The Grave Digger Database

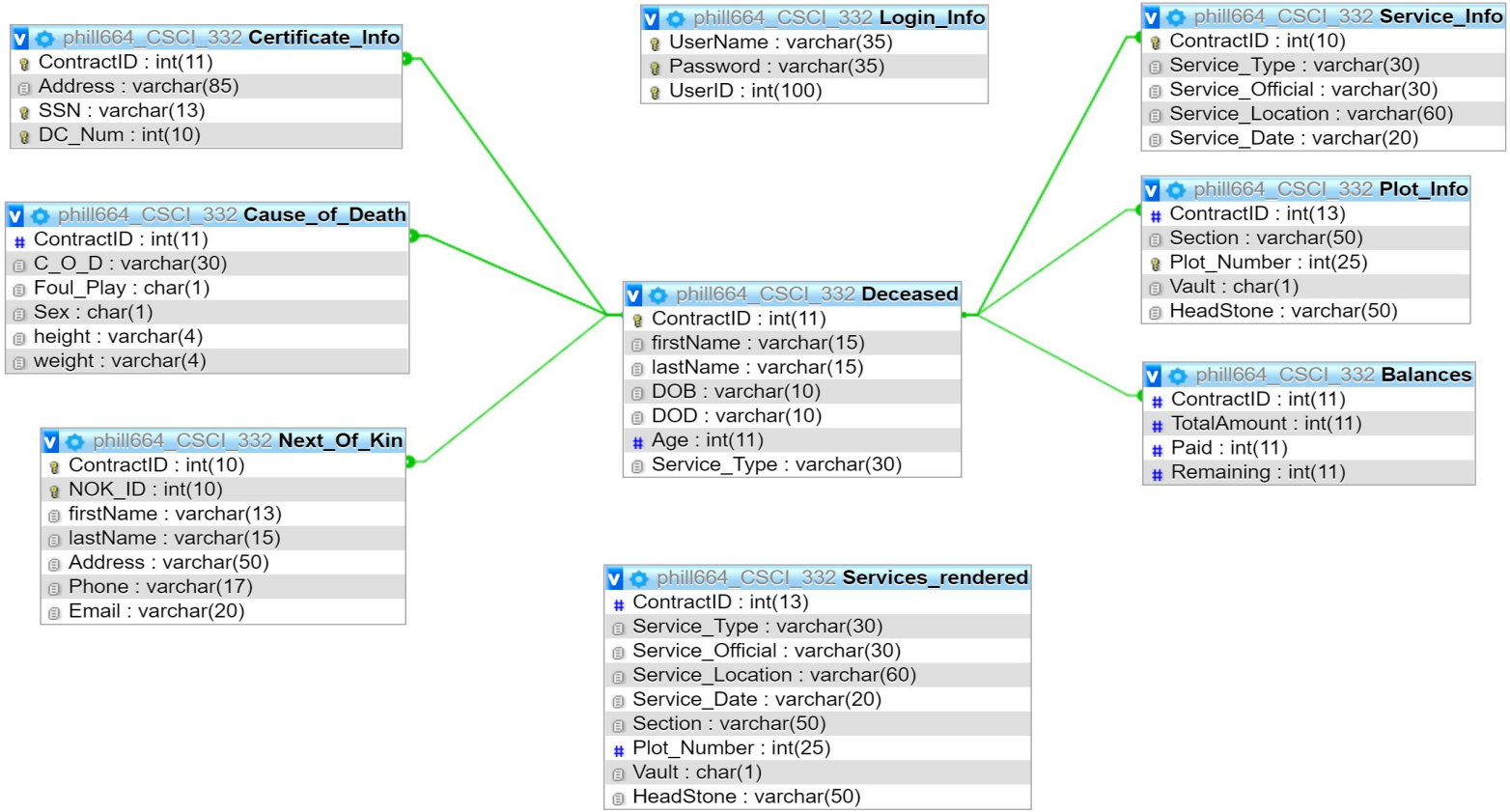
Phillip Byrd

Purpose

- This Database is modeled after the business that my family started 25 years ago.
  - They Own and Operate a cemetery and funeral home in South Alabama.

- This database was designed to be a tool for the funeral home and cemetery to keep track of the information on the services they provide to families.
  - The expected user would be an employee working at the funeral home.
  - The information includes personal information and service information.

ER Diagram


phill664_CSCI_332 Certificate_Info
ContractID : int(11)
Address : varchar(85)
SSN : varchar(13)
DC_Num : int(10)
phill664_CSCI_332 Login_Info
UserName : varchar(35)
Password : varchar(35)
UserID : int(100)
phill664_CSCI_332 Service_Info
ContractID : int(10)
Service_Type : varchar(30)
Service_Official : varchar(30)
Service_Location : varchar(60)
Service_Date : varchar(20)
phill664_CSCI_332 Cause_of_Death
ContractID : int(11)
C_O_D : varchar(30)
Foul_Play : char(1)
Sex : char(1)
height : varchar(4)
weight : varchar(4)
phill664_CSCI_332 Plot_Info
ContractID : int(13)
Section : varchar(50)
Plot_Number : int(25)
Vault : char(1)
HeadStone : varchar(50)
phill664_CSCI_332 Deceased
ContractID : int(11)
firstName : varchar(15)
lastName : varchar(15)
DOB : varchar(10)
DOD : varchar(10)
Age : int(11)
Service_Type : varchar(30)
phill664_CSCI_332 Balances
ContractID : int(11)
TotalAmount : int(11)
Paid : int(11)
Remaining : int(11)
phill664_CSCI_332 Next_Of_Kin
ContractID : int(10)
NOK_ID : int(10)
firstName : varchar(13)
lastName : varchar(15)
Address : varchar(50)
Phone : varchar(17)
Email : varchar(20)
phill664_CSCI_332 Services_rendered
ContractID : int(13)
Service_Type : varchar(30)
Service_Official : varchar(30)
Service_Location : varchar(60)
Service_Date : varchar(20)
Section : varchar(50)
Plot_Number : int(25)
Vault : char(1)
HeadStone : varchar(50)

Normalization

- My database is 4th Normal Form
  - All the relationships in my database depend on each other
  - In particular, the contract ID in the deceased table.
    - This column is the main primary key that all other table rely on as a foreign key.

Isolation level

- Typically Only one person would be in the system at a time editing and adding data to the system.

- For this reason I set the integrity level to the Default of **Repeatable read**

- There did not seem to be any need to use the serializable or read uncommitted for these conditions.

Integrity Enforcement

- ContractID in the Deceased table is unique and (auto-incremented) as my Primary Key.
  - All other table have a column ContractID and reference the Deceased table to maintain consistency with all the data

TRIGGER

- Once an entry is made in the deceased table the contractID is used to create a row in the Balances table to help generate financial reports and it's then passed to other tables in simple forms to store their information.

Function CALCULATOR

```
DELIMITER $$
CREATE DEFINER=`phill664`@`localhost` FUNCTION `Calculator`(`id` INT(10)) RETURNS int(11)
BEGIN

    DECLARE totalAmt INT(25);
    DECLARE ser_type VARCHAR(15);
    DECLARE vault CHAR(5);
    DECLARE section INT(25);

    SET totalAmt = 0;
    SET ser_type = (select Service_Type from Services_rendered Where ContractID= id);
    SET vault = (select Vault from Services_rendered Where ContractID= id);
    SET section = (select Section from Services_rendered Where ContractID= id);

    IF ser_type = "Burial" THEN SET totalAmt = totalAmt + 5000;
    ELSE SET totalAmt = totalAmt + 2500;
    END IF;

    IF vault = 'Y' THEN SET totalAmt = totalAmt + 500;
    ELSE SET totalAmt = totalAmt + 250;
    END IF;

    IF section < 150 THEN SET totalAmt = totalAmt + 700;
    ELSEIF section > 151 AND section < 350 THEN SET totalAmt = totalAmt+ 500;
    ELSE SET totalAmt = totalAmt + 250;
    END IF;

    RETURN(totalAmt);
END$$
DELIMITER ;
```

Procedure UpdateBAL

```
DELIMITER $$
CREATE DEFINER=`phill664`@`localhost` PROCEDURE `UpdateBAL`(IN `ID` INT(10), IN `AMT` INT(10), IN `PAID` INT(10))
UPDATE Balances SET
Balances.TotalAmount= AMT,
Balances.Paid=PAID,
Balances.Remaining=(AMT-PAID)
WHERE Balances.ContractID=ID$$
DELIMITER ;
```

Trigger Balance Maker

```
CREATE TRIGGER `BalanceMaker` AFTER INSERT ON `Deceased`
 FOR EACH ROW BEGIN

INSERT INTO Balances VALUES(NEW.ContractID,0,0,0);



END
```

Trigger DelALL

```
CREATE TRIGGER `delALL` BEFORE DELETE ON `Deceased`
 FOR EACH ROW BEGIN

        DELETE FROM Service_Info WHERE Service_Info.ContractID = OLD.ContractID;
        DELETE FROM Plot_Info WHERE Plot_Info.ContractID = OLD.ContractID;
        DELETE FROM Next_Of_Kin WHERE Next_Of_Kin.ContractID = OLD.ContractID;
        DELETE FROM Certificate_Info WHERE Certificate_Info.ContractID = OLD.ContractID;
        DELETE FROM Cause_of_Death WHERE Cause_of_Death.ContractID = OLD.ContractID;
        DELETE FROM Balances WHERE Balances.ContractID = OLD.ContractID;



    END
```

View

```
CREATE VIEW `Services_rendered`  AS  select

`p`.`ContractID` AS `ContractID`,`s`.`Service_Type`
AS `Service_Type`,`s`.`Service_Official`
AS `Service_Official`,`s`.`Service_Location`
AS `Service_Location`,`s`.`Service_Date`
AS `Service_Date`,`p`.`Section`
AS `Section`,`p`.`Plot_Number`
AS `Plot_Number`,`p`.`Vault`
AS `Vault`,`p`.`HeadStone`
AS `HeadStone`
from (`Plot_Info` `p` join `Service_Info` `s`) where (`p`.`ContractID` = `s`.`ContractID`) ;
```

Report

```
SHOWS NUMBER OF CONTRACTS IN THE SYSTEM:

    select COUNT(*) AS deadCount from Deceased

SHOWS SUM OF ALL SERVICES RENDERED IN SYSTEM:

    select SUM(TotalAmount) AS DEADSum from Balances

SHOWS THE NUMBER OF BURIALS IN THE SYSTEM:

    select Count(Service_Type) AS BurialCOUNT from Deceased WHERE Service_Type = 'Burial'

SHOWS THE NUMBER OF CREMATIONS IN THE SYSTEM:

select Count(Service_Type) AS CremationCOUNT from Deceased WHERE Service_Type = 'Cremation'
```

TIME FOR A RUN THROUGH

The Grave Digger Database