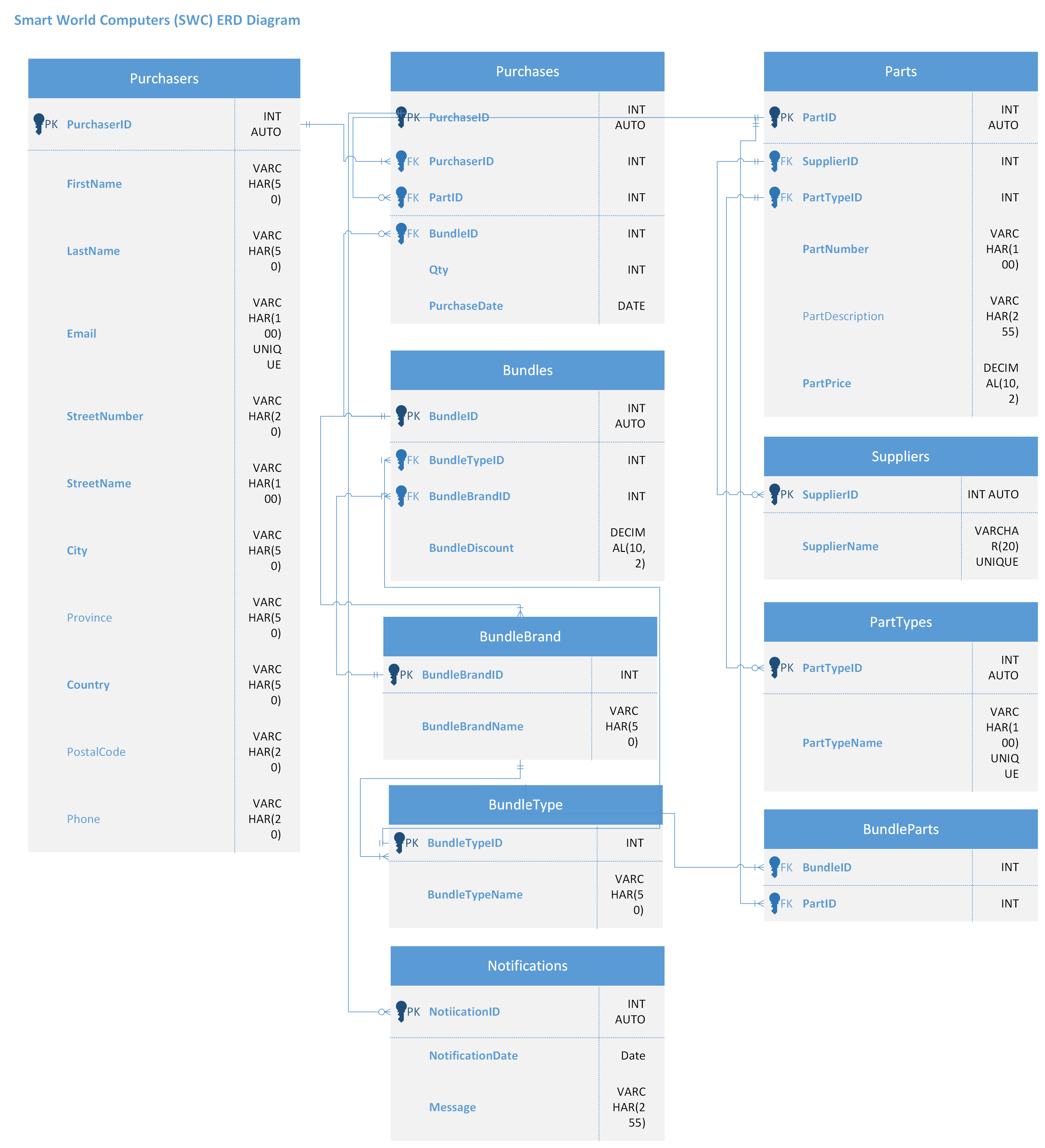
**Smart World Computers (SWC) ERD Diagram**

Completed by:   
Jaanvi Jaanvi, Zainab el Bukhatwa, Youyuan Zhang, Ravi Jha (Observer), Marina Paskal



**Constraints**

**-- PurchaserID = PK**

**-- Purchaser first name length >= 3**

**-- Unique email**

**-- Default country name**

**-- Phone number length == 10**  
CREATE TABLE Purchasers (

PurchaserID INT IDENTITY(1,1) PRIMARY KEY,

**FirstName VARCHAR(50) NOT NULL CHECK (LEN(FirstName) >= 3), -- Purchaser first name length >= 3**

LastName VARCHAR(50) NOT NULL,

**Email VARCHAR(100) NOT NULL UNIQUE, --unique email**

StreetNumber VARCHAR(20) NOT NULL,

StreetName VARCHAR(100) NOT NULL,

City VARCHAR(50) NOT NULL,

Province VARCHAR(50) NOT NULL,

**Country VARCHAR(50) DEFAULT 'Canada',**

PostalCode VARCHAR(20) NOT NULL,

**Phone VARCHAR(20) NOT NULL CHECK (LEN(Phone) = 10)-- Index for Performance**

);

**CREATE INDEX IDX\_Phone ON Purchasers(Phone); -- Index for Performance**

CREATE UNIQUE INDEX IDX\_Email ON Purchasers(Email); -- Create unique index for the purchaser's email

**-- PartID PK**

**-- PartTypeID FK**

**-- SupplierID FK**

**-- Part description length >= 3**

**-- Unit price cannot be negative**

CREATE TABLE Parts (

PartID INT IDENTITY(1,1) PRIMARY KEY,

SupplierID INT NOT NULL FOREIGN KEY REFERENCES Suppliers(SupplierID),

PartTypeID INT NOT NULL FOREIGN KEY REFERENCES PartTypes(PartTypeID),

PartNumber VARCHAR(100) NOT NULL UNIQUE,

**PartDescription VARCHAR(255) NOT NULL CHECK (LEN(PartDescription) >= 3), --Part description length >= 3**

PartUnitPrice DECIMAL(10, 2) NOT NULL CHECK (PartUnitPrice >= 0) **-- Unit price cannot be negative**

);

**-- PurchaseID PK**

**-- PurchaserID FK**

**-- PartID FK**

**-- BundleID FK**

**-- Quantity must be positive  
-- Only PartID or BundleID can be selected**

CREATE TABLE Purchases (

PurchaseID INT IDENTITY(1,1),

PurchaserID INT NOT NULL,

PartID INT NULL,

BundleID INT NULL,

PurchaseDate DATE NOT NULL,

Qty INT NOT NULL CHECK (Qty > 0), **-- quantity must be positive**

FOREIGN KEY (PurchaserID) REFERENCES Purchasers(PurchaserID),

FOREIGN KEY (PartID) REFERENCES Parts(PartID),

FOREIGN KEY (BundleID) REFERENCES Bundles(BundleID),

CONSTRAINT CHK\_PartOrBundle

CHECK ((PartID IS NULL AND BundleID IS NOT NULL) OR (PartID IS NOT NULL AND BundleID IS NULL))

);

**-- PartTypeID PK**

CREATE TABLE PartTypes (

PartTypeID INT IDENTITY(1,1) PRIMARY KEY,

PartTypeName VARCHAR(50) NOT NULL UNIQUE

);

**-- BundleBrandID PK**

CREATE TABLE BundleBrand (

BundleBrandID INT IDENTITY(1,1) PRIMARY KEY,

BundleBrandName VARCHAR(50) NOT NULL UNIQUe

);

**-- BundleTypeID PK**

CREATE TABLE BundleType (

BundleTypeID INT IDENTITY(1,1) PRIMARY KEY,

BundleTypeName VARCHAR(50) NOT NULL UNIQUE

);

**-- SupplierID PK**

CREATE TABLE Suppliers (

SupplierID INT IDENTITY(1,1) PRIMARY KEY,

SupplierName VARCHAR(120) NOT NULL UNIQUE

);

**-- BundleID PK  
-- BundleTypeID FK**

**-- BundleBrandID FK**  
CREATE TABLE Bundles (

BundleID INT IDENTITY(1,1) PRIMARY KEY,

BundleTypeID INT NOT NULL FOREIGN KEY REFERENCES BundleType(BundleTypeID),

BundleBrandID INT NOT NULL FOREIGN KEY REFERENCES BundleBrand(BundleBrandID),

BundleDiscount DECIMAL(10, 2) NOT NULL

);

**-- BundleID and PartID make CK  
-- BundleID FK**

**-- PartID FK**

CREATE TABLE BundleParts (

BundleID INT NOT NULL FOREIGN KEY REFERENCES Bundles(BundleID),

PartID INT NOT NULL FOREIGN KEY REFERENCES Parts(PartID),

CONSTRAINT CK\_BundleParts PRIMARY KEY (BundleID, PartID) -- Define composite key constraint

);

**-- NotificationID PK**

CREATE TABLE Notifications (

NotificationID INT IDENTITY(1,1) PRIMARY KEY,

NotificationDate DATETIME2 DEFAULT GETDATE(),

Message VARCHAR(255));  
  
**Views:**-- Question 10 Create a View to Display all purchase details with extended amount

CREATE VIEW View\_PurchaseDetailsWithExtendedAmount AS

SELECT

p.PartNumber,

s.SupplierName,

pt.PartTypeName,

p.PartDescription,

pcr.FirstName,

pc.PurchaseDate,

p.PartUnitPrice,

pc.Qty,

p.PartUnitPrice \* pc.Qty AS Extended\_Amount

FROM

(SELECT

t1.PurchaseID,

t1.PurchaserID,

COALESCE(t1.PartID, t2.PartID) AS PartID,

t1.BundleID,

t1.PurchaseDate,

t1.Qty

FROM Purchases t1

LEFT JOIN BundleParts t2 ON t1.BundleID = t2.BundleID

WHERE COALESCE(t1.PartID, t2.PartID) IS NOT NULL) pc

JOIN Parts p ON pc.PartID = p.PartID

JOIN Purchasers pcr ON pc.PurchaserID = pcr.PurchaserID

JOIN Suppliers s ON p.SupplierID = s.SupplierID

JOIN PartTypes pt ON p.PartTypeID = pt.PartTypeID;

GO

-- Test Question 10 View to Display all purchase details with extended amount

print 'Test Question 10 View to Display all purchase details with extended amount';

SELECT \*

FROM View\_PurchaseDetailsWithExtendedAmount

ORDER BY FirstName, PurchaseDate, PartNumber;

GO

-- Question 11\_A Create a View to display desktop bundles total coast

CREATE VIEW vw\_DesktopBundlesTotalCost AS

SELECT bp.BundleID, SUM(p.PartUnitPrice) AS total\_cost

FROM BundleParts bp JOIN Parts p ON bp.PartID = p.PartID

GROUP BY bp.BundleID;

Go

-- Test Question 11\_A DesktopBundleTotalCost view

print 'Test Question 11\_A DesktopBundleTotalCost view';

SELECT \* FROM vw\_DesktopBundlesTotalCost

ORDER BY BundleID;  
  
**Stored Procedure and Trigger**

--Question 11\_B Stored Procedure "adds a new purchase record to the Purchases table."

GO

print 'create a Stored Procedure to add a new purchase record to the Purchases table.';

CREATE PROCEDURE sp\_AddNewPurchase

@purchaserID INT,

@partID INT = NULL, -- Default NULL means it might be a bundle pwe4waurchase

@bundleID INT = NULL, -- Default NULL means it might be a part purchase

@purchaseDate DATE,

@qty INT

AS

BEGIN

-- Validate inputs to respect CHK\_PartOrBundle constraint

IF (@partID IS NOT NULL AND @bundleID IS NOT NULL) OR (@partID IS NULL AND @bundleID IS NULL)

BEGIN

RAISERROR('Each purchase must be for either a part or a bundle, not both or neither.', 16, 1);

RETURN;

END

-- Insert the new purchase record

INSERT INTO Purchases (PurchaserID, PartID, BundleID, PurchaseDate, Qty)

VALUES (@purchaserID, @partID, @bundleID, @purchaseDate, @qty);

END;

GO  
  
-- To execute the procedure using script  
  
USE SWC\_DB;

GO

EXEC sp\_AddNewPurchase

@purchaserID = 1,

@partID = 1,

@bundleID = 5,

@purchaseDate = '2024-09-01',

@qty = 1;

--Question 11\_B Trigger "inserts a purchaser detail into the Notifications table whenever a new purchase with a quantity greater than 100 is made in the Purchases table."

print '\*\*\* create trigger to inserts a purchaser detail into the Notifications table whenever a new purchase with a quantity greater than 100 is made in the Purchases table. \*\*\* ';

CREATE TRIGGER trg\_LargePurchaseNotification

ON Purchases

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

-- Check if the inserted purchase has a Qty greater than 100

IF EXISTS (SELECT \* FROM inserted WHERE Qty > 100)

BEGIN

INSERT INTO Notifications (Message)

SELECT 'Large purchase made by PurchaserID ' + CAST(i.PurchaserID AS VARCHAR) + ' on ' + CAST(i.PurchaseDate AS VARCHAR) + '. Quantity: ' + CAST(i.Qty AS VARCHAR)

FROM inserted i

WHERE i.Qty > 100;

END

END;

GO  
  
-- Check the Notifications Table  
  
-- Select \* from Notifications;