Part 13: Datenschutz: Zugriffsrechte in SQL

Literatur:

- Elmasri/Navathe:Fundamentals of Database Systems, 3rd Edition, 1999. Chap. 22, "Database Security and Authorization"
- Silberschatz/Korth/Sudarshan: Database System Concepts, 3rd Edition, McGraw-Hill, 1999. Section 19.1, "Security and Integrity"
- Kemper/Eickler: Datenbanksysteme (in German), Ch. 12, Oldenbourg, 1997.
- Lipeck: Skript zur Vorlesung Datenbanksysteme (in German), Univ. Hannover, 1996.
- Date/Darwen: A Guide to the SQL Standard, Fourth Edition, Addison-Wesley, 1997.
- van der Lans: SQL, Der ISO-Standard (in German, there is an English version), Hanser, 1990.
- Oracle8 SQL Reference, Oracle Corporation, 1997, Part No. A58225-01.
- Oracle8 Concepts, Release 8.0, Oracle Corporation, 1997, Part No. A58227-01.
- Don Chamberlin: A Complete Guide to DB2 Universal Database. Morgan Kaufmann, 1998.
- Microsoft SQL Server Books Online: Accessing and Changing Data, Administering SQL Server.
- K. Nagel: Informationsbroschüre zum Bundesdatenschutzgesetz, ("Information about the German Federal Data Privacy Law", in German), 10. Auflage, Oldenbourg, 2001.

Lernziele

Nach diesem Kapitel sollten Sie Folgendes können:

- Einige Vorgehensweisen von Hackern erklären.
- Sicherheitsrelevante DBMS Funktionen erklären.
- Die SQL Befehle `GRANT` und `REVOKE` benutzen.
- Einige Grundzüge des deutschen Datenschutzrechtes beim Aufbau von Datenbanken berücksichtigen.

  Oder zumindest erkennen, wann Sie weitere Informationen einholen müssen.

Inhalt

1. Anforderungen
2. Deutsches Datenschutzrecht
3. Die SQL-Befehle GRANT und REVOKE
4. Oracle
5. DB2
6. SQL Server

DB-Sicherheit: Motivation (1)

- Information: wichtiger Aktivposten von Firmen. Darf nicht in die Hände der Konkurrenz gelangen.

  Z.B. Kundendaten, Einkaufspreise, Produkt-Zusammensetzungen.

- Es gibt Gesetze zum Schutz vertraulicher Daten.

  Wenn man z.B. zuläßt, dass ein Hacker Zugriff auf Kreditkartennummern bekommt und diese Daten dann mißbraucht, können hohe Schadenersatzforderungen entstehen. Außerdem: schlechte Reputation.

- Die Vertraulichkeit von Daten muss auch innerhalb der Firma geschützt werden (z.B. Gehälter).

  Man muss auch verhindern, dass Angestellte einfach alle Daten mitnehmen können, wenn sie die Firma verlassen.

DB-Sicherheit: Motivation (2)

- Falls es einem Hacker gelingen sollte, alle Daten zu löschen, oder vollständig durcheinander zu bringen, entsteht ein gewaltiger Schaden für die Firma.

  Man sollte mindestens noch Sicherungskopien haben, auch offline, und an einem anderen Ort untergebracht (wegen einem möglichen Brand im Rechnerraum). Aber die Wiederherstellung dauert, und die letzten Änderungen sind schwer zu rekonstruieren.

- Unautorisierte Änderungen / Verfälschungen der Daten müssen verhindert werden, z.B. sollten Angestellte nicht ihr eigenes Gehalt ändern können.

  Bestimmte Geschäftsregeln — wer darf was tun — sollten durch das Informationssystem automatisch sichergestellt werden.

Sicherheits-Anforderungen (1)

Es sollte möglich sein ...

- sicher zu stellen, dass nur die tatsächlich legitimierten Benutzer Zugriff auf die Datenbank haben.

  → Benutzer Identifikation/Authentifikation

- festzulegen, welcher Nutzer welche Operationen auf welchen DB-Objekten durchführen darf.

  → Benutzer-Autorisierung

- die Aktionen der Benutzer zu protokollieren.

  → Auditing

Sicherheits-Anforderungen (2)

Es sollte auch möglich sein ...

- die Benutzung von Resourcen (Platten-Platz, CPU-Zeit) für jeden Nutzer zu begrenzen (Quotas).

  Sonst kann ein einzelner Nutzer die ganze Datenbank zum Stillstand bringen.

- Gruppen von Benutzern mit den gleichen Rechten zu verwalten.

  Umgekehrt kann ein Nutzer auch unterschiedliche Rollen haben.

- mehrere Administratoren mit unterschiedlichen Befugnissen zu haben (nicht nur einen einzelnen Nutzer "root", der alles darf).

Sicherheits-Anforderungen (3)

Es sollte auch möglich sein ...

- die Vertraulichkeit und Integrität der Daten auch in einer Netzwerk-Umgebung zu garantieren.

  Oft geschieht die Client-Server Kommunikation unverschlüsselt.

- sicher zu stellen, dass niemand direkten Zugriff auf die Daten hat (und damit die Zugriffskontrolle der Datenbank umgeht).

- darauf zu vertrauen, dass nicht durch Programmierfehler im DBMS Sicherheitslücken bestehen (oder sie wenigstens schnell korrigiert werden).

Benutzer-Authentifikation (1)

- Üblich ist, dass sich Benutzer mit Passworten gegenüber dem System ausweisen.
- Zum Teil führen die DBMS selbst eine Benutzer-Identifikation durch, zum Teil verlassen sie sich auf das Betriebssystem.

  Beide Möglichkeiten haben Vor- und Nachteile: Es ist bequem, nicht mehrfach Passworte eingeben zu müssen, und es ist besser, wenn Anwendungsprogramme keine Passworte enthalten. Aber in einer Client-Server-Umgebung kann man nicht so sicher sein, dass der Benutzer auf dem Client tatsächlich die Person ist, die er/sie behauptet, zu sein (und dass der Computer wirklich der richtige Computer ist). Manche Client-Betriebssysteme haben ein schwächeres Sicherheitssystem als der Server (bei manchen PCs kann jeder eine Boot-CD einlegen).

Benutzer-Authentifikation (2)

- Es ist gefährlich, wenn des gleiche Passwort immer wieder genutzt wird.

  Wenn ein Hacker das Passwort irgendwie beobachten kann, kann er es auch nutzen. Banken senden ihren Kunden typischerweise Listen mit Einmal-Passworten (TANs).

- Einige Administratoren zwingen die Nutzer, ihre Passworte in regelmäßigen Abständen zu ändern.

  Z.B. jeden Monat. Dies kann aber zu schwächeren Passworten führen, (und zu häufigeren Problemen mit vergessenen Passworten), als wenn der Benutzer ein neues Passwort nur wählt, wenn er dazu bereit ist. Wenn ein System den Benutzer zwingt, sein Passwort zu ändern, prüft es üblicherweise auch, dass neues und altes Passwort sich deutlich unterscheiden, und dass der Benutzer nicht zu schnell zurück wechselt.

Benutzer-Authentifikation (3)

- Trojaner sind Programme, die neben der Hauptfunktion, für die sie eingesetzt werden, im Verborgenen noch andere (schlechte) Dinge tun.
- Speichern Sie nie ein Passwort für einen anderen Computer auf der Festplatte Ihres Computers. Das gilt auch für Cookies, die Passworte ersetzen.

  Wenn Sie sich auf diese Art bei dem anderen Rechner einloggen können, ohne ein Passwort einzugeben, kann das auch jemand, der sich (z.B. mit Trojaner) die entsprechenden Daten von Ihrem System besorgt. Wenn das Passwort in einer Datei steht, ist es besonders einfach, und betrifft auch Passworte, die Sie nicht nutzen, wenn der Trojaner aktiv ist. Je nach Betriebssystem könnte er aber auch Tastendrücke mitloggen oder den Hauptspeicher anschauen.

Benutzer-Authentifikation (4)

- Passworte sollten nicht leicht zu raten sein.

  Der Name Ihrer Freundin/Ihres Freundes, Ihrer Lieblings-Popgruppe (Autor, Schauspieler, Film, Urlaubsland, etc.), Ihre Telefonnummer, Adresse, Geburtsdatum wären besonders leicht. Falls es einem Hacker gelingt, an eine verschlüsselte Version Ihres Passwortes zu kommen, kann er mit einer schnellen Verschlüsselungsroutine viele Worte probieren (selbst wenn eine direkte Entschlüsselung nicht möglich ist). Ein Hacker würde in so einem Fall den Duden durchprobieren, alle Namen von Personen, Popgruppen, relativ lange Folgen von Ziffern, zumindest alle kurzen Folgen von Kleinbuchstaben, etc. Außerdem alles auch rückwärts und auf verschiedene andere Arten leicht verändert. Deswegen wird empfohlen, dass ein gutes Passwort nicht zu kurz sein sollte, und Buchstaben, Ziffern, und Sonderzeichen enthalten sollte. Um eine möglichst zufällige Buchstabenfolge zu bekommen, kann man sich einen Satz denken und davon die Anfangsbuchstaben nehmen.

Benutzer-Authentifikation (5)

- Wenn ein System mehrere Anmeldeversuche mit falschem Passwort entdeckt, sollte es einen Alarm auslösen und eventuell den Account sperren.

  Außerdem gibt es oft eine künstliche Verzögerung, bevor das Programm dem Benutzer mitteilt, dass die Anmeldedaten falsch waren. So kann ein Hackerprogramm nicht viele Worte in kurzer Zeit probieren. Das Sperren des Accounts bestraft aber den korrekten Nutzer.

- Viele DBMS haben Default-Passworte für den Administrator. Diese müssen sofort geändert werden.

  Z.B. hat in Oracle der automatisch angelegte Administrator-Account `SYSTEM` das Passwort `MANAGER`, und das Passwort für `SYS` (kann alles) ist `CHANGE_ON_INSTALL`. Jeder Hacker weiß das. In SQL Server hat der mächtigste Account `sa` direkt nach der Installation kein Passwort.

Benutzer-Authentifikation (6)

- Man sollte Gast-Accounts löschen oder sperren.

  Oracle hat einen Account `SCOTT` mit Passwort `TIGER`. SQL Server hat einen Account `guest`, für Windows-Nutzer, die der Datenbank unbekannt sind. Prüfen Sie die Accounts im System in regelmäßigen Abständen und sperren Sie alle, die nicht wirklich benötigt werden.

- Das die meisten DBMS Client-Server Systeme sind, ist der DB-Server automatisch am Netz, sobald Ihr Rechner mit dem Internet verbunden ist.

  In dieser Hinsicht verhält sich der DB-Server ähnlich wie ein Web-Server. Selbst wenn der Hacker sich nicht beim Betriebssystem anmelden kann, kann er sich vielleicht bei der Datenbank anmelden. Oracle wartet normalerweise auf Port 1521 auf Verbindungen, die Default-SID ist "`ORCL`".

Benutzer-Authentifikation (7)

- Wird ein Passwort unverschlüsselt über das Internet verschickt, können es Hacker eventuell lesen.
  - ◇ Beim klassischen Ethernet kann man alle Pakete mitlesen, die über das Netz geschickt werden.

    Mit modernen Switches ist das nicht mehr möglich, aber erst, nachdem der Switch die Adressen der angeschlossenen Rechner gelernt hat. Man kann sich also nicht darauf verlassen, dass andere Rechner, die an das lokale Netz angeschlossen sind, die Datenpakete nicht auch erhalten.
  - ◇ Die Route, die Datenpakete über das globale Internet nehmen, ist kaum vorhersehbar.

    Ein Gateway wird vielleicht von einem Hacker betrieben, oder ist schon von einem Hacker "geknackt".

Benutzer-Authentifikation (8)

- Wenn Sie ein Passwort eingeben, achten Sie darauf, dass Sie auch mit dem richtigen Programm bzw. dem richtigen Computer verbunden sind.

  Früher gab es Programme, die wie ein UNIX Login Prompt aussahen, aber das Passwort an einen Hacker schickten.

  Der Suchpfad für Kommandos darf nur vertrauenswürdige Verzeichnisse enthalten (z.B. nicht "."). Sonst bekommt man vielleicht ein ganz anderes Programm, wenn man z.B. "sqlplus" aufruft.

  Es gibt viele "Phishing" ("password fishing") EMails, die z.B. behaupten, von der eigenen Bank zu kommen, und zur Eingabe von PIN und TAN auffordern. Tatsächlich kommt man mit der URL aber auf eine Hacker-Webseite, die wie die echte Webseite der Bank aussieht.

Benutzer-Authentifikation (9)

- Man sollte möglichst nicht das gleiche Passwort für verschiedene Systeme (auch Webshops) nutzen.

  Angestellte des Webshops können Ihr Passwort möglicherweise im Klartext lesen. Es ist zwar üblich, Passworte nur verschlüsselt (z.B. MD5-Hash) zu speichern, aber es gibt keine Garantie, dass jeder Shop-Betreiber das auch so macht. Als Schutz gegen vorberechnete Listen von Hashes von Wörterbüchern ("Rainbow Table") wird dem Passwort eine längere, zufällige Zeichenkette angehängt ("Salt"), die zusammen mit dem Hashwert in der Datenbank gespeichert wird.

- Sagen Sie Ihr Passwort (auch PIN etc.) nie offiziell klingenden Unbekannten am Telefon!

  Z.B. dem technischen Support des Rechenzentrums, das gerade auf LDAP umstellt, und Ihnen die Mühe ersparen möchte, ins Rechenzentrum zu kommen, und dort Ihr Passwort noch einmal einzutippen.

Zugriffsrechte (1)

- Oft haben verschiedene Nutzer einer Datenbank auch verschiedene Rechte.
- Kommandos bestehen aus
  - "Subject" (der Benutzer, der es ausführt),
  - "Verb" (die Operation, z.B. "`INSERT`), und
  - "Objekt" (typischerweise eine Tabelle).
- In SQL kann man mit dem `GRANT`-Kommando (s.u.) definieren, welche dieser Tripel erlaubt sind.

  Das DBMS speichert also eine Menge von Tripeln (u, o, d), aus Benutzer u, Operation o (im wesentlichen `SELECT`, `INSERT`, `UPDATE`, `DELETE`), Objekt/Tabelle o. Eigentlich Quadrupel: Wer hat das Recht erteilt?

Zugriffsrechte (2)

- Das Subjekt-Verb-Objekt Modell für die Zugriffsrechte hat aber Einschränkungen:
  - Kommandos wie "`CREATE TABLE`" beziehen sich nicht auf existierende Objekte, aber man muss ihre Benutzung einschränken können.

    Prinzip: Jeder sollte nur das tun können, was er tun muss.
  - Für DB-Administratoren sollen die normalen Zugriffsbeschränkungen nicht gelten. Es soll aber auch nicht jeder Administrator alles können.

    Große Firmen haben mehrere Administratoren.

Zugriffsrechte (3)

- Während das Subjekt-Verb-Objekt Modell sehr portabel ist (schon im SQL-86 Standard enthalten), sind die Erweiterungen zur Lösung der obigen Probleme meist vorhanden, aber sehr systemabhängig:
  - Oracle hat neben Objektrechten nach dem obigen Modell noch eine große Anzahl von Systemrechten (wer darf welches Kommando/Verb benutzen, ggf. auch für beliebige Objekte).
  - Andere Systeme unterscheiden oft nur wenige Benutzerklassen, die verschieden privilegiert sind.

Zugriffsrechte (4)

- Sichten und serverseitige Prozeduren ("stored procedures") erlauben es, Objekte zu verkapseln:
  - ◇ Z.B. ist es möglich, dass ein Benutzer für eine Tabelle selbst kein `SELECT`-Recht hat, aber über eine Sicht bestimmte Zeilen/Spalten oder aggregierte Daten sehen darf.
  - ◇ Es ist auch möglich, dass ein Benutzer kein direktes `INSERT`-Recht für eine Tabelle hat, aber eine Prozedur benutzen kann, die Daten in diese Tabelle nach zusätzlichen Prüfungen einfügt.

Sicherheitsmodelle

- Es gibt zwei grundlegend verschiedene Sicherheitsmodelle. Normale Datenbanksysteme implementieren nur das erste:
  - ◇ "Discretionary Access Control" erlaubt den Administratoren, die Zugriffsrechte nach Belieben zu vergeben.
  - ◇ "Mandatory Access Control" basiert auf einer Klassifizierung von Benutzern und Daten.

    Z.B. "confidential", "secret", "top secret". Benutzer können nur Daten auf der eigenen oder einer niedrigeren Sicherheitsebene lesen, und nur Daten auf der eigenen oder einer höheren Sicherheitsebene schreiben.

Auditing

- In manchen Systemen kann man die Aktionen der Benutzer mitprotokollieren lassen.

  Das könnte Ärger mit dem Betriebsrat geben, eventuell auch mit dem Datenschutz. Man beachte, dass nicht nur erfolgreich ausgeführte Kommandos protokolliert werden müssen, sondern auch Kommandos, die z.B. wegen nicht ausreichender Zugriffsrechte abgelehnt wurden.
- So kann man wenigstens hinterher herausfinden, wer für ein Problem verantwortlich ist.
- Eventuell fallen bei einer Kontrolle auch merkwürdige Muster auf, bevor wirklich ein Schaden eintritt.

  Man muss sehr selektiv mitprotokollieren, wenn man im Protokoll wirklich noch etwas Interessantes finden will.

Datensicherheit

- Niemand (der nicht ohnehin DBA Rechte hat) sollte direkten Zugriff auf die Daten haben (und damit die Zugriffskontrolle der Datenbank umgehen).

  Aus Performance-Gründen werden die Daten normalerweise unverschlüsselt in Betriebssystem-Dateien gespeichert. Wer auf diese Dateien Zugriff hat, kann auch ohne DB-Account die Daten darin lesen.
- Backup Bänder müssen weggeschlossen werden.
- In Deutschland wurde ein PC gestohlen, der ein AIDS-Register enthielt. In den USA sind mehrere Notebooks der CIA mit geheimen Unterlagen verschwunden.

Inhalt

1. Anforderungen
2. Deutsches Datenschutzrecht
3. Die SQL-Befehle GRANT und REVOKE
4. Oracle
5. DB2
6. SQL Server

Datenschutzrecht (1)

- In diesem Abschnitt sollen die Vorschriften des Bundesdatenschutzgesetzes erklärt werden.

  Ich bin kein Jurist, und der zur Verfügung stehende Platz ist nicht ausreichend für eine vollständige Erklärung. Alle Angaben sind ohne Gewähr. Wenn es wirklich wichtig ist, informieren Sie sich bitte bei einem Juristen.

- Das Gesetz soll davor schützen, dass man durch Umgang mit seinen personenbezogenen Daten in seinem verfassungsmäßig garantierten Persönlichkeitsrecht beeinträchtigt wird (freie Selbstbestimmung bei der Entfaltung der Persönlichkeit).

Datenschutzrecht (2)

- Es soll vermieden werden, dass "Bürger nicht mehr wissen können, wer was wann bei welcher Gelegenheit über sie weiß."
- Man soll normalerweise selbst über die Preisgabe und Verwendung seiner personenbezogenen Daten bestimmen können ("Informationelle Selbstbestimmung").
- Personenbezogene Daten sind "Einzelangaben über persönliche oder sächliche Verhältnisse einer bestimmten oder bestimmbaren natürlichen Person".

Datenschutzrecht (3)

- Das Gesetz bezieht sich nicht auf aggregierte oder anonymisierte Daten.
- Ebenfalls ausgenommen sind Daten, die ausschließlich für persönliche oder familiäre Tätigkeiten erhoben/verarbeitet werden.
- Das Gesetz gilt auch für "nicht-automatisierte Dateien".

  Es ist also nicht richtig, dass das Gesetz grundsätzlich erst dann anwendbar ist, wenn die Daten in einem Computer gespeichert werden. Die Daten müssen aber gleichartig aufgebaut sein und nach bestimmten Merkmalen zugänglich sein (Kartei).

Datenschutzrecht (4)

Besonders schützungswerte, sensitive Daten:

- "Besondere Arten personenbezogener Daten" sind:
  - ◇ rassische und ethnische Herkunft
  - ◇ politische Meinungen
  - ◇ religiöse oder philosophische Überzeugungen
  - ◇ Gewerkschaftszugehörigkeit
  - ◇ Gesundheit
  - ◇ Sexualleben
- Für diese Daten gelten teilweise verschärfte Vorschriften (s.u.). Sie müssen in jeder Einverständniserklärung explizit genannt werden.

Datenschutzrecht (5)

Verbot mit Ausnahmen:

- Personenbezogene Daten dürfen nur abgespeichert werden, wenn
  - ◇ der Betroffene zugestimmt hat,
  - ◇ die Daten aufgrund eines Gesetzes gespeichert werden müssen,
  - ◇ die Daten zur Erfüllung eines Vertrages (o.ä.) mit dem Betroffenen gespeichert werden müssen,
  - ◇ ... (Fortsetzung siehe nächste Folie)

Datenschutzrecht (6)

Verbot mit Ausnahmen, Forts.:

- Zulässige Gründe für die Speicherung personenbezogender Daten, Forts.:
  - ◇ wenn es zur Wahrung berechtigter Interessen der Firma nötig ist, und es keinen Grund zur Annahme gibt, dass schutzwürdige Interessen des Betroffenen überwiegen,
  - ◇ die Daten allgemein zugänglich sind oder veröffentlicht werden dürften.

    Sofern keine schutzwürdigen Interessen entgegenstehen.

Datenschutzrecht (7)

Zweckbindung der Daten:

- Daten müssen für einen speziellen Zweck gesammelt werden und dürfen später nur unter bestimmten Voraussetzungen für andere Zwecke verwendet werden:
  - ◇ Wahrung berechtigter Interessen (wie oben)
  - ◇ Daten allgemein zugänglich (wie oben)
  - ◇ Wahrung berechtigter Interessen eines Dritten,

    Mit Einschränkungen, z.B. bei strafbaren Handlungen, oder Angaben des Arbeitgebers auf arbeitsrechtliche Verhältnisse.

Datenschutzrecht (8)

Zweckbindung der Daten, Forts.:

- Gründe für Nutzung von Daten zu anderen Zwecken als bei der ursprünglichen Erhebung, Forts.:
  - ◇ Abwehr von Gefahren für die staatliche Sicherheit, Verfolgung von Straftaten
  - ◇ Werbung/Marktforschung mit Einschränkungen

    Wenn es sich eine Liste von Angehörigen einer Personengruppe handelt, die nur die Zugehörigkeit zu der Gruppe, Berufsbezeichnung, Namen, akademische Titel, Anschrift und Geburtsjahr enthält, und kein Grund zur Annahme besteht, dass der Betroffene ein schutzwürdiges Interesse hat, das dem entgegensteht.
  - ◇ Für Forschungszwecke (unter Bedingungen).

Datenschutzrecht (9)

Benachrichtigungspflicht:

- Personen müssen darüber informiert werden,
  - ◇ dass Daten über sie gespeichert werden,
  - ◇ welche Arten von Daten das sind,
  - ◇ was der Zweck der Datenverarbeitung ist,
  - ◇ wer die Daten speichert (Stelle/Firma)

  (falls sie es nicht ohnehin schon wissen).
- Es gibt aber viele Ausnahmen von der Benachrichtigungspflicht, siehe nächste Folie.

Datenschutzrecht (10)

Ausnahmen von der Benachrichtigungspflicht:

- Daten müssen per Gesetz gespeichert werden
- Daten entstammen einer öffentlichen Quelle, unverhältnismäßiger Aufwand
- Daten müssen geheim gehalten werden (Gesetz oder überwiegendes Interesse eines Dritten)
- Würde Geschäftszweck erheblich gefährden (und kein überwiegendes Interesse des Betroffenen)
- Werbezwecke mit eingeschränkten Daten (s.o.), unverhältnismäßiger Aufwand.

Datenschutzrecht (11)

Datenschutzbeauftragter:

- Firmen oder öffentliche Stellen, die personenbezogende Daten verarbeiten, müssen einen Datenschutzbeauftragten bestellen.

  Bei Firmen gilt das nur, wenn mindestens 10 Personen ständig mit der Verarbeitung dieser Daten beschäftigt sind. Handelt es sich aber um besonders schützenswerte Daten, oder handelt die Firma mit den Daten, ist ein Datenschutzbeauftragter unbedingt nötig.
- Der Datenschutzbeauftragte muss die erforderliche Fachkunde und Zuverlässigkeit besitzen.
- Er ist in Ausübung dieser Tätigkeit weisungsfrei.

Datenschutzrecht (12)

Datenschutzbeauftragter, Forts.:

- Der Datenschutzbeauftragte
  - ◇ überwacht die ordnungsgemäße Anwendung der Datenverarbeitungsprogramme,
  - ◇ schult die Benutzer in den Vorschriften des Datenschutzes,
  - ◇ ist rechtzeitig über Vorhaben zur Verarbeitung personenbezogener Daten zu informieren.
- Es gibt auch externe Berater als Datenschutzbeauftragte.

Datenschutzrecht (13)

Meldepflicht:

- Die Verarbeitung personenbezogener Daten ist den zuständigen Aufsichtsbehörden zu melden, sofern man keinen Datenschutzbeauftragten hat.

  Die Meldepflicht entfällt aber, wenn unter 10 Personen mit der Verarbeitung der Daten beschäftigt sind, und die Betroffenen ihr Einverständnis erklärt haben, oder es der Abwicklung eines Vertrages oder ähnlichen Vertrauensverhältnisses dient.
- Hat die Firma einen Datenschutzbeauftragten, so muss dieser entsprechend informiert werden.
- Falls man mit den Daten handelt, gilt die Meldepflicht an die Behörden auf jeden Fall.

Datenschutzrecht (14)

Inhalte der Meldepflicht:

- Verantwortliche Stelle (Firma, DV-Leiter)
- Zweck der Datenerhebung und -Verarbeitung
- Betroffene Personengruppen
- Daten oder Datenkategorien
- Empfänger, denen Daten mitgeteilt werden könnten
- Regelfristen für die Löschung der Daten
- geplante Datenübermittlung in Drittstaaten
- Maßnahmen zur Datensicherheit

Datenschutzrecht (15)

Vorabkontrolle:

- Während normalerweise diese Meldung ausreicht, ist bei besonders kritischen Fällen eine Vorabkontrolle notwendig:
  - ◇ bei besonders sensitiven Daten (Folie 13-29)
  - ◇ wenn die Verarbeitung der Daten dazu bestimmt ist, Persönlichkeit, Fähigkeiten, Leistung des Betroffenen zu bewerten.
- Zuständig ist der Datenschutzbeauftragte

  Kann sich in Zweifelsfällen an die Behörde wenden.

Datenschutzrecht (16)

Verbot von rein automatischen Entscheidungen:

- Es ist verboten, Entscheidungen über Menschen, die besonders negative Konsequenzen für sie haben können, rein automatisch (durch ein Computerprogramm) zu fällen.

  Mindestens müssen die Betroffenen darüber informiert werden, und die Möglichkeit bekommen, ihren Standpunkt darzustellen. Der Fall muss dann anschließend erneut betrachtet werden (durch einen Menschen).

- Wenn der Betroffene fragt, muss die grundlegende Logik der Entscheidung erklärt werden.

Datenschutzrecht (17)

Recht auf Information:

- Personen, über die Daten gespeichert werden, können die speichernde Stelle fragen,
  - welche Daten genau über sie gespeichert werden,
  - was die Quelle für diese Daten war,
  - zu welchem Zweck die Daten gespeichert sind,
  - an wen die Daten übermittelt wurden.
- Normalerweise müssen diese Fragen beantwortet werden, und in den meisten Fällen ohne Gebühr.

Datenschutzrecht (18)

Recht auf Information, Forts.:

- Die Polizei und ähnliche Behörden sind vom Recht auf Information natürlich ausgenommen.
- Man kann aber die Bundesdatenschutzbeauftragten bitten, die Daten zu prüfen.

  Weitere Ausnahmen: Wenn eine Firma mit Daten handelt, braucht sie die Frage, an wen die Daten übermittelt wurden, nicht zu beantworten (Geschäftsgeheimnis). Ggf. hat ein Richter zu entscheiden, was wichtiger ist. Firmen, die mit Daten handeln, können eine Gebühr für die Beantwortung der Fragen verlangen, wenn mit der Antwort Geld verdient werden kann. Die Gebühr kann aber niemals höher als die tatsächlichen Kosten sein. Gibt es Grund zur Annahme, dass die gespeicherten Daten inkorrekt sind, kann keine Gebühr verlangt werden.

Datenschutzrecht (19)

Recht auf Korrektur:

- Falsche Daten müssen berichtigt werden.

  Wenn die betroffene Person behauptet, die Daten seien falsch, und es läßt sich nicht feststellen, ob die Daten tatsächlich richtig oder falsch sind, müssen die Daten blockiert werden (können nicht mehr benutzt werden, sind als gelöscht markiert). Dies gilt nicht für Firmen, die mit Daten handeln, aber sie müssen den Daten eine Gegendarstellung der betroffenen Person beifügen und dürfen die Daten nicht ohne diese Gegendarstellung weitergeben.

- Falls die falschen Daten bereits weitergegeben wurden, muss der Empfänger informiert werden.

  Es sei denn, der Aufwand dafür wäre unverhältnismäßig hoch im Vergleich zum Schaden für die betroffene Person.

Datenschutzrecht (20)

Pflicht zur Löschung:

- Daten müssen gelöscht werden, wenn sie
  - ◇ illegal gespeichtert wurden,
  - ◇ besonders sensitiv sind, und nicht bewiesen werden kann, dass sie korrekt sind,

    Siehe Folie 13-29, hier auch Information über illegale Aktivitäten.
  - ◇ nicht mehr für den ursprünglichen Zweck nötig sind.
- U.u. reicht es, sie als gelöscht zu markieren.

  Sie dürfen dann nicht mehr zugreifbar sein, können aber für im Gesetz beschriebene Zwecke wiederhergestellt werden.

Datenschutzrecht (21)

Technische Anforderungen:

- Um Mißbrauch der Daten zu verhindern, ist (in vernüftigen Grenzen) sicherzustellen, dass
  - ◇ nur zulässige Nutzer Zugriff auf die Rechner haben (Schutz vor Hackern),
  - ◇ diese Nutzer nur in den Grenzen ihrer Zugriffsrechte damit arbeiten können,
  - ◇ festgestellt werden kann, an wen (welche Firmen oder Behörden) Daten übermittelt wurden,

Datenschutzrecht (22)

Technische Anforderungen, Forts.:

- Es ist weiter sicherzustellen, dass
  - ◇ Daten bei der Übermittlung geschützt sind,

    Z.B. sollten Daten nur verschlüsselt über öffentliche Netze übertragen werden.
  - ◇ es möglich ist, festzustellen, wer die Daten eingegeben, geändert, gelöscht (?) hat,
  - ◇ die Daten vor Beschädigung/Verlust geschützt sind.

    Wenn z.B. festgehalten werden muss, an wen Daten übermittelt wurden, darf die Firma nicht einfach sagen können, "leider ist uns die Platte kaputt gegangen, auf der diese Daten standen."

Datenschutzrecht (23)

Schadenersatz/Strafe:

- Wenn eine Firma/Behörde jemanden durch illegale oder inkorrekte Verarbeitung seiner/ihrer Daten schädigt, muss sie Schadenersatz zahlen.
- Dies gilt aber nicht, wenn sie einen angemessenen Standard an Sorgfalt erfüllt hat.

  Für Behörden gilt diese Einschränkung nicht, dafür ist die maximale Entschädigung begrenzt.
- Wenn dieses Gesetz vorsätzlich verletzt wurde, um damit Geld zu verdienen oder jemanden zu schädigen, ist die Maximalstrafe 2 Jahre Gefängnis.

Inhalt

1. Anforderungen
2. Deutsches Datenschutzrecht
3. Die SQL-Befehle GRANT und REVOKE
4. Oracle
5. DB2
6. SQL Server

GRANT Kommando (1)

- In SQL können Zugriffsrechte für Datenbankobjekte (Tabellen, Sichten, etc.) an andere Benutzer mit dem `GRANT` Kommando gegeben werden.
- `GRANT` war schon im SQL-86 Standard enthalten. Es hat die Form

  `GRANT ⟨Rechte⟩ ON ⟨Objekt⟩ TO ⟨Nutzer⟩`
- Das DBMS speichert diese Tripel im Systemkatalog ab, und prüft bei jedem Zugriff, ob der aktuelle Nutzer das entsprechende Recht hat.

  Es wird noch zusätzlich vermerkt, wer das Recht vergeben hat.

GRANT Kommando (2)

- Angenommen, die Tabelle `AUFGABEN` gehört dem Benutzer `BRASS`.
- Er kann Lese- und Einfüge-Rechte ("privileges") für diese Tabelle den Benutzern `MEIER` und `JUNG` mit folgendem Kommando geben:

```
GRANT SELECT, INSERT ON AUFGABEN TO MEIER, JUNG
```

- Dann können `MEIER` und `JUNG` z.B. Folgendes tun:
  - ◇ `SELECT * FROM BRASS.AUFGABEN`
  - ◇ `INSERT INTO BRASS.AUFGABEN VALUES (...)`

GRANT Kommando (3)

- `MEIER` und `JUNG` können die einmal eingegebenen Tabellenzeilen aber nicht ändern oder löschen.

  Auch nicht die Zeilen, die sie selbst eingegeben haben. Die Zeilen werden Bestandteil der Tabelle, die `BRASS` gehört. Wenn man nicht selbst etwas programmiert (z.B. eine Spalte mit `DEFAULT USER` (in Oracle), für die nicht explizit ein Wert angegeben werden kann: s.u.), ist nicht nachzuvollziehen, wer eine bestimmte Zeile eingegeben hat. Manche DBMS haben Erweiterungen für diesen Zweck.
- Man kann später z.B. dem Benutzer `MEIER` noch zusätzlich die fehlenden Rechte geben:

```
GRANT UPDATE, DELETE ON AUFGABEN TO MEIER
```

  Die bereits vorher vergebenen Rechte bleiben dabei bestehen.

GRANT Kommando (4)

Rechte für Tabellen/Sichten in SQL-92:

- `SELECT`: Lesezugriff (Anfragen an die Tabelle).

  In SQL Server kann man `SELECT` für einzelne Spalten vergeben. Das ist nicht im SQL-92 Standard und geht nicht in Oracle und DB2. Aber Sichten haben den gleichen Effekt.

- `INSERT`: Einfügen neuer Zeilen.

- `INSERT`$(A_1, \ldots, A_n)$: Nur für die Spalten A_i können Werte angegeben werden, die übrigen werden mit dem jeweiligen Defaultwert gefüllt ($\rightarrow$ `CREATE TABLE`).

  `INSERT` nur für bestimmte Spalten ist Teil des SQL-92 Standards, aber nur in Oracle möglich (nicht in SQL Server und DB2). Mit Sichten kann man den gleichen Effekt erreichen.

GRANT Kommando (5)

Rechte für Tabellen/Sichten in SQL-92, Forts.:

- `UPDATE`: Änderung von Tabelleneinträgen.

- `UPDATE`$(A_1, \ldots, A_n)$: Nur Daten in den Spalten A_i können geändert werden.

- `DELETE`: Löschung von Tabellenzeilen.

- `REFERENCES`: Anlegen von Fremdschlüsseln, die auf diese Tabelle verweisen.

  Wenn Benutzer A in seiner Tabelle R auf die Tabelle S des Benutzers B verweist, aber B keine `DELETE`-Rechte an R bekommt, kann B referenzierte Zeilen aus seiner eigenen Tabelle S nicht mehr löschen. A kann auch prüfen, ob ein Schlüsselwert in S existiert.

GRANT Kommando (6)

Rechte für Tabellen/Sichten in SQL-92, Forts.:

- `REFERENCES`$(A_1, \ldots, A_n)$: Nur dieser Schlüssel darf in Fremdschlüsseln referenziert werden.

  Das ist nur wichtig, wenn eine Tabelle mehrere Schlüssel hat. Unterstützt in Oracle und DB2 (nicht in SQL Server).

Rechte für Domains/Zeichensätze, etc. in SQL-92:

- `USAGE`: Verwendung z.B. der Domain (benutzerdefinierter Datentyp) in `CREATE TABLE` Anweisungen.

  In keinem der drei DBMS (Oracle, DB2, SQL Server) unterstützt.

GRANT Kommando (7)

Weitere Rechte für Tabellen (nicht im Standard):

- `ALTER`: Recht, die Tabellendefinition zu ändern.

  Unterstützt in Oracle und DB2, nicht in SQL Server.

- `INDEX`: Recht, einen Index für de Tabelle anzulegen.

  Unterstützt in Oracle und DB2, nicht in SQL Server.

Rechte für Prozeduren/Packages (nicht im Standard):

- `EXECUTE`: Recht, die Prozedur auszuführen.

  Unterstützt in allen drei DBMS.

- `BIND`: Neuoptimierung von SQL-Anweisungen.

  Nur in DB2 für “Packages” (SQL-Anweisungen eines Programms).

GRANT Kommando (8)

Rechte für Schema Objekte (nur DB2):

- `ALTERIN`: Recht, jedes Objekt (z.B. Tabelle) des Schemas zu ändern.
- `CREATEIN`: Recht, Objekte im Schema anzulegen.
- `DROPIN`: Recht, Objekte im Schema zu löschen.

Rechte für Directory Objekte (nur Oracle):

- `READ`: Recht, Dateien im Directory zu lesen.

Alle Rechte auf dieser Folie sind nicht im SQL Standard enthalten.

GRANT Kommando (9)

`GRANT ALL PRIVILEGES`:

- Anstatt einzelne Rechte aufzulisten, geht auch:

  `GRANT ALL PRIVILEGES ON` ⟨`Object`⟩ `TO` ⟨`Users`⟩
- Dies sind alle Rechte, die der Benutzer, der das `GRANT`-Kommando ausführt, selber für ⟨`Object`⟩ hat.

`TO PUBLIC`:

- Um Rechte an alle Benutzer der Datenbank zu geben (auch an zukünftige Nutzer), schreibt man:

```
GRANT SELECT ON COURSES TO PUBLIC
```


GRANT Kommando (10)

`WITH GRANT OPTION`:

- Man kann einem Benutzer ein Recht auch mit der Möglichkeit geben, das Recht weiterzugeben.
- Dazu hängt man die Klausel "`WITH GRANT OPTION`" an das `GRANT`-Kommando an:

```
GRANT SELECT ON COURSES TO BRASS
WITH GRANT OPTION
```

- Dies erlaubt dem Benutzer `BRASS` auch, die `GRANT`-Option an andere Benutzer weiterzugeben (die es dann selbst weitergeben können, u.s.w.).

GRANT Kommando (11)

Besitzer eines Objektes:

- Der Besitzer eines Objektes (z.B. einer Tabelle), d.h. der Benutzer, der die Tabelle angelegt hat, hat alle Rechte an dem Objekt inklusive `GRANT OPTION`.

  Direkt nach Anlegen der Tabelle hat nur der Besitzer Rechte an ihr. Andere Benutzer bekommen diese Rechte nur durch explizite `GRANT`s. Allerdings können Benutzer mit Administrator-Rechten auf Tabellen oft auch ohne die mit `GRANT` vergebenen Rechte zugreifen.
- Der Besitzer eines Objektes kann es wieder löschen (`DROP TABLE`).

  Dieses Recht ist nicht in den anderen Rechten enthalten. Mit `DELETE` kann man nur den Inhalt der Tabelle löschen.

GRANT Kommando (12)

`CONTROL`-Recht in DB2:

- Dies gibt alle Rechte am Objekt mit `GRANT OPTION`, und erlaubt auch, das Objekt zu löschen.

  In Oracle und SQL Server gibt es kein `CONTROL`-Recht. Es entspricht den Rechten des Besitzers einer Tabelle.

- Wenn ein Benutzer eine Tabelle neu anlegt, hat er/sie das `CONTROL`-Recht für diese Tabelle.

  Bei Sichten bekommt der Benutzer das `CONTROL`-Recht nur, wenn er/sie es auch für die verwendeten Basistabellen und Sichten hat. Das `CONTROL`-Recht wird immer ohne `GRANT OPTION` vergeben. Es kann nur durch Administratoren vergeben werden. Um ein `CONTROL`-Recht vergeben zu können, muss man die `SYSADM` oder `DBADM`-Stufe ("Authority") haben. Das `CONTROL`-Recht hat Bedeutung für `REVOKE` (s.u.).

Zugriffsrechte entziehen (1)

- Vergebene Zugriffsrechte können wieder entzogen (zurückgenommen) werden mit einem Kommando, das dem `GRANT`-Befehl sehr ähnlich ist:

  `REVOKE ⟨Rights⟩ ON ⟨Object⟩ FROM ⟨Users⟩`

- `⟨Rights⟩`: Liste von einzelnen Rechten (z.B. `SELECT`), durch Kommata getrennt, oder "`ALL PRIVILEGES`".
- `⟨Object⟩`: Name eines DB-Objektes (z.B. Tabelle).
- `⟨Users⟩`: Liste von Nutzernamen (durch Komma getrennt) oder "`PUBLIC`".

Zugriffsrechte entziehen (2)

- Beispiel:

  `REVOKE INSERT ON AUFGABEN FROM MEIER`

  Wenn der Benutzer `MEIER` vor diesem Kommando `SELECT` und `INSERT`-Rechte an `AUFGABEN` hatte, hat er hinterher nur noch `SELECT`-Rechte.

- Benutzer können nur Rechte entziehen, die sie vorher vergeben haben.

  Deswegen speichert das DBMS in seinen internen Tabellen nicht nur das Tripel Benutzer-Recht-Objekt sondern das Quadrupel (A, P, O, B): Benutzer A hat das Recht P für Objekt O an Benutzer B vergeben.

Zugriffsrechte entziehen (3)

- Man kann ein an `PUBLIC` vergebenes Recht nicht selektiv einzelnen Nutzern entziehen.

  Man kann es natürlich zurücknehmen, aber dann verlieren es alle Nutzer. Da `PUBLIC` auch zukünftige Nutzer enthält, speichert die DB nur, dass es an `PUBLIC` vergeben wurde (nicht einzeln für jeden Nutzer).

- Man kann aber "`ALL PRIVILEGES`" vergeben, und die Rechte später selektiv zurücknehmen.

  "`ALL PRIVILEGES`" sind die Rechte, die der Benutzer aktuell hat. Sie werden jeweils einzeln in der Systemtabelle gespeichert.

- Wenn eine Tabelle gelöscht wird, werden auch alle dafür vergebenen Zugriffsrechte gelöscht.

  Wird sie dann neu angelegt, kann nur der Besitzer darauf zugreifen.

Zugriffsrechte entziehen (4)

- Hat A ein Recht "WITH GRANT OPTION" an B gegeben, und B es an C weitergegeben, und A entzieht das Recht dann B, so verliert es rekursiv auch C:

$$A \xcancel{\longrightarrow} B \longrightarrow C$$

  In SQL-92 benötigt dies allerdings CASCADE, siehe unten.

- Falls C das Recht allerdings zusätzlich auf einem anderen Pfad bekommen hat (z.B. direkt von A), behält C das Recht:

$$\begin{tikzcd} A \arrow[r, "\times" marking] \arrow[rr, bend right] & B \arrow[r] & C \end{tikzcd}$$

Zugriffsrechte entziehen (5)

- Das REVOKE Kommando soll die gleiche Situation wieder herstellen, als hätte B das Recht nie gehabt.

  Nur bezüglich der Zugriffsrechte: Wenn B das Recht genutzt hat, um Tabellen zu ändern, bleiben diese Änderungen natürlich bestehen.

- SQL-86 hatte kein REVOKE-Kommando.

  Das rekursive Entziehen ist nicht ganz einfach: Man muss Pfade im Graphen der einzelnen Weitergaben verfolgen. Insbesondere geht das nicht mit einer (klassischen) SQL-Anfrage. Man braucht eine deduktive Datenbank oder rekursive Sichten in SQL.

Zugriffsrechte entziehen (6)

Systemabhängige Details:

- In DB2 kann man nur mit CONTROL-Recht für ein Objekt X Rechte für X entziehen.

  Es ist etwas komisch, dass man mit der GRANT OPTION das Recht weitergeben kann, aber dann nicht mehr zurücknehmen.

- Auf diese Art braucht DB2 nicht darüber Buch zu führen, auf welchem Weg ein Nutzer das Recht bekommen hat.

  Wenn ein Nutzer mit CONTROL-Recht das Recht entzieht, ist es weg (außer ggf. über Gruppen/PUBLIC).

Zugriffsrechte entziehen (7)

Systemabhängige Details, Forts.:

- In SQL-92 und SQL Server muss man CASCADE zum Kommando hinzufügen, um Rechte rekursiv zu entziehen, z.B.

```
REVOKE INSERT ON AUFGABEN FROM MEIER CASCADE
```

  In SQL Server, muss CASCADE immer angegeben werden, wenn man ein Recht entziehen will, das WITH GRANT OPTION verliehen wurde. In SQL-92 muss man dies nur angeben, wenn der Nutzer das Recht tatsächlich weitergegeben hat. In SQL-92 kann man alternativ RESTRICT statt CASCADE angeben: Dann wird das REVOKE nur ausgeführt, wenn keine Rechte von anderen Nutzern davon abhängen. SQL Server versteht RESTRICT nicht. Oracle und DB2 kennen gar kein CASCADE/RESTRICT.

Zugriffsrechte entziehen (8)

Systemabhängige Details, Forts.:

- SQL-92 hat auch "REVOKE GRANT OPTION FOR ...".

  Dies wird nur von SQL Server verstanden, nicht von Oracle oder DB2. SQL Server verlangt, dass dann auch CASCADE angegeben wird.

- In Oracle muss man "CASCADE CONSTRAINTS" hinzufügen, wenn man ein REFERENCES-Recht zurücknehmen will, das genutzt wurde, um Fremdschlüssel anzulegen.
  - ◇ Die Fremdschlüssel-Integritätsbedingungen werden dann gelöscht.

Inhalt

1. Anforderungen
2. Deutsches Datenschutzrecht
3. Die SQL-Befehle GRANT und REVOKE
4. Oracle
5. DB2
6. SQL Server

Nutzer und DB-Schemata (1)

- In Oracle muss man "⟨Benutzer⟩.⟨Objekt⟩" schreiben, um auf DB Objekte (Tabellen, Sichten, etc.) von anderen Nutzern zuzugreifen, z.B.:

```
SELECT * FROM BRASS.AUFGABE
```

- Natürlich geht das nur, wenn der Nutzer BRASS das SELECT-Recht an der Tabelle AUFGABE vergeben hat.

  Entweder direkt an Sie (den Nutzer, der den Befehl eingibt) oder an PUBLIC (eine weitere Alternative wären Rollen/Gruppen, s.u.). Wenn ein Nutzer, der nicht einmal das SELECT-Recht für eine Tabelle hat, versucht, auf diese zuzugreifen, gibt Oracle die gleiche Fehlermeldung, wie wenn die Tabelle nicht existiert. So ist auch die Information über die Existenz der Tabelle geschützt.

Nutzer und DB-Schemata (2)

- In Oracle ist der Nutzer-Name gleichzeitig Schema-Name. D.h. jeder Nutzer hat genau ein Schema, und jedes Schema gehört genau einem Nutzer.
- Eine eindeutige Identifikation eines Objektes (Tabelle etc.) innerhalb der Datenbank besteht also aus Nutzer-Name und Objekt-Name.
- Wenn man in Oracle zwei Mengen von Tabellen trennen will (also zwei Schemata braucht), muss man zwei Benutzer-Accounts verwenden.

  Oracle unterstützt Schemata nicht unabhängig von Benutzerkonten.

Synonyme (1)

- Da die zweiteiligen Namen etwas umständlich sind, kann man in Oracle Abkürzungen definieren:

  `CREATE SYNONYM AUFGABE FOR BRASS.AUFGABE`

  Schreibt man nun "`AUFGABE`" (ohne vorangestellten Nutzer), wird dies intern durch "`BRASS.AUFGABE`" ersetzt.
- Dieses Synonym (Abkürzung, Macro) gilt nur für den Nutzer, der es definiert hat.
- Ein DBA kann aber auch "`PUBLIC`" Synonyme definieren. Dann sieht es so aus, als würde die Tabelle "`AUFGABE`" unter jedem Benutzer-Account existieren.

  Obwohl es tatsächlich nur eine Tabelle ist.

Synonyme (2)

- Solche "`PUBLIC`" Synonyme werden für den Systemkatalog ("Data Dictionary") genutzt.

  Z.B. ist `CAT` (Liste aller eigenen Tabellen) ein Synonym für die Sicht `SYS.USER_CATALOG` (dem Benutzer `SYS` gehört der Systemkatalog).
- Selbst wenn ein "`PUBLIC`" Synonym X definiert ist, kann ein Nutzer seine eigene Tabelle X anlegen.

  Dann ist X für diesen Nutzer seine eigene Tabelle. Er kann auf das Objekt aber noch mit "⟨`User`⟩.⟨`Table`⟩" zugreifen.
- Löschen von Synonymen: `DROP SYNONYM AUFGABE`
- Synonyme sind nicht Teil des SQL-92 Standards.

System-Rechte (1)

- Neben den oben beschriebenen Zugriffsrechten für Tabellen etc. ("object privileges") hat Oracle auch System-Rechte ("system privileges").
- Diese beziehen sich auf die Ausführung bestimmter Kommandos, nicht auf Datenbank-Objekte.
- Z.B. ist das System-Recht "`CREATE TABLE`" nötig, um dieses Kommando ausführen zu können.

  Ein Nutzer, der nur Daten eingeben soll, braucht keine Tabellen anlegen zu können. Für ein sicheres System sollte jeder Nutzer nur die Kommandos ausführen konnen, die er für seine Aufgabe sinnvoll benötigen könnte. Mit Objektrechten alleine könnte die Nutzung von `CREATE TABLE` nicht eingeschränkt werden.

System-Rechte (2)

- Um sich bei Oracle anmelden zu können, braucht man das System-Recht "`CREATE SESSION`".

  Ein Benutzerkonto kann gesperrt werden, indem dieses Recht entzogen wird. Die Tabellen etc. des entsprechenden Schema bleiben erhalten, und können von anderen Nutzern (mit entsprechenden Zugriffsrechten) genutzt werden.
- Viele System-Rechte sind nur für DBAs gedacht, z.B. gibt "`SELECT ANY TABLE`" Lesezugriff für alle Tabellen in der Datenbank.

  Auch wenn man das `SELECT`-Objekt-Recht für die Tabelle nicht hat.

System-Rechte (3)

- Da die DBA-Rechte in viele verschiedene System-Rechte aufgeteilt sind, kann man mehrere DBAs mit unterschiedlichen Aufgabenbereichen haben.

  Z.B. muss ein Mitarbeiter, der für die Backups zuständig ist, nicht unbedingt das "`CREATE USER`" Recht haben, um auch neue Benutzer anlegen zu können. Selbstverständlich erlaubt das Modell aber auch, einen DBA mit allen System-Rechten zu haben.

- Es gibt mehr als 90 verschiedene System-Rechte.

  Im wesentlichen gibt es für jedes Administrations-Kommando ein entsprechendes System-Recht. Verschiedene Arten von `CREATE`-Befehlen sind auch System-Rechte (da sie sonst nicht eingeschränkt werden könnten). Die meisten Befehle haben eine `ANY`-Version als System-Recht (erlaubt das Kommando auf Objekte aller Nutzer anzuwenden).

System-Rechte (4)

- Besitzt ein Nutzer ein System-Recht "`WITH ADMIN OPTION`" hat, kann er/sie es weitergeben:

  ```
  GRANT CREATE TABLE TO SCOTT
  ```

  Fügt man "`WITH ADMIN OPTION`" hinzu, kann anschließend auch `SCOTT` das "`CREATE TABLE`"-Recht vergeben.

- Wenn ein Systemrecht einem Nutzer A entzogen wird, der es "`WITH ADMIN OPTION`" hatte, wird das Recht nicht rekursiv Nutzern B entzogen, die es von A bekommen haben.

  Vermutlich heißt es deswegen nicht "`GRANT OPTION`". Aber es ist sehr ähnlich ("`GRANT OPTION`" gibt es nur für Objekt-Rechte).

Roles (1)

- It is difficult to grant privileges to many users one by one. In one way or another, all modern DBMS support groups of users with similar privileges.
- Oracle has the concept of "roles", which are sets of privileges that can be granted to users:

  ```
  CREATE ROLE ⟨Name⟩
  ```

- Only those with the system privilege "`CREATE ROLE`" can execute this command (e.g. only the DBA).

Roles (2)

- Access rights are granted to a role (e.g. "`STAFF`") in the same way as they are granted to a user:

  ```
  GRANT SELECT ON COURSES TO STAFF;
  ```

- Granting access rights to a role is treated in the same way as granting it to specific users (DBA rights are not needed).
- Roles can be granted to users:

  ```
  GRANT STAFF TO JIM, MARY
  ```

Roles (3)

- When a user A is granted a role R, A receives all privileges that were or will be granted to R.

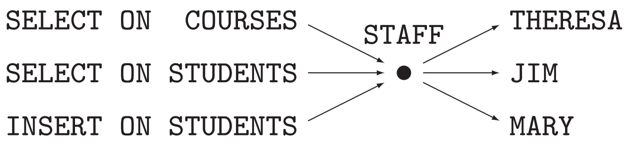
  But if "STAFF" is not one of the default roles of these users, which are automatically activated when they log in, they must explicity execute "SET ROLE STAFF" in every session in which they want to use these privileges. (It seems that this is not enforced in Oracle 8.0.)

- Only the owner of the role or a user who has received it with admin option can grant the role to another user.

  `GRANT STAFF TO THERESA WITH ADMIN OPTION`

Roles (4)

- Roles are a level of indirection between privileges and users, intended to simplify the administration of a group of users with the same privileges:



- When further privileges are granted to "STAFF", these become automatically available to THERESA, JIM, and MARY.

Roles (5)

- A role can be granted to another role, e.g.

  `GRANT STAFF TO REG_OFFICE`

- Then users with the role "REG_OFFICE" also have all the privileges granted to "STAFF".

  I.e. "REG_OFFICE" is more powerful, it implies staff.

- Roles can be protected by passwords. Then the SET ROLE command requires a password.

Roles (6)

- Several roles are predefined in Oracle 8, e.g.
  - ◇ CONNECT: Basic usage rights.

    This corresponds to the system privileges: CREATE SESSION, ALTER SESSION, CREATE DATABASE LINK, CREATE SYNONYM, CREATE TABLE, CREATE CLUSTER, CREATE VIEW, CREATE SEQUENCE.
  - ◇ RESOURCE: Rights for advanced users.

    This includes e.g. CREATE TABLE, CREATE PROCEDURE, CREATE TRIGGER. Students in this course were granted CONNECT and RESOURCE (but UNLIMITED TABLESPACE was revoked).
  - ◇ DBA: Right to do everything.
- In older Oracle versions, users were classified into these three types.

Creating Users (1)

User Authentication:

- Oracle can perform the user authentication itself. One must specify a user name and a password:

```
CREATE USER BRASS IDENTIFIED BY ABC_78
```

Passwords have the same syntax as table names: They are not case-sensitive and "..." is needed to include special characters.

- Oracle can also rely on the authentication done by the operating system or a network service:

```
CREATE USER OPS$BRASS IDENTIFIED EXTERNALLY
```

So when the UNIX user `BRASS` logs into Oracle (with empty username/password), he becomes the Oracle user `OPS$BRASS`.

Creating Users (2)

- A user created as explained above has no rights, not even the system privilege to connect to the DB.
- The necessary privileges can be given e.g. with:

```
GRANT CONNECT, RESOURCE TO BRASS
```

- After the `GRANT`, these roles can be made default roles, so that they are automatically activated when the user logs in:

```
ALTER USER BRASS DEFAULT ROLE ALL
```

It seems that roles without a password automatically become default roles (?). So this command might not be necessary.

Tablespaces and Quotas (1)

- A tablespace is a database file or a collection of DB files (storage space, container for tables).
- All tablespaces are listed in the system catalog table `DBA_TABLESPACES`.

  E.g. use "`SELECT TABLESPACE_NAME FROM DBA_TABLESPACES`" to list all tablespaces. This query must be executed by a DBA. All users have read access to `USER_TABLESPACES` (tablespaces that are accessible by the current user). The files for each tablespace are listed in `DBA_DATA_FILES`. It has e.g. the columns `FILE_NAME`, `FILE_ID`, `TABLESPACE_NAME`, `BYTES`. See also `DBA_FREE_SPACE`/`USER_FREE_SPACE` and `DBA_FREE_SPACE_COALESCED`.

- The tablespace "`SYSTEM`" contains e.g. the data dictionary (collection of system tables).

Tablespaces and Quotas (2)

-
```
CREATE USER BRASS IDENTIFIED BY MY_PASSWORD
            DEFAULT TABLESPACE USER_DATA
            TEMPORARY TABLESPACE TEMPORARY_DATA
            QUOTA 2M ON USER_DATA
            QUOTA UNLIMITED ON TEMPORARY_DATA
```
- A tablespace can be defined when a table is created.

  Otherwise it is stored in the user's `DEFAULT TABLESPACE` (which is `SYSTEM` if it is not set in the `CREATE USER`).

- Without quota (and "`UNLIMITED TABLESPACE`"), the user cannot create tables on the tablespace.

  Use: `REVOKE UNLIMITED TABLESPACE FROM BRASS`

Changing and Deleting Users

- If a user has forgotten his/her password:

  `ALTER USER BRASS IDENTIFIED BY NEW_PASSWORD`

- A user without tables can be deleted in this way:

  `DROP USER BRASS`

- To delete the user including all his/her data, use:

  `DROP USER BRASS CASCADE`

- The following command ensures that the user can no longer log in, but leaves his/her data untouched:

  `ALTER USER BRASS ACCOUNT LOCK`

Some Predefined Users (1)

- `SYS`: Owner of the system tables (data dictionary).

  Most powerful account. Default password: `CHANGE_ON_INSTALL`.

- `SYSTEM`: The default database administrator.

  For most administration tasks. Default password: `MANAGER`.

- `SCOTT`: Guest and demonstration account.

  Default password: `TIGER`. Sometimes there are additional accounts used in tutorials: `ADAMS`, `BLAKE`, `CLARK`, `JONES`.

- `OUTLN`: Schema contains information for optimizer.

  Default password: `OUTLN`.

Some Predefined Users (2)

- `DBSNMP`: Information for the "intelligent agent".

  It is used for remote administration via the "enterprise manager". Default password: `DBSNMP`.

- One should check the list of users in the system table `ALL_USERS` and lock all users that are currently not needed (or change their passwords).

  There is also a table `DBA_USERS` with more information. The list of users created during the installation can change with new versions. Also, when one installs additional software (e.g. the Oracle application manager), more accounts are created.

- Hackers know all the default passwords!

External Password File

- Whereas the above passwords are stored in the database (encrypted), there usually is an additional file that contains passwords of administrators who need e.g. to start up the database.

  When the database is not running, passwords stored in the database cannot be accessed. If you use `CONNECT INTERNAL` in the server manager (`svrmgrl`) or `CONNECT SYS AS SYSDBA`, the default password is `ORACLE`. Actually, the `SYS` password in the password file and in the database can be different. The password file is generated by the `orapwd` utility program. Later, every user granted `SYSDBA`/`SYSOPER` rights is also stored in the password file. Instead of using a password file, you can use OS authentication. This depends on the parameter `REMOTE_LOGIN_PASSWORDFILE`.

Other Security Features (1)

- The resource usage of DB users can be restricted by creating a "profile" for them. This defines e.g.
  - ◇ How many concurrent sessions the user can have (number of windows with DB applications).
  - ◇ After what idle time he/she is logged off.
  - ◇ How much CPU time and how many logical reads (disk accesses) is allowed per session/per call.
  - ◇ After what time a password must be changed.
  - ◇ Which function is used to check the password complexity.

Other Security Features (2)

- Oracle also has an `AUDIT` command for defining which user actions are logged in system tables, so that one can later find out who did what.
  - ◇ E.g. all insertions should be logged that were executed (not refused):

```
AUDIT INSERT ON SCOTT.EMP
BY SESSION WHENEVER SUCCESSFUL;
```

"`BY SESSION`" means that only one record is written for an entire session that did this operation (default). Alternative: "`BY ACCESS`".

  - ◇ E.g. log all unsuccessful login attempts:

```
AUDIT CONNECT WHENEVER NOT SUCCESSFUL;
```


Overview

1. Requirements
2. German Data Privacy Law
3. GRANT and REVOKE in SQL
4. Oracle
5. DB2
6. SQL Server

Users in DB2 (1)

- DB2 uses the authentication mechanism of the underlying operating system (DB2 itself does not manage passwords).
- DB2 has something similar to the "system privileges" of Oracle, they are called "Database Authorities" and "Instance-Level Authorities" in DB2.
- A DB2 instance can manage several databases.

  Instance: server process. Database: Collection of DB schemas (and their table data). E.g. use "`CONNECT TO SAMPLE`" (a specific DB) after logging into DB2.

Users in DB2 (2)

- An operating system user can connect to a database if he/she has the "CONNECT" authority.
- E.g. an administrator can "create" a user in the database by issuing this command:

```
GRANT CONNECT ON DATABASE TO BRASS
```

- This right can also be granted to "PUBLIC", in which case every OS user can connect to the database.

Database Authorities in DB2

- CONNECT: Right to log into the database.
- CREATETAB: Right to create tables.

  No right is required for view creation.
- BINDADD: Right to create packages.

  I.e. to compile application programs with embedded SQL.
- IMPLICIT_SCHEMA: Create tables in new schemas.
- CREATE_NOT_FENCED: Extend DBMS (add functions).
- DBADM: Access and modify all DB objects, grant any object privilege or DB authority except DBADM.

Groups in DB2

- DB2 has no roles, but it understands the concept of groups of the underlying operating system.

  E.g. UNIX and Windows NT have groups of users.
- Privileges can be granted not only to users, but also to groups.

  Privileges granted to groups are not used when binding a package.
- So there are three ways a user might get a privilege:
  - It can be granted to this user personally,
  - to a group in which he/she is member,
  - or to PUBLIC.

Instance-Level Authorities

- The most powerful right in DB2 is the SYSADM authority.

  The account under which DB2 is installed plus all members of its group get SYSADM authority. It cannot be granted or revoked, but the group membership can be changed in the OS.
- There are also two other groups with fewer rights:
  - SYSCTRL: Can e.g. create or delete databases and tablespaces (plus all SYSMAINT privileges).
  - SYSMAINT: Can e.g. start or stop the database, do backups and restore the database after a crash.

Overview

1. Requirements
2. German Data Privacy Law
3. GRANT and REVOKE in SQL
4. Oracle
5. DB2
6. SQL Server

Creating New Users (1)

- SQL Server has two authentication mechanisms:
  - ◇ Windows NT Users or Groups can be mapped to SQL server logins.

    Then Windows NT does the authentication, there is no need to enter again a password. Users from trusted clients (which must run Windows NT) can also be mapped to SQL Server logins.
  - ◇ SQL Server Authentication (with password).

    In this case, SQL Server requires login name and password. This is the only possibility if SQL Server runs on Windows 95/98.
- One SQL Server Instance can manage several databases. The databases can have different users.

Creating New Users (2)

- It is necessary to distinguish between a login into the server, and the user in a specific database.

  Each login has a default DB to which it will be connected. The username in a DB may be different from the server login, and in different DBs different usernames can be used.
- Logins / users can be created with the Enterprise Manager (graphical interface).

  Alternatively, one can use the system procedures `sp_addlogin` (SQL Server authentication) and `sp_grantlogin` (Windows NT authentication) to create a login and `sp_grantdbaccess` to allow the login to access a specific database (also needed for the default DB of the login).

Special Users

- There is a login "`sa`" (system/server administrator). Everything can be done under this login.

  It uses SQL Server authentication (no password by default!).
- Every database has a user "`dbo`" (DB owner).

  Any member of the `sysadmin` sever role (e.g. "`sa`") is mapped to this user when he/she connects to a database.
  If a table is referenced without specifying a user, SQL Server first tries to find it in the account/schema of the current user, and then in the account/schema of "`dbo`". This eliminates the need for synonyms as used in Oracle.
- Some databases may have a "`guest`" user.

  A server login not mapped to a db user is mapped to `guest`.

Roles

- SQL Server has the concept of roles (user groups) which seems to be basically the same as in Oracle.

  In addition SQL Server also uses Windows NT groups.

- However, some roles are special and contain administration rights which cannot explicitly be granted.
- Fixed server roles allow administration of the server. The most powerful is “`sysadmin`” (e.g. “`sa`”).
- Fixed database roles allow administration of a DB. The most powerful one is “`db_owner`” (e.g. “`dbo`”).

System Privileges

- In addition, SQL Server has system privileges basically as in Oracle (managed via `GRANT/REVOKE`):
  - ◇ `CREATE DATABASE`
  - ◇ `CREATE DEFAULT`
  - ◇ `CREATE PROCEDURE`
  - ◇ `CREATE RULE`
  - ◇ `CREATE TABLE`
  - ◇ `CREATE VIEW`
  - ◇ `BACKUP DATABASE`
  - ◇ `BACKUP LOG`

Fixed Server Roles

- `sysadmin`: Perform any activity in SQL Server.
- `securityadmin`: Manage logins.
- `serveradmin`: Configure server-wide settings.
- `setupadmin`: Execute some system stored procedures, e.g. `sp_serveroption`.
- `processadmin`: Kill processes of other users.
- `diskadmin`: Manage the disk files.
- `dbcreator`: Create and alter databases.

Fixed Database Roles

- `db_owner`: Perform any activity in the database.
- `db_accessadmin`: Manage users of the database.
- `db_securityadmin`: Manage roles and permissions.
- `db_ddladmin`: Create, modify, drop any DB object.
- `db_backupoperator`: Make a backup copy of the DB.
- `db_datareader`: Read all tables in the database.
- `db_datawriter`: Modify all tables in the database.
- `denydatareader`, `denydatawriter`: These roles forbids to read/modify any table in the database.

DENY Statement

- SQL Server has a statement which is the opposite of `GRANT`, e.g.:

  `DENY SELECT ON COURSES TO BRASS`

- The idea is that `BRASS` might be member of a group or role, which has the right, and you might want to declare `BRASS` as an exception.

  In an older version of SQL Server, `REVOKE` worked like `DENY` now. When an SQL92-conforming `REVOKE` was introduced, the old was called `DENY`.

- So the `DENY` on the user level takes precedence over the `GRANT` on the group/role/public level.