CS_340_400_Summer2016
7 Aug 2016
Chris Kearns
Final Project: http://web.engr.oregonstate.edu/~kearnsc/Diamonds/

Outline:

This project is a greatly simplified local block chain node for trading diamonds and in particular, Diamond Buyers Inc., a fictitious firm engaging in the facilitation of diamond identification, grading, ownership transfer with buyers and sellers, the resulting commissions, and ultimately, the full traceability of each Diamond asset though it's entire life cycle from initial fabrication or identification date up to and including the present. This project can easily be converted for use with any high value asset that is subject to fraudulent copying, transfer, or substitution with inferior "knock offs", etc. Potential uses include titanium alloy aircraft parts, digital assets, financial securities, obligation contracts (insurance) and many more.

The goal is to be able to create, transact, and report the "Blockchain" or "Ownership Chain" of various current and previous ownership attributes while providing a level of confidentiality about who is/was or will be the owner of the contracted asset, similar in fashion to most financial instrument transactions as carried out via trusted third parties today (banks, clearing houses, stock exchanges, etc.) Proof of this project achieving this goal can be seen by selecting an asset and then clicking the "Show Blockchain for This Asset" button at the bottom of the scrollable menu initially rendered by the above URL (showLedgerByID.php).

What is omitted is the distributed instance of the Ledger and the individual block transaction / concurrence / consolidation mechanisms over multiple nodes (servers), which I intend to continue to develop.

Database Outline In Words:

A Ledger has an id and a date_time stamp where each id uniquely represents a consummated Contract in the Ledger. Ledger.id is referenced by Contract.L_ID.

A Contract has an id, Asset_ID number, Buyers Account number (B_Acct_ID) , Seller's Account number (S_Acct_ID), Transaction price (Trans_at), Commission paid on the transaction (Com_pd), Effective Date (Eff_date), and a Ledger Id (L_ID) number. Contracts are uniquely identified by their id. *Contract.L_ID references Ledger.id. Contract.id is referenced by Contract_Asset.Contract_ID and Contract_Customer.Contract_ID.*

Assets have an id, Description, Carrot, Cut, Clarity, Color, Create_Date, and Owned_By attributes. The Asset's id uniquely identifies it. In practice, a diamond can have an identifying id that easily has as many as 20 numerals and additionally this id can be an alpha-numeric string. (I elected to go with BIGINT for now.) *Asset.id is referenced by Contract_Asset.Asset.id.*

Customers have id, Fname, Lname, Addr_1, Addr_2, City, State, Zip, and Phone attributes. The id uniquely identifies each Customer. *Customers.id is referenced by Contract_Customers.Customer_ID and Account.C_ID.*

Account(s) have an id, Customer id (C_ID), and a Balance. The id uniquely identifies each Account. *Account.C_ID references Customers.id.*

There is one Ledger that relates to many Contract(s). All Contract(s) must be included in the Ledger.

Each Asset (Diamonds) can relate to many Contract(s) as the Asset changes ownership over time. A Contract must associate with at least one Asset* and every Asset must have at least one Contract associated with it. (A onetime exception exists for each Asset when it is first entered into the "system" and assigned to an owner, as one cannot programmatically instance a Contract on an Asset that does not yet exist). This exception lasts for the period of time before a Contract is first consummated on the new Asset.

* A contract can be consummated and recorded for more than one Asset at a time as a "lot". This can be handled as an array of Asset id's, and to this end I have included a junction table for the Asset - Contract relationship but could have gotten away with simply referencing Asset.id to Contract.Asset_ID. Since the implementation, as seen in RecordTrans.php currently only handles one Asset per contract, the ER Diagram reflects this as a 1 to many relationship.

Each of the many Contract(s) must be associated with two distinct Customers that cannot be the same. [*Since a trigger is required to do this, and as a student, I can't enable a trigger, I handled this constraint with php. See course Piazza post 96.*] A Customer may have 0 to many Contracts associated with themselves.

Each Customer must have at least one Account. Each Account has one Customer associated with it (notwithstanding joint accounts, etc. as Accounts have a unique id attribute.)

Table Creation Queries (Contents of Blockchain-definitions.sql less comments):

```
SET foreign_key_checks = 0;
DROP TABLE IF EXISTS Customers;
DROP TABLE IF EXISTS Account;
DROP TABLE IF EXISTS Asset;
DROP TABLE IF EXISTS Contract;
DROP TABLE IF EXISTS Ledger;
DROP TABLE IF EXISTS Contract_Customers;
DROP TABLE IF EXISTS Contract_Asset;
SET foreign_key_checks = 1;

CREATE TABLE Customers (
        id INT AUTO_INCREMENT PRIMARY KEY,
        Lname VARCHAR(30) NOT NULL,
        Fname VARCHAR(30) NOT NULL,
```

```
        Addr_1 VARCHAR(30) NOT NULL,
        Addr_2 VARCHAR(30),
        City VARCHAR(25) NOT NULL,
        State VARCHAR(2) NOT NULL,
        Zip INT UNSIGNED NOT NULL,
        Phone BIGINT UNSIGNED NOT NULL
)ENGINE=InnoDB AUTO_INCREMENT = 200000;

CREATE TABLE Account (
        id INT AUTO_INCREMENT PRIMARY KEY,
        C_ID INT NOT NULL,
        Balance DECIMAL(10,2) NOT NULL,
        INDEX Customer_Account_index(C_ID),
        FOREIGN KEY(C_ID) REFERENCES Customers(id) ON DELETE CASCADE ON UPDATE CASCADE
)ENGINE=InnoDB AUTO_INCREMENT = 396400;

CREATE TABLE Asset (
        id BIGINT AUTO_INCREMENT PRIMARY KEY,
        Description VARCHAR(255) NOT NULL,
        Carrot DECIMAL(7,4) NOT NULL,
        Cut VARCHAR(10) NOT NULL,
        Clarity VARCHAR(5) NOT NULL,
        Color CHAR NOT NULL,
        Create_Date DATE NOT NULL,
        Owned_By INT
)ENGINE=InnoDB AUTO_INCREMENT = 500000;

CREATE TABLE Ledger (
        id INT AUTO_INCREMENT PRIMARY KEY,
        date_time DATETIME NOT NULL
)ENGINE = InnoDB AUTO_INCREMENT = 1000;

CREATE TABLE Contract(
        id INT AUTO_INCREMENT PRIMARY KEY,
        Asset_ID BIGINT UNSIGNED NOT NULL,
        B_Acct_ID INT UNSIGNED NOT NULL,
        S_Acct_ID INT UNSIGNED NOT NULL,
        Eff_Date DATETIME NOT NULL,
        Trans_at DECIMAL(13,2) UNSIGNED NOT NULL,
        Com_pd DECIMAL(12,2) UNSIGNED NOT NULL,
        L_ID INT NOT NULL,
        FOREIGN KEY (L_ID) REFERENCES Ledger(id)
                ON DELETE CASCADE ON UPDATE CASCADE
)ENGINE=InnoDB AUTO_INCREMENT = 12000;

CREATE TABLE Contract_Customers (
        Contract_ID INT NOT NULL,
        Customer_ID INT NOT NULL,
```

```
        PRIMARY KEY(Contract_ID, Customer_ID),
        FOREIGN KEY (Contract_ID) REFERENCES Contract(id)
                ON DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY (Customer_ID) REFERENCES Customers(id)
                ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE Contract_Asset (
        Contract_ID INT NOT NULL,
        Asset_ID BIGINT NOT NULL,
        PRIMARY KEY(Contract_ID, Asset_ID),
        FOREIGN KEY (Contract_ID) REFERENCES Contract(id)
                ON DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY (Asset_ID) REFERENCES Asset(id)
                ON DELETE CASCADE ON UPDATE CASCADE
);
```

General Use Queries (As found in the various php files that comprise the website)

Sql queries
Integral php statements.
Clarifying comments.
File_name.php

AddNewAsset.php

```
INSERT INTO Asset (Description, Carrot, Cut, Clarity, Color, Create_Date, Owned_By)
 VALUES( [Description input], [Carrot input], [Cut input], [Clarity input], [Color input], [Date input],
[Immutable Owned_By input from index.php SELECT] )
```

AddNewCustomer.php

```
LOCK TABLES Customers WRITE, Account WRITE

INSERT INTO Customers (Lname, Fname, Addr_1, Addr_2, City, State, Zip, Phone)
VALUES([Lname input], [Fname input], [Addr_1 input], [Addr_2 input], [City input], [State input], [Zip
input], [Phone input])

$temp = $mysqli->insert_id;

INSERT INTO Account (C_ID, Balance) VALUES([Immutable $temp insert_id from above], [Balance input])

UNLOCK TABLES
```

EditAsset.php

SELECT id, Description, Carrot, Cut, Clarity, Color, Create_Date, Owned_By
FROM Asset WHERE id=[Immutable Asset_ID input from index.php SELECT]

//Owned_By above is informational only to user and immutable.

UPDATE Asset SET Description=[Description input], Carrot=[Carrot input], Cut=[Cut input],
Clarity=[Clarity input], Color=[Color input], Create_Date=[Date input]
WHERE id=[Immutable Asset_id input from SELECT id above]

index.php

SELECT id FROM Account

SELECT id FROM Customers

SELECT Phone FROM Customers ORDER BY Phone ASC

SELECT id FROM Asset ORDER BY id ASC

SELECT id FROM Account ORDER BY id ASC

SELECT id FROM Asset ORDER BY id ASC

SELECT id FROM Account ORDER BY id ASC

SELECT id FROM Customers ORDER BY id ASC

SELECT id FROM Account ORDER BY id ASC

SELECT id FROM Asset ORDER BY id ASC

RecordTrans.php

SELECT id, Owned_By FROM Asset ORDER BY id ASC -- yields Asset_ID select drop down list.

SELECT id FROM Account ORDER BY id ASC -- yields Buyer's Account_ID select drop down list.

SELECT Owned_By FROM Asset WHERE id = [Buyer's Account ID from above SELECT]

while($stmt->fetch()){ $S_Acct_ID = $Owned_By; }

LOCK TABLES Ledger WRITE, Contract WRITE, Contract_Asset WRITE, Contract_Customers WRITE, Asset WRITE, Account WRITE

INSERT INTO Ledger (date_time) VALUES(NOW())

$temp = $mysqli->insert_id;

```
INSERT INTO Contract (Asset_ID, B_Acct_ID, S_Acct_ID, Eff_Date, Trans_at, Com_pd, L_ID)
VALUES([Asset_ID from above SELECT], [B_Acct_ID from above SELECT], [$S_Acct_ID from above],
[Eff_Date input], [Trans_at input], [Com_pd input], [$temp from above])

INSERT INTO Contract_Asset (Contract_ID, Asset_ID) VALUES([$temp from above], [Asset_ID input])

INSERT INTO Contract_Customers (Contract_ID, Customer_ID)
VALUES ([$temp from above], (SELECT C_ID FROM Account WHERE id = [B_Acct_ID input]))

INSERT INTO Contract_Customers (Contract_ID, Customer_ID)
VALUES([$temp from above], (SELECT C_ID FROM Account WHERE id = [$S_Acct_ID from above]))

UPDATE Asset SET Owned_By=[B_Acct_ID input] WHERE id=[Asset_ID input]

SELECT Balance FROM Account WHERE id=[B_Acct_ID input]

while( $stmt->fetch() ) { $B_Balance = $Balance; }

UPDATE Account SET Balance=[$B_Balance calculated from above] WHERE id=[B_Acct_ID input]

SELECT Balance FROM Account WHERE id=[$S_Acct_ID from above]

while( $stmt->fetch() ) { $S_Balance = $Balance; }

UPDATE Account SET Balance=[$S_Balance calculated from above] WHERE id=[$S_Acct_ID from above]

UNLOCK TABLES
```

showAccount.php

```
SELECT A.id AS Account, A.Balance, Cu.id AS 'Cust_ID', Cu.Lname, Cu.Fname, Cu.Addr_1, Cu.Addr_2,
        Cu.City, Cu.State, Cu.Zip, Cu.Phone
FROM Customers Cu
INNER JOIN Account A ON Cu.id = A.C_ID
```

showAcctByCID.php

```
SELECT Cu.id, A.id AS Account, A.Balance, Cu.Lname, Cu.Fname, Cu.Addr_1, Cu.Addr_2,  Cu.City,
        Cu.State, Cu.Zip, Cu.Phone
FROM Customers Cu
INNER JOIN Account A ON Cu.id = A.C_ID
WHERE Cu.id = [Customer_ID input from index.php SELECT]
```

showAcctByID.php

```
SELECT A.id AS 'Account', A.Balance, Cu.id AS 'Cust_ID', Cu.Lname, Cu.Fname, Cu.Addr_1,  Cu.Addr_2,
        Cu.City, Cu.State, Cu.Zip, Cu.Phone
FROM Customers Cu
INNER JOIN Account A ON Cu.id = A.C_ID
WHERE A.id = [Account_ID from index.php SELECT]
```

showAcctByPhone.php

```
SELECT Cu.id, A.id AS Account, A.Balance, Cu.Lname, Cu.Fname, Cu.Addr_1, Cu.Addr_2, Cu.City,
        Cu.State, Cu.Zip, Cu.Phone
FROM Customers Cu
INNER JOIN Account A ON Cu.id = A.C_ID
WHERE Cu.Phone = [Phone from index.php SELECT]
```

showAllAssets.php

```
SELECT id, Description, Carrot, Cut, Clarity, Color, Create_Date, Owned_By
FROM Asset
ORDER BY Asset.id ASC
```

showAssetByAccID.php

```
SELECT id, Description, Carrot, Cut, Clarity, Color, Create_Date, Owned_By
FROM Asset
WHERE Owned_By = [Asset_ID from index.php SELECT]
ORDER BY Asset.id ASC
```

showAssetByID.php

```
SELECT AST.id, CA.Contract_ID, CO.Trans_at AS 'Value', CO.B_Acct_ID, CO.Eff_Date, CU.Lname,
        CU.Fname
FROM Asset AST
INNER JOIN Contract_Asset CA ON CA.Asset_ID = AST.id
INNER JOIN Contract CO ON CO.id = CA.Contract_ID
INNER JOIN Contract_Customers CC ON CC.Contract_ID = CO.id
INNER JOIN Customers CU ON CU.id = CC.Customer_ID
WHERE AST.id = [Asset_ID from index.php SELECT]
ORDER BY CO.Eff_Date DESC LIMIT 1
```

showLedger.php

```
SELECT L.id, L.date_time, CO.id as ID, CO.Asset_ID, CO.Trans_at, CO.Com_pd, AST.Description,
        AST.Owned_By
FROM Ledger L
INNER JOIN Contract CO ON CO.L_ID = L.id
INNER JOIN Contract_Asset CA ON CA.Contract_ID = CO.id
```

```
INNER JOIN Asset AST ON AST.id = CA.Asset_ID
ORDER BY L.id ASC
```

showLedgerByID.php

```
SELECT L.id, L.date_time, CO.id as ID, CO.Asset_ID, CO.Trans_at, CO.Com_pd, AST.Description,
        AST.Owned_By
FROM Ledger L
INNER JOIN Contract CO ON CO.L_ID = L.id
INNER JOIN Contract_Asset CA ON CA.Contract_ID = CO.id
INNER JOIN Asset AST ON AST.id = CA.Asset_ID
WHERE CO.Asset_ID = [Asset_ID input from index.php SELECT]
ORDER BY L.id ASC
```

showTCPaid.php

```
SELECT  SUM(Trans_at) AS 'Contracted', SUM(Com_pd) AS 'Total_Commissions'
FROM Contract
WHERE Eff_Date >= [begDate input from index.php SELECT)]
AND
Eff_Date <= [endDate input from index.php SELECT)]
```

UpdateAccBalance.php

```
SELECT id, C_ID, Balance FROM Account WHERE id=[id from index.php SELECT]
UPDATE Account SET Balance=[Balance input] WHERE id=[id input originally from index.php SELECT]
```

UpdateCustomer.php

```
SELECT id, Lname, Fname, Addr_1, Addr_2, City, State, Zip, Phone
FROM Customers WHERE id=[immutable id input from index.php SELECT]

UPDATE Customers SET Lname=[Lname input], Fname=[Fname input], Addr_1=[Addr_1 input],
Addr_2=[Addr_2 input], City=[City input], State=[State input], Zip=[Zip input], Phone=[Phone input]
WHERE id=[immutable id input]
```

Files included in student submission:

cs340_400_Summer16_KearnsC_Final_Project.pdf - This document.

Blockchain-definition.sql	Table creation import file (Importable via phpMyAdmin).
Blockchain-intitialData.sql	Initial data load for testing. (Importable via phpMyAdmin).
Blockchain-queries.sql	Various queries for testing via phpMyAdmin.

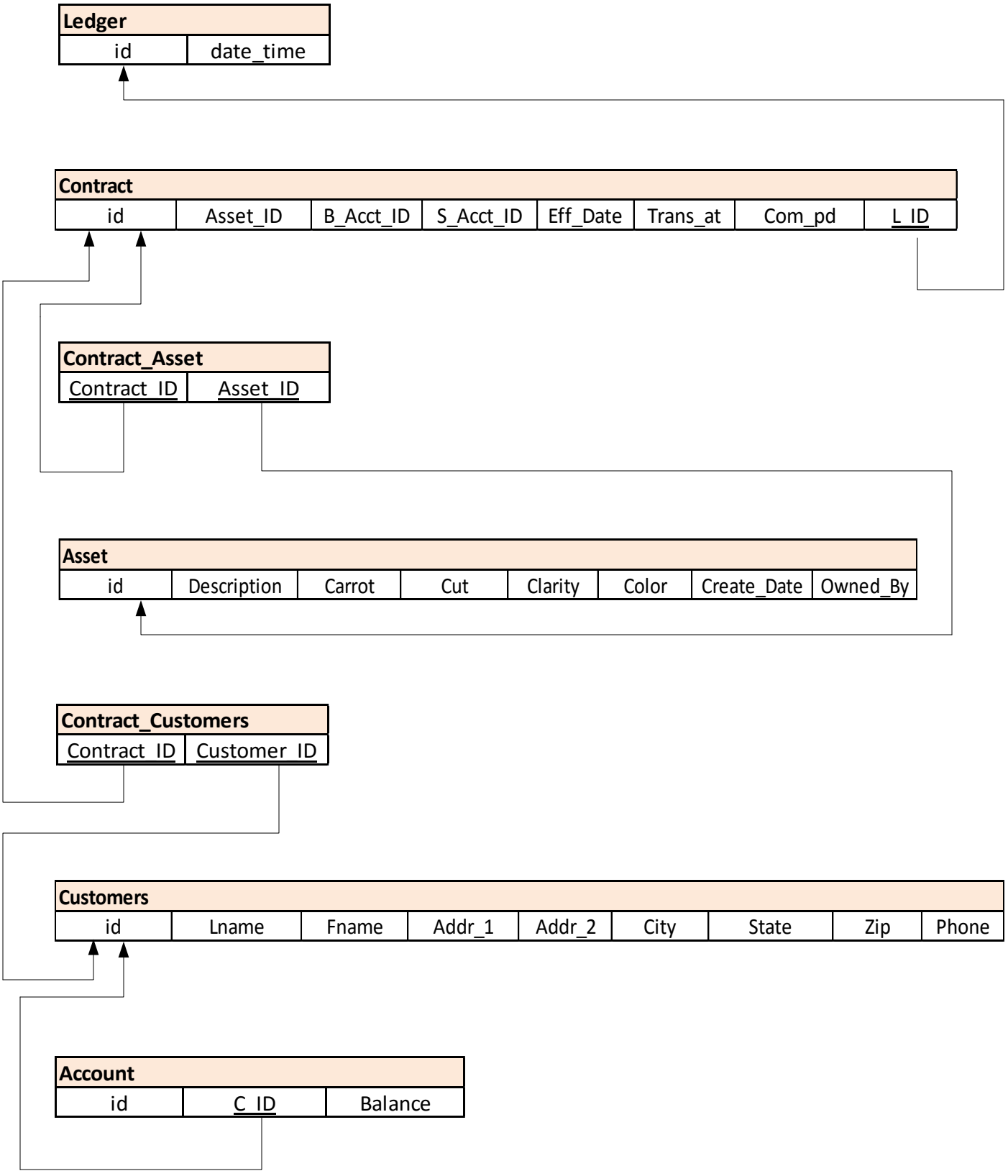
AddNewAsset.php
AddNewCustomer.php

EditAsset.php
index.php
RecordTrans.php
showAccount.php
showAcctByCID.php
showAcctByID.php
showAcctByPhone.php
showAllAssets.php
showAssetByAccID.php
showAssetByID.php
showLedger.php
showLedgerByID.php
showTCPaid.php
UpdateAccBalance.php
UpdateCustomer.php

style.css
favicon.ico

ER Diagram and Schema are below.

SCHEMA for BLOCKCHAIN LOCAL NODE



ER DIAGRAM for BLOCKCHAIN LOCAL NODE