Functions in Programming

- a *subroutine* associates a set of statements with a name
  - may have one or more *parameters*
  - may have a *return value*
- subroutines with a return value are called *functions*

  though this terminology is used sloppily – often any subroutine is called a function, and in object-oriented languages like Java, any operation defined on an object is called a *method*, whether or not it returns a value
- the *header* (or *prototype*) of a function defines its parameters and return value, including types

```
int square ( int n ) {
  return n*n;
}
```

square: *int* → *int*

```
int mult ( int a, int b ) {
  return a*b;
}
```

mult: *int* ✕ *int* → *int*

Functions in Programming

- functions in programming are often not true mathematical functions
  - many functions in programming are really *partial functions* which map a subset of *A* to *B*

```
// n >= 1
int square ( int n ) {
  return n*n;
}
```

square: *int* → *int*

  - functions in programming don't always return only a single value for particular parameter values

```
int random ( int n ) {
  …
}
```

random: *int* → *int*

First-Class Functions

- in mathematics, the elements in a set can be anything – including other sets, ordered pairs, and functions
- some programming languages support *first-class functions* where
  - a function can be passed as a parameter to a function
  - a function can be returned from a function
  - a function can be assigned to a variable and used later

  just like any other type

Applications of First-Class Functions

- e.g. define a function corresponding to the Σ operator

```
sum ( f, a, b ) {
  total = 0;
  for ( i = a ; i <= b ; i++ ) {
    total += f(i);
  }
  return total;
}
```

$$\sum_{i=a}^{b} f(i)$$

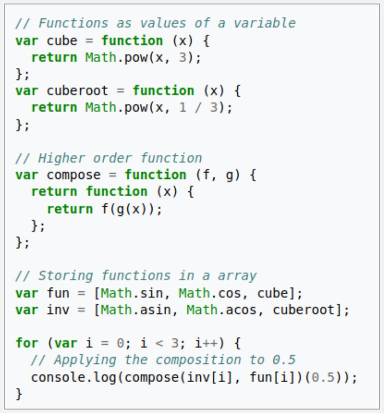
then

```
sum( function(n) { return n*n; }, 1, 100 )
```

```
square = function(n) { return n*n; }
```

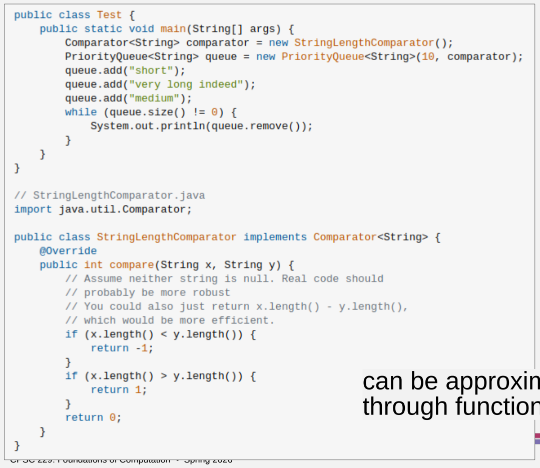
```
sum(square,1,100)
```

First-Class Functions in JavaScript



```javascript
// Functions as values of a variable
var cube = function (x) {
  return Math.pow(x, 3);
};
var cuberoot = function (x) {
  return Math.pow(x, 1 / 3);
};

// Higher order function
var compose = function (f, g) {
  return function (x) {
    return f(g(x));
  };
};

// Storing functions in a array
var fun = [Math.sin, Math.cos, cube];
var inv = [Math.asin, Math.acos, cuberoot];

for (var i = 0; i < 3; i++) {
  // Applying the composition to 0.5
  console.log(compose(inv[i], fun[i])(0.5));
}
```

First-Class Functions (More or Less) in Java

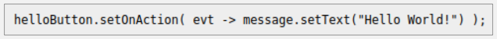


```java
public class Test {
    public static void main(String[] args) {
        Comparator<String> comparator = new StringLengthComparator();
        PriorityQueue<String> queue = new PriorityQueue<String>(10, comparator);
        queue.add("short");
        queue.add("very long indeed");
        queue.add("medium");
        while (queue.size() != 0) {
            System.out.println(queue.remove());
        }
    }
}

// StringLengthComparator.java
import java.util.Comparator;

public class StringLengthComparator implements Comparator<String> {
    @Override
    public int compare(String x, String y) {
        // Assume neither string is null. Real code should
        // probably be more robust
        // You could also just return x.length() - y.length(),
        // which would be more efficient.
        if (x.length() < y.length()) {
            return -1;
        }
        if (x.length() > y.length()) {
            return 1;
        }
        return 0;
    }
}
```

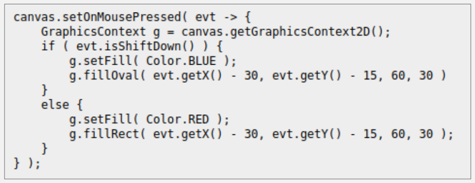
can be approximated
through functional interfaces

First-Class Functions (More or Less) in Java

- newer versions of Java support lambda expressions
  - can be subroutines – they don't have to be true functions



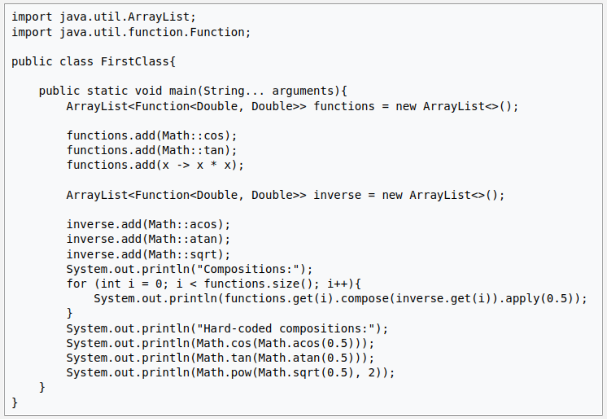
```java
helloButton.setOnAction( evt -> message.setText("Hello World!") );
```



```java
canvas.setOnMousePressed( evt -> {
    GraphicsContext g = canvas.getGraphicsContext2D();
    if ( evt.isShiftDown() ) {
        g.setFill( Color.BLUE );
        g.fillOval( evt.getX() - 30, evt.getY() - 15, 60, 30 )
    }
    else {
        g.setFill( Color.RED );
        g.fillRect( evt.getX() - 30, evt.getY() - 15, 60, 30 );
    }
} );
```

and define a generic functional interface `Function`

First-Class Functions (More or Less) in Java



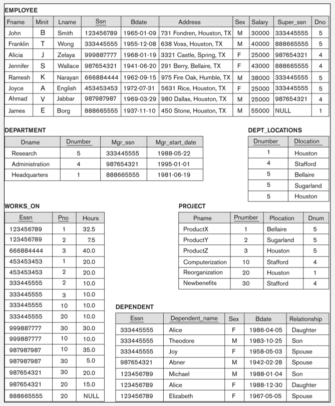
```java
import java.util.ArrayList;
import java.util.function.Function;

public class FirstClass{

    public static void main(String... arguments){
        ArrayList<Function<Double, Double>> functions = new ArrayList<>();

        functions.add(Math::cos);
        functions.add(Math::tan);
        functions.add(x -> x * x);

        ArrayList<Function<Double, Double>> inverse = new ArrayList<>();

        inverse.add(Math::acos);
        inverse.add(Math::atan);
        inverse.add(Math::sqrt);
        System.out.println("Compositions:");
        for (int i = 0; i < functions.size(); i++){
            System.out.println(functions.get(i).compose(inverse.get(i)).apply(0.5));
        }
        System.out.println("Hard-coded compositions:");
        System.out.println(Math.cos(Math.acos(0.5)));
        System.out.println(Math.tan(Math.atan(0.5)));
        System.out.println(Math.pow(Math.sqrt(0.5), 2));
    }
}
```

Databases

- database
  - the data – a structured collection of information
- *structured data* is highly organized and easily searchable
  - conforms to a defined data model or schema
  - *relational model* is based on tables (with rows and columns)
    - most common
    - general purpose
  - object-oriented database stores objects (instance variables and methods)
- *unstructured data* lacks a defined data model or schema
  - does not fit neatly into data tables (or some other predefined format)
  - e.g. text, images, videos, documents

Relational Databases

- a relation is a cross product of sets
- a relational database consists of tables
  - data types define sets
  - each table is a relation
- each row in a table groups together data values about a particular thing
  - duplicate rows are not allowed (must differ by at least one value)
    - *primary key* uniquely identifies each row – PK must be unique
  - order of the rows doesn't matter
  - order of the columns doesn't matter
  - data values are *atomic*



EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

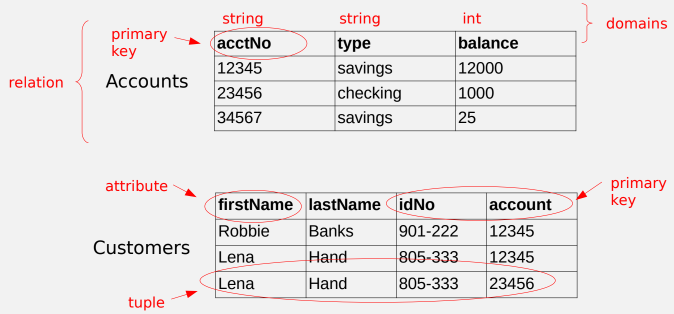
PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

The Relational Model: Structure



string string int

domains

primary key

relation

Accounts

acctNo	type	balance
12345	savings	12000
23456	checking	1000
34567	savings	25

attribute

primary key

Customers

firstName	lastName	idNo	account
Robbie	Banks	901-222	12345
Lena	Hand	805-333	12345
Lena	Hand	805-333	23456

tuple

relation schema

Accounts(acctNo,type,balance)
Customers(firstName,lastName,idNo,account)

database schema
(with primary key)

SQL

- SQL is the standard language for interacting with relational databases
  - supported by databases from many different vendors (though there are incompatibilities due to incomplete implementations and non-standard extensions)
- technical detail: bags vs sets
  - in the relational model, relations are sets
    - no duplicate tuples
    - order of tuples doesn't matter
  - in practice, tables are implemented as bags
    - order of rows doesn't matter
    - *duplicates are not automatically removed from query results*

Basic Queries

```
SELECT a1,a2,...,an
FROM R
WHERE C
```

- (only) table and column names are case-sensitive
- resulting table has
  - the rows from R for which C is true
  - the columns a1,a2,…,an

```
SELECT DISTINCT a1,a2,...,an
FROM R
WHERE C
```

  - as the last step, remove duplicate rows (duplicates identified based only on columns a1,a2,...an)

Basic Queries

SELECT

- can contain * (all columns), arithmetic expressions
- can rename individual columns of the result using `AS`

FROM

- can rename the whole relation using `AS`
- can contain `R,S` (cross product), `R JOIN S ON C`, `R NATURAL JOIN S`
  - `JOIN` keeps all columns of R and S – equivalent to `FROM R,S WHERE C`
  - `NATURAL JOIN` keeps only one copy of each join column
- qualify column name with relation name or alias to disambiguate same-named columns

WHERE

- can contain arithmetic expressions, <, <=, >, >=, =, <>, `AND`, `OR`, `NOT`, `IS NULL`, `IS NOT NULL`, `LIKE`, `NOT LIKE`
  - wildcards % (0 or more), _ (single character) allowed in `LIKE`/`NOT LIKE` patterns – used for pattern-matching within column values

Data Modification

- insert rows into a table

```
INSERT INTO R(A1,…,An)
VALUES (v1,…,vn)
```

  - uses specified values for attributes listed and default values for the others

- delete rows from a table

```
DELETE
FROM R
WHERE <condition>
```

  - all rows satisfying the condition are deleted
    - without WHERE, all rows are deleted (but not the table itself)

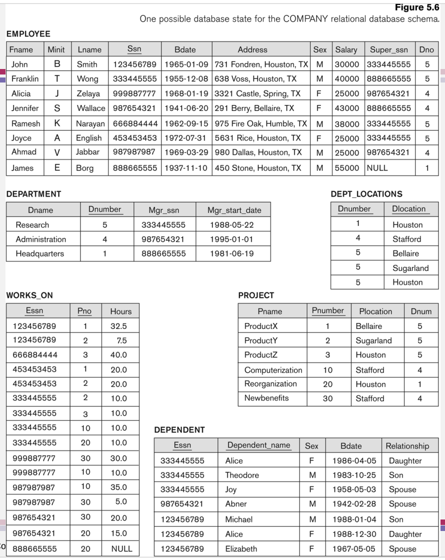


Figure 5.6
One possible database state for the COMPANY relational database schema.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

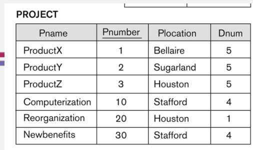
PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

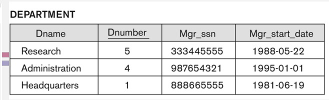
Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

```
SELECT DISTINCT Dependent_name
FROM DEPENDENT
WHERE Sex=’F’

SELECT *
FROM DEPT_LOCATIONS JOIN PROJECT ON Dnumber=Dnum

SELECT *
FROM WORKS_ON JOIN PROJECT ON Pno=Pnmber
WHERE Plocation='Houston' AND Hours >= 10

SELECT Fname,Lname
FROM EMPLOYEE JOIN DEPARTMENT ON Super_ssn=Mgr_ssn
WHERE Dno=Dnumber

SELECT Fname,Lname,Dependent_name,Relationship
FROM EMPLOYEE NATURAL JOIN DEPENDENT

SELECT *
FROM DEPARTMENT NATURAL JOIN DEPT_LOCATIONS
```



PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4



DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

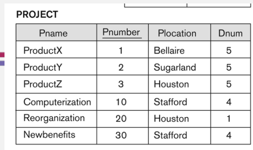
Assuming that all default values are NULL, are these legal INSERT statements?

```
INSERT INTO PROJECT(Pname,Plocation,Dnum)
            VALUES ("ProductA","Austin",1)
```
🚫

```
INSERT INTO PROJECT(Pnumber,Pname,Plocation,Dnum)
            VALUES (40,"ProductA","Austin",1)
```
🙂

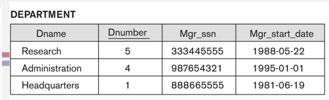
```
INSERT INTO PROJECT(Pnumber,Pname,Plocation,Dnum)
            VALUES (40,"ProductA",NULL,1)
```
🙂

```
INSERT INTO PROJECT(Pnumber,Pname,Plocation,Dnum)
            VALUES (10,"ProductA","Austin",1)
```
🚫



PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4



DEPARTMENT

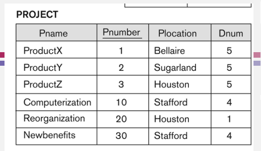
Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

Assuming that all default values are NULL, are these legal INSERT statements?

```
INSERT INTO PROJECT(Pnumber,Pname,Plocation,Dnum)
            VALUES ("ProductA",40,"Austin",1)
```
🚫

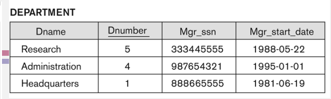
```
INSERT INTO PROJECT(Pnumber,Pname)
            VALUES (40,"ProductA")
```
🙂

Quiz



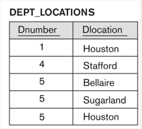
PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4



DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19



DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Assuming that all default values are NULL, are these legal INSERT statements?

```
INSERT INTO PROJECT(Pnumber,Plocation,Dnum)
SELECT * FROM DEPT_LOCATIONS
```

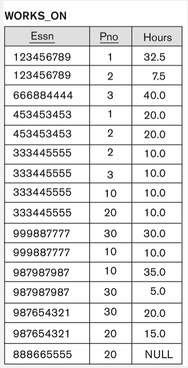
🚫

```
INSERT INTO PROJECT(Pnumber,Plocation,Dnum)
SELECT Dnumber+50,Dlocation,Dnumber FROM DEPT_LOCATIONS
```

🚫

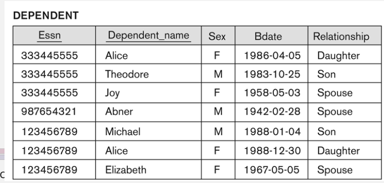
How many rows will be deleted?

```
DELETE FROM WORKS_ON
WHERE Hours < 10

DELETE FROM WORKS_ON W
WHERE NOT EXISTS ( SELECT *
                   FROM DEPENDENT D
                   WHERE D.Essn=W.Essn )
```



WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL



DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse