

IN3020&4020 – Database Systems (2022)

(Between Parts 2 and 3)

Lecture 23: Database security

21 April

Egor V. Kostylev

IFI, University of Oslo

egork@ifi.uio.no



Big Picture (reminder)

Syllabus of the course:

- (Extended) Intro and SQL recap



Part 1. Query processing in relational databases



Part 2. Transaction management in relational databases



- **Database security**

Part 3. NoSQL DBMS

- Advanced DBMS architecture (IN4020 only)

Lecture schedule

#	Date	Part	Topic
...	...	...	...
23	21.04 (Thu)	(no part)	Database security
24	25.04 (Mon)	Part 3: NoSQL	Overview
25	28.04 (Thu)	Part 3: NoSQL	Knowledge Graphs
26	02.05 (Mon)	Part 3: NoSQL	RDF and SPARQL
27	05.05 (Thu)	Part 3: NoSQL	Neo4j and Cypher
–	(skipped)	Part 3: NoSQL	Column-based (Hadoop&MapReduce)
–	(skipped)	Emerging technologies	DBs and ML
28	09.05 (Mon)	for IN4020	Advanced DBMSs (Digital? V. Goebel, IN5040)
29	12.05 (Thu)	Wrap-up	Summary

Exam: Monday, 30 May (see next slide)

Exam formal details:

- **Time:** 30 May 2022, 3pm **Duration:** 4 hours
- **Place:** Silurveien 2 Sal 3D / Home **System:** Inspira
- **Withdrawal deadline:** 1 May

Exam rules and contents:

- no **collaboration** (standard plagiarism rules)
- most likely, the slides will be available (to be confirmed)
 - if so, the content will be in the spirit of the mandatories
- in any case, if you are comfortable with **mandatories**, you should be fine with the exam
- (**last-year exam** was a bit special, so do not look at it much)

Materials to read about database security

1. *Fundamentals of Database Systems* by R. Elmasri and S.B. Navathe (7th edition):
Section 30 (the only in Part 12)
(more than my presentation)
2. *Database Systems: the Complete Book* by H. Garcia-Molina, J. Ullman & J. Widom (2nd edition): Parts of Section 10
3. IN2090 – Databaser og datamodellering (Leif Harald Karlsen):
Lecture 12 (basics)
<https://www.uio.no/studier/emner/matnat/ifi/IN2090/h21/timeplan/>
4. etc.

1. Introduction and main concepts
2. Discretionary access control (DAC)
3. Mandatory access control (MAC)
4. Other control measures

1. Introduction and main concepts

Database Security: Issues and threats

Database security a broad area with many issues:

- Legal and ethical issues
(e.g., private information access rights)
- Policy issues
(e.g., governmental policies regarding credits, medical records)
- System-related issues
(e.g., system levels on which security should be enforces)
- Security categorisation and organisation
(classes of users vs. classes of information)

Threats to databases:

- Loss of integrity: Improper modification of information
- Loss of availability: Legitimate user cannot access data objects
- Loss of confidentiality: Unauthorised disclosure of confidential information

Database Security: Not an isolated concern

Applications often contain much **more** than just the database:
the database works as part of a **network of services**
(other applications, programs, Web servers, etc.)

- Database security can therefore **not only** focus on the database
- Security holes can occur in **all joints** (frontend, backend, network, etc.)
- Safety is therefore always a **comprehensive task**
- Must therefore ensure **each component**
and the **interaction** between them
- Must be clear on what **assumptions**
about other components each part of the system makes

DB-related threat example: SQL injection

SQL injection:

- Attacker **changes** SQL statement to their advantage
(e.g., commonly, adds **conditions** to the **WHERE** clause)
- May lead to **unauthorised** data manipulation
or execution of **system-level** commands

Example:

```
SELECT * FROM Users WHERE  
    username = 'jake' AND password = 'jakespasswd'
```

can be **changed** to

```
SELECT * FROM Users WHERE  
    username = 'jake' AND (password = 'jakespasswd' OR 'x' = 'x')
```

with obvious **consequences**

Database multi-user security mechanisms

DBMS must provide techniques to enable certain **users** or **user groups** to access selected **portions** of a database

There are several **types** of security **mechanisms**; most fundamental:

- **Discretionary** security mechanisms

Used to **grant privileges** to users to access certain data

Commonly used in **practice**

- **Mandatory** security mechanisms

Classify data pieces and **users** into various security **classes**

Give access according to a corresponding **security policy**

Used (and extended) in **professional DBMSs** (e.g., **Oracle**)

We will briefly consider such mechanisms for **access control** below

Control measures for data security

Access control (our main focus below):

- handled by **user accounts** (with **passwords**) and data labelling to ensure the **security mechanism**

Inference control (briefly at the end):

- ensure that information about **individuals** cannot be **derived** key in **statistical databases**

Flow control (briefly at the end):

- prevents information from **flowing** from authorised to unauthorised users disallow **covert channels**

Data encryption (briefly at the end):

- used to protect sensitive data during **transmission** **symmetric/asymmetric encryption**, **digital signatures** and **certificates**

Role of database administrator

Database administrator (DBA):

- central **authority** for **administering** database system
- **superuser** or system account
- **central role** for ensuring security (especially access control)

DBA-privileged commands for **access control**:

- **account creation**
- **privilege granting** and **revocation** (discretionary authorisation)
- **security level assignment** (mandatory authorisation)

Security-related system tables and audit

User must **log in** using assigned **username** and **password**

DBMS has **encrypted tables** for

- **users**, including accounts and passwords
- **login sessions**, including user IDs, beginning and end (may include several transactions)
- **system log**, including user IDs and session IDs

If **tampering** is suspected, **database audit** is performed:
reviewing **log** to examine all accesses and operations
to determine the **possible culprit**

Causes for data sensitivity

Sensitivity of data:

measure of the **importance** assigned to the data by its owner
denoting its **need for protection**

Different parts of data in database may have **different sensitivity**

Factors that can cause **sensitivity status** of a piece of data:

- **inherently sensitive** (e.g., a patient's diagnosis)
- from a **sensitive source** (e.g., from a secret informer)
- **declared sensitive** (e.g., by the data owner)
- has a **sensitive part** (e.g., salary attribute)
- sensitive in **relation** to **previously disclosed** data
(i.e., disclose sensitive information
in conjunction with previously given data)

Security vs. precision

Security:

ensures that sensitive data is kept **safe** from **corruption** and **non-authorised access**

Precision (or Availability):

To make available as much nonsensitive data as possible

(In both cases, sensitivity is **relative** to a specific user)

The **ideal combination** is to maintain
perfect security with **maximum precision**
contradictory in practice, so the best **trade-off** should be found

Security as a part of privacy

Security:

issues related to **access to resources**

Privacy goes **beyond**:

issues related to appropriate **use of information**—
that is, the ability of individuals to control the terms
under which their **personal information** is acquired and used;
for example

- preventing **storage** without a need
- ensuring **appropriate use**

General Data Protection Regulation (GDPR):

an **EU** (and **EEA**) **data protection** and **privacy law**
(should always be considered when considering privacy issues)

2. Discretionary access control

Discretionary access control: Idea

Part of SQL standard and commonly implemented
(sometimes partially)

Discretionary access control (DAC):

- based on granting and revoking privileges to users (or group of users)
- SQL has statements GRANT and REVOKE for granting and revoking

(Revoking may be necessary when temporal access is needed)

There are two levels of privileges: account level and relation level

Account-level privileges

Account-level privileges—

that is, privileges applying to the **whole database**:

- privileges for
 - data definition** (CREATE/DROP/ALTER TABLE/VIEW)
 - data modification** (INSERT/UPDATE/REMOVE)
 - data access** (SELECT)
- **granted/revoked** by the DBA

Examples:

```
GRANT CREATETAB TO UserName
```

```
REVOKE CREATETAB FROM UserName
```

Relation- (table-) level privileges

Relation-level privileges—

that is, privileges applying to **specific tables** and **views**:

- privileges for
 - data modification** (INSERT/UPDATE/REMOVE)
 - data access** (SELECT)
 - constraints** (FOREIGN KEY)
- can be seen as a **access matrix** (Users $\times$ Tables)
- (originally) **granted/revoked** by the **owner** (usually the **creator**)

Example:

```
GRANT INSERT, DELETE
      ON Employee, Department TO UserName
```

Propagation of relation-level privileges

Relation-level privileges can be propagated to other users
by WITH GRANT OPTION modifier

Example:

```
GRANT SELECT ON Employee, Department TO UserName  
WITH GRANT OPTION
```

Revoking is automatically cascading

Example:

```
REVOKE SELECT ON Employee FROM UserName
```

automatically recursively revokes all privileges given by UserName

Should be careful with multiple grantees and should prevent cycles

Some DBMSs allow for horizontal (i.e., number of users) and
vertical (i.e., depth) limits on propagations

Fine-grained privileges via views

Views are a DAC mechanism by themselves:
can give fine-grained access to certain attributes and/or tuples

Examples:

Consider owner UserA of relation R and other user UserB

- UserA can create view V1 of R
that includes only attributes UserA wants UserB to access
and grants SELECT privilege on V1 to UserB
- UserA can also create view V2 of R
that selects only tuples UserA wants UserB to access
and grants SELECT privilege on V2 to UserB

3. Mandatory access control

Mandatory access control: Idea

Mandatory access control (MAC) idea:

- **classify** subjects (i.e., data elements such as attribute values and tuples) and **objects** (i.e., users) to security **classes**
- classes are (possibly partially) **ordered** (i.e., **multilevel security**)
- compose the **classes** of involved subjects and objects using the order when deciding to **give access**

Typical (simple) security classes

- Top secret (TS)
- Secret (S)
- Confidential (C)
- Unclassified (U)

with (total) **order** $TS \geq S \geq C \geq U$

Bell-LaPadula access control model

Two **rules** are enforced

(general rules, may be applied to any type of data)

- Simple security property:

subject (e.g., user) S can **read object** (e.g., value) O

only if $class(S) \geq class(O)$

that is, **no one** can read data with **higher security**

- Star property:

subject (e.g., user) S can **write object** (e.g., value) O

only if $class(S) \leq class(O)$

that is, no one can write data with lower security

(prevents **information flow** from higher to lower classifications)

MAC model for relational databases

Mandatory access control (MAC) in DBMSs:

- less commonly adopted due to complexity (mostly professional)
- objects are attribute values in tuples and tuples themselves
- classification of a tuple is
the highest classification of its attribute values

Example:

EMPLOYEE

Name	Salary	JobPerformance	TC
Smith U	40000 C	Fair S	S
Brown C	80000 S	Good C	S

(We omit the commands for assigning classes,
they are not in the SQL standard anyway)

Reading in MAC model for relational databases

Filtering is used for **reading** (i.e., **SELECT** queries):

- **replacing** by **NULLs** attribute values with higher classification than user's
- **removing** whole tuples when **nothing** left

Example:

EMPLOYEE

Name	Salary	JobPerformance	TC
Smith U	40000 C	Fair S	S
Brown C	80000 S	Good C	S

Query **SELECT * FROM Employee** issued by user of **class C** gives

EMPLOYEE

Name	Salary	JobPerformance	TC
Smith U	40000 C	NULL C	C
Brown C	NULL C	Good C	C

Reading in MAC model for relational databases

Filtering is used for **reading** (i.e., **SELECT** queries):

- **replacing** by **NULLs** attribute values with higher classification than user's
- **removing** whole tuples when **nothing** left

Example:

EMPLOYEE

Name	Salary	JobPerformance	TC
Smith U	40000 C	Fair S	S
Brown C	80000 S	Good C	S

Query **SELECT * FROM Employee** issued by user of **class U** gives

EMPLOYEE

Name	Salary	JobPerformance	TC
Smith U	NULL U	NULL U	U

Writing in MAC model for Relational Databases

Poly-instantiation can be used for **writing** (e.g., **UPDATE** queries):

- updates on the **lower level** is **not allowed** (star property)
- updates on the **same level** is **as usual**
- updates on the **higher level** creates **several copies**
(need to be **careful**, details are omitted)

Example:

EMPLOYEE

Name	Salary	JobPerformance	TC
Smith U	40000 C	Fair S	S
Brown C	80000 S	Good C	S

Query **UPDATE** Employee SET Job_performance = 'Excellent'
WHERE Name = 'Smith' issued by user of **class C** gives

EMPLOYEE

Name	Salary	JobPerformance	TC
Smith U	40000 C	Fair S	S
Smith U	40000 C	Excellent C	C
Brown C	80000 S	Good C	S

DAC vs. MAC

Discretionary access control (DAC):

- more flexible
- more vulnerable
- no flow control

Mandatory access control (MAC):

- more rigid
- higher degree of protection
- ensures flow control

Professional DBMSs (e.g., Oracle) implement more advanced models, often generalising both DAC and MAC:

- Role-based access control
- Label-based (e.g., for tuples) access control

4. Other control measures

Inference control in statistical databases

Statistical database

is a **usual** database with special use:

intuitively, (usual) users are

- assumed to retrieve only **statistical information**
- forbidden to retrieve **individual data**

Essentially, implemented so that only **statistical** (i.e., **aggregate**) queries are allowed; for **example**

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

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

Inference control in statistical databases

Forbidding non-aggregate queries is **not enough**

Example:

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

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

If count is 1, the average is the **exact number**

Can be resolved by **several methods**, for example:

- fixing a **minimum threshold** on number of tuples
- introducing **noise**
- prohibit **sequences** of queries for the same population
(e.g., table)

Flow control

Flow control:

- regulates the **flow** (i.e., **distribution**) of information among accessible objects
- ensures that information contained in some objects does not flow explicitly or implicitly into **less protected** objects

Flow policy:

- specifies **channels** along which information is allowed to move
- **covert channels** (improperly) allows to pass from higher to **lower** classification

We have seen an **example**:

MAC

Encryption of sensitive data

Encryption converts data into cyphertext

- performed by applying an encryption algorithm to data using a pre-specified encryption key
- resulting data can only be decrypted using a decryption key to recover original data

Encryption protocols and algorithms

- a big separate research field
- classic approach: symmetric (secret key) protocols
requires the same secret key for encryption and decryption
- modern approach: asymmetric (public key) protocols
RSA scheme (Rivest, Shamir, and Adleman, 1978):
encryption with public key, decryption with private key
deep mathematics (number theory, one-side functions)

What have we learned?

Brief intro to database security

Next time: Part 3 (NoSQL)