BEMM459J - Database Technologies for Business Analytics

Professor Nav Mustafee

INDIVIDUAL ASSIGNMENT ON

POLYGLOT PERSISTENCE

Candidate number: 091713

TABLE OF CONTENTS

A.1 CHOOSE AN ORGANISATION

A.1.1. DESCRIPTION OF THE PROBLEM TO SOLVE

Constructora Kanono S.A. de C.V. (http://www.kanono.mx/) is a company with 20 years of operation that focuses on managing construction projects and supervision. The company receive and generates hundreds of invoices a year. These invoices are generated by different departments that generally have their own computer system or traditional file-based systems in the best of cases. In the worst of cases, they keep the information in folders with physical papers. What is more, there are official invoices created by the government and invoices that do not have government intervention. This has generated two significant problems. The first is that it is not known with certainty how much clients owe, and it is not possible to visualise how much a company owes, how much it owes per project or dates, etc. This leads to project delays, inaccurate and late analysis, which translates into poor decision making. The other problem is that due to the nature of the business, it is necessary to have quick access to the invoices, either by the government or by the clients; they can request invoices generated ten years ago.

A.1.2. USE OF THE APPLICATION

For this reason, I will create an invoicing module for Constructora Kanono S.A. of CV, where you can store the invoices generated by the government (XML), and record the non-government invoices generated in paper or PDF. In both types of invoices, payments made can be recorded, and several visualisations can be performed. Such as pending invoices by project, by company, by dates, etc.

Over time it could grow to an enterprise resource planning (ERP) specialised for the construction industry in Mexico. This application will be built on top of a database management system (DBMS) that integrates the required business functions such as accounting and finance, banks, Invoicing, Customer Relationship Management (CRM), inventory, purchase, construction management, Material Requirements Planning (MRP) and human resources. It is expected to have visibility into transactions, improve relationships with customers and suppliers, reduce processing costs, time and errors, allowing employees to work in an integrated and collaborative manner.

A.1.3. TYPE OF DATABASES AND JUSTIFICATION

RDBMS and NoSQL databases will be used for the development of the invoicing module. SQL to register the payments, input data for non-government invoices and the NoSQL to store government invoices.

The use of DBMS has essential advantages such as consistency, redundancy control, space optimisation, data integrity, greater security, and finally, improvement of data accessibility, creating different types of data visualisation and therefore increasing productivity.

Specifically, the SQL database will be used for its ACID properties; for Constructora Kanono S.A. de C.V it is essential to keep control of payments as an internal control. Atomicity is required, so new changes to an invoice or payment are not allowed until a transaction is complete; consistency is essential to maintain integrity constraints.

MongoDB (NoSQL) database will be used to store government invoices. The Tax Administration Department in Mexico (SAT for its acronym in Spanish) generates invoices in two files, a PDF and an XML with a digital signature (Base64) necessary to verify the authenticity of an invoice. It is necessary to have access to these data in case of inquiries made by suppliers or clients, or a government; these invoices do not have a specific structure, there are different versions over time; finally, it is essential to think about scalability. These invoices have to be kept for decades.

A.1.4. OTHER RELEVANT INFORMATION

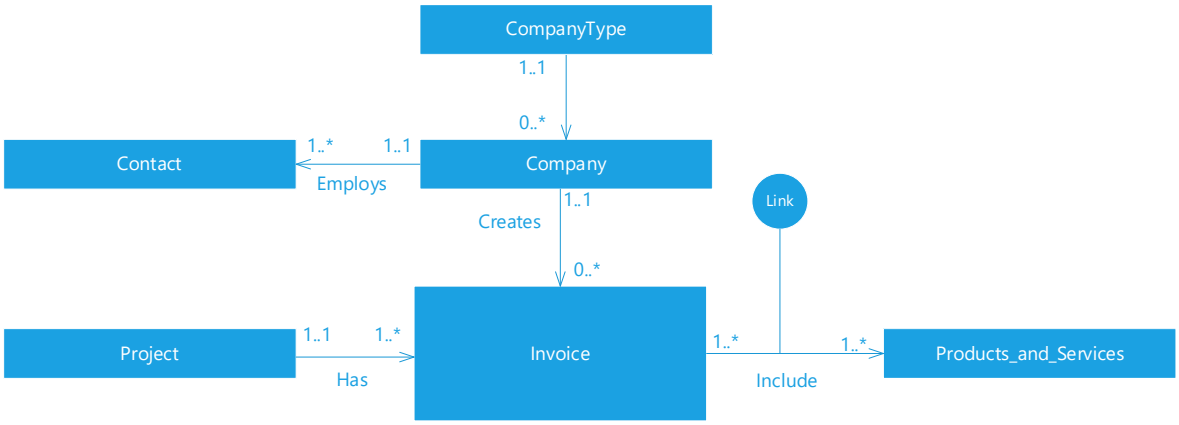
The government invoices are in XML format this will be transformed in Python so they can be stored in MongoDB as a JSON file for future retrieval, and at the same time, that invoice will be saved into the SQL database to keep track of payments and be able to visualise it with the non-government invoices. The non-government invoices will be stored directly in SQL. As the application will run on Python, we will take advantage of the different libraries such as Pandas or Matplot. The main objective of this application is to process and present information to make the best decisions and courses of action.

It is important to note that this will be an actual application under the requirements of Constructora Kanono S.A. de C.V. This is the first step in future developments I will add more entities, attributes and deploy it as a Software as a service (SaaS), and the government invoices will be added through an API (https://www.facturapi.io/) that automatically downloads the invoices created, because of time, data privacy and cost issues it will not be included at this time.

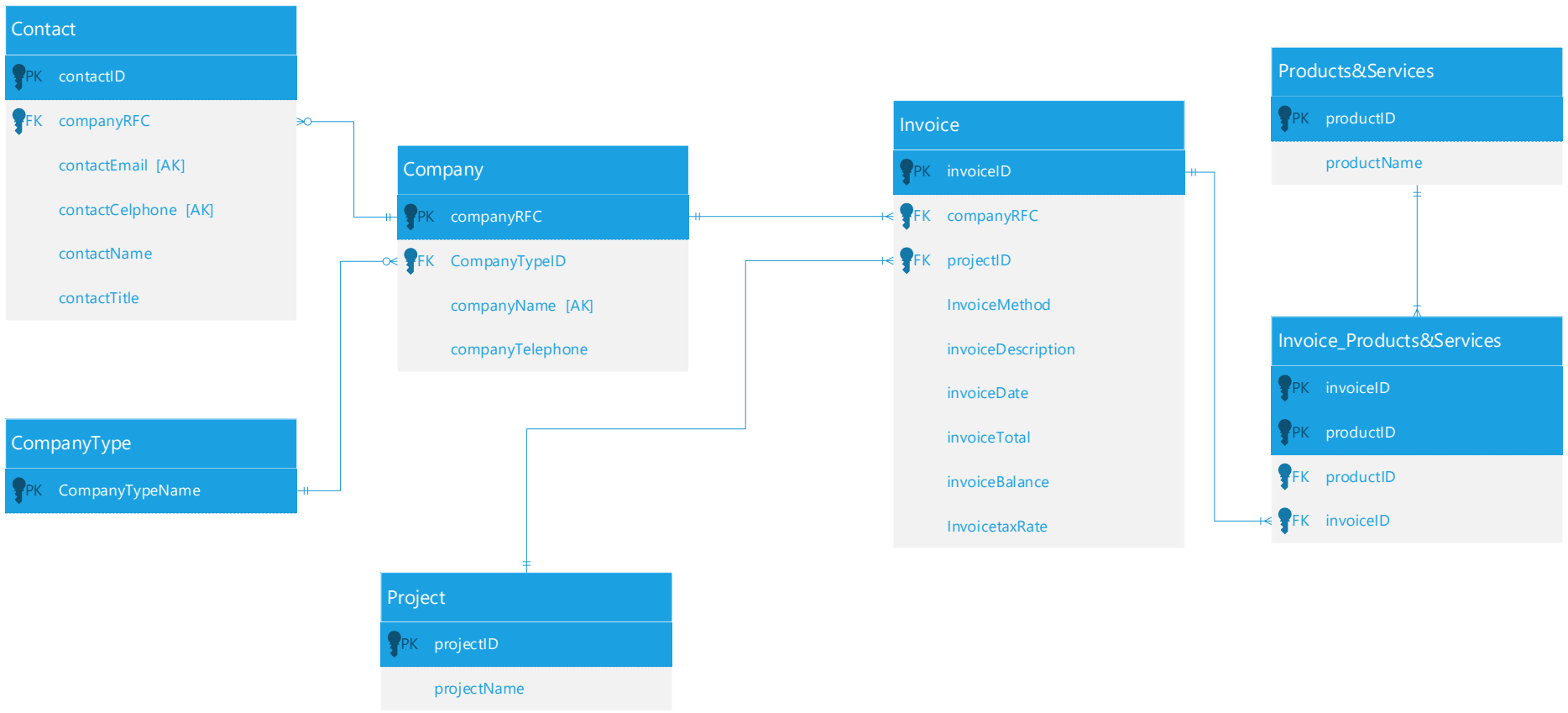
A.2. CONCEPTUAL DATABASE DESIGN (ER MODEL) WEEKS 2 AND 3

A.2.1 ENTITY-RELATIONSHIP DIAGRAM MODEL (ENTITIES)

The following diagrams show the entities of the RDBMS: The diagrams contain the primary information that the invoice must have plus contacts that is the start to be able to develop a Customer relationship management (CRM). For this stage, it is only essential to have the description of the invoice, the total amount, the amount paid. Regarding Products and Services, we need to know which Products and Services are included (according to a government list). However, it is not important to have the price and quantity of each one.



A.2.2 ENTITY RELATIONSHIP DIAGRAM MODEL (ENTITIES AND ATTRIBUTES)



A.2.3 DOMAIN ATTRIBUTES AND PRIMARY {PK} /ALTERNATE KEYS [AK]

Conceptual modelling stage Domain attributes.

Entity	Attribute Name	Description	Domain attribute and/or Data type	Null	Multi-values
Invoice	invoiceID (Sarah Allen)	Uniquely identifies each invoice	Integer Autoincrement	No	No
	invoiceDescription	Invoice Description	Text (30 characters)	No	No
	invoiceDate	Invoice Date	Date format	No	No
	invoiceTotal	Total Amount of the Invoice	Numeric(10,2)	No	No
	invoiceBalance	Amount Paid of the Invoice	Text (30 characters)	Yes	No
	invoiceMethod	Invoice Method	Text. Value can be Cash,Cheque,Wire Transfer, Credit Card, Debit Card	No	No
	invoicetaxRate	Tax Rate	Value can be either .16 or 0	No	No
Company	companyRFC (Sarah Allen)	Uniquely string that identifies every company	Text (13)	No	No
	companyName	Company legal name	Text (30 characters)	No	No
	companyTelephone	Company telephone	Number (10 characters)	Yes	No
ComanyType	companyTypeName (Sarah Allen)	Company type . Can be Supplier or Customer	Text	No	No
Project	projectID (Sarah Allen)	Uniquely identifier each project	Integer	No	No
	projectName	Name of the project	Text (30 characters)	No	No
Contact	contactID (Sarah Allen)	Uniquely identifies every contact	Integer	No	No
	contactEmail [AK]	Contact Email	Text (30 characters) and must include '@'	No	No
	contactCelphone [AK]	Contact Celphone	Integer (10 characters)	No	No
	contactName	Contact Name	Text (30 characters)	No	No
	contactTitle	Contact Title	Text (30 characters)	Yes	No
Product&Services	productID (Sarah Allen)	Uniquely identifies each product or service	Integer	No	No
	productName	Product/Service name	Text (30 characters)	No	No

A.2.4 LOGICAL DATABASE DESIGN WITH 3NF AND REFERENCIAL INTEGRITY

Logical database design using conform to the first, second and third normal forms and referential integrity constraints.

Table 1:	
Project	(projectID, projectName)
Primary key	projectID
Table 2:	
CompanyType	(CompanyTypeName)
Primary key	CompanyTypeName
Table 3:	
Company	(companyRFC, companyName, companyTelephone, companyTypeID)
Primary key	companyRFC
Alternate key	companyName
Foreign key	CompanyTypeID REFERENCES CompanyType (CompanyTypeName) ON DELETE CASCADE ON UPDATE CASCADE
Table 4:	
Contact	(contactID, contactEmail, contactCelphone, contactName, contactTitle, companyRFC)
Primary key	contactID
Alternate key	contactEmail
Alternate key	contactCelphone
Foreign key	(companyRFC) REFERENCES Company(companyRFC) ON UPDATE CASCADE ON DELETE CASCADE
Table 4:	
Invoice	(invoiceID, invoiceDescription, invoiceDate, invoiceTotal, invoiceBalance, invoiceMethod, invoicetaxRate, companyRFC, projectID)
Primary key	invoiceID
Foreign key	(companyRFC) REFERENCES Company(companyRFC) ON UPDATE RESTRICT ON DELETE RESTRICT
Foreign key	(projectID) REFERENCES Project(projectID) ON UPDATE RESTRICT ON DELETE RESTRICT
Table 5:	
Product&Services	(productID, productName)
Primary key	productID
Table 6:	
Invoice_Products&Services	(invoiceID, productID)
Primary key	invoiceID, productID
Foreign key	(invoiceID) REFERENCES Invoice(invoiceID) ON UPDATE RESTRICT ON DELETE RESTRICT
Foreign key	(productID) REFERENCES Products&Services(productID) ON UPDATE RESTRICT ON DELETE RESTRICT

A.2.5 DDL(DATA DEFINITION LANGUAGE)

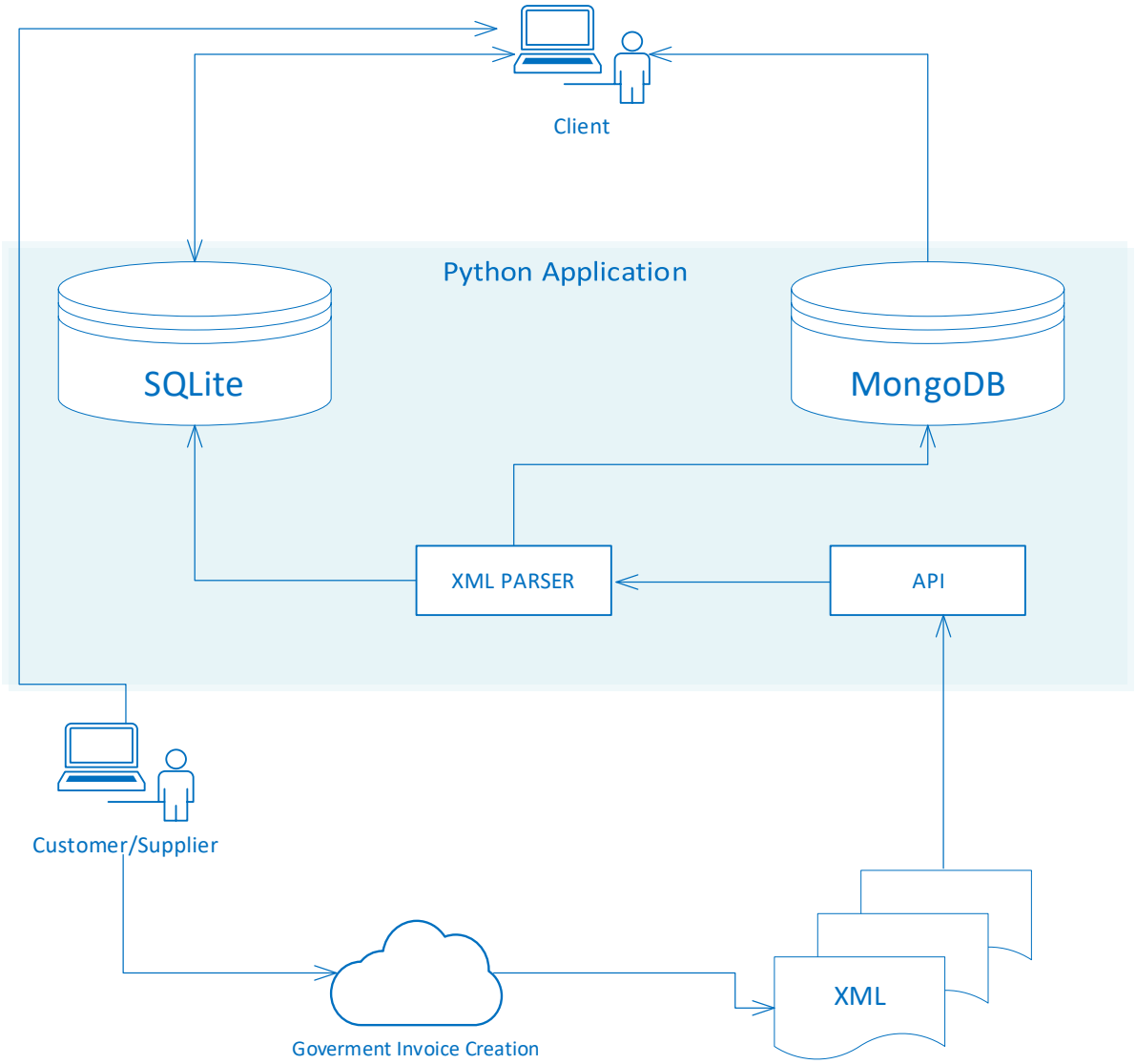
```
create database invoices;
use invoices;

create table invoices
. open "Z:\\OneDrive\\Python\\BEMM459_RDBMS_NoSQL-main\\TareaFinal\\invoicing.db"

PRAGMA foreign_keys = ON;

CREATE TABLE Project (
 projectID   INTEGER   PRIMARY KEY AUTOINCREMENT,
 projectName TEXT (30) NOT NULL
);

CREATE TABLE CompanyType (
 CompanyTypeID TEXT PRIMARY KEY
      NOT NULL
);

CREATE TABLE Company(
companyRFC TEXT (13) PRIMARY KEY,
companyName TEXT (30) NOT NULL,
companyTelephone INTEGER,
companyType TEXT CHECK( companyType IN ('Supplier','Customer')) NOT NULL
);

CREATE TABLE Company (
companyRFC TEXT (13) PRIMARY KEY,
companyName TEXT (30) NOT NULL,
companyTelephone INTEGER,
CompanyTypeID INTEGER   REFERENCES CompanyType (CompanyTypeID) ON DELETE CASCADE
ON UPDATE CASCADE NOT NULL
);

CREATE TABLE Contact (
contactID INTEGER   PRIMARY KEY AUTOINCREMENT,
companyRFC TEXT (13),
contactName TEXT (30) NOT NULL,
contactTitle TEXT (30),
contactEmail TEXT (20) NOT NULL,
contactCelphone INTEGER   NOT NULL,
FOREIGN KEY (companyRFC) REFERENCES Company (companyRFC) ON UPDATE CASCADE
ON DELETE CASCADE
);


CREATE TABLE Invoice (
invoiceID INTEGER   PRIMARY KEY AUTOINCREMENT,
companyRFC TEXT (13),
projectID INTEGER,
invoiceDescription TEXT (30) NOT NULL,
invoiceDate DATE NOT NULL,
invoiceTotal NUMERIC (10, 2) NOT NULL,
invoiceBalance NUMERIC (10, 2) NOT NULL CHECK (invoiceBalance >= 0),
invoiceMethod TEXT CHECK (invoiceMethod IN ('Cash', 'Cheque', 'Wire Transfer', 'Credit Card', 'Debit Card'))
NOT NULL DEFAULT 'Wire Transfer',
invoicetaxRate REAL CHECK (invoicetaxRate IN (0, 0.16) ) NOT NULL DEFAULT '.16',
FOREIGN KEY (companyRFC) REFERENCES Company (companyRFC) ON UPDATE RESTRICT ON DELETE RESTRICT,
FOREIGN KEY (projectID) REFERENCES Project (projectID) ON UPDATE RESTRICT ON DELETE RESTRICT
);

CREATE TABLE Invoice_ProductsServices (
invoiceID INTEGER NOT NULL,
productID INTEGER NOT NULL,
PRIMARY KEY (invoiceID, productID),
FOREIGN KEY (invoiceID) REFERENCES Invoice (invoiceID) ON UPDATE RESTRICT ON DELETE RESTRICT,
FOREIGN KEY (productID) REFERENCES ProductsServices (productID) ON UPDATE RESTRICT
ON DELETE RESTRICT
);

CREATE TABLE ProductsServices (
productID   INTEGER   PRIMARY KEY AUTOINCREMENT,
productName TEXT (30) NOT NULL
);
```

A.3. NOSQL

The NoSQL database (MongoDB) will be used to store government invoices. Each invoice is a document which is an XML file. Therefore it will be transformed into a JSON file and store in a single collection. It is crucial noting that those invoices cannot be deleted or updated once they are stored as they represent a legal document. In future developments, it has to be added a function to detect if there are invoice duplicates. It can be done by running a simple cryptographic hash function to the Digital Signature and compare it to new added invoices. Also, it is essential to detect if an invoice was cancelled and change the status. Even cancelled invoices has to be stored.

A.4. IMPLEMENTATION OF POLYGLOT PERSISTENCE

1. An invoice is created by a Supplier or to a Customer. If it is a non-government invoice, it will be manually input into the SQL database; if it is a government invoice, it will be created in the cloud, where it will be digitally signed.
2. The XML invoice will be automatically downloaded by an API (in the future). It will be stored in the MongoDB database and in the SQL database. The attributes that are Not Null will be asked before (like project), or the invoice will be created with default values (like tax rate).
3. The final user (client) will be able to update data and balances in the SQL database and visualise data in the SQL and NoSQL databases.



B. SUPPORTING DOCUMENTATION

B.1. VIDEO PRESENTATION

5 min video.

https://youtu.be/gk2-6PdH44k

B.2. CODE SUBMISSION

Files are in the following link.

https://www.dropbox.com/sh/aqarsuewzbgl9wg/AADKgcGT8V_HR9jozN9SPB3Ma?dl=0

5. REFERENCES

Connolly, T., Begg , C. E., & Holowczak , R. (2008). Business Database Systems. In *Business Database Systems*. Addison-Wesley Publishing Company.

How to insert XML data to MongoDB using Python. (2020). https://www.etutorialspoint.com/index.php/292-how-to-insert-xml-data-to-mongodb-using-python

Sarah Allen, S. C., Ittay Eyal, Giulia Fanti, Bryan Ford, James Grimmelmann, Ari Juels, Kari Kostiainen, Sarah Meiklejohn, Andrew Miller. (2020). Design Choices for Central Bank Digital Currency

Sullivan, D. (2015). *NoSQL for mere mortals* https://go.oreilly.com/yale-university/library/view/-/9780134029894/?ar