SOLUTIONS TO REVIEW QUESTIONS

AND EXERCISES

FOR PART 1 – BACKGROUND (CHAPTERS 1 – 3)

Solutions to Review Questions and Exercises

Chapter 1 Introduction to Databases

Review Questions

1.1 *List four examples of database systems other than those listed in Section 1.1.*

Some examples could be:

- A system that maintains component part details for a car manufacturer;
- An advertising company keeping details of all clients and adverts placed with them;
- A training company keeping course information and participants' details;
- An organization maintaining all sales order information.

1.2 *Discuss each of the following terms:*

Data	For end users, this constitutes all the different values connected with the various objects/entities that are of concern to them. See also Section 1.3.4.
Database	See Section 1.3.1
Database Management System	See Section 1.3.2
Database Application Program	See Section 1.3.3
Data Independence	This is essentially the separation of underlying file structures from the programs that operate on them, also called program-data independence. See also Sections 1.2.2 and 1.3.1.
Security	The protection of the database from unauthorized users, which may involve passwords and access restrictions. See also Section 1.6.
Integrity	The maintenance of the validity and consistency of the database by use of particular constraints that are applied to the data. See also Section 1.6.
Views	These present only a subset of the database that is of particular interest to a user. Views can be customized, for example, field names may change, and they also provide a level of security preventing users from seeing certain data. See also Section 1.3.3.

1.3 *Describe the approach taken to the handling of data in the early file-based systems. Discuss the disadvantages of this approach.*

Focus was on applications for which programs would be written, and all the data required would be stored in a file or files owned by the programs. See also Section 1.2.

Clearly, each program was responsible for only its own data, which could be repeated in other program's data files. Different programs could be written in different languages, and would not be able to access another program's files. This would be true even for those programs written in the same language, because a program needs to know the file structure before it can access it. See also Section 1.2.2.

1.4 *Describe the main characteristics of the database approach and contrast it with the file-based approach.*

Focus is now on the data first, and then the applications. The structure of the data is now kept separate from the programs that operate on the data. This is held in the system catalog or data dictionary. Programs can now share data, which is no longer fragmented. There is also a reduction in redundancy, and achievement of program-data independence. See also Section 1.3.

1.5 *Describe the five components of the DBMS environment and discuss how they relate to each other.*

See Section 1.3.3.

1.6 *Discuss the roles of the following personnel in the database environment:*

Data Administrator	See Section 1.4.1
Database Administrator	See Section 1.4.1
Logical Database Designer	See Section 1.4.2
Physical Database Designer	See Section 1.4.2
Application Developer	See Section 1.4.3
End-Users	See Section 1.4.4

1.7 Discuss the three generations of DBMSs.

The CODASYL and hierarchical approaches represented the first generation of DBMSs. They were based on the concept that smaller components come together as parts of larger components, and so on, until the final product is assembled. This structure, which conforms to an upside down tree, is also known as a hierarchical structure.

Relational DBMSs are referred to as second-generation DBMSs. In 1970, E. F. Codd of the IBM Research Laboratory produced his highly influential paper on the relational data model ("A relational model of data for large shared data banks," Codd, 1970). This paper was very timely and addressed the disadvantages of the former approaches. Many experimental relational DBMSs were implemented thereafter.

In response to the increasing complexity of database applications, two "new" systems have emerged: the object-oriented DBMS (OODBMS) and the object-relational DBMS (ORDBMS). However, unlike previous models, the actual composition of these models is not clear. This evolution represents third generation DBMSs.

1.8 *Discuss the advantages and disadvantages of database management systems.*

See Section 1.6

Exercises

1.9 *Interview some users of database systems. Which DBMS features do they find most useful and why? Which DBMS facilities do they find least useful and why? What do these users perceive to be the advantages and disadvantages of the DBMS?*

Select a variety of users for a particular DBMS. If the users are using different DBMSs, group the answers for the different systems, which will give an overall picture of specific systems.

1.10 *Write a small program (using pseudocode, if necessary) that allows entry and display of client details including a client number, name, address, telephone number, preferred number of rooms, and maximum rent. The details should be stored in a file. Enter a few records and display the details. Now repeat this process but rather than writing a special program, use any DBMS that you have access to. What can you conclude from these two approaches?*

The program can be written in any appropriate programming language, such as Pascal, FORTRAN, C. It should adhere to basic software engineering principles including being well-structured, modular, and suitably commented. It is important to appreciate the process involved even in developing a small program such as this. The DBMS facilities to structure, store, and retrieve data are used to the same effect. The differences in the approaches, such as the effort involved, potential for extension, ability to share the data should be noted.

1.11 *Study the DreamHome case study presented in Section 10.4 and Appendix A. In what ways would a DBMS help this organization? What data can you identify that needs to be represented in the database? What relationships exist between the data? What queries do you think are required?*

It may be useful to review the file-based approach and the database approach here before tackling the first part of the exercise. Careful reading and thinking about how people might use the applications should help in carrying out the rest of the exercise.

1.12 *Study the Wellmeadows Hospital case study presented in Appendix B.3. In what ways would a DBMS help this organization? What data can you identify that needs to be represented in the database? What relationships exist between the data?*

The approach used for Exercise 1.10 should be used for this exercise also.

1.13. *Discuss what you consider to be the three most important advantages for the use of a DBMS for a company like DreamHome and provide a justification for your selection. Discuss what you consider to be the three most important disadvantages for the use of a DBMS for a company like DreamHome and provide a justification for your selection.*

Students should review section 1.6 to make a list of advantages and disadvantes of using a DBMS.

1.14 *Using any Web browser, look at some of the following Web pages and discover the wealth of information available there:*
(a) http://www.oracle.com
(b) http://www.microsoft.com/sql and http://www.microsoft.com/access
(c) http://www.ibm.com/db2
(d) http://www.mysql.com
(e) http://en.wikipedia.org/wiki/database and http://en.wikipedia.org/wiki/DBMS

Students should visit the Web pages listed above to gain understanding about the type of information covered on each Web site. Pages a-d are the major database vendor Web sites while the final two links provide information on databases and DBMS.

Chapter 2 Database Environment

Review Questions

2.1 *Discuss the concept of data independence and explain its importance in a database environment.*

See Section 2.1.5

2.2 *To address the issue of data independence, the ANSI-SPARC three-level architecture was proposed. Compare and contrast the three levels of this model.*

See Section 2.1

2.3 *What is a data model? Discuss the main types of data models.*

An integrated collection of concepts for describing and manipulating data, relationships between data, and constraints on the data in an organization. See Section 2.3.

Object-based data models such as the Entity-Relationship model (see Section 2.3.1). Record-based data models such as the relational data model, network data model, and hierarchical data model (see Section 2.3.2). Physical data models describe how data is stored in the computer (see Section 2.3.3).

2.4 *Discuss the function and importance of conceptual modeling.*

See Section 2.3.4.

2.5 *Describe the types of facility you would expect to be provided in a multi-user DBMS.*

Data Storage, Retrieval and Update	Authorization Services
A User-Accessible Catalog	Support for Data Communication
Transaction Support	Integrity Services
Concurrency Control Services	Services to Promote Data Independence
Recovery Services	Utility Services

See also Section 2.4

2.6 *Of the facilities described in your answer to Question 2.5, which ones do you think would* ***not*** *be needed in a standalone PC DBMS? Provide justification for your answer.*

Concurrency Control Services - only single user.
Authorization Services - only single user, but may be needed if different individuals are to use the DBMS at different times.
Utility Services - limited in scope.
Support for Data Communication - only standalone system.

2.7 *Discuss the function and importance of the system catalog.*

See Section 2.4, Service (2) – User-accessible catalog.

Exercises

2.8 *Analyze the DBMSs that you are currently using. Determine each system's compliance with the functions that we would expect to be provided by a DBMS. What type of language does each system provide? What type of architecture does each DBMS use? Check the accessibility and extensibility of the system catalog. Is it possible to export the system catalog to another system?*

To do this you will need to obtain appropriate information about each system. There should be manuals available or possibly someone in charge of each system who could supply the necessary information.

2.9 *Write a program that stores names and telephone numbers in a database. Write another program that stores names and addresses in a database. Modify the programs to use external, conceptual, and internal schemas. What are the advantages and disadvantages of this modification?*

The programs can be written in any suitable language and should be well structured and appropriately commented. Two distinct files result. The structures can be combined into one containing name, address, and telNo, which can be the representation of both the internal and conceptual schemas. The conceptual schema should be created separately with a routine to map the conceptual to the internal schema. The two external schemas also must be created separately with routines to map the data between the external and the conceptual schema. The two programs should then use the appropriate external schema and routines.

2.10 *Write a program that stores names and dates of birth in a database. Extend the program so that it stores the format of the data in the database; in other words, create a system catalog. Provide an interface that makes this system catalog accessible to external users.*

Again, the program can be written in any suitable language. It should then be modified to add the data format to the original file. This should not be difficult, if the original program is well structured. The interface for other users operates on the data dictionary and is separate from the original program. A menu-based interface is adequate.

Chapter 3 Database Architectures and the Web

Review Questions

3.1 *What is meant by the term 'client-server architecture' and what are the advantages of this approach? Compare the client-server architecture with two other architectures.*

The client is a process that requires some resource, and the server provides the resource. Neither need reside on the same machine. Advantages include:

- Better performance
- Likely reduction in hardware costs
- Reduction in communication costs
- Better consistency

See also Section 3.1.

3.2 *Compare and contrast the two-tier client-server architecture for traditional DBMSs with the three-tier client-server architecture. Why is the latter architecture more appropriate for the Web?*

See Figures 3.5 and 3.6. Architecture maps quite naturally to the Web with a Web browser acting as 'thin' client and Web server acting as an application server (with database server as third layer).

3.3 What is an n-tier architecture?

The three-tier architecture can be expanded to *n* tiers, with additional tiers providing more flexibility and scalability.

3.4 What is middleware? Provide a classification service for middleware.

Middleware is a generic term used to describe software that mediates with other software and allows for communication between disparate applications in a heterogeneous system. The need for middleware arises when distributed systems become too complex to manage efficiently without a common interface. The need to make heterogeneous systems work efficiently across a network and be flexible enough to incorporate frequent modifications led to the development of middleware, which hides the underlying complexity of distributed systems.

3.5 What is a TP Monitor? What advantages does a TP Monitor bring to an OLTP environment?

A TP Monitor forms the middle tier of a three-tier architecture. TP Monitors provide significant advantages, including:

- *Transaction routing:* The TP Monitor can increase scalability by directing transactions to specific DBMSs.
- *Managing distributed transactions:* The TP Monitor can manage transactions that require access to data held in multiple, possibly heterogeneous, DBMSs. For example, a transaction may require to update data items held in an Oracle DBMS at

site 1, an Informix DBMS at site 2, and an IMS DBMS as site 3. TP Monitors normally control transactions using the X/Open Distributed Transaction Processing (DTP) standard. A DBMS that supports this standard can function as a resource manager under the control of a TP Monitor acting as a transaction manager. We discuss distributed transactions and the DTP standard in Chapters 23 and 24.

- *Load balancing:* The TP Monitor can balance client requests across multiple DBMSs on one or more computers by directing client service calls to the least loaded server. In addition, it can dynamically bring in additional DBMSs as required to provide the necessary performance.
- *Funneling:* In environments with a large number of users, it may sometimes be difficult for all users to be logged on simultaneously to the DBMS. In many cases, we would find that users generally do not need continuous access to the DBMS. Instead of each user connecting to the DBMS, the TP Monitor can establish connections with the DBMSs as and when required, and can funnel user requests through these connections. This allows a larger number of users to access the available DBMSs with a potentially much smaller number of connections, which in turn would mean less resource usage.
- *Increased reliability:* The TP Monitor acts as a *transaction manager*, performing the necessary actions to maintain the consistency of the database, with the DBMS acting as a *resource manager*. If the DBMS fails, the TP Monitor may be able to resubmit the transaction to another DBMS or can hold the transaction until the DBMS becomes available again.

3.6 What is a Web service?

Web services allow applications to integrate with other applications across the Internet and may be a key technology that supports B2B (Business to Business) interaction. Unlike other Web-based applications, Web services have no user interface and are not aimed at Web browsers. Web services instead share business logic, data, and processes through a programmatic interface across a network. In this way, it is the applications that interface and not the users. Developers can then add the Web service to a Web page (or an executable program) to offer specific functionality to users.

3.7 What technologies and standards are used to develop Web services and how do they relate to each other?

Key to the Web services approach is the use of widely accepted technologies and standards, such as:

- XML (extensible Markup Language).
- SOAP (Simple Object Access Protocol) is a communication protocol for exchanging structured information over the Internet and uses a message format based on XML. It is both platform- and language-independent.
- WSDL (Web Services Description Language) protocol, again based on XML, is used to describe and locate a Web service.
- UDDI (Universal Discovery, Description, and Integration) protocol is a platform-independent, XML-based registry for businesses to list themselves on the Internet. It

was designed to be interrogated by SOAP messages and to provide access to WSDL documents describing the protocol bindings and message formats required to interact with the Web services listed in its directory.

Figure 3.9 illustrates the relationship between these technologies. From the database perspective, Web services can be used both from within the database (to invoke an external Web service as a *consumer)* and the Web service itself can access its own database (as a *provider)* to maintain the data required to provide the requested service.

3.8 What is a service-oriented architecture?

The SOA approach attempts to design loosely coupled and autonomous *services* that can be combined to provide flexible composite business processes and applications. SOA principles provide a unique design approach for building Web services for SOA:

• *loose coupling:* services must be designed to interact on a loosely coupled basis;

• *reusability:* logic that can potentially be reused is designed as a separate service;

• *contract:* services adhere to a communications contract that defines the information exchange and any additional service description information, specified by one or more service description documents;

• *abstraction:* beyond what is described in the service contract, services hide logic from the outside world;

• *composability:* services may compose other services, so that logic can be represented at different levels of granularity thereby promoting reusability and the creation of abstraction layers;

• *autonomy:* services have control over the logic they encapsulate and are not dependent upon other services to execute this governance;

• *stateless:* services should not be required to manage state information, as this can affect their ability to remain loosely-coupled;

• *discoverability:* services are designed to be outwardly descriptive so that they can be found and assessed via available discovery mechanisms.

3.9 Provide an example of a service-oriented architecture.

An example of SOA is shown in Figure 3.10(b).

3.10 Describe the main components in a DBMS.

The major software components in a DBMS environment are depicted in Figure 3.14.

The main components in a DBMS are:

- *Query processor.* This is a major DBMS component that transforms queries into a series of low-level instructions directed to the database manager.
- *Database manager (DM).* The DM interfaces with user-submitted application programs and queries. The DM accepts queries and examines the external and conceptual schemas to determine what conceptual records are required to satisfy the request. The DM then places a call to the file manager to perform the request.
- *File manager*. The file manager manipulates the underlying storage files and manages the allocation of storage space on disk. It establishes and maintains the list of structures and indexes defined in the internal schema. If hashed files are used, it

calls on the hashing functions to generate record addresses. However, the file manager does not directly manage the physical input and output of data. Rather, it passes the requests on to the appropriate access methods, which either read data from or write data into the system buffer (or *cache*).

- *DML preprocessor*. This module converts DML statements embedded in an application program into standard function calls in the host language. The DML preprocessor must interact with the query processor to generate the appropriate code.
- *DDL compiler*. The DDL compiler converts DDL statements into a set of tables containing metadata. These tables are then stored in the system catalog while control information is stored in data file headers.
- *Catalog manager*. The catalog manager manages access to and maintains the system catalog. The system catalog is accessed by most DBMS components.

3.11 Describe the internal architecture of Oracle.

Oracle is based on the client–server architecture. The Oracle server consists of the database (the raw data, including log and control files) and the instance (the processes and system memory on the server that provide access to the database). An instance can connect to only one database. The database consists of a logical structure, such as the database schema, and a physical structure, containing the files that make up an Oracle database.

Exercises

3.12 Examine the documentation sets of Microsoft SQL Server, Oracle, and IBM's DB2 system to identify their support for the following:
(a) client–server architecture
(b) Web services
(c) service-oriented architecture

Microsoft SQL Server

Microsoft SQL Server is designed to work effectively in a number of environments:

- As a two-tier or multitier client/server database system
- As a desktop database system

In a two-tier client/server system, users run an application on their local computer, known as a client, that connects over a network to the server running SQL Server. The client application runs both business logic and the code to display output to the user, and is also known as a thick client.

SQL Server 2008 enables data to be consumed from custom applications developed using Microsoft .NET and Visual Studio and from within service-oriented architectures (SOA) and business processes through Microsoft BizTalk Server.

Web services extend the World Wide Web infrastructure to provide the means for software to connect to other software applications. Applications access Web services via ubiquitous Web protocols and data formats such as HTTP, XML, and SOAP, with no need to worry about how each Web service is implemented. Web services combine the best aspects of component-based development and the Web, and are a cornerstone of the Microsoft .NET programming model.

Microsoft System Center plays a central role in Microsoft's vision to help IT organizations benefit from self-managing, dynamic systems. System Center solutions are tuned to simplify management of the systems and applications your company already has implemented, including Microsoft SQL Server, Microsoft Exchange Server, Microsoft Biztalk Server, Internet Information Services and the Microsoft .NET Framework.

Oracle

In the Oracle database system environment, the database application and the database are separated into two parts: a front-end or **client** portion, and a back-end or **server** portion--hence the term **client/server architecture.** The client runs the database application that accesses database information and interacts with a user through the keyboard, screen, and pointing device, such as a mouse. The server runs the Oracle software and handles the functions required for concurrent, shared data access to an Oracle database.

Although the client application and Oracle can be run on the same computer, greater efficiency can often be achieved when the client portions and server portion are run by different computers connected through a network. The following sections discuss possible variations in the Oracle client/server architecture.

Most organizations implement service-oriented architecture (SOA) with the hope of gaining more business agility through reuse of shared services. As reuse begins to take hold within the organization, however, it becomes critical to manage consumption of services or your SOA can quickly spin out of control. Oracle Web Services Manager provides a solution for governing the interactions with shared services through security and operational policy management and enforcement to ensure service reuse remains under control.

Leading companies are gaining operational efficiencies and business agility through adaptable, re-usable business processes and services built on a truly flexible Service-Oriented Architecture (SOA) foundation. Oracle SOA products allow you to build, deploy, and manage SOA with integrated, best-in-class technology that provides:

- Comprehensive and Pre-Integrated SOA Platform—Complete set of service and process infrastructure components for building, deploying, and managing SOAs
- Closed-Loop Governance—Comprehensive, end-to-end lifecycle governance of services
- Extreme performance and scalability—In-memory transactions, real-time event processing, and high-volume data transfer on top of a highly scalable application server
- Integrated Security—Centralized policy management, enterprise-grade, end-to-end security

Oracle SOA Suite, which now includes the former BEA AquaLogic Service Bus, is Oracle Fusion Middleware's strategic product for SOA. Oracle plans to continue to develop and support Oracle WebLogic Integration, and expects to converge this product with Oracle's strategic solutions over time. Existing deployments of this product will benefit from complementary products such as Oracle SOA Suite.

DB2

Local and remote application processes can work with the same database. A remote application is one that initiates a database action from a machine that is remote from the database machine. Local applications are directly attached to the database at the server machine.

Web service providers are described by Web Services Description Language (WSDL) documents. You can use the Web services wrapper to access Web service providers.

A Web services client application can obtain access to a DB2® Version 9 database with a Web services description language (WSDL) interface. You can create a WSDL interface to DB2® Version 9 data by using the Web services Object Runtime Framework (WORF), also known as Document Access Definition Extension (DADX) files. After you define the operations to access DB2 data with the DADX file, then you deploy the DADX file and its runtime environment (IBM® Web Service SOAP provider or Apache Axis version 1.2) to a supported Java™ Web application server environment (Apache Jakarta Tomcat or IBM® WebSphere® Application Server). After you have the DB2 Web service tested and deployed, any Web services client can start using the DB2 Web service.

Web services consumer - the user-defined functions

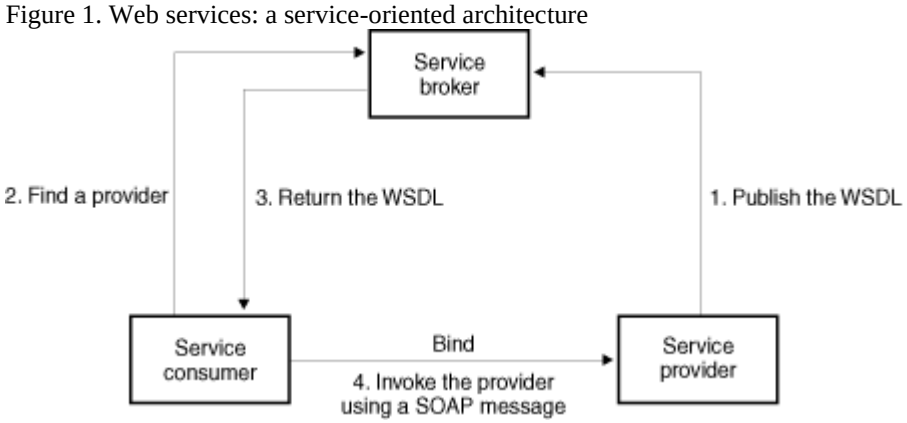
When DB2 Version 9 becomes the consumer, Web services can take advantage of the optimization that is built within the database. By using SQL statements, you can consume and integrate Web services data. By using SQL to access Web services data, you can reduce some application programming efforts because the data can be manipulated within the context of an SQL statement before that data is returned to the client application. You can convert an existing WSDL interface into a DB2 table or scalar function by using tools that are provided in WebSphere® Studio version 5 and later. During the execution of an SQL statement, you establish a connection with the Web service provider, and then you receive a response document as a relational table or a scalar value.

Web service providers are described by Web Services Description Language (WSDL) documents. You can use the Web services wrapper to access Web service providers.

The Figure 1 diagram shows the architecture of Web services.

1. A Web service provider implements a service and publishes the WSDL information to a service broker, such as UDDI.
2. The service consumer can then use the service broker to find a Web service provider.
3. When the service consumer finds a Web service provider, the service consumer binds to the service provider so that the consumer can use the Web service.
4. The consumer invokes the service by exchanging SOAP (simple object access protocol) messages between the requester and provider.

Figure 1. Web services: a service-oriented architecture



The SOAP specification defines the layout of an XML-based message. A SOAP message is contained in a SOAP envelope. The envelope consists of an optional SOAP header and a mandatory SOAP body. The SOAP header can contain information about the message, such as encryption information or authentication information. The SOAP body contains the message. The SOAP specification also defines a default encoding for programming language bindings, which is called the SOAP encoding.

3.13 Search the Web for a number of Web services other than the ones discussed in Section 3.2. What do these services have in common? Identify whether the services access a database.

The Google Maps API provides Web services to make use of the Google Maps application. The following Web site contains a number of examples: http://maps.forum.nu/ The following site contains information on using a database with Google maps: http://groups.google.com/group/Google-Maps-API/web/using-databases-with-gmaps-apps

Amazon provides a number of Web services discussed on the following Web site: http://aws.amazon.com/