FIDO'S

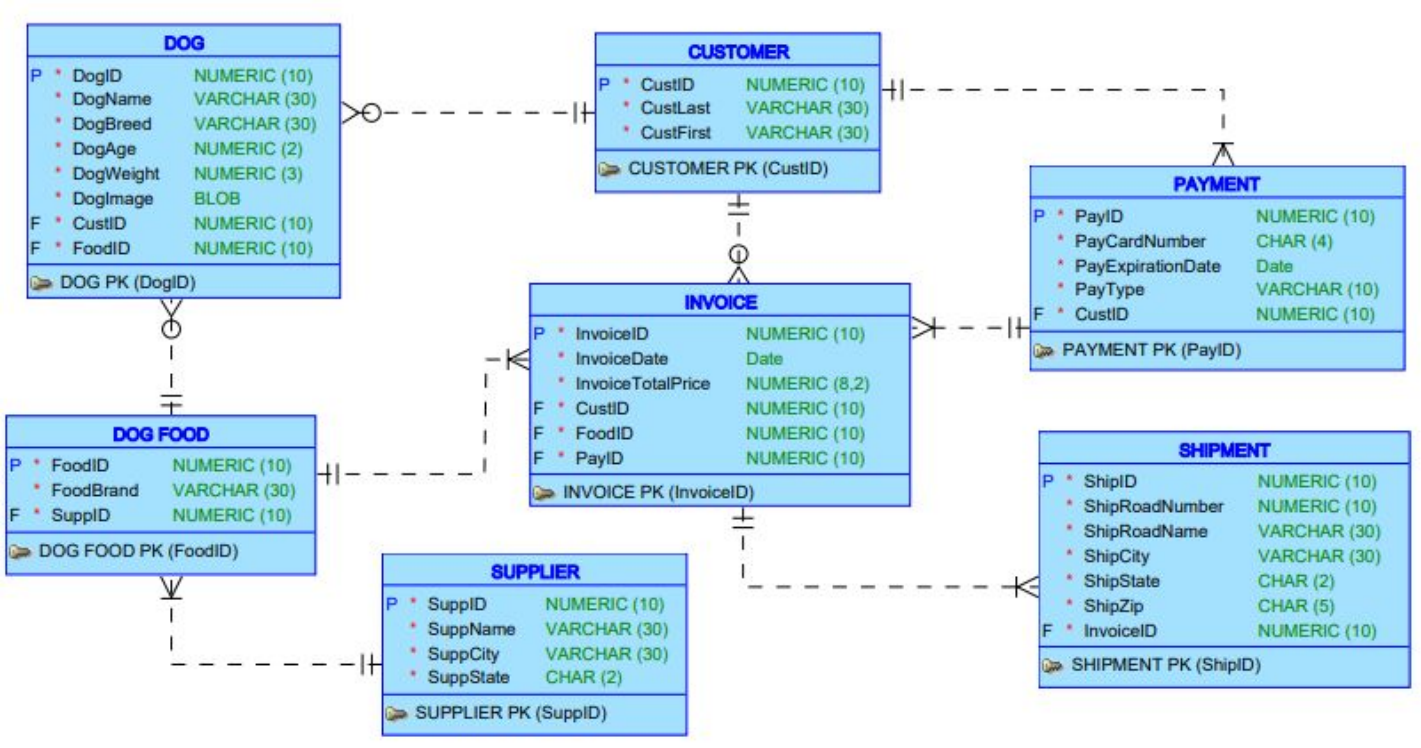
Fido's Store

Brandon Dave
Samantha Stetz

Purpose

- A solution to store information about an Owner, their dog, and the dog's favorite food.
- A solution to store information about a Supplier of a dog food
- A solution to store information to sell dog food to an Owner including information on Shipping

Entity-Relationship Diagram



Schema

CUSTOMER(CustID, CustLast, CustFirst)

DOG(DogID, DogName, DogBreed, DogAge, DogWeight, DogImage, CustID, FoodID)

DOGFOOD(FoodID, FoodBrand, SuppID)

SUPPLIER(SuppID, SuppName, SuppCity, SuppState)

INVOICE(InvoiceID, InvoiceDate, InvoiceTotalPrice, CustID, foodID, payID)

SHIPMENT(ShipID, ShipRoadNumber, ShipRoadName, ShipCity, ShipState, ShipZip, InvoiceID)

PAYMENT(PayID, PayCardNumber, PayExpirationDate, PayType, CustID)

Three Table Join SQL Statement

- Display the owners of dogs and the respective dog's favorite food that are over 50 pounds

```
SELECT CustFirst AS CUSTOMER_FIRST_NAME, CustLast AS CUSTOMER_LAST_NAME, DogName AS
DOG_NAME, DogWeight AS DOG_WEIGHT, FoodBrand AS FAVORITE_FOOD
FROM CUSTOMER, DOG, DOGFOOD
WHERE CUSTOMER.CustID = DOG.CustID
AND DOG.FoodID = DOGFOOD.FoodID
AND DOG.DogWeight >= 50
ORDER BY DogWeight;
```

Script Output

	CUSTFIRST	CUSTLAST	DOGNAME	DOGBREED	DOGWEIGHT	FOODBRAND
1	Greg	Edgar	Ralphy	Bulldog	52	WeAreFamily
2	Jane	Felix	Tommy	Poodle	55	Blue Square
3	Greg	Edgar	Mobi	Bulldog	55	WeAreFamily
4	John	Doe	Chuckie	German Shephard	68	WeAreFamily
5	Jane	Felix	Bean	Golden Retriever	77	Blue Square

Two Table Statistical SQL Statement

- Display unique dog food brands purchased in the last 3 months along with their average cost and total point of times they were purchased

```
Select DOGFOOD.FOODID, DOGFOOD.FOODBRAND as FAVORITE_FOOD, AVG(INVOICETOTALPRICE) as
"QUARTERLY_COSTS", COUNT(INVOICE.FOODID) as "AMOUNT_OF_TIMES_PURCHASED"
From INVOICE, DOGFOOD
WHERE InvoiceDate >= add_months(sysdate, -3)
and INVOICE.FOODID = DOGFOOD.FOODID
GROUP BY DOGFOOD.FOODID, DOGFOOD.FOODBRAND
ORDER BY FOODID;
```

Script Output

	FOODID	FOODBRAND	Quarterly Costs	Amount of times purchased
1	1	Blue Square	23.99	3
2	2	WeAreFamily	22.99	2
3	5	Old Paws	20.99	1
4	6	Bedigree	15.99	1

Nested Subquery Statement

- Display the highest invoice total price and the city name it was shipped to

```
SELECT ShipCity, InvoiceTotalPrice
FROM INVOICE, SHIPMENT
WHERE INVOICE.InvoiceID = SHIPMENT.InvoiceID
AND SHIPMENT.ShipCity IN
            (SELECT ShipCity
            FROM SHIPMENT, INVOICE
            WHERE INVOICE.InvoiceID = SHIPMENT.InvoiceID
            AND TO_CHAR(INVOICE.InvoiceDate, 'MM') = 3 )
AND INVOICE.InvoiceTotalPrice IN
    (SELECT MAX(InvoiceTotalPrice)
    FROM INVOICE
    WHERE TO_CHAR(INVOICE.InvoiceDate, 'MM') = 3 );
```

Script Output

SHIPCITY	INVOICETOTALPRICE
Beavercreek	26.99

Automated PL/SQL Statement

- Automate Dog Age increment and Display Dog's over the age of 5

```
DECLARE
CURSOR DogAgeCursor IS
SELECT DogName, DogAge
FROM DOG
WHERE DogAge > 5;
DogName DOG.DogName%type;
NewDogAge DOG.DogAge%type;
BEGIN
--Update dog ages to be one year older
UPDATE DOG
SET DogAge = DogAge + 1;
--Intro messages
DBMS_OUTPUT.PUT_LINE('A year has passed!');
DBMS_OUTPUT.PUT_LINE('DOGS OVER 5 YEARS OLD:');
OPEN DogAgeCursor;
LOOP
FETCH DogAgeCursor INTO DogName,NewDogAge;
EXIT WHEN DogAgeCursor%NOTFOUND;
DBMS_OUTPUT.PUT_LINE(DogName||' is now '||NewDogAge||' years old!');
END LOOP;
CLOSE DogAgeCursor;
END;
```

Script Output

```
A year has passed!
DOGS OVER 5 YEARS OLD:
Chuckie is now 8 years old!
Bean is now 6 years old!
Ralphy is now 6 years old!
Mickey is now 9 years old!
```

SQL View

- Create a view for the client to display all the customers and their respective dogs along with the dog's favorite dog food

```
CREATE MATERIALIZED VIEW vwCustomerDogFavoriteFood
AS SELECT CustFirst as FIRST_NAME, CustLast as LAST_NAME, DOGNAME as DOG_NAME,
DOGFOOD.FOODBRAND as FAVORITE_FOOD
FROM CUSTOMER, DOG, DOGFOOD
WHERE CUSTOMER.CUSTID = DOG.CUSTID
and DOG.FOODID = DOGFOOD.FOODID
Order by CUSTOMER.CUSTID;
```

SQL View Display: vwCustomerDogFavoriteFood

VWCUSTOMERDOGFAVORITEFOOD | bdd_CS3700.sql

Columns | Data | Indexes | Grants | Dependencies | Details | SQL

Sort.. | Filter:

	FIRST_NAME	LAST_NAME	DOG_NAME	FAVORITE_FOOD
1	John	Doe	Chuckie	WeAreFamily
2	Jane	Felix	Bean	Blue Square
3	Jane	Felix	Tommy	Blue Square
4	Greg	Edgar	Mobi	WeAreFamily
5	Greg	Edgar	Ralphy	WeAreFamily
6	Matt	Euler	Mickey	Bedigree
7	Kristin	Proof	Carlos	Special Paws
8	Kristin	Proof	Draco	Special Paws

SQL View

- Create a view for the client to display the invoices of customers and the dog food they purchased on that day

```
CREATE MATERIALIZED VIEW vwCustomerInvoiceDogFood
AS SELECT CustFirst as FIRST_NAME, CustLast as LAST_NAME,
INVOICE.INVOICEDATE AS INVOICE_DATE, INVOICE.INVOICETOTALPRICE AS PURCHASE_COSTS,
DOGFOOD.FOODBRAND as PURCHASED_FOOD
FROM CUSTOMER, INVOICE, DOGFOOD
WHERE CUSTOMER.CUSTID = INVOICE.CUSTID
and INVOICE.FOODID = DOGFOOD.FOODID
ORDER BY INVOICEDATE;
```

SQL View Display: vwCustomerInvoiceDogFood

VWCUSTOMERINVOICEDOGFOOD | bdd_CS3700.sql

Columns | Data | Indexes | Grants | Dependencies | Details | SQL

Sort.. | Filter:

	FIRST_NAME	LAST_NAME	INVOICE_DATE	PURCHASE_COSTS	PURCHASED_FOOD
1	Kristin	Proof	06-JAN-21	32.95	Special Paws
2	Jane	Felix	01-FEB-21	23.99	Blue Square
3	Jane	Felix	01-MAR-21	23.99	Blue Square
4	Matt	Euler	12-MAR-21	15.99	Bedigree
5	John	Doe	23-MAR-21	26.99	WeAreFamily
6	Jane	Felix	01-APR-21	23.99	Blue Square
7	Greg	Edgar	02-APR-21	18.99	WeAreFamily
8	Kristin	Proof	16-APR-21	20.99	Old Paws

Lesson Learned

We discovered the importance of thought out, succinct naming conventions for tables. Twice, we were presented with the option to rename tables which for OWNER turning into CUSTOMER was easy enough; however, changing ORDER into something meaningful that still drove the purpose of its existence was challenging. INVOICE was chosen as the final option due to wanting to keep with a singular form of a word. An easy option presented was ORDERS, but it did not meet the criteria of being a singular form.

Questions?

The client of Fido's comes back after reviewing the purposes of the database, and now requests the developers to implement a way to track if an invoice is paid. What are a couple of way you could achieve this?

Solution

- Create a PL/SQL to deduct from INVOICETOTALPRICE as Payments are made
  - INVOICETOTALPRICE was designed as a hard-coded value. DOGFOOD could incorporate a COST attribute that could be used to calculate INVOICETOTALPRICE for multiple purchases of DOGFOOD.
- Create a new INVOICE field that describes whether payments have been performed
- Alternatively, the new field could describe how much has been paid to the INVOICETOTALPRICE