PAYAP UNIVERSITY

SQL
Structured Query Language

Vaidė Narváez

Computer Information Systems

November 30th, 2011

PAYAP UNIVERSITY

Introduction

- "S-Q-L" or "sequel", **S**tructured **Q**uery **L**anguage
- SQL is the standard language for creating, querying and modifying relational databases.

- Declarative language: says *"what to do"* rather than *"how to do it*
- Avoids a lot of data-manipulation details needed in procedural languages like C++ or Java.
- DBMSs figure out the "best way to execute a query - *query optimization*

PAYAP UNIVERSITY

History of SQL

- 1970 – E. Codd develops relational database concept
- 1974-1979 – System R with Sequel (later SQL) created at IBM Research Lab, available commercially in 1981
- 1979 – Oracle markets first relational DB with SQL
- 1986 – ANSI SQL standard released
- 1989, 1992, 1999, 2003, 2008 – Major ANSI standard updates
- Current – SQL is supported by most major database vendors

PAYAP UNIVERSITY

History of SQL

- 1970 – E. Codd develops relational database concept
- 1974-1979 – System R with Sequel (later SQL) created at IBM Research Lab, available commercially in 1981
- 1979 – Oracle markets first relational DB with SQL
- 1986 – ANSI SQL standard released
- 1989, 1992, 1999, 2003, 2008 – Major ANSI standard updates
- Current – SQL is supported by most major database vendors

  Different SQL dialects:
  - MS SQL Server - T-SQL,
  - Oracle - PL/SQL,
  - MS Access - JET SQL, etc.

PAYAP UNIVERSITY

Classification

SQL commands can be classified into three types:

1. DDL – Data Definition Language
   (create, alter, drop tables)
2. DML – Data Manipulation Language
   (update, insert, delete, query data)
3. DCL – Data Control Language
   (grant/revoke privileges, create/remove transactions)

PAYAP UNIVERSITY

Example schema

BEERS(name, manufacturer)
BARS(name, address,license)
DRINKERS(name, address, phone)
LIKES(drinker, beer)
SELLS(bar, beer, price)
FREQUENTS(drinker, bar)

PAYAP UNIVERSITY

Relations in SQL

Three kinds of relations:

- Stored relations - *tables*
- Relations defined by computation - *views*
- Temporary tables

temporary tables are never declared!

PAYAP UNIVERSITY

Data Types

All attributes must have a data type

- Character strings of fixed and varying length: `CHAR(n)`, `VARCHAR(n)`
- Bit strings of fixed or varying length: `BIT(n)`, `BIT VARYING(n)`
- `BOOLEAN: TRUE, FALSE, UNKNOWN`
- Integer values: `INT`, `INTEGER`, `SHORTINT`
- Floating point numbers: `FLOAT`, `REAL`, `DECIMAL(n,d)`, `NUMERIC(n,d)`
- Date and time:
  - `DATE`: 'yyyy-mm-dd'
  - `TIME`: 'hh:mm:ss[.ss...]'
- `DATETIME` or `TIMESTAMP`: 'yyyy-mm-dd hh:mm:ss[.ss...]'

PAYAP UNIVERSITY

Some SQL Data types

String	CHARACTER (CHAR)	Stores string values containing any characters in a character set. CHAR is defined to be a fixed length.
	CHARACTER VARYING (VARCHAR)	Stores string values containing any characters in a character set, but of definable variable length.
	BINARY LARGE OBJECT (BLOB)	Stores binary string values in hexadecimal format. BLOB is defined to be a variable length.
Number	NUMERIC	Stores exact numbers with a defined precision and scale.
	INTEGER (INT)	Stores exact numbers with a predefined precision and scale of zero.
Temporal	TIMESTAMP	Stores a moment an event occurs, using a definable fraction of a second precision.
Boolean	BOOLEAN	Stores truth values, TRUE, FALSE, or UNKNOWN.

PAYAP UNIVERSITY

DDL: CREATE SCHEMA

CREATE SCHEMA defines a portion of the database owned by a particular user or that belong to the same application.

PAYAP UNIVERSITY

DDL: CREATE SCHEMA

CREATE SCHEMA defines a portion of the database owned by a particular user or that belong to the same application.

CREATE SCHEMA company AUTHORIZATION jsmith;
CREATE SCHEMA beersAndBars;

PAYAP UNIVERSITY

DDL: CREATE TABLE

CREATE TABLE defines a new table and its columns.

PAYAP UNIVERSITY

DDL: CREATE TABLE

CREATE TABLE defines a new table and its columns.

```
CREATE TABLE tablename
( {column definition   [table constraint] } . , . .
[ON COMMIT {DELETE | PRESERVE} ROWS] );

where column definition ::=
column_name
        {domain name | datatype [(size)] }
        [column_constraint_clause . . .]
        [default value]
        [collate clause]

and table constraint ::=
        [CONSTRAINT constraint_name]
        Constraint_type [constraint_attributes]
```

PAYAP UNIVERSITY

CREATE TABLE example

BEERS(name, manufacturer)

```
CREATE TABLE Beers(
name VARCHAR(20),
manufacturer VARCHAR(100)
);
```

PAYAP UNIVERSITY

CREATE TABLE example

SELLS(bar, beer, price)

```
CREATE TABLE Sells(
bar VARCHAR(20),
beer VARCHAR(20),
price DECIMAL(5,2)
);
```

PAYAP UNIVERSITY

Default values

```
CREATE TABLE Beers(
name VARCHAR(20),
manufacturer VARCHAR(100),
alcohol_level DECIMAL(4,2) DEFAULT 8.00
);
```

PAYAP UNIVERSITY

Declaring Keys

```
CREATE TABLE Beers (
name CHAR(20) UNIQUE,
manufacturer CHAR(20)
);
```

```
CREATE TABLE Sells (
bar CHAR(20),
beer VARCHAR(20),
price REAL,
PRIMARY KEY (bar, beer)
);
```

PAYAP UNIVERSITY

Foreign Keys

Two ways to declare a foreign key:

- If single attribute:
  `REFERENCES table_name (attribute_name)`
- Otherwise:
  `FOREIGN KEY (attribute_names) REFERENCES table_name (attribute_names)`

PAYAP UNIVERSITY

Foreign Keys

Two ways to declare a foreign key:

- If single attribute:
  `REFERENCES table_name (attribute_name)`
- Otherwise:
  `FOREIGN KEY (attribute_names) REFERENCES table_name (attribute_names)`

```
CREATE TABLE Sells (
bar CHAR(20),
beer VARCHAR(20),
price REAL,
PRIMARY KEY (bar, beer),
FOREIGN KEY(beer) REFERENCES Beers(name),
FOREIGN KEY(bar) REFERENCES Bars(name)
);
```

PAYAP UNIVERSITY

CREATE TABLE steps

- Identify data types for attributes
- Identify columns that can and cannot be null
- Identify columns that must be unique (candidate keys)
- Identify primary key,foreign key attributes
- Determine default values
- Identify constraints on columns (domain specifications)
- Create the table and associated indexes

PAYAP UNIVERSITY

CREATE TABLE examples

```
CREATE TABLE CUSTOMER_T
                (CUSTOMER_ID                    NUMBER(11, 0) NOT NULL,
                 CUSTOMER_NAME                  VARCHAR2(25)  NOT NULL,
                 CUSTOMER_ADDRESS               VARCHAR2(30),
                 CITY                           VARCHAR2(20),
                 STATE                          VARCHAR2(2),
                 POSTAL_CODE                    VARCHAR2(9),
CONSTRAINT CUSTOMER_PK PRIMARY KEY (CUSTOMER_ID));

CREATE TABLE ORDER_T
                (ORDER_ID                       NUMBER(11, 0) NOT NULL,
                 ORDER_DATE                     DATE DEFAULT SYSDATE,
                 CUSTOMER_ID                    NUMBER(11, 0),
CONSTRAINT ORDER_PK PRIMARY KEY (ORDER_ID),
CONSTRAINT ORDER_FK FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER_T(CUSTOMER_ID));

CREATE TABLE PRODUCT_T
                (PRODUCT_ID                     INTEGER          NOT NULL,
                 PRODUCT_DESCRIPTION            VARCHAR2(50),
                 PRODUCT_FINISH                 VARCHAR2(20)
                                CHECK (PRODUCT_FINISH IN ('Cherry', 'Natural Ash', 'White Ash',
                                  'Red Oak', 'Natural Oak', 'Walnut')),
                 STANDARD_PRICE                 DECIMAL(6,2),
                 PRODUCT_LINE_ID                INTEGER,
CONSTRAINT PRODUCT_PK PRIMARY KEY (PRODUCT_ID));

CREATE TABLE ORDER_LINE_T
                (ORDER_ID                       NUMBER(11,0)  NOT NULL,
                 PRODUCT_ID                     NUMBER(11,0)  NOT NULL,
                 ORDERED_QUANTITY               NUMBER(11,0),
CONSTRAINT ORDER_LINE_PK PRIMARY KEY (ORDER_ID, PRODUCT_ID),
CONSTRAINT ORDER_LINE_FK1 FOREIGN KEY(ORDER_ID) REFERENCES ORDER_T(ORDER_ID),
CONSTRAINT ORDER_LINE_FK2 FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT_T(PRODUCT_ID));
```

PAYAP UNIVERSITY

Attributes and data types

```
CREATE TABLE PRODUCT_T
            (PRODUCT_ID              INTEGER        NOT NULL,
             PRODUCT_DESCRIPTION     VARCHAR2(50),
             PRODUCT_FINISH          VARCHAR2(20)
                        CHECK (PRODUCT_FINISH IN ('Cherry', 'Natural Ash', 'White Ash',
                                'Red Oak', 'Natural Oak', 'Walnut')),
             STANDARD_PRICE          DECIMAL(6,2),
             PRODUCT_LINE_ID         INTEGER,
CONSTRAINT PRODUCT_PK PRIMARY KEY (PRODUCT_ID));
```

PAYAP UNIVERSITY

Constraints

```
CREATE TABLE PRODUCT_T
            (PRODUCT_ID                 INTEGER         NOT NULL,
             PRODUCT_DESCRIPTION        VARCHAR2(50),
             PRODUCT_FINISH             VARCHAR2(20)
                        CHECK (PRODUCT_FINISH IN ('Cherry', 'Natural Ash', 'White Ash',
                                'Red Oak', 'Natural Oak', 'Walnut')),
             STANDARD_PRICE             DECIMAL(6,2),
             PRODUCT_LINE_ID            INTEGER,
CONSTRAINT PRODUCT_PK PRIMARY KEY (PRODUCT_ID));
```

PAYAP UNIVERSITY

Constraints

```
CREATE TABLE PRODUCT_T
            (PRODUCT_ID                 INTEGER        NOT NULL,
             PRODUCT_DESCRIPTION        VARCHAR2(50),
             PRODUCT_FINISH             VARCHAR2(20)
                        CHECK (PRODUCT_FINISH IN ('Cherry', 'Natural Ash', 'White Ash',
                                'Red Oak', 'Natural Oak', 'Walnut')),
             STANDARD_PRICE             DECIMAL(6,2),
             PRODUCT_LINE_ID            INTEGER,
CONSTRAINT PRODUCT_PK PRIMARY KEY (PRODUCT_ID));
```



```
CREATE TABLE ORDER_LINE_T
            (ORDER_ID                   NUMBER(11,0)  NOT NULL,
             PRODUCT_ID                 NUMBER(11,0)  NOT NULL,
             ORDERED_QUANTITY           NUMBER(11,0),
CONSTRAINT ORDER_LINE_PK PRIMARY KEY (ORDER_ID, PRODUCT_ID),
CONSTRAINT ORDER_LINE_FK1 FOREIGN KEY(ORDER_ID) REFERENCES ORDER_T(ORDER_ID),
CONSTRAINT ORDER_LINE_FK2 FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT_T(PRODUCT_ID));
```

PAYAP UNIVERSITY

Attributes values

```
CREATE TABLE ORDER_T
                (ORDER_ID                    NUMBER(11, 0) NOT NULL,
                 ORDER_DATE                  DATE DEFAULT SYSDATE,
                 CUSTOMER_ID                 NUMBER(11, 0),
CONSTRAINT ORDER_PK PRIMARY KEY (ORDER_ID),
CONSTRAINT ORDER_FK FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER_T(CUSTOMER_ID));
```

PAYAP UNIVERSITY

Attributes values

```
CREATE TABLE ORDER_T
              (ORDER_ID                  NUMBER(11, 0) NOT NULL,
               ORDER_DATE                DATE DEFAULT SYSDATE,
               CUSTOMER_ID               NUMBER(11, 0),
CONSTRAINT ORDER_PK PRIMARY KEY (ORDER_ID),
CONSTRAINT ORDER_FK FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER_T(CUSTOMER_ID));
```



```
CREATE TABLE PRODUCT_T
              (PRODUCT_ID                INTEGER        NOT NULL,
               PRODUCT_DESCRIPTION       VARCHAR2(50),
               PRODUCT_FINISH            VARCHAR2(20)
                         CHECK (PRODUCT_FINISH IN ('Cherry', 'Natural Ash', 'White Ash',
                                'Red Oak', 'Natural Oak', 'Walnut')),
               STANDARD_PRICE            DECIMAL(6,2),
               PRODUCT_LINE_ID           INTEGER,
CONSTRAINT PRODUCT_PK PRIMARY KEY (PRODUCT_ID));
```

PAYAP UNIVERSITY

Foreign Keys

```
CREATE TABLE CUSTOMER_T
              (CUSTOMER_ID                NUMBER(11, 0) NOT NULL,
               CUSTOMER_NAME              VARCHAR2(25)  NOT NULL,
               CUSTOMER_ADDRESS           VARCHAR2(30),
               CITY                       VARCHAR2(20),
               STATE                      VARCHAR2(2),
               POSTAL_CODE                VARCHAR2(9),
CONSTRAINT CUSTOMER_PK PRIMARY KEY (CUSTOMER_ID));
```

PAYAP UNIVERSITY

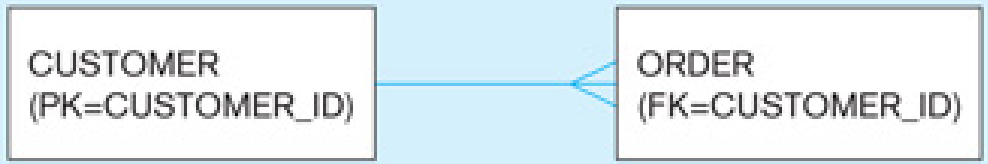
Foreign Keys

```
CREATE TABLE CUSTOMER_T
              (CUSTOMER_ID              NUMBER(11, 0) NOT NULL,
               CUSTOMER_NAME            VARCHAR2(25)  NOT NULL,
               CUSTOMER_ADDRESS         VARCHAR2(30),
               CITY                     VARCHAR2(20),
               STATE                    VARCHAR2(2),
               POSTAL_CODE              VARCHAR2(9),
CONSTRAINT CUSTOMER_PK PRIMARY KEY (CUSTOMER_ID));
```



```
CREATE TABLE ORDER_T
              (ORDER_ID                 NUMBER(11, 0) NOT NULL,
               ORDER_DATE               DATE DEFAULT SYSDATE,
               CUSTOMER_ID              NUMBER(11, 0),
CONSTRAINT ORDER_PK PRIMARY KEY (ORDER_ID),
CONSTRAINT ORDER_FK FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER_T(CUSTOMER_ID));
```

Data integrity constraints



Restricted Update: A customer ID can only be deleted if it is not found in ORDER table.

```
CREATE TABLE CUSTOMER_T
                (CUSTOMER_ID          INTEGER DEFAULT 'C999'   NOT NULL,
                 CUSTOMER_NAME        VARCHAR(40)              NOT NULL,
                 . . .
CONSTRAINT CUSTOMER_PK PRIMARY KEY (CUSTOMER_ID),
ON UPDATE RESTRICT);
```

Cascaded Update: Changing a customer ID in the CUSTOMER table will result in that value changing in the ORDER table to match.

```
. . . ON UPDATE CASCADE);
```

Set Null Update: When a customer ID is changed, any customer ID in the ORDER table that matches the old customer ID is set to NULL.

```
. . . ON UPDATE SET NULL);
```

Set Default Update: When a customer ID is changed, any customer ID in the ORDER tables that matches the old customer ID is set to a predefined default value.

```
. . . ON UPDATE SET DEFAULT);
```

PAYAP UNIVERSITY

Changing and removing tables

DROP TABLE table_name;

DROP TABLE beer

PAYAP UNIVERSITY

Changing and removing tables

DROP TABLE table_name;

```
DROP TABLE beer
```

ALTER TABLE table_name alter_table_action;

```
ALTER TABLE beer ADD COLUMN alcohol_level DECIMAL(4,2);
ALTER TABLE beer ADD alcohol_level DECIMAL(4,2);
ALTER TABLE beer DROP alcohol_level;
```

PAYAP UNIVERSITY

ALTER TABLE actions

ADD [COLUMN] column_definition;

ALTER [COLUMN] column_name SET DEFAULT default_value;

ALTER [COLUMN] column_name DROP DEFAULT;

DROP [COLUMN] column_name [RESTRICT][CASCADE];

ADD table_constraint;

PAYAP UNIVERSITY

DML: INSERT

INSERT is used to populate tables with data

```
INSERT INTO beers VALUES('Bud Lite', 'Anheuser-Busch');

INSERT INTO product (product_ID, product_description,product_finish,
standard_price, product_line_Id) VALUES (1, End Table, Cherry, 175, 8);
```

PAYAP UNIVERSITY

DML: DELETE

```
DELETE FROM beers;
```

```
DELETE FROM beers WHERE manufacturer = 'Anheuser-Busch';
```

PAYAP UNIVERSITY

DML: UPDATE

```
UPDATE Sells SET price = 5.2 WHERE beer = 'Bud Lite' and bar='Tuskers';
```

DML:SELECT

Clauses of the SELECT statement:

- SELECT
  List the columns (and expressions) that should be returned from the query
- FROM
  Indicate the table(s) or view(s) from which data will be obtained
- WHERE
  Indicate the conditions under which a row will be included in the result
- GROUP BY
  Indicate categorization of results
- HAVING
  Indicate the conditions under which a category (group) will be included
- ORDER BY
  Sorts the result according to specified criteria

PAYAP UNIVERSITY

SELECT by example

What beers are made by Anheuser-Busch?
BEERS(name, manufacturer)

```
SELECT name FROM Beers WHERE manufacturer = 'Anheuser-Busch';

SELECT * FROM Beers WHERE manufacturer = 'Anheuser-Busch';

SELECT name AS beer FROM Beers WHERE manufacturer = 'Anheuser-Busch';
```

PAYAP UNIVERSITY

SELECT by example (Renaming)

SELLS(bar,beer, price)

```
SELECT bar, beer, price*120 AS priceInYen
FROM Sells;
```

SELECT by example:AND, OR, NOT

Find the price Joe's Bar charges for Bud.
SELLS(bar,beer, price)

SELECT by example:AND, OR, NOT

Find the price Joe's Bar charges for Bud.
SELLS(bar,beer, price)

```
SELECT price FROM Sells
WHERE bar = 'Joe"s Bar' AND beer = 'Bud';
```

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to
!=	Not equal to

PAYAP UNIVERSITY

Pattern Matching in SQL

A condition can compare a string to a pattern by:
`attribute LIKE pattern` or
`attribute NOT LIKE pattern`

% - is used to represent any collection of characters
_ - is used to represent exactly one character

PAYAP UNIVERSITY

Pattern Matching in SQL

A condition can compare a string to a pattern by:
`attribute LIKE pattern` or
`attribute NOT LIKE pattern`

% - is used to represent any collection of characters
_ - is used to represent exactly one character

Find drinkers whose phone has exchange 555.
DRINKERS(name, address, phone)

PAYAP UNIVERSITY

Pattern Matching in SQL

A condition can compare a string to a pattern by:
`attribute LIKE pattern` or
`attribute NOT LIKE pattern`

% - is used to represent any collection of characters
_ - is used to represent exactly one character

Find drinkers whose phone has exchange 555.
DRINKERS(name, address, phone)

```
SELECT name FROM Drinkers WHERE phone LIKE '%555-_ _ _ _;
```

PAYAP UNIVERSITY

SELECT by example:Alias

```
SELECT B.name AS title
FROM bars AS B;
```

PAYAP UNIVERSITY

Sorting Results: ORDER BY

Sorts the rows of the result in ascending or descending order.

Retrieve the beer prices at each bar, sort the results first by bar, and within a bar by price

PAYAP UNIVERSITY

Sorting Results: ORDER BY

Sorts the rows of the result in ascending or descending order.

Retrieve the beer prices at each bar, sort the results first by bar, and within a bar by price

```
SELECT bar, beer, price
FROM sells
ORDER BY bar, price;
```

PAYAP UNIVERSITY

SELECT with functions

`COUNT, MIN, MAX, SUM, AVG`

How many different beers are sold at *Tuskers*?

PAYAP UNIVERSITY

SELECT with functions

`COUNT, MIN, MAX, SUM, AVG`

How many different beers are sold at *Tuskers*?

```
SELECT COUNT(*)
FROM sells
WHERE bar='Tuskers';
```

PAYAP UNIVERSITY

SELECT with functions

`COUNT, MIN, MAX, SUM, AVG`

How many different beers are sold at *Tuskers*?

```
SELECT COUNT(*)
FROM sells
WHERE bar='Tuskers';
```

What is the first manufacturer name in the beers table (alphabeticaly)?

PAYAP UNIVERSITY

SELECT with functions

`COUNT, MIN, MAX, SUM, AVG`

How many different beers are sold at *Tuskers*?

```
SELECT COUNT(*)
FROM sells
WHERE bar='Tuskers';
```

What is the first manufacturer name in the beers table (alphabeticaly)?

```
SELECT MIN (manufacturer)
FROM beers;
```

Find the average price of Bud.

PAYAP UNIVERSITY

SELECT with functions

`COUNT, MIN, MAX, SUM, AVG`

How many different beers are sold at *Tuskers*?

```
SELECT COUNT(*)
FROM sells
WHERE bar='Tuskers';
```

What is the first manufacturer name in the beers table (alphabeticaly)?

```
SELECT MIN (manufacturer)
FROM beers;
```

Find the average price of Bud.

```
SELECT AVG(price)
FROM Sells
WHERE beer = 'Bud';
```

PAYAP UNIVERSITY

DISTINCT values

```
SELECT manufacturer
FROM beers;
```

```
SELECT DISTINCT manufacturer
FROM beers;
```

PAYAP UNIVERSITY

IN and NOT IN

```
SELECT bar,beer, price
FROM sells
WHERE bar IN ('Tuskers', 'NorthGate', 'Warm up');
```

PAYAP UNIVERSITY

Categorize Results: GROUP BY

Groups rows in an intermediate results table where the values in those rows are the same for one or more columns

Count the number of beer types sold per bar.

```
SELECT bar, COUNT(beer)
FROM sells
GROUP BY bar;
```

PAYAP UNIVERSITY

Categorize Results: GROUP BY

Groups rows in an intermediate results table where the values in those rows are the same for one or more columns

Count the number of beer types sold per bar.

```
SELECT bar, COUNT(beer)
FROM sells
GROUP BY bar;
```

Find the average price for each beer

```
SELECT beer, AVG(price)
FROM Sells
GROUP BY beer;
```

PAYAP UNIVERSITY

Categorize Results: GROUP BY

Find, for each drinker, the average price of Bud at the bars they frequent.

PAYAP UNIVERSITY

Categorize Results: GROUP BY

Find, for each drinker, the average price of Bud at the bars they frequent.

```
SELECT drinker, AVG(price)
FROM Frequents, Sells
WHERE beer = 'Bud' AND Frequents.bar = Sells.bar
GROUP BY drinker;
```

PAYAP UNIVERSITY

Qualifying Results: HAVING

Can only be used following GROUP BY and acts as a secondary WHERE clause, returning only those groups that meet the specified condition

Find only the bars with more than 3 different beers

```
SELECT bar, COUNT(beer)
FROM sells
GROUP BY bar
HAVING COUNT(beer) > 3;
```

PAYAP UNIVERSITY

Qualifying Results: HAVING

Find the average price of those beers that are either served in at least 3 bars or manufactured by Anheuser-Busch.

PAYAP UNIVERSITY

Qualifying Results: HAVING

Find the average price of those beers that are either served in at least 3 bars or manufactured by Anheuser-Busch.

```
SELECT beer, AVG(price)
FROM Sells
GROUP BY beer
HAVING COUNT(*) >= 3 OR
beer IN (
SELECT name
FROM Beers
WHERE manufacturer = 'Anheuser-Busch' );
```