL], **eJob** Job)

Account (**accNum** NUMBER [PRIMARY KEY], **accType** VARCHAR2(20) [CHECK IN ("current", "savings")], **balance** NUMBER [IS NOT NULL], ***bID***ref Branch, **inRate** NUMBER [IS NOT NULL], **limitOfFreeOD** NUMBER [IS NOT NULL], **openDate** DATE [IS NOT NULL])

CustomerAccount (***custID*** ref Customer, ***accNum*** ref Account)

- A Critical Review of the Rationale for the Proposed Object-Relational Database Design

**Structured Types Used and Why**

The ‘Employee’ and ‘Customer’ entities in the relational database share many attributes (*street, city, postCode, title, firstName, surName, niNum*) as well has having some similar attributes (*empHomePhone, custHomePhone, empMobile1, custMobile1, empMobile2, custMobile2*). Therefore, the structured types ‘Name’, ‘Address’ and ‘Phone’ were created with their relevant attributes assigned to them. Further to this a structured type ‘Person’ was created that used these three structured types and also has the *niNum* attribute. This allowed the final structured types ‘Employee and ‘Customer’ (used to make their respective tables) to inherit from ‘Person’ and maintain all those shared attributes as well as the attributes unique to them. The ‘Address’ type was also able to be used in the ‘Branch’ structured type for the ‘Branch’ table as each branch also had a street, city and post code attribute in the original design. Separate to this, the structured type ‘Job’ was created to further group attributes in the ‘Employee’ entity.

**Data Types Used and Why**

For most text-based attributes the data type VARCHAR2(20) was used. VARCHAR2 was picked over VARCHAR as recommended by Oracle as it can store a larger number of bytes of characters. A maximum of 20 characters was chosen as a sensible number of characters for these attributes. Exceptions to this were for the *title, p\_code,* and all phone number attributes where a different maximum was chosen. A maximum of 8 characters was chosen for *titles* and *p\_code* to reduce size as they are always 8 or less characters. For the phone number attributes (*homePhone, bPhone*), as well as the data type for the nested table of mobile numbers, VARCHAR2(15) was chosen as the international standard can support up to 15 characters. The data type NUMBER was used for all attributes that would be stored as only numbers and was chosen as it is Oracle’s recommended number-based data type. Finally, the DATE data type was chosen for the two date attributes (*joinDate* and *openDate* in the structured types ‘Job’ and ‘Account’ respectively).

**Inheritance Used and Why**

In the redesigned system, the structured subtypes ‘Employee’ and ‘Customer’ inherit from the supertype ‘Person’. The reason for this is that they shared many attributes in the original relational model and some of these were grouped into ‘Person’. If new attributes are required they can be added to the ‘Person’ supertype once (and therefore inherited by the two subtypes) so that time will be saved when updating the database.

**References Used and Why**

There are 4 references used in the proposed object-relational database: in the ‘CustomerAccount’ structured type there are references to *custID* in the ‘Customer’ type and to *accNum* in the ‘Account’ type, as well as two references made to *bID* in the ‘Branch’ type by the ‘Job’ type and the ‘Account’ type. These references were chosen as replacements to the foreign key attributes in the original relational database, allowing tuples to be referred to by separate structured types.

**Constraints Used and Why**

On most attributes the only constraint was the ‘NOT NULL’ constraint. This is to ensure that the fields are not empty so that each query returns actual data. There are, however, are few exceptions to this. 4 of the tables had a single ‘PRIMARY KEY’ constraint that corresponds to the primary keys used in the original relational database. In the ‘Account’ table the attribute *accType* is also constrained by a CHECK to ensure it is either a “current” or “savings” account. The *niNum* attributes in both the ‘Employee’ and ‘Customer’ tables have the ‘UNIQUE’ constraint to ensure that there are no duplicate values for that attribute within its respective table. Also in the ‘Employee’ table is the *position* attribute from the ‘Job’ type which uses the CHECK constraint to ensure it is one of the specified positions as described in the scenario: Head, Manager, Leader, Accountant, Cashier.

**Collections Used and Why**

In both the ‘Employee’ and ‘Customer’ entities in the relational database there were two attributes to store mobile phone numbers. For the object-relation database a nested table was created to store multiple mobile phone numbers and is used by the ‘Phone’ type; a nested table was chosen over a varray as it allows more mobile numbers to be added later if needed. A varray’s size must be specified when the attribute is defined and not changed later, this would cause issues if an employee or customer wished to add more mobile numbers than the varray could store.

**Task 3** - SQL Statements for Creating and Populating the Database

--Create name type

CREATE OR REPLACE TYPE name\_typ AS OBJECT (

title VARCHAR2(8),

firstName VARCHAR2(20),

surName VARCHAR2(20));

/

--Create address type

CREATE OR REPLACE TYPE address\_typ AS OBJECT (

street VARCHAR2(20),

city VARCHAR2(20),

p\_code VARCHAR2(8));

/

--Create nested table of mobile phones

CREATE OR REPLACE TYPE mobilePhones\_nested AS TABLE OF VARCHAR2(15)

/

--Create phone type

CREATE OR REPLACE TYPE phone\_typ AS OBJECT (

homePhone VARCHAR2(15)

mobilePhone mobilePhones\_nested);

/

--Create branch type for Branch table

CREATE OR REPLACE TYPE branch\_typ AS OBJECT (

bID NUMBER,

bAddress address\_typ,

bPhone VARCHAR2(15));

/

--Create job type

CREATE OR REPLACE TYPE job\_typ AS OBJECT (

position VARCHAR2(20),

salary NUMBER,

bID ref branch\_typ,

joinDate DATE);

/

--Create person type to be supertype to Customer and Employee

CREATE OR REPLACE TYPE person\_typ AS OBJECT (

pName name\_typ,

pAddress address\_typ,

pPhone phone\_typ,

niNum VARCHAR(20))

NOT FINAL;

/

--Create customer subtype for Customer table

CREATE OR REPLACE TYPE customer\_typ UNDER person\_typ (

custID NUMBER);

/

--Create employee subtype for Employee table

CREATE OR REPLACE TYPE employee\_typ UNDER person\_typ (

empID NUMBER,

supervisorID NUMBER,

eJob job\_typ);

/

--Create account type for Account table

CREATE OR REPLACE TYPE account\_typ AS OBJECT (

accNum NUMBER,

accType VARCHAR2(20),

balance NUMBER,

bID ref branch\_typ

inRate NUMBER,

limitOfFreeOD NUMBER,

openDate DATE);

/

--Create customerAccount type for CustomerAccount table

CREATE OR REPLACE TYPE customerAccount\_typ AS OBJECT (

custID ref customer\_typ,

accNum ref account\_typ);

/

--Create Branch table

CREATE TABLE branchTable OF branch\_typ (

bID PRIMARY KEY,

CONSTRAINT bStreet\_const CHECK (bAddress.street IS NOT NULL),

CONSTRAINT bCity\_const CHECK (bAddress.city IS NOT NULL),

CONSTRAINT bP\_code CHECK (bAddress.p\_code IS NOT NULL)

CONSTRAINT bPhone\_const CHECK (bPhone IS NOT NULL));

/

--Create Account table

CREATE TABLE accountTable OF account\_typ (

accNum PRIMARY KEY,

CONSTRAINT accType\_const CHECK(accType IN ("current", "savings")),

CONSTRAINT balance\_const CHECK(balance IS NOT NULL),

CONSTRAINT inRate\_const CHECK(inRate IS NOT NULL),

CONSTRAINT openDate\_const CHECK(openDate IS NOT NULL));

/

--Create CustomerAccount table

CREATE TABLE customerAccountTable OF customerAccount\_typ;

/

--Create Customer table

CREATE TABLE customerTable OF customer\_typ (

custID PRIMARY KEY,

CONSTRAINT cTitle\_const CHECK(pName.title IS NOT NULL),

CONSTRAINT cFirstName\_const CHECK(pName.firstName IS NOT NULL),

CONSTRAINT cSurName\_const CHECK(pName.surName IS NOT NULL),

CONSTRAINT cStreet\_const CHECK(pAddress.street IS NOT NULL),

CONSTRAINT cCity\_const CHECK(pAddress.city IS NOT NULL),

CONSTRAINT cP\_Code\_const CHECK(pAddress.p\_code IS NOT NULL),

CONSTRAINT cNiNum\_const UNIQUE(niNum));

/

--Create Employee table

CREATE TABLE employeeTable OF employee\_typ (

empID PRIMARY KEY,

CONSTRAINT supervisorID\_const CHECK(supervisorID IS NOT NULL),

CONSTRAINT eTitle\_const CHECK(pName.title IS NOT NULL),

CONSTRAINT eFirstName\_const CHECK(pName.firstName IS NOT NULL),

CONSTRAINT eSurName\_const CHECK(pName.surName IS NOT NULL),

CONSTRAINT eStreet\_const CHECK(pAddress.street IS NOT NULL),

CONSTRAINT eCity\_const CHECK(pAddress.city IS NOT NULL),

CONSTRAINT eP\_Code\_const CHECK(pAddress.p\_code IS NOT NULL),

CONSTRAINT eNiNum\_const UNIQUE(niNum),

CONSTRAINT ePosition\_const CHECK (eJob.position IN ("Head", "Manager", "Accountant", "Leader", "Cashier")),

CONSTRAINT eSalary\_const CHECK(eJob.salary IS NOT NULL),

CONSTRAINT eJoinDate\_const CHECK(eJob.joinDate IS NOT NULL));

/

**Task 4** - Database Queries

**Question 4a.**

Find employees with 'on' in first name who live in Glasgow:

SELECT

e.pName.firstName AS "First Name",

e.pName.surName AS "Last Name"

FROM

employeeTable e

WHERE

e.pAddress.city = 'Glasgow'

AND e.pName.firstName LIKE '%on%';

Output:

|  |  |
| --- | --- |
| First Name | Last Name |
| Bradon | Edwards |
| Jason | Armstrong |

**Question 4b.**

Find the number of savings account at each branch:

SELECT

COUNT(a.accType) AS "Number of Savings Accounts",

a.bID.bAddress.street AS "Branch Street",

a.bID.bAddress.city AS "Branch City",

a.bID.bAddress.p\_code AS "Branch Post Code"

FROM

accountTable a

WHERE

a.accType = 'savings'

GROUP BY a.bID;

Output:

|  |  |  |  |
| --- | --- | --- | --- |
| Number of Savings Accounts | Branch Street | Branch City | Branch Post Code |
| 2 | Westerfield | Zhenghu | WN8 OW0 |
| 1 | Mariners Cove | Föglö | RZ4 XA0 |
| 1 | Chive | Vereya | GD0 IQ8 |
| 1 | Scoville | Buka | IH0 XO9 |
| 1 | Trailsway | Yashalta | NL5 QW0 |
| 1 | Northfield | Rey | ZS6 DK4 |
| 1 | Commercial | Santo Antonio do Monte | BD9 HG2 |
| 1 | Moland | Samashki | AG6 JX8 |
| 2 | Magdeline | Sumqayit | KF6 RN6 |
| 1 | Hauk | Longjin | GU6 FQ8 |

**Question 4g.**

Find the number of eployees who are supervised by Mr William, who is supervised by Mrs Smith:

SELECT

CONCAT(CONCAT(e.pName.title, ' '), e.pName.surName) AS "Supervisor Name",

(SELECT COUNT(e.supervisorID) FROM employeeTable e WHERE e.supervisorID.empID = 301) AS "Number of Employees Supervised",

(SELECT CONCAT(CONCAT(e.supervisorID.pName.title, ' '), e.supervisorID.pName.surName) FROM employeeTable e WHERE e.supervisorID.empID = 309) AS "Supervisor's Supervisor"

FROM

employeeTable e

WHERE

e.empID = 301; **Task 5** - A Critical Analysis of the Object-Relational Model Compared Against the Relational Model

Relational

Adv -

Disadv -

Object

Adv -

Disadv -

**Task 6** - Drop Statements

--DROP TYPES

DROP TYPE name\_typ FORCE;

/

DROP TYPE address\_typ FORCE;

/

DROP TYPE mobilePhones\_nested FORCE;

/

DROP TYPE phone\_typ FORCE;

/

DROP TYPE branch\_typ FORCE;

/

DROP TYPE job\_typ FORCE;

/

DROP TYPE person\_typ FORCE;

/

DROP TYPE customer\_typ FORCE;

/

DROP TYPE employee\_typ FORCE;

/

DROP TYPE account\_typ FORCE;

/

DROP TYPE customerAccount\_typ FORCE;

/

--DROP TABLES

DROP TABLE branchTable PURGE;

/

DROP TABLE accountTable PURGE;

/

DROP TABLE customerAccountTable PURGE;

/

DROP TABLE customerTable PURGE;

/

DROP TABLE employeeTable PURGE;

/