Chapter 7
JDBC

Java Database Connectivity

Overview

- Database overview
- JDBC

JDBC

Java Database Connectivity

*JDBC Goal

- ???
  - Relational databases
- Yet be independent of the actual database, e.g.:
  - ????

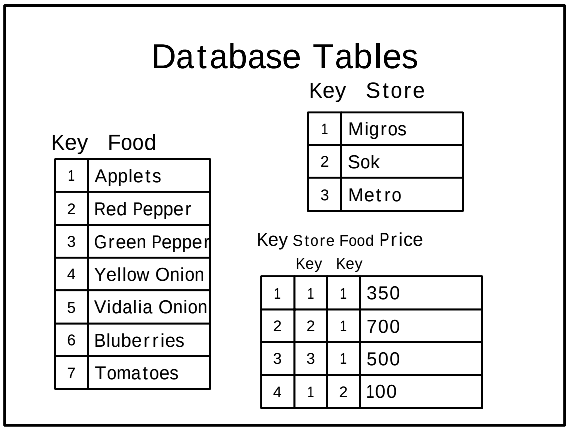
(RDBMS)

- Based on the concept of tables of data
- Each table is called a relation
- Relational DBs sit on a sound theoretical foundation based on E.F. Codd's relational calculus
  - operations on relational DBs result in new (temporary) tables of results
  - important (but costly) JOIN operation

Database Tables

Key	Food
1	Applets
2	Red Pepper
3	Green Pepper
4	Yellow Onion
5	Vidalia Onion
6	Bluberries
7	Tomatoes

Key	Store
1	Migros
2	Sok
3	Metro

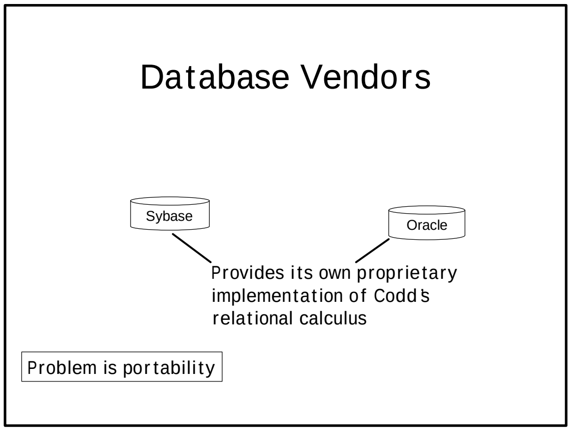


Database Tables

Key	Food
1	Applets
2	Red Pepper
3	Green Pepper
4	Yellow Onion
5	Vidalia Onion
6	Bluberries
7	Tomatoes

Key	Store
1	Migros
2	Sok
3	Metro

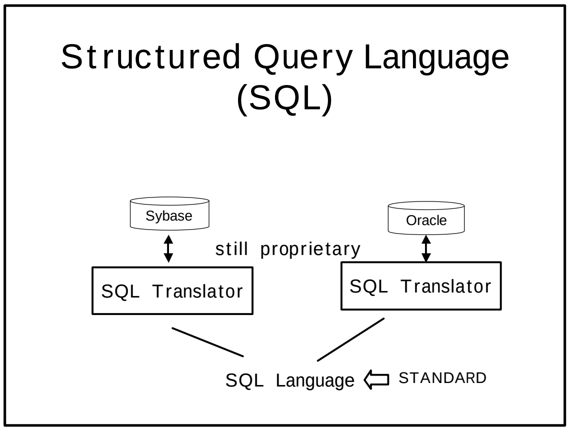
Key	Store	Food	Price
	Key	Key	
1	1	1	350
2	2	1	700
3	3	1	500
4	1	2	100



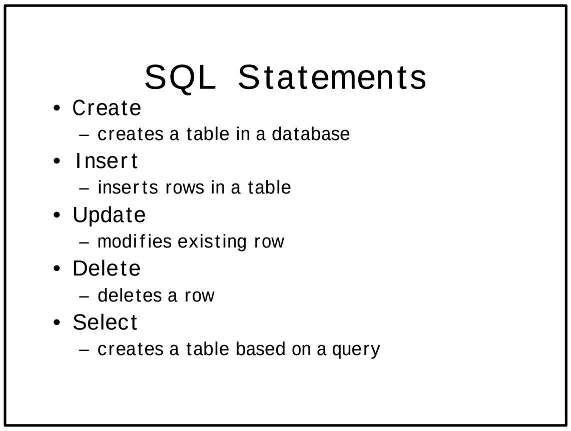
Database Vendors

Provides its own proprietary implementation of Codd's relational calculus

Problem is portability

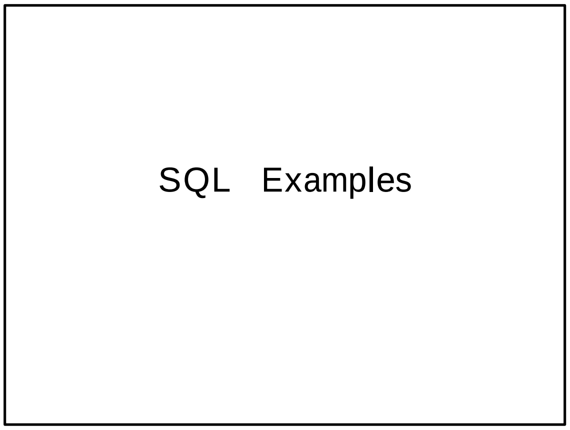


Structured Query Language (SQL)

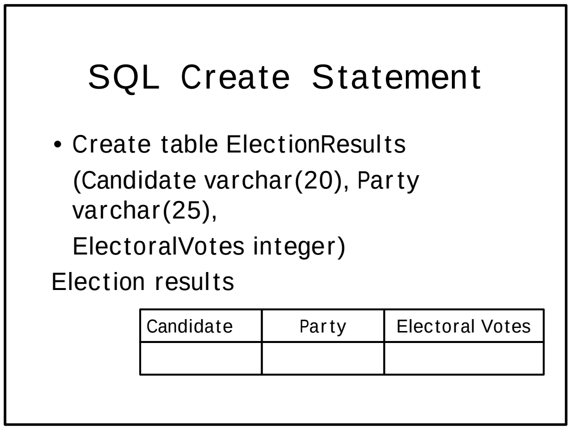


SQL Statements

- Create
  - creates a table in a database
- Insert
  - inserts rows in a table
- Update
  - modifies existing row
- Delete
  - deletes a row
- Select
  - creates a table based on a query



SQL Examples



SQL Create Statement

- Create table ElectionResults (Candidate varchar(20), Party varchar(25), ElectoralVotes integer)

Election results

Candidate	Party	Electoral Votes

SQL Data Types

SQL Type	Java
bigint	long
bit	boolean
char	String
integer	int
time	java.sql.Time
varchar	String

SQL Insert Statement

```
Insert into ElectionResults
  values(‘Clinton’, ‘Democratic’, 223)
```

ElectionResults

Candidate	Party	Electoral Votes
Clinton	Democratic	223

SQL SELECT (Query)

- An SQL Query:

```
SELECT <fields>
FROM <Table>
WHERE <Criteria>
```

Candidate	Party	Electoral Votes
Clinton	Democratic	223
Bush	Republican	134
Perot	Independent	0

SELECT . . .

Candidate	Party	Electoral Votes
Clinton	Democratic	223
Bush	Republican	134
Perot	Independent	0

SQL:

```
SELECT Candidate, Party
FROM ElectionResults
WHERE ElectrolVotes > 0
```

Temporary Relation
(Result Set)

Candidate	Party
Clinton	Democratic
Bush	Republican

SQL - the UPSIDE

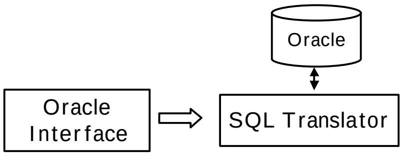
- SQL is a standard accepted by all major vendors

SQL - the DOWNSIDE

- SQL is not a “full” programming language
- SQL does not have all of the “Big 3”
  - sequence
  - iteration
  - conditional

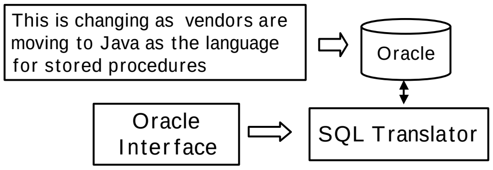
SQL - the DOWNSIDE

- Users are locked into using the proprietary interfaces provided by vendors

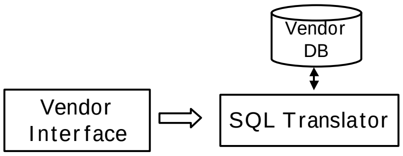


SQL - the DOWNSIDE

- Users are locked into using proprietary stored procedures that greatly improve DB performance

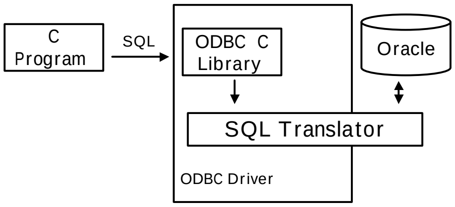

This is changing as vendors are moving to Java as the language for stored procedures

SQL - Getting Open



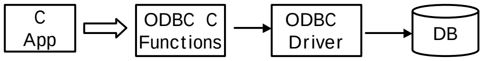
Open Database Connectivity (ODBC)

- Microsoft interface for accessing DBs from C programs



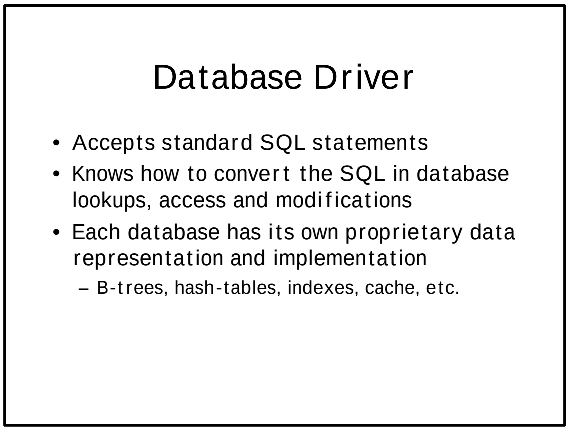
*ODBC

- Microsoft introduced ODBC for accessing relational databases from the C language
- C application programs pass SQL statement to C functions that implement the ODBC interface
- The object code (driver) is provided by the vendor (or third party)



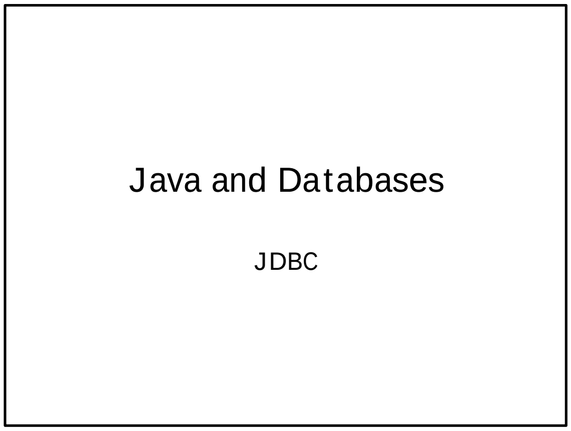
Drivers

- Allow a specific piece of hardware to connect to a system
- Support standard interfaces
- Hide the details of how a specific piece of hardware implements interface
  - video card drivers
  - scanner drivers
  - … database drivers



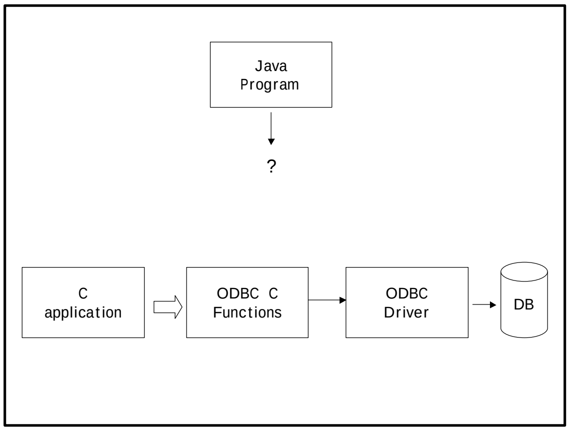
Database Driver

- Accepts standard SQL statements
- Knows how to convert the SQL in database lookups, access and modifications
- Each database has its own proprietary data representation and implementation
  - B-trees, hash-tables, indexes, cache, etc.



Java and Databases

JDBC



Java Program

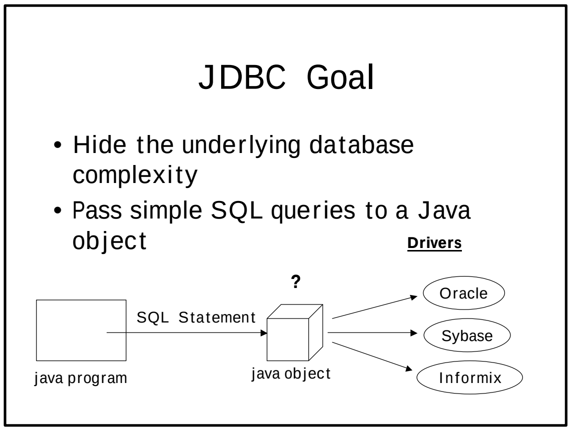
?

C application

ODBC C Functions

ODBC Driver

DB



JDBC Goal

- Hide the underlying database complexity
- Pass simple SQL queries to a Java object

Drivers

?

SQL Statement

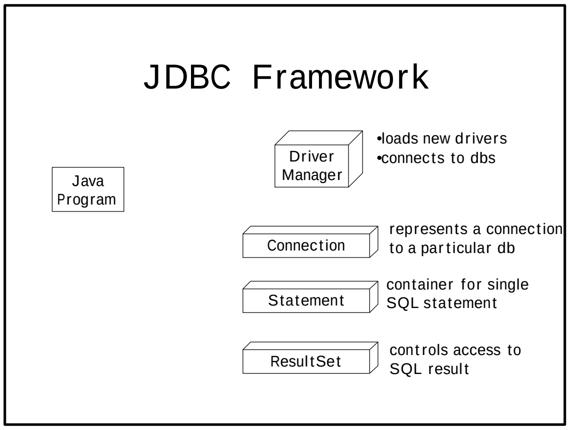
java program

java object

Oracle

Sybase

Informix



JDBC Framework

Java Program

Driver Manager

•loads new drivers
•connects to dbs

Connection

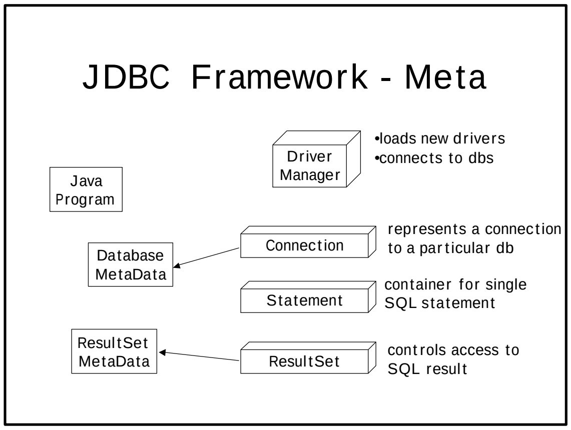
represents a connection to a particular db

Statement

container for single SQL statement

ResultSet

controls access to SQL result



JDBC Framework - Meta

Java Program

Driver Manager

•loads new drivers
•connects to dbs

Database MetaData

Connection

represents a connection to a particular db

Statement

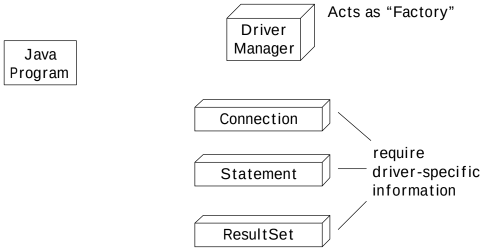
container for single SQL statement

ResultSet MetaData

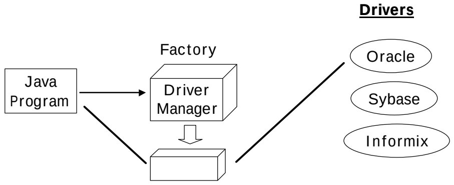
ResultSet

controls access to SQL result

JDBC Framework



DriverManager

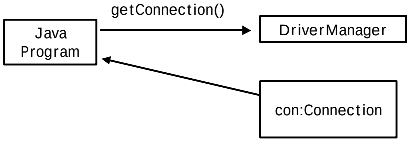


DriverManager asks all drivers if they know
how to connect to specified DB.
The first one that can, returns a Connection object.

Registering Drivers with DriverManager

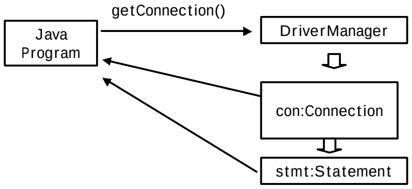
- DriverManager will look for "sql.drivers" property in system properties
- Common technique is to load the driver directly using:
  - Class.forName("acme.db.Driver")

Connection



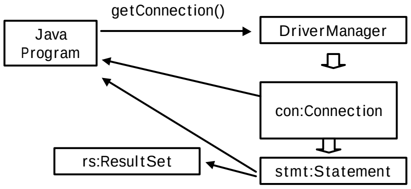
```
String url = "jdbc:odbc:mydb";
String user = "fred";
String passwd = "zorkon";
Connection con = DriverManager.getConnection(url, user, passwd);
```

Statement



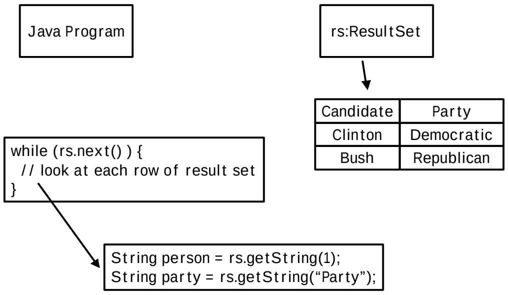
```
Connection con = DriverManager.getConnection(url, user, passwd);
Statement stmt = con.createStatement();
```

ResultSet



```
ResultSets rs = stmt.executeQuery("SELECT Candidate, Party
  FROM ElectionResults WHERE ElectoralVotes > 0");
```

*ResultSet Iteration

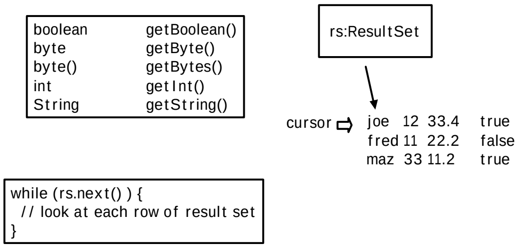


Candidate	Party
Clinton	Democratic
Bush	Republican

```
while (rs.next() ) {
  // look at each row of result set
}
```

```
String person = rs.getString(1);
String party = rs.getString(“Party”);
```

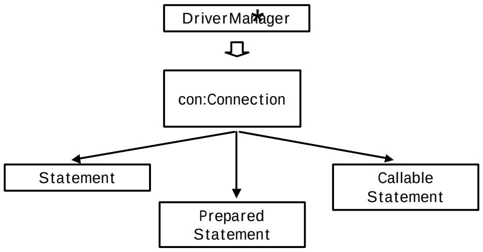
ResultSet Methods



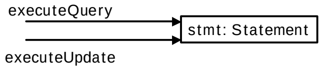
boolean	getBoolean()
byte	getByte()
byte()	getBytes()
int	getInt()
String	getString()

joe	12	33.4	true
fred	11	22.2	false
maz	33	11.2	true

```
while (rs.next() ) {
  // look at each row of result set
}
```



*Ad-hoc Queries with the Statement Object

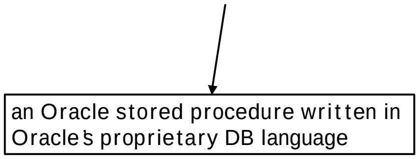


→ s: Statement

```
Statement s = conn.createStatement();

s.executeUpdate(“CREATE TABLE employee” +
        “(name char(30), age integer, salary integer)”)
```

*for using stored procedures


an Oracle stored procedure written in Oracle’s proprietary DB language

Using Java for stored procedures

syntactic style

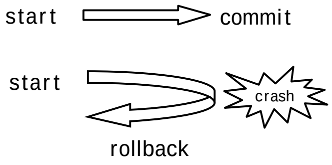
The Java QuickUpdate.class
•can be loaded into the database
•or run external to the database!!

MetaData

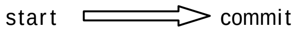
- You can use java.sql.DatabaseMetaData class to query a database about its services
- Can ask about:
  - kinds of joins
  - stored procedures
  - transaction support

Transactions

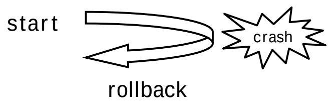
- A unit of work in a database
- DB software guarantees correctness when DB changes are made



DB Transactions



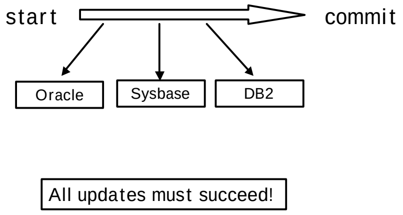
- All major commercial databases support transactions
- Allows programmers to group several updates as ONE transaction
- All or none will occur



ACID Properties (of transactions)

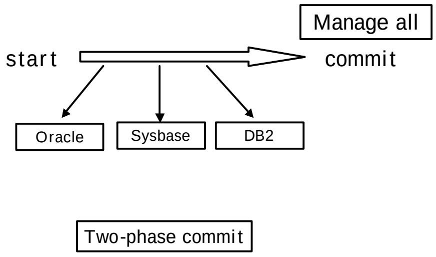
- Atomic
  - transaction is indivisable unit of work
- Consistent
  - it it cannot leave system in stable state it must return system to initial state
- Isolated
  - runs independently of other transactions
  - effects not seen until it commits
- Durable
  - persistent – effect survives system failure

Distributed Transactions



All updates must succeed!

Transaction Monitors

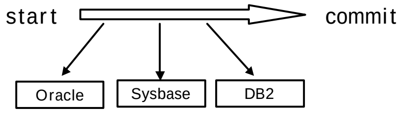


Two-phase commit

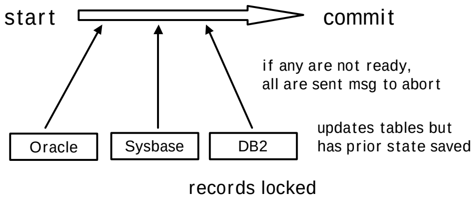
Transaction Delimiters

System	Start		Abort
CICS	SYNCPOINT	SYNCPOINT	ROLLBACK
Tuxedo	TPBEGIN	TPCOMMIT	TPABORT
OSI TP	C-BEGIN	C-COMMIT	C-ROLLBACK

*Two-Phase Commit – Phase I



*Two-Phase Commit – Phase I

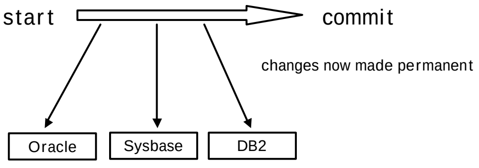


if any are not ready,
all are sent msg to abort

updates tables but
has prior state saved

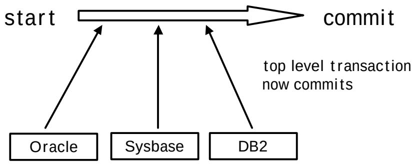
records locked

*Two-Phase Commit – Phase II



changes now made permanent

*Two-Phase Commit – Phase II



Transactions

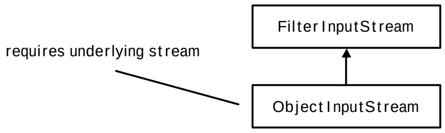
- JDBC user can choose between
  - auto-commit mode: every statement is committed immediately
  - transaction mode:
    - statements are ended either with
      - Connection.commit(), or
      - Connection.abort()
- Mode is switched with
  - Connection.setAutoCommint(boolean)

JDBC Example JDBC and Object Serialization Application Servers

Overview

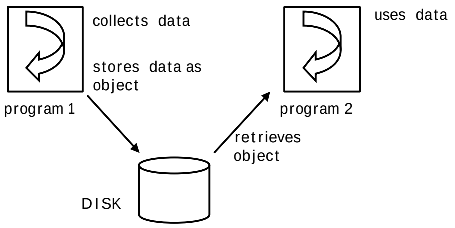
- Save an object to a database
- Using
  - JDBC connectivity
  - Object serialization
- Code example from Java Developers Journal, vol.2, Issue 3.
- www.JavaDevelopersJournal.com

ObjectStreams are Filter Streams



```
FileInputStream fin = new FileInputStream("my.dat");
ObjectInputStream oin = new ObjectInputStream(fin);
```

Object Persistence



Relational Database

ElectionResults

Candidate	Party	Electoral Votes

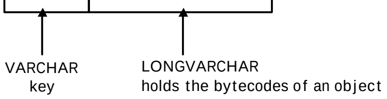
Fields must be defined as as SQL type

- BIGINT
- INTEGER
- CHAR
- VARCHAR
- LONGVARCHAR
- . . .

Relation for Objects

tObject

ObjName	Object



Use PreparedStatement

```
//set up an SQL PreparedStatement
String sql =“Insert into tObject
                    values (Frame1’, ?)”;
PreparedStatement prep=c.prepareStatement(sql);
```

PreparedStatement

Insert into tObject values (Frame1’, ?)”;

Nth ?
starting with 1

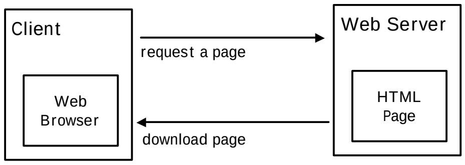
Prepared Statement methods

```
setBoolean(int, boolean);
setDouble(int, double);
setInt(int, int);
setLong(int, long);
setBinaryStream(int, InputStream, int);
```

From Web Server to Application Server

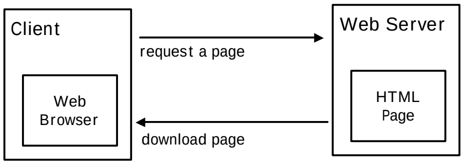
*Static HTML

Designed for file transfer
Slow: download page & close connection
Messages coded as text



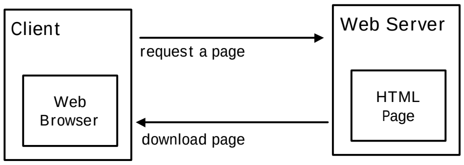
HTTP

Server remembers nothing about the client
Scalable: connections are short-lived
Reliable: client fails – no problem
No database connections



Java Applets

Can be download to any browser

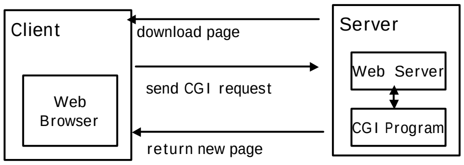


HTML Scripting Languages

- Code can be nested in HTML
  - useful for consistency checking with data entry
- JavaScript
  - not related to the JDK
  - originally known as “LiveScript” from Netscape
- JScript and VBScript from Microsoft

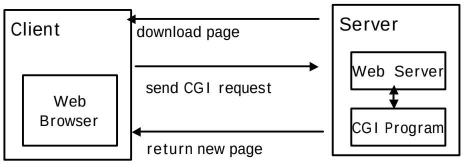
CGI (Common Gateway Interface)

- A protocol for calling programs vs. fetching documents
- Extends the http protocol



CGI Request

http://www.foo.com/cgi-bin/myProg?param1=Bill+Gates&param2= 76



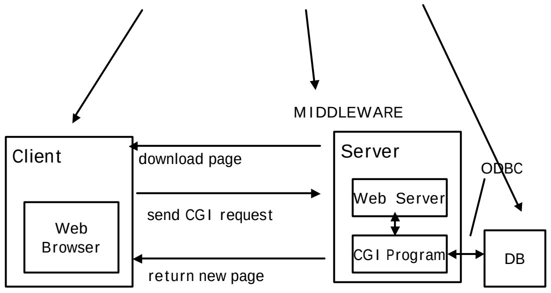
CGI and Forms

- HTML allows forms to be included in a web page
  - text boxes, radio buttons, list boxes.
- The browser can extract user input and pack it into a URL – sent it as a CGI request
- Many search engines use CGI

HTML Forms

```
<FORM METHOD="POST"
  ACTION="http://www.foo.com/cgi-bin/myProg">
<P>
<INPUT TYPE="text" NAME="Credit Card">
<INPUT TYPE="submit" NAME="OK" VALUE="OK">
```

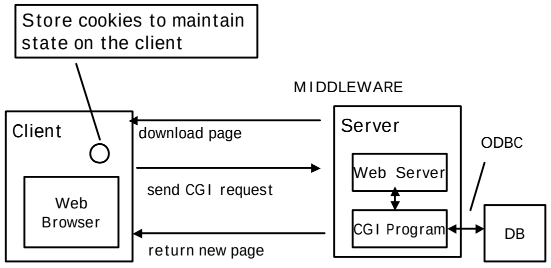
Three Tier Architecture



CGI Problems

- CGI is slow
  - http is slow
  - a new process is started for each request (not good for scalability)
  - request on a LAN = 0.5 – 1.0 secs
    - 100x slower than CORBA or RMI
- Difficult to maintain
  - uses ENVIRONMENT variables
  - PERL and other script languages difficult to understand and maintain

CGI (Partial) Solutions



Use CGI for ...

- Simple input via Forms
  - when HTML GUI is satisfactory
- User submits query and waits for reply
  - database lookup takes a long time

Don't Use CGI . . .

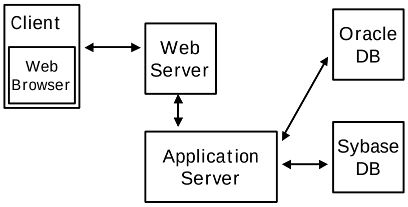
- For transactions
  - no state maintained on server
  - too slow
- For complex transactions

Application Servers

Application Server

- Does not have one fixed definition
  - Name began to appear around 1995
- All the major vendors have one
  - IBM
  - Oracle
  - Netscape
  - Sun
- Smaller vendors:
  - NetDynamics (SUN), Weblogic (BEA), Kiva (Netscape)

N-Tier Architecture



Client

Web Browser

Web Server

Application Server

Oracle DB

Sybase DB

Application Server provides services

- Scalability
  - caching, multiple servers, connection pools
- Multiple database connections
  - JDBC, data caching, business objects
- Transaction support across multiple DBs
- Security
- Software productivity tools
- Multi-tier management support
- Standards support:
  - CORBA, RMI, IIOP, XML

END

Chapter 7 JDBC

Java Database Connectivity - 2

Data Caching Issues

- Caching all data
- Caching no data
- Intermediate models

Data Caching Issues - 1

JDBC data objects

- can cache all of their data locally
  - * appropriate when the data being accessed is fairly stable
  - * is only updated from one client at any given time

Data Caching Issues - 2

JDBC data objects

- can have no cached data (ie, each request is serviced by generating an SQL query to the database that gets the current data from the source)
  - - inappropriate for remote db applications
    - - additional overhead

Data Caching Issues - 3

JDBC data objects

- intermediate caching schemes
  - involve data updates of varying frequency
  - only subset of data served from db is cached

Remote Data Servers

- Keeps the client-side data access lean and free from complex data logic
  - – client has a simplified data access layer
- Helps to provide security
  - – prevents the bytecode reverse-engineering attempts to determine the proprietary structure of databases
  - – prohibits physical access to db server

How can we do this?

- Using message-passing techniques
- Using distributed objects implemented in CORBA, Java RMI, …

1- Message passing with data server

- establish a data server that can respond to messages from clients and access the data referenced in the messages
  - We can model this system after Message and MessageHandler classes and write a Do() method on GetResourceMsg class.

Two-level data caching

Caching occurs both on client agent and in objects on data server.

Constraints:

- Frequency of updating each data item on next data level?
- Caching scheme used in next data level?
- Nature of connection to next data level?
  - bandwidth, reliability, and effect on effective throughput
- Allowed frequency of updates on local cache w/o imposing unreasonable overhead on d access times?

More constraints

- On data server
  - Is data server the single entry point for data clients or are there multiple data servers?
- On data clients
  - Are we only the client accessing d server?
  - Can we use network bandwidth issues alone to decide our caching scheme, or do we have to consider updates from others?

END