24.5 Introduction to Statistical Database Security

Statistical databases are used mainly to produce statistics about various populations. The database may contain confidential data about individuals, which should be protected from user access. However, users are permitted to retrieve statistical information about the populations, such as averages, sums, counts, maximums, minimums, and standard deviations. The techniques that have been developed to protect the privacy of individual information are beyond the scope of this book. We will illustrate the problem with a very simple example, which refers to the relation shown in Figure 24.3. This is a PERSON relation with the attributes Name, Ssn, Income, Address, City, State, Zip, Sex, and Last_degree.

A **population** is a set of tuples of a relation (table) that satisfy some selection condition. Hence, each selection condition on the PERSON relation will specify a particular population of PERSON tuples. For example, the condition Sex = 'M' specifies the male population; the condition ((Sex = 'F') AND (Last_degree = 'M.S.' OR Last_degree = 'Ph.D.')) specifies the female population that has an M.S. or Ph.D. degree as their highest degree; and the condition City = 'Houston' specifies the population that lives in Houston.

Statistical queries involve applying statistical functions to a population of tuples. For example, we may want to retrieve the number of individuals in a population or the average income in the population. However, statistical users are not allowed to retrieve individual data, such as the income of a specific person. **Statistical database security** techniques must prohibit the retrieval of individual data. This can be achieved by prohibiting queries that retrieve attribute values and by allowing only queries that involve statistical aggregate functions such as COUNT, SUM, MIN, MAX, AVERAGE, and STANDARD DEVIATION. Such queries are sometimes called **statistical queries**.

It is the responsibility of a database management system to ensure the confidentiality of information about individuals, while still providing useful statistical summaries of data about those individuals to users. Provision of **privacy protection** of users in a statistical database is paramount; its violation is illustrated in the following example.

In some cases it is possible to **infer** the values of individual tuples from a sequence of statistical queries. This is particularly true when the conditions result in a

PERSON

Name	Ssn	Income	Address	City	State	Zip	Sex	Last_degree

Figure 24.3
The PERSON relation schema for illustrating statistical database security.

population consisting of a small number of tuples. As an illustration, consider the following statistical queries:

Q1: **SELECT COUNT** (*) **FROM** PERSON
WHERE <condition>;

Q2: **SELECT AVG** (Income) **FROM** PERSON
WHERE <condition>;

Now suppose that we are interested in finding the Salary of Jane Smith, and we know that she has a Ph.D. degree and that she lives in the city of Bellaire, Texas. We issue the statistical query Q1 with the following condition:

(Last_degree='Ph.D.' AND Sex='F' AND City='Bellaire' AND State='Texas')

If we get a result of 1 for this query, we can issue Q2 with the same condition and find the Salary of Jane Smith. Even if the result of Q1 on the preceding condition is not 1 but is a small number—say 2 or 3—we can issue statistical queries using the functions MAX, MIN, and AVERAGE to identify the possible range of values for the Salary of Jane Smith.

The possibility of inferring individual information from statistical queries is reduced if no statistical queries are permitted whenever the number of tuples in the population specified by the selection condition falls below some threshold. Another technique for prohibiting retrieval of individual information is to prohibit sequences of queries that refer repeatedly to the same population of tuples. It is also possible to introduce slight inaccuracies or *noise* into the results of statistical queries deliberately, to make it difficult to deduce individual information from the results. Another technique is partitioning of the database. Partitioning implies that records are stored in groups of some minimum size; queries can refer to any complete group or set of groups, but never to subsets of records within a group. The interested reader is referred to the bibliography at the end of this chapter for a discussion of these techniques.

24.6 Introduction to Flow Control

Flow control regulates the distribution or flow of information among accessible objects. A flow between object X and object Y occurs when a program reads values from X and writes values into Y. **Flow controls** check that information contained in some objects does not flow explicitly or implicitly into less protected objects. Thus, a user cannot get indirectly in Y what he or she cannot get directly in X. Active flow control began in the early 1970s. Most flow controls employ some concept of security class; the transfer of information from a sender to a receiver is allowed only if the receiver's security class is at least as privileged as the sender's. Examples of a flow control include preventing a service program from leaking a customer's confidential data, and blocking the transmission of secret military data to an unknown classified user.

A **flow policy** specifies the channels along which information is allowed to move. The simplest flow policy specifies just two classes of information—confidential (C)

and nonconfidential (*N*)—and allows all flows except those from class *C* to class *N*. This policy can solve the confinement problem that arises when a service program handles data such as customer information, some of which may be confidential. For example, an income-tax computing service might be allowed to retain a customer's address and the bill for services rendered, but not a customer's income or deductions.

Access control mechanisms are responsible for checking users' authorizations for resource access: Only granted operations are executed. Flow controls can be enforced by an extended access control mechanism, which involves assigning a security class (usually called the *clearance*) to each running program. The program is allowed to read a particular memory segment only if its security class is as high as that of the segment. It is allowed to write in a segment only if its class is as low as that of the segment. This automatically ensures that no information transmitted by the person can move from a higher to a lower class. For example, a military program with a secret clearance can only read from objects that are unclassified and confidential and can only write into objects that are secret or top secret.

Two types of flow can be distinguished: *explicit flows,* occurring as a consequence of assignment instructions, such as $Y{:=}\, f(X_1, X_n,)$, and *implicit flows* generated by conditional instructions, such as if $f(X_{m+1,\ldots,} X_n)$ then $Y{:=}\, f\,(X_1, X_m)$.

Flow control mechanisms must verify that only authorized flows, both explicit and implicit, are executed. A set of rules must be satisfied to ensure secure information flows. Rules can be expressed using flow relations among classes and assigned to information, stating the authorized flows within a system. (An information flow from *A* to *B* occurs when information associated with *A* affects the value of information associated with *B*. The flow results from operations that cause information transfer from one object to another.) These relations can define, for a class, the set of classes where information (classified in that class) can flow, or can state the specific relations to be verified between two classes to allow information to flow from one to the other. In general, flow control mechanisms implement the controls by assigning a label to each object and by specifying the security class of the object. Labels are then used to verify the flow relations defined in the model.

24.6.1 Covert Channels

A covert channel allows a transfer of information that violates the security or the policy. Specifically, a **covert channel** allows information to pass from a higher classification level to a lower classification level through improper means. Covert channels can be classified into two broad categories: timing channels and storage. The distinguishing feature between the two is that in a **timing channel** the information is conveyed by the timing of events or processes, whereas **storage channels** do not require any temporal synchronization, in that information is conveyed by accessing system information or what is otherwise inaccessible to the user.

In a simple example of a covert channel, consider a distributed database system in which two nodes have user security levels of secret (S) and unclassified (U). In order

for a transaction to commit, both nodes must agree to commit. They mutually can only do operations that are consistent with the *-property, which states that in any transaction, the *S* site cannot write or pass information to the *U* site. However, if these two sites collude to set up a covert channel between them, a transaction involving secret data may be committed unconditionally by the *U* site, but the *S* site may do so in some predefined agreed-upon way so that certain information may be passed from the *S* site to the *U* site, violating the *-property. This may be achieved where the transaction runs repeatedly, but the actions taken by the *S* site implicitly convey information to the *U* site. Measures such as locking, which we discussed in Chapters 22 and 23, prevent concurrent writing of the information by users with different security levels into the same objects, preventing the storage-type covert channels. Operating systems and distributed databases provide control over the multiprogramming of operations that allows a sharing of resources without the possibility of encroachment of one program or process into another's memory or other resources in the system, thus preventing timing-oriented covert channels. In general, covert channels are not a major problem in well-implemented robust database implementations. However, certain schemes may be contrived by clever users that implicitly transfer information.

Some security experts believe that one way to avoid covert channels is to disallow programmers to actually gain access to sensitive data that a program will process after the program has been put into operation. For example, a programmer for a bank has no need to access the names or balances in depositors' accounts. Programmers for brokerage firms do not need to know what buy and sell orders exist for clients. During program testing, access to a form of real data or some sample test data may be justifiable, but not after the program has been accepted for regular use.

24.7 Encryption and Public Key Infrastructures

The previous methods of access and flow control, despite being strong control measures, may not be able to protect databases from some threats. Suppose we communicate data, but our data falls into the hands of a nonlegitimate user. In this situation, by using encryption we can disguise the message so that even if the transmission is diverted, the message will not be revealed. **Encryption** is the conversion of data into a form, called a **ciphertext**, which cannot be easily understood by unauthorized persons. It enhances security and privacy when access controls are bypassed, because in cases of data loss or theft, encrypted data cannot be easily understood by unauthorized persons.

With this background, we adhere to following standard definitions:[6]

- *Ciphertext*: Encrypted (enciphered) data.

[6]These definitions are from NIST (National Institute of Standards and Technology) from http://csrc.nist.gov/publications/nistpubs/800-67/SP800-67.pdf.

- *Plaintext (or cleartext)*: Intelligible data that has meaning and can be read or acted upon without the application of decryption.
- *Encryption*: The process of transforming plaintext into ciphertext.
- *Decryption*: The process of transforming ciphertext back into plaintext.

Encryption consists of applying an **encryption algorithm** to data using some prespecified **encryption key**. The resulting data has to be **decrypted** using a **decryption key** to recover the original data.

24.7.1 The Data Encryption and Advanced Encryption Standards

The **Data Encryption Standard** (DES) is a system developed by the U.S. government for use by the general public. It has been widely accepted as a cryptographic standard both in the United States and abroad. DES can provide end-to-end encryption on the channel between sender *A* and receiver *B*. The DES algorithm is a careful and complex combination of two of the fundamental building blocks of encryption: substitution and permutation (transposition). The algorithm derives its strength from repeated application of these two techniques for a total of 16 cycles. Plaintext (the original form of the message) is encrypted as blocks of 64 bits. Although the key is 64 bits long, in effect the key can be any 56-bit number. After questioning the adequacy of DES, the NIST introduced the **Advanced Encryption Standard** (AES). This algorithm has a block size of 128 bits, compared with DES's 56-block size, and can use keys of 128, 192, or 256 bits, compared with DES's 56-bit key. AES introduces more possible keys, compared with DES, and thus takes a much longer time to crack.

24.7.2 Symmetric Key Algorithms

A symmetric key is one key that is used for both encryption and decryption. By using a symmetric key, fast encryption and decryption is possible for routine use with sensitive data in the database. A message encrypted with a secret key can be decrypted only with the same secret key. Algorithms used for symmetric key encryption are called **secret-key algorithms**. Since secret-key algorithms are mostly used for encrypting the content of a message, they are also called **content-encryption algorithms**.

The major liability associated with secret-key algorithms is the need for sharing the secret key. A possible method is to derive the secret key from a user-supplied password string by applying the same function to the string at both the sender and receiver; this is known as a *password-based encryption algorithm.* The strength of the symmetric key encryption depends on the size of the key used. For the same algorithm, encrypting using a longer key is tougher to break than the one using a shorter key.

24.7.3 Public (Asymmetric) Key Encryption

In 1976, Diffie and Hellman proposed a new kind of cryptosystem, which they called **public key encryption**. Public key algorithms are based on mathematical

functions rather than operations on bit patterns. They address one drawback of symmetric key encryption, namely that both sender and recipient must exchange the common key in a secure manner. In public key systems, two keys are used for encryption/decryption. The *public key* can be transmitted in a non-secure way, whereas the *private key* is not transmitted at all. These algorithms—which use two related keys, a public key and a private key, to perform complementary operations (encryption and decryption)—are known as **asymmetric key encryption algorithms**. The use of two keys can have profound consequences in the areas of confidentiality, key distribution, and authentication. The two keys used for public key encryption are referred to as the **public key** and the **private key**. The private key is kept secret, but it is referred to as a *private key* rather than a *secret key* (the key used in conventional encryption) to avoid confusion with conventional encryption. The two keys are mathematically related, since one of the keys is used to perform encryption and the other to perform decryption. However, it is very difficult to derive the private key from the public key.

A public key encryption scheme, or *infrastructure*, has six ingredients:

1. **Plaintext.** This is the data or readable message that is fed into the algorithm as input.
2. **Encryption algorithm.** This algorithm performs various transformations on the plaintext.
3. and 4. **Public and private keys.** These are a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the encryption algorithm depend on the public or private key that is provided as input. For example, if a message is encrypted using the public key, it can only be decrypted using the private key.
5. **Ciphertext.** This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts.
6. **Decryption algorithm.** This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

As the name suggests, the public key of the pair is made public for others to use, whereas the private key is known only to its owner. A general-purpose public key cryptographic algorithm relies on one key for encryption and a different but related key for decryption. The essential steps are as follows:

1. Each user generates a pair of keys to be used for the encryption and decryption of messages.
2. Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private.
3. If a sender wishes to send a private message to a receiver, the sender encrypts the message using the receiver's public key.

4. When the receiver receives the message, he or she decrypts it using the receiver's private key. No other recipient can decrypt the message because only the receiver knows his or her private key.

The RSA Public Key Encryption Algorithm. One of the first public key schemes was introduced in 1978 by Ron Rivest, Adi Shamir, and Len Adleman at MIT and is named after them as the **RSA scheme**. The RSA scheme has since then reigned supreme as the most widely accepted and implemented approach to public key encryption. The RSA encryption algorithm incorporates results from number theory, combined with the difficulty of determining the prime factors of a target. The RSA algorithm also operates with modular arithmetic—mod n.

Two keys, d and e, are used for decryption and encryption. An important property is that they can be interchanged. n is chosen as a large integer that is a product of two large distinct prime numbers, a and b, $n = a \times$ b. The encryption key e is a randomly chosen number between 1 and n that is relatively prime to $(a - 1) \times (b - 1)$. The plaintext block P is encrypted as P^e where $P^e = P \bmod n$. Because the exponentiation is performed mod n, factoring P^e to uncover the encrypted plaintext is difficult. However, the decrypting key d is carefully chosen so that $(P^e)^d \bmod n = P$. The decryption key d can be computed from the condition that $d \times e = 1 \bmod ((a - 1) \times (b - 1))$. Thus, the legitimate receiver who knows d simply computes $(P^e)^d \bmod n = P$ and recovers P without having to factor P^e.

24.7.4 Digital Signatures

A digital signature is an example of using encryption techniques to provide authentication services in electronic commerce applications. Like a handwritten signature, a **digital signature** is a means of associating a mark unique to an individual with a body of text. The mark should be unforgettable, meaning that others should be able to check that the signature comes from the originator.

A digital signature consists of a string of symbols. If a person's digital signature were always the same for each message, then one could easily counterfeit it by simply copying the string of symbols. Thus, signatures must be different for each use. This can be achieved by making each digital signature a function of the message that it is signing, together with a timestamp. To be unique to each signer and counterfeitproof, each digital signature must also depend on some secret number that is unique to the signer. Thus, in general, a counterfeitproof digital signature must depend on the message and a unique secret number of the signer. The verifier of the signature, however, should not need to know any secret number. Public key techniques are the best means of creating digital signatures with these properties.

24.7.5 Digital Certificates

A digital certificate is used to combine the value of a public key with the identity of the person or service that holds the corresponding private key into a digitally signed

statement. Certificates are issued and signed by a certification authority (CA). The entity receiving this certificate from a CA is the subject of that certificate. Instead of requiring each participant in an application to authenticate every user, third-party authentication relies on the use of digital certificates.

The digital certificate itself contains various types of information. For example, both the certification authority and the certificate owner information are included. The following list describes all the information included in the certificate:

1. The certificate owner information, which is represented by a unique identifier known as the distinguished name (DN) of the owner. This includes the owner's name, as well as the owner's organization and other information about the owner.
2. The certificate also includes the public key of the owner.
3. The date of issue of the certificate is also included.
4. The validity period is specified by 'Valid From' and 'Valid To' dates, which are included in each certificate.
5. Issuer identifier information is included in the certificate.
6. Finally, the digital signature of the issuing CA for the certificate is included. All the information listed is encoded through a message-digest function, which creates the digital signature. The digital signature basically certifies that the association between the certificate owner and public key is valid.

24.8 Privacy Issues and Preservation

Preserving data privacy is a growing challenge for database security and privacy experts. In some perspectives, to preserve data privacy we should even limit performing large-scale data mining and analysis. The most commonly used techniques to address this concern are to avoid building mammoth central warehouses as a single repository of vital information. Another possible measure is to intentionally modify or perturb data.

If all data were available at a single warehouse, violating only a single repository's security could expose all data. Avoiding central warehouses and using distributed data mining algorithms minimizes the exchange of data needed to develop globally valid models. By modifying, perturbing, and anonymizing data, we can also mitigate privacy risks associated with data mining. This can be done by removing identity information from the released data and injecting noise into the data. However, by using these techniques, we should pay attention to the quality of the resulting data in the database, which may undergo too many modifications. We must be able to estimate the errors that may be introduced by these modifications.

Privacy is an important area of ongoing research in database management. It is complicated due to its multidisciplinary nature and the issues related to the subjectivity in the interpretation of privacy, trust, and so on. As an example, consider medical and legal records and transactions, which must maintain certain privacy

requirements while they are being defined and enforced. Providing access control and privacy for mobile devices is also receiving increased attention. DBMSs need robust techniques for efficient storage of security-relevant information on small devices, as well as trust negotiation techniques. Where to keep information related to user identities, profiles, credentials, and permissions and how to use it for reliable user identification remains an important problem. Because large-sized streams of data are generated in such environments, efficient techniques for access control must be devised and integrated with processing techniques for continuous queries. Finally, the privacy of user location data, acquired from sensors and communication networks, must be ensured.

24.9 Challenges of Database Security

Considering the vast growth in volume and speed of threats to databases and information assets, research efforts need to be devoted to the following issues: data quality, intellectual property rights, and database survivability. These are only some of the main challenges that researchers in database security are trying to address.

24.9.1 Data Quality

The database community needs techniques and organizational solutions to assess and attest the quality of data. These techniques may include simple mechanisms such as quality stamps that are posted on Web sites. We also need techniques that provide more effective integrity semantics verification and tools for the assessment of data quality, based on techniques such as record linkage. Application-level recovery techniques are also needed for automatically repairing incorrect data. The ETL (extract, transform, load) tools widely used to load data in data warehouses (see Section 29.4) are presently grappling with these issues.

24.9.2 Intellectual Property Rights

With the widespread use of the Internet and intranets, legal and informational aspects of data are becoming major concerns of organizations. To address these concerns, watermarking techniques for relational data have been proposed. The main purpose of digital watermarking is to protect content from unauthorized duplication and distribution by enabling provable ownership of the content. It has traditionally relied upon the availability of a large noise domain within which the object can be altered while retaining its essential properties. However, research is needed to assess the robustness of such techniques and to investigate different approaches aimed at preventing intellectual property rights violations.

24.9.3 Database Survivability

Database systems need to operate and continue their functions, even with reduced capabilities, despite disruptive events such as information warfare attacks. A DBMS,

in addition to making every effort to prevent an attack and detecting one in the event of occurrence, should be able to do the following:

- **Confinement.** Take immediate action to eliminate the attacker's access to the system and to isolate or contain the problem to prevent further spread.
- **Damage assessment.** Determine the extent of the problem, including failed functions and corrupted data.
- **Reconfiguration.** Reconfigure to allow operation to continue in a degraded mode while recovery proceeds.
- **Repair.** Recover corrupted or lost data and repair or reinstall failed system functions to reestablish a normal level of operation.
- **Fault treatment.** To the extent possible, identify the weaknesses exploited in the attack and take steps to prevent a recurrence.

The goal of the information warfare attacker is to damage the organization's operation and fulfillment of its mission through disruption of its information systems. The specific target of an attack may be the system itself or its data. While attacks that bring the system down outright are severe and dramatic, they must also be well timed to achieve the attacker's goal, since attacks will receive immediate and concentrated attention in order to bring the system back to operational condition, diagnose how the attack took place, and install preventive measures.

To date, issues related to database survivability have not been sufficiently investigated. Much more research needs to be devoted to techniques and methodologies that ensure database system survivability.

24.10 Oracle Label-Based Security

Restricting access to entire tables or isolating sensitive data into separate databases is a costly operation to administer. **Oracle Label Security** overcomes the need for such measures by enabling row-level access control. It is available in Oracle Database 11g Release 1 (11.1) Enterprise Edition at the time of writing. Each database table or view has a security policy associated with it. This policy executes every time the table or view is queried or altered. Developers can readily add label-based access control to their Oracle Database applications. Label-based security provides an adaptable way of controlling access to sensitive data. Both users and data have labels associated with them. Oracle Label Security uses these labels to provide security.

24.10.1 Virtual Private Database (VPD) Technology

Virtual Private Databases (VPDs) is a feature of the Oracle Enterprise Edition that adds predicates to user statements to limit their access in a transparent manner to the user and the application. The VPD concept allows server-enforced, fine-grained access control for a secure application.

VPD provides access control based on policies. These VPD policies enforce object-level access control or row-level security. It provides an application programming

interface (API) that allows security policies to be attached to database tables or views. Using PL/SQL, a host programming language used in Oracle applications, developers and security administrators can implement security policies with the help of stored procedures.[7] VPD policies allow developers to remove access security mechanisms from applications and centralize them within the Oracle Database.

VPD is enabled by associating a security "policy" with a table, view, or synonym. An administrator uses the supplied PL/SQL package, DBMS_RLS, to bind a policy function with a database object. When an object having a security policy associated with it is accessed, the function implementing this policy is consulted. The policy function returns a predicate (a WHERE clause) which is then appended to the user's SQL statement, thus *transparently* and *dynamically* modifying the user's data access. Oracle Label Security is a technique of enforcing row-level security in the form of a security policy.

24.10.2 Label Security Architecture

Oracle Label Security is built on the VPD technology delivered in the Oracle Database 11.1 Enterprise Edition. Figure 24.4 illustrates how data is accessed under Oracle Label Security, showing the sequence of DAC and label security checks.

Figure 24.4 shows the sequence of discretionary access control (DAC) and label security checks. The left part of the figure shows an application user in an Oracle Database 11g Release 1 (11.1) session sending out an SQL request. The Oracle DBMS checks the DAC privileges of the user, making sure that he or she has SELECT privileges on the table. Then it checks whether the table has a Virtual Private Database (VPD) policy associated with it to determine if the table is protected using Oracle Label Security. If it is, the VPD SQL modification (WHERE clause) is added to the original SQL statement to find the set of accessible rows for the user to view. Then Oracle Label Security checks the labels on each row, to determine the subset of rows to which the user has access (as explained in the next section). This modified query gets processed, optimized, and executed.

24.10.3 How Data Labels and User Labels Work Together

A user's label indicates the information the user is permitted to access. It also determines the type of access (read or write) that the user has on that information. A row's label shows the sensitivity of the information that the row contains as well as the ownership of the information. When a table in the database has a label-based access associated with it, a row can be accessed only if the user's label meet certain criteria defined in the policy definitions. Access is granted or denied based on the result of comparing the data label and the session label of the user.

Compartments allow a finer classification of sensitivity of the labeled data. All data related to the same project can be labeled with the same compartment. Compartments are optional; a label can contain zero or more compartments.

[7] Stored procedures are discussed in Section 5.2.2.

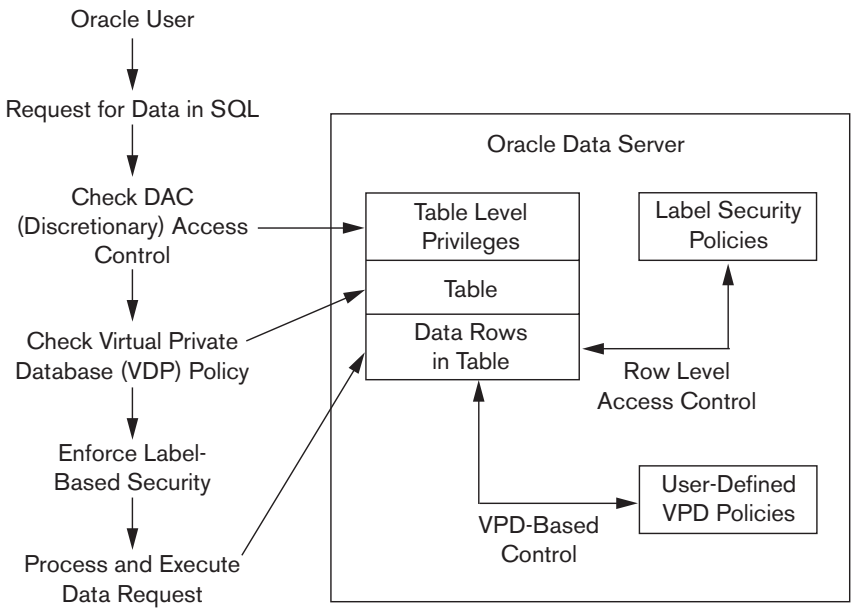

Figure 24.4
Oracle Label Security architecture.
Source: Oracle (2007)

Groups are used to identify organizations as owners of the data with corresponding group labels. Groups are hierarchical; for example, a group can be associated with a parent group.

If a user has a maximum level of SENSITIVE, then the user potentially has access to all data having levels SENSITIVE, CONFIDENTIAL, and UNCLASSIFIED. This user has no access to HIGHLY_SENSITIVE data. Figure 24.5 shows how data labels and user labels work together to provide access control in Oracle Label Security.

As shown in Figure 24.5, User 1 can access the rows 2, 3, and 4 because his maximum level is HS (Highly_Sensitive). He has access to the FIN (Finance) compartment, and his access to group WR (Western Region) hierarchically includes group WR_SAL (WR Sales). He cannot access row 1 because he does not have the CHEM (Chemical) compartment. It is important that a user has authorization for all compartments in a row's data label to be able to access that row. Based on this example, user 2 can access both rows 3 and 4, and has a maximum level of S, which is less than the HS in row 2. So, although user 2 has access to the FIN compartment, he can only access the group WR_SAL, and thus cannot acces row 1.

24.11 Summary

In this chapter we discussed several techniques for enforcing database system security. We presented different threats to databases in terms of loss of integrity, availability, and confidentiality. We discussed the types of control measures to deal with these problems: access control, inference control, flow control, and encryption. In

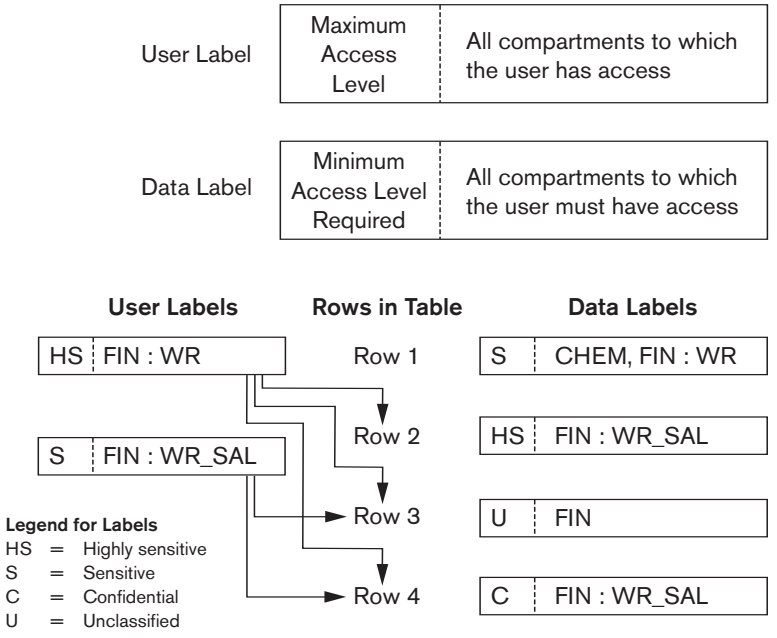

Figure 24.5
Data labels and user labels in Oracle.
Source: Oracle (2007)

the introduction we covered various issues related to security including data sensitivity and type of disclosures, providing security vs. precision in the result when a user requests information, and the relationship between information security and privacy.

Security enforcement deals with controlling access to the database system as a whole and controlling authorization to access specific portions of a database. The former is usually done by assigning accounts with passwords to users. The latter can be accomplished by using a system of granting and revoking privileges to individual accounts for accessing specific parts of the database. This approach is generally referred to as discretionary access control (DAC). We presented some SQL commands for granting and revoking privileges, and we illustrated their use with examples. Then we gave an overview of mandatory access control (MAC) mechanisms that enforce multilevel security. These require the classifications of users and data values into security classes and enforce the rules that prohibit flow of information from higher to lower security levels. Some of the key concepts underlying the multilevel relational model, including filtering and polyinstantiation, were presented. Role-based access control (RBAC) was introduced, which assigns privileges based on roles that users play. We introduced the notion of role hierarchies, mutual exclusion of roles, and row- and label-based security. We briefly discussed the problem of controlling access to statistical databases to protect the privacy of individual information while concurrently providing statistical access to populations of records. We explained the main ideas behind the threat of SQL Injection, the methods in which it can be induced, and the various types of risks associated with it. Then we gave an

idea of the various ways SQL injection can be prevented. The issues related to flow control and the problems associated with covert channels were discussed next, as well as encryption and public-private key-based infrastructures. The idea of symmetric key algorithms and the use of the popular asymmetric key-based public key infrastructure (PKI) scheme was explained. We also covered the concepts of digital signatures and digital certificates. We highlighted the importance of privacy issues and hinted at some privacy preservation techniques. We discussed a variety of challenges to security including data quality, intellectual property rights, and data survivability. We ended the chapter by introducing the implementation of security policies by using a combination of label-based security and virtual private databases in Oracle 11g.

Review Questions

24.1. Discuss what is meant by each of the following terms: *database authorization, access control, data encryption, privileged (system) account, database audit, audit trail.*

24.2. Which account is designated as the owner of a relation? What privileges does the owner of a relation have?

24.3. How is the view mechanism used as an authorization mechanism?

24.4. Discuss the types of privileges at the account level and those at the relation level.

24.5. What is meant by granting a privilege? What is meant by revoking a privilege?

24.6. Discuss the system of propagation of privileges and the restraints imposed by horizontal and vertical propagation limits.

24.7. List the types of privileges available in SQL.

24.8. What is the difference between *discretionary* and *mandatory* access control?

24.9. What are the typical security classifications? Discuss the simple security property and the *-property, and explain the justification behind these rules for enforcing multilevel security.

24.10. Describe the multilevel relational data model. Define the following terms: *apparent key, polyinstantiation, filtering.*

24.11. What are the relative merits of using DAC or MAC?

24.12. What is role-based access control? In what ways is it superior to DAC and MAC?

24.13. What are the two types of mutual exclusion in role-based access control?

24.14. What is meant by row-level access control?

24.15. What is label security? How does an administrator enforce it?

24.16. What are the different types of SQL injection attacks?

24.17. What risks are associated with SQL injection attacks?

24.18. What preventive measures are possible against SQL injection attacks?

24.19. What is a statistical database? Discuss the problem of statistical database security.

24.20. How is privacy related to statistical database security? What measures can be taken to ensure some degree of privacy in statistical databases?

24.21. What is flow control as a security measure? What types of flow control exist?

24.22. What are covert channels? Give an example of a covert channel.

24.23. What is the goal of encryption? What process is involved in encrypting data and then recovering it at the other end?

24.24. Give an example of an encryption algorithm and explain how it works.

24.25. Repeat the previous question for the popular RSA algorithm.

24.26. What is a symmetric key algorithm for key-based security?

24.27. What is the public key infrastructure scheme? How does it provide security?

24.28. What are digital signatures? How do they work?

24.29. What type of information does a digital certificate include?

Exercises

24.30. How can privacy of data be preserved in a database?

24.31. What are some of the current outstanding challenges for database security?

24.32. Consider the relational database schema in Figure 3.5. Suppose that all the relations were created by (and hence are owned by) user *X*, who wants to grant the following privileges to user accounts *A*, *B*, *C*, *D*, and *E*:

a. Account *A* can retrieve or modify any relation except DEPENDENT and can grant any of these privileges to other users.

b. Account *B* can retrieve all the attributes of EMPLOYEE and DEPARTMENT except for Salary, Mgr_ssn, and Mgr_start_date.

c. Account *C* can retrieve or modify WORKS_ON but can only retrieve the Fname, Minit, Lname, and Ssn attributes of EMPLOYEE and the Pname and Pnumber attributes of PROJECT.

d. Account *D* can retrieve any attribute of EMPLOYEE or DEPENDENT and can modify DEPENDENT.

e. Account *E* can retrieve any attribute of EMPLOYEE but only for EMPLOYEE tuples that have Dno = 3.

f. Write SQL statements to grant these privileges. Use views where appropriate.

24.33. Suppose that privilege (a) of Exercise 24.32 is to be given with GRANT OPTION but only so that account *A* can grant it to at most five accounts, and each of these accounts can propagate the privilege to other accounts but *without* the GRANT OPTION privilege. What would the horizontal and vertical propagation limits be in this case?

24.34. Consider the relation shown in Figure 24.2(d). How would it appear to a user with classification *U*? Suppose that a classification *U* user tries to update the salary of 'Smith' to $50,000; what would be the result of this action?

Selected Bibliography

Authorization based on granting and revoking privileges was proposed for the SYSTEM R experimental DBMS and is presented in Griffiths and Wade (1976). Several books discuss security in databases and computer systems in general, including the books by Leiss (1982a) and Fernandez et al. (1981), and Fugini et al. (1995). Natan (2005) is a practical book on security and auditing implementation issues in all major RDBMSs.

Many papers discuss different techniques for the design and protection of statistical databases. They include McLeish (1989), Chin and Ozsoyoglu (1981), Leiss (1982), Wong (1984), and Denning (1980). Ghosh (1984) discusses the use of statistical databases for quality control. There are also many papers discussing cryptography and data encryption, including Diffie and Hellman (1979), Rivest et al. (1978), Akl (1983), Pfleeger and Pfleeger (2007), Omura et al. (1990), Stallings (2000), and Iyer at al. (2004).

Halfond et al. (2006) helps understand the concepts of SQL injection attacks and the various threats imposed by them. The white paper Oracle (2007a) explains how Oracle is less prone to SQL injection attack as compared to SQL Server. It also gives a brief explanation as to how these attacks can be prevented from occurring. Further proposed frameworks are discussed in Boyd and Keromytis (2004), Halfond and Orso (2005), and McClure and Krüger (2005).

Multilevel security is discussed in Jajodia and Sandhu (1991), Denning et al. (1987), Smith and Winslett (1992), Stachour and Thuraisingham (1990), Lunt et al. (1990), and Bertino et al. (2001). Overviews of research issues in database security are given by Lunt and Fernandez (1990), Jajodia and Sandhu (1991), Bertino (1998), Castano et al. (1995), and Thuraisingham et al. (2001). The effects of multilevel security on concurrency control are discussed in Atluri et al. (1997). Security in next-generation, semantic, and object-oriented databases is discussed in Rabbiti et al. (1991), Jajodia and Kogan (1990), and Smith (1990). Oh (1999) presents a model for both discretionary and mandatory security. Security models for Web-based applications and role-based access control are discussed in Joshi et al. (2001). Security issues for managers in the context of e-commerce applications and the need for risk assessment models for selection of appropriate security control measures are discussed in

Farahmand et al. (2005). Row-level access control is explained in detail in Oracle (2007b) and Sybase (2005). The latter also provides details on role hierarchy and mutual exclusion. Oracle (2009) explains how Oracle uses the concept of identity management.

Recent advances as well as future challenges for security and privacy of databases are discussed in Bertino and Sandhu (2005). U.S. Govt. (1978), OECD (1980), and NRC (2003) are good references on the view of privacy by important government bodies. Karat et al. (2009) discusses a policy framework for security and privacy. XML and access control are discussed in Naedele (2003). More details can be found on privacy preserving techniques in Vaidya and Clifton (2004), intellectual property rights in Sion et al. (2004), and database survivability in Jajodia et al. (1999). Oracle's VPD technology and label-based security is discussed in more detail in Oracle (2007b).