Database security

Advantages of using databases

- Shared access
  - Many users use one common, centralized data set
- Minimal redundancy
  - Individual users do not have to collect and maintain their own sets of data
- Data consistency
  - A change to a data value affects all users of the data value
- Data integrity
  - Data values are protected against accidental or malicious undesirable changes
- Controlled access
  - Only authorized users are allowed to view or to modify data values

Security requirements (1/2)

- Physical integrity
  - Immunity to physical problems
    - e.g. power failures
  - Ability to reconstruct the database if destroyed in a catastrophe
- Logical integrity
  - Structure is preserved
- Element integrity
  - Data in each element is accurate
- Auditability
  - It is possible to track who or what has accessed (or modified) which elements in the database

Security requirements (2/2)

- Access control
  - A user/role is allowed to access only authorized data/queries
  - Different users/roles can be restricted to different modes
    - e.g. read or write
- User authentication
  - Every user/role is positively identified
  - Fundamental for audit trails and for permissions to access data
- Availability
  - Users/roles can access the database in general and all the data for which they are authorized

Two-phase updates

- Problem
  - Failures during updates may render databases incoherent
    - Requirement → atomicity (from ACID)
  - Logical integrity problem

- Solution: Atomicity with two-phase updates

Two-phase update

- 1st phase: intent phase
  - The DBMS gathers resources it needs to perform the update
  - It does everything to prepare for the update, but makes no changes to the database
  - Committing: writes a commit flag to the database
    - Point of no return
    - After, the DBMS begins making permanent changes
- 2nd phase: commit phase
  - Makes the permanent changes in the database
  - It lasts until finishing all changes prepared in the first phase
  - When it finishes, the database changed to a new stable, coherent state

Redundancy / internal consistency

- Error detection and correction codes
  - Parity bits, Hamming codes, cyclic redundancy checks
  - Can be applied to different data elements
    - Fields, records, entire database
  - More space
    - To store error detection/correction information
- Shadow fields
  - Duplication of fields or records
  - Requires substantial storage space

Concurrency / consistency

- Accesses by two users of the same DBMS must be constrained so that neither interferes with each other
  - Simple locking: multiple readers, one writer
  - But simple locking may not be enough on query-update cycles
- Solution
  - Treat every query-update cycle as a single atomic operation (a transaction)
    - e.g. flight booking
  - Synchronization should be applied to transactions
    - Two concurrent transactions cannot write (and sometimes read) the same field/record

Monitors

- DBMS unit responsible for the DB structural integrity
  - Checks entered values to ensure their consistency with field, record or database consistency constraints
- Types of monitors
  - Range comparisons
    - Tests is values belong to an acceptable range
  - State constraints
    - Describe the condition of the entire database
      - e.g. the commit flag
    - Impose integrity restriction rules
      - e.g. to detect duplicate records
  - Transition constraints
    - Describe required conditions before changing the database

Database activity monitoring

- DBMS usage supervision
  - To detect abuses
  - To detect unusual/suspect activity or operations
- DBMS independent
  - Not part of the DBMS
  - External observation of DBMS activity
- Monitoring sensors
  - Network activity
  - Local SQL commands performed
  - Log analysis

Sensitive data

- Data requiring (extra) protection
  - From loss (disclosure)
  - From misuse
  - From modification
  - From unauthorized access
- Risks
  - Privacy and welfare of individuals
  - Business activities
  - Security-related activities

Sensitive data

- Some databases contain sensitive data
  - Data that should not be made public
    - e.g. clinical records of patients
- Sensitivity depends on: BD purpose + DB data
  - Some record fields, entire records/tables, entire database
    - e.g. personal health record (HER) with all detected pathologies, treatments and interventions
    - e.g. clinical records of an AIDS table
    - e.g. defense-related databases
- Complexity
  - Simple cases: all or nothing
    - Everything sensitive, nothing sensitive
  - Complicated cases: part of the DB elements are sensitive
    - In some cases, sensitivity is extended to the simple existence of a field data or record

Sensitive data:

Factors that make data sensitive

- Inherently sensitive
  - The value itself may be so revealing that it is sensitive
- From a sensitive source
  - The value may reveal the identity of its source
- Declared sensitive
  - The value was explicitly declared sensitive
- Belongs to a sensitive record
  - Value of a record was explicitly marked as sensitive
- Sensitive given previously disclosed information
  - By itself, the data is not sensitive, but together with other data, the whole can be sensitive

Types of disclosures (of sensitive data)

- Exact data
  - The exact value of a sensitive datum
  - The most serious disclosure
- Bounds
  - Sensitive data item is > lower bound or < upper bound
  - Sometimes bounds are used to protect (hide) sensitive data
    - By providing bounds to elements instead of their exact value
- Negative result
  - By getting a negative result for a query on a sensitive value, a user can conclude that the value has a particular set of values
    - e.g. from a list of effective voters we can conclude who didn't vote

Types of disclosures (of sensitive data)

- Existence
  - The existence of a sensitive field in a record can be, by itself, sensitive information
    - Because it may reveal a hidden data gathering & processing activity
- Probable value
  - By crossing the results of several queries we can infer a probability for an element value

Inference

- Definition
  - A way to infer, or derive, sensitive data from non-sensitive data

Inference attacks

- Direct attack
  - Uses queries with a blend of selection rules that use sensitive fields and non-sensitive fields
  - The DBMS can be deceived by the selection rules with non-sensitive fields, which are not intended to select particular records
- Indirect attack
  - Inference of particular values from statistical values computed over several records
    - Counts, sums, averages

Inference attacks

- Tracker attack
  - The database may conceal data when a small number of records make up the large proportion of the data revealed
  - A tracker attack can fool the DBMS by using different queries that reveal data and, by combining the results, the attacker can get the desired information

K-anonymity

L. Sweeney, "K-anonymity: A Model for Protecting Privacy", Int. Journal on Uncertainty, Fuzziness and Knowledge-based Systems. 2002

- Definition
  - No query can deliver an anonymity set with less than k entries
  - The anonymity set is the set of all possible subjects
- Privacy-critical attributes
  - (Unique) identifiers
  - Quasi-identifiers
    - When combined can produce unique tuples
  - Sensitive attributes
    - Potentially unique per subject
    - Disease, salary, crime committed

Multilevel security:
Goal

- Tag information items with security classifications
  - e.g unclassified, confidential, secret, top secret
- Tag queries with security levels
  - The security level of the entity responsible for the query
- Prevent queries from observing values of fields with a different security classification
  - Or from observing meaningful values

Multilevel security: Poli-instanciation

- A record with a particular key field may be duplicated in different security levels
  - Possibly with different values

- This reduces the precision of the database information
  - The correctness of the information depends on the entity performing the query
  - Duplicates know can legitimately occur

Multilevel security: Separation strategies (1)

- Partitioning
  - Different security levels, different databases
  - Queries are directed to the appropriate DB
- Advantages
  - Easy to implement
- Disadvantages
  - Redundancy of information
  - Problems in the access to records with fields with different security levels

Multilevel security: Separation strategies (2)

- Encryption
  - Fields are encrypted with a security-level key
- Advantages
  - Single database, same database structure
- Disadvantages
  - Decryption on each query with the adequate security level key
  - Randomized encryption: equal fields should not produce the same cryptogram
    - Otherwise statistics and known-plaintext attacks disclose values
    - Solution: different keys per record or different IVs per record
  - No encrypted values should be updated by providing another encrypted value

Multilevel security: Separation strategies (3)

- Integrity lock
  - Each data item is formed by three parts:
    - Data item, sensitivity label, checksum
  - The sensitivity label should be
    - Unforgeable (cannot be changed)
    - Unique (cannot be copied to another data item)
    - Concealed (cannot be observed)
- Advantages
  - Can use a regular DBMS
  - Trusted stored procedures are enough to implement them
- Disadvantages
  - Space for storing sensitivity labels and checksums

Laws for the protection of personal data

- Each country has its how set of laws
  - There is not a global consensus
- In Portugal this is supervised by CNPD
  - Comissão Nacional de Proteção de Dados
  - All data processing involving personal data gathered from individuals needs to be submitted to CNPD for authorization
- European Directive for Data Protection
  - To be applyed from May 25, 2018