فهرست

امنیت

۱,۱۷ مقدمه

معمولاً امنیت داده را همان جامعیت داده می‌پندارند. ولی در اصل این دو مقوله کاملاً متفاوت هستند. امنیت، با حفاظت داده‌ها در مقابل دسترسی غیرمجاز سروکار دارد، در حالی که جامعیت با درستی داده‌ها سروکار دارد. این تفاوت را می‌توان در دو جمله نشان داد:

- امنیت به معنی حفاظت داده در مقابل کاربران غیر مجاز است.
- جامعیت به معنای حفاظت داده‌ها در مقابل کاربران مجاز است (!).

به عبارت دیگر، امنیت یعنی اطمینان از اینکه کاربران مجاز به انجام آنچه را که قصد انجام آن‌را دارند، هستند. در حالی که جامعیت یعنی حصول اطمینان از اینکه کارهایی که کاربران (مجاز) انجام می‌دهند صحیح است و درستی و صحت داده‌ها را خدشه دار نمی‌کند.

البته شباهت‌هایی هم بین آن‌ها وجود دارد، در هر دو مورد سیستم مدیریت پایگاه داده باید از وجود پاره‌ای قواعد و محدودیت‌ها[1] که کاربران نباید نقض[2] کنند، باخبر باشند. در هر دو مورد، این قواعد و محدودیت‌ها باید به طور اعلانی[3] توسط یک زبان مناسب توصیف شوند و در کاتالوگ سیستم نگهداری شوند. و دست آخر اینکه در هر دو مورد سیستم مدیریت پایگاه داده باید روی عملیات کاربران نظارت

---

[1] Constraints
[2] violate
[3] declaratively

داشته باشد تا مطمئن شود که قواعد و محدودیت‌ها اعمال شده‌اند. در این فصل به موضوع امنیت می‌پردازیم. توجه : علت اصلی جدا نمودن این دو مبحث این است که جامعیت یک موضوع اساسی است (در صورت نقض جامعیت، دیگر پایگاه داده‌ها درست نخواهد بود) در حالی که موضوع امنیت در درجه دوم مطرح می‌گردد گرچه امنیت در مقوله‌هایی مثل دست‌یابی به اینترنت، تجارت الکترونیکی اهمیت ویژه‌ای دارد.

مسئله امنیت جنبه‌های مختلفی دارد که در اینجا به برخی از آن‌ها می‌پردازیم.

- جنبه‌های قانونی، اجتماعی و اخلاقی (برای مثال آیا شخصی که میزان اعتبار یک مشتری را پرسیده مجاز است چنین درخواستی را انجام دهد؟)
- کنترل‌های فیزیکی (برای مثال آیا کامپیوتر یا اتاق پایانه قفل است یا به شکل دیگر محافظت می‌شود؟)
- پرسش‌های سیاستی (برای مثال تصمیم گیری مدیران برای اینکه چه کسی به چه چیزی دسترسی داشته باشد).
- مسائل عملیاتی (برای مثال اگر از کلمه عبور استفاده می‌گردد، به چه صورت کلمات عبور حفظ می‌شوند، و هر چند وقت یک‌بار تغییر می‌کنند؟)
- کنترل‌های سخت افزاری (برای مثال، آیا سرویس دهنده ویژگی‌های امنیتی، مثل کلیدهای محافظت حافظه یا حالت عملیات حفاظت شده را فراهم می‌سازد؟)
- پشتیبانی سیستم عامل (برای مثال، آیا سیستم عامل محتویات حافظه اصلی و فایل‌های دیسک را پس از اتمام کار با آن‌ها پاک می‌کند یا

خیر؟ همچنین در مورد فایل کارنامه ترمیم[1] نیز همین کار را انجام می‌دهد؟)

و سرانجام

- نکاتی که به خود سیستم مدیریت پایگاه داده‌ها بازمی گردد (برای مثال، آیا سیستم مدیریت پایگاه داده‌ها دارای مفهوم تملک داده‌ها[2] است؟)

بنا به دلایل روشن، در این فصل بیشتر توجه خود را تنها معطوف به موضوعات دسته آخر می‌کنیم.

در حال حاضر، سیستم‌های مدیریت پایگاه داده‌های مدرن از یک یا دو روش امنیت داده‌ها پشتیبانی می‌کنند، این دو روش عبارتند از کنترل محتاطانه۳ (یا روش اختیاری DAC) و کنترل اجباری٤ (روش اجباری). در هر دو مورد، واحد داده یا شئ داده۵ که نیاز به محافظت دارد، می‌تواند محدوده‌ای از کل پایگاه داده و تا یک جزء خاص از یک تاپل خاص باشد. تفاوت این دو روش مختصراً به شرح زیر است.

- در روش اختیاری یا کنترل محتاطانه، کاربر معمولاً حقوق دست‌یابی (یا امتیازات[6]) متفاوتی روی اشیاء مختلف دارد. به علاوه، چندین محدودیت وراثتی وجود دارد، که مشخص می‌کند کدام کاربران چه مجوزهایی را روی کدام اشیاء دارند (برای مثال، کاربر U_1 ممکن است بتواند شئ A را ببیند اما نتواند شئ B را ببیند، در حالی که کاربر U_2 ممکن است قادر باشد شئ B را ببیند اما نتواند شئ A را ببیند) بنابراین طرح‌های محتاطانه بسیار انعطاف پذیر هستند.

---

[1] Recovery log
[2] Data ownership
[3] Discretionary control
[4] Mandatory control
[5] Data object
[6] privilege

- در مقایسه در روش اجباری یا کنترل اجباری، هرشی داده با یک سطح رده بندی[1] مشخص برچسب گذاری می‌شود و به هر کاربر یک سطح مجوز[2] داده می‌شود ، سپس یک شئ داده می‌تواند توسط کاربرانی که مجوز مناسب دارند، مورد دست‌یابی قرار گیرد. سیاست‌های امنیتی طرح‌های اجباری ذاتاً ارثی است و بنابراین بسیار دقیق هستند (اگر کاربر U_1 بتواند شئ A را ببیند ولی نتواند شئ B را ببیند، آنگاه سطح محرمانه شئ B بالاتر از شئ A است و در نتیجه کاربر U_2 نمی‌تواند شئ B را ببیند ولی شئ A را نبیند).

روش اختیاری (شماهای محتاطانه) در بخش ۱۷,۲ و روش اجباری (شماهای اجباری) را در بخش ۱۷,۳ بحث می‌کنیم.

حال فارغ از اینکه با کدام روش اجباری یا اختیاری سروکار داریم، تمامی تصمیمات مانند اینکه کدامیک از کاربران مجازند که چه عمل‌هایی را روی چه شئ های داده انجام دهند، تصمیمات سیاستی هستند نه تصمیمات تکنیکی. بنابراین این سری تصمیمات خارج از بحث سیستم‌های مدیریت پایگاه داده می‌باشند. سیستم‌های مدیریت پایگاه داده تنها می‌توانند این چنین تصمیمات را اعمال کنند. پیرو این موضوع:

- نتایج این سیاست‌های امنیتی (تصمیمات) باید (الف) برای سیستم معلوم باشد (این کار با تعریف یکسری قواعد و محدودیت‌های امنیتی توسط زبانی مناسب انجام می‌پذیرد) و (ب) باید توسط سیستم به یاد آورده شود (این کار با ذخیره محدودیت‌ها در کاتالوگ سیستم انجام می‌پذیرد).
- باید ابزارهایی برای بررسی درخواست‌های دست‌یابی در مقابل محدودیت‌ها و قواعد قابل اعمال در کاتالوگ سیستم وجود داشته

---

[1] Classification level
[2] Clearance level

باشـــد (منظـــور از «درخواســـت دسترســـی»، ترکیبـــی از عملیـــات درخواستی، به اضافه شئ درخواستی بـه اضـافه کـاربر درخواسـت کننده است). این بررسی توسط زیر سیستم امنیتـی سیسـتم مـدیریت پایگاه داده انجام می‌پذیرد.

- برای اینکه تعیین شـود کـه کـدام محـدودیت بـه یـک **درخواسـت دست‌یابی** قابل اعمال است، سیستم باید بتوانـد منبـع درخواسـت را تشخیص دهد. یعنی اینکه باید کاربر درخواست کننده تشخیص داده شود، به همین علت وقتی کاربران وارد سیستم می‌شوند بایـد شناسـه (برای گفتن اینکه چه کسی هستند) و رمز عبور (برای اثبـات ادعـای خویش) خود را وارد نمایند. رمـز عبـور را تنهـا سیسـتم و کـاربران قانونی (کاربرانی که در سیستم شناسه دارند) می‌دانند. فرایند بررسـی رمز عبور (یعنی فرایند شناسایی کاربر **احراز هویت**[1] نام دارد. توجه: یادآوری می‌شود که روش‌های احراز هویت دیگری نیـز وجـود دارد که به مراتب از روش بررسی رمز عبور بهتر و حرفه‌ای‌تر هستند، این روش‌ها شامل دستگاه‌های بیومتریک می‌باشند: دستگاه‌های تشـخیص اثر انگشت، دستگاه‌های تشـخیص الگـوی شـبکه چشـم، تشـخیص صدا، دستگاه‌های تشـخیص دهنـده امضـا و غیـره. ایـن دسـتگاه‌هـا می‌توانند برای تشخیص ویژگی‌هایی به کـار رونـد کـه توسـط هـیچ کسی قابل ربودن نیست)

توجه کنید که در مـورد شناسـه‌هـای کـاربران، هـر تعـداد از کـاربران متمـایز می‌توانند شناسه مشترکی داشته باشند. به این ترتیب سیستم می‌تواند **واسـط کـاربری**[2] (که می‌تواند به عنوان **نقش‌ها**[3] نیز شناخته شود) را پشتیبانی کند. بنابراین می‌توان راهـی

---

[1] authentication
[2] User groups
[3] rules

را فراهم نمود که مثلاً افرادی که در یک اداره هستند، بر روی اشیاء یکسان، مجوزهـای یکسانی داشته باشند. عملیاتی از قبیـل حـذف و اضـافه کـردن بـر روی ایـن گـروه‌هـا می‌تواند مستقل از مجوزهای آن گروه بر روی اشیاء داده صورت پذیرد. توجه: توانـایی رده بندی کاربران به صورت سلسله مراتبی، ابزاری قدرتمند را فراهم می‌نمایـد کـه بـه وسیله آن‌ها می‌توان هزاران کاربر و شئ را مدیریت نمـود. امـا، توجـه کنیـد کـه مکـان ذخیره رکوردهای امنیتی که در انها مشخص می‌شود کدام کـاربر در کـدام گـروه قـرار دارد، کاتالوگ سیستم است، و خود این رکوردها باید تحت کنترل امنیتی مناسـب قـرار بگیرند.

۲,۱۷ کنترل دست‌یابی اختیاری

همان‌طور که در بخش پیش نیز گفته شد، اغلب سیستم‌هـای مـدیریت پایگـاه داده‌ها (DBMS) از روش کنترل اختیاری یا اجباری و یا هر دو استفاده مـی‌کننـد. در واقع می‌توان گفت که بسیاری از سیستم‌های مـدیریت پایگـاه داده از روش اختیـاری و برخی از روش اجباری اسـتفاده مـی‌کننـد. بنـابراین در عمـل بیشـتر از روش اختیـاری استفاده می‌شود و از این رو ابتدا این روش را مورد بررسی قرار می‌دهیم.

همـان‌طـور کـه قـبلاً متـذکر شـدیم، بـه زبـانی نیازمنـدیم تـا بـه وسـیله آن محدودیت‌های امنیتی (محدوده اختیارات) را تعریف کنیم. در عمل تعریف کارهایی که اجازه داریم انجام دهیم ساده‌تر از تعریف کارهایی است که اجازه نـداریم انجـام دهـیم. بنابراین، زبان‌های امنیتی تعریف مجوزها را بـه جـای تعریـف محـدودیت‌هـا و قواعـد امنیتی پشتیبانی می‌کنند که در واقع برعکس محدودیت‌های امنیتـی اسـت (اگـر چیـزی مجاز باشد، محدودیت نیست). بنابراین، ابتدا یک زبان فرضی را برای تعریف مجوزهـا تعریف می‌کنیم. مثال ساده‌ای در اینجا آمده است:

```
AUTORITY SA3
      GRANT RETRIEVE {S#, SNAME, CITY},
DELETE
```

```
ON S
TO Jim, Fred, Mary;
```

همان‌طور که در این مثال ملاحظه می‌کنید، هر مجوز چهار جزء دارد:

- نام مجوز (در این مثال SA3)
- مجموعه‌ای از امتیازات که به وسیله فراکرد[1] GRANT مشخص می‌گردد.
- حیطه اعمال مجوز که توسط فراکرد ON مشخص می‌گردد.
- مجموعه‌ای از کاربران که امتیازات خاصی را روی حیطه‌های معینی کسب می‌کنند، توسط فراکرد TO مشخص می‌گردند.

نحو[2] کلی به شکل زیر است:

```
AUTHORITY <authority name>
    GRANT < privilege comma list>
    ON <relvar name>
    TO < user ID comma list>;
```

توضیح: نام قاعده امنیتی <authority name>و مجموعه کاربران < user ID comma list> واضح می‌باشند (فقط ALL به معنی شناسه تمام کاربران شناخته شده است). هر امتیاز < privilege > یکی از موارد زیر است:

```
RETRIEVE [{<attribute name comma list>}]
INSERT [{<attribute name comma list>}]
DELETE
UPDATE [{<attribute name comma list>}]
ALL
```

بازیابی بدون قید و شرط RETRIEVE، درج بدون قید و شرط INSERT، حذف DELETE و به هنگام سازی بدون قید و شرط UPDATE مشخص هستند و لازم نیست که توضیحی داده شود. اگر در جلوی دستور RETRIEVE نام یکسری از صفات لیست شود آن گاه امتیازی که در مجوز امنیتی مشخص می‌گردد تنها بر روی این صفات اعمال می‌گردد (شخص تنها اجازه بازیابی این صفات را دارد). به طور

---

[1] clause
[2] syntax

مشابه این لیست صفات برای INSERT و UPDATE نیز وجـود دارد. مشخصـه ALL خلاصه تمام امتیازات می‌باشد: یعنی هم امتیاز بازیابی (تمـام صـفات)، هـم امتیـاز درج (تمام صفات)، هم امتیاز حذف و هم به هنگام سازی (نیز تمام صـفات). توجـه: بـرای سادگی از این پرسش که آیا برای انجام انتساب‌هـای رابطـه‌ای نیـاز بـه امتیـاز خاصـی است؟، صرف‌نظر می‌کنیم. همچنین توجه خود را به عملیات دست‌کـاری داده معطـوف می‌کنیم. اما در عمل عملیات دیگری مانند تعریف و حذف حیطه‌هـا عمـل و همچنـین عملیات تعریف و حذف امتیازات، وجود دارد که نیاز به مجاز شماری دارند.

اگر کاربری سعی کند عملیاتی را روی شئ داده‌ای انجام دهد کـه مجـوز ایـن کار را ندارد، چه اتفاقی می‌افتد؟ ساده‌ترین گزینه این است که از این کار جلـوگیری بـه عمل آید (و البته، پیام خطای مناسبی صادر شود). این کار در عمل بسیار متداول اسـت و بدین خاطر می‌توان آن را پیش فرض در نظر گرفـت. در مواقـع حسـاس‌تـر، ممکـن است فعالیت‌های دیگری مناسب‌تر باشد. برای مثال، می‌توان برنامـه را خاتمـه داد و یـا صفحه کلید کاربر را قفل نمود. همچنین ممکن است این تلاش‌ها در یک فایل ثبت بنام نظارت بر خطر، ثبت گردند تا متعاقباً برای تحلیل نقض‌هـای امنیتـی و بـه عنـوان یـک بازدارنده نفوذهای غیر قانونی، مورد استفاده قرار بگیرند.

البته به روشی برای حذف مجوزها نیاز داریم.

DROP AUTHORITY <authority name>;

برای سادگی، فرض می‌کنیم با حذف یک متغیر رابطه[1] ای، بـه طـور خودکـار تمام مجوزها روی آن متغیر رابطه نیز حذف خواهد شد.

در اینجا مثال‌های دیگری از مجوزها آمده است که نیاز به توضیح ندارد.

1. AUTHORITY EX1
GRANT RETRIEVE {P#, PNAME, WEIGHT}
ON P
TO Jacques, Anne, Charley;

---

[1] relvar

کاربران Jacques , Anne , Charley می‌توانند یک زیرمجموعه عمودی از متغیر رابطه‌ای پایه P را ببینند. در واقع این کاربران تنها می‌توانند تعدادی از فیلدهای جدول مبنای P را بازیابی کنند. بنابراین این مثال، مثالی از مجوز **مستقل از مقدار**[1] می‌باشد.

```
2. VAR LS VIEW
       (P WHERE CITY = 'London') {SNAME, STATUS}
   AUTHORITY EX2
       GRANT RETRIEVE, DELETE, UPDATE {SNAME,
STATUS}
       ON LS
       TO Dan, Misha;
```

متغیر رابطه‌ای LSدر واقع یک دید بر روی یک جدول مبنا S است که «تهیه کنندگان لندن» را مشخص می‌کند. بنابراین کاربران Dan , Misha تنها می‌توانند یک زیر مجموعه افقی از رابطه مبنای S را ببینند. این مثال، یک مثال از مجوز **وابسته به مقدار**[2] می‌باشد. همچنین نکته آن که، درست است که کاربران Dan , Misha می‌توانند تاپل های مشخصی از رابطه عرضه کنندگان (S) را حذف کنند (از طریق دید LS) اما آن‌ها نمی‌توانند درج انجام دهند و همچنین نمی‌توانند صفات S# و CITY را به هنگام (UPDATE) سازند.

```
3. VAR SSPPO VIEW
       (S JOIN SP JOIN) P WHERE CITY = 'Oslo' {p#}
                                   {ALL BUT P#, QTY};
  AUTHORITY EX3
       GRANT RETRIEVE
       ON SSPPO
       TO Lars;
```

این مثال، نمونه‌ای دیگر از مثال وابسته به مقدار است. کاربر Lars می‌تواند اطلاعات عرضه کننده‌ای را بازیابی کند که قطعاتی را در'Oslo' تولید کرده باشد.

---

[1] Value-independent
[2] Value=dependent

```
4. VAR SSQ VIEW
       (SUMMARIZE SP PER S {S#} ADD SUM) QTY AS SQ;
  AUTHORITY EX4
       GRANT RETRIEVE
       ON SSQ
       TO Fidel;
```

کاربر Fidel می‌توانند کل کمیت‌های حمل را به ازای هر عرضه کننده ببیند، اما نمی‌تواند هر یک از کمیت‌های حمل را ببیند. بنابراین، کاربر Fidel خلاصه‌ی آماری داده‌های پایه مورد نظر را می‌بیند.

```
5. AUTHORITY EX5
       GRANT RETRIEVE, UPDATE {STATUS}
       ON S
       WHEN DAY () IN {' Mon', 'Tue', 'Wed', 'Thu', 'Fri'}
       AND NOW () ≥ TIME '09:00:00'
       AND NOW () ≤ TIME '17:00:00'
       TO ACCOUNTING;
```

در اینجا نحو تعریف قاعده امنیتی را گسترش می‌دهیم و فراکرد جدیدی را برای تعریف این قاعده معرفی می‌کنیم. فراکرد when، که به وسیله آن می‌توان کنترل‌های متن[1] ویژه‌ای را مشخص نمود. همچنین در اینجا فرض می‌کنیم که سیستم مدیریت پایگاه داده‌ها دو عملگر بدون عمل وند (دو تابع) را فراهم می‌سازد. این دو عملگر (تابع) DAY() و NOW() هستند که تفسیر روشنی دارند. مجوز EX5 تضمین می‌کند که مقادیر وضعیت عرضه کنندگان می‌توانند توسط کاربر ACCOUNTING(یعنی هر کاربری که در بخش حسابداری است)، تنها در روزهای و ساعت‌های کاری هفته، تغییر کند. بنابراین، این مثال، نمونه‌ای از مجوز وابسته به متن است زیرا یک درخواست دست‌یابی بسته به متن ممکن است اجازه داده شود و ممکن است اجازه داده نشود. در این مثال ترکیبی از روز هفته و ساعت از روز مشخص می‌کنند که دسترسی امکان پذیر است یا خیر.

---

[1] Context controls

مثال‌های دیگری از عملگرهای (توابع) توکار[1] که احتمالاً سیستم باید پشتیبانی کند و می‌تواند برای مجوزهای وابسته به متن مفید باشد عبارتند از:

TODAY (); Value = the current date
USER (); Value = the ID of current user
TERMINAL (); Value = the ID of the originating terminal for the current request

حال احتمالاً از نظر مفهومی متوجه شده‌اید که تمام مجوزها (مجاز شماری‌ها) با هم از لحاظ منطقی OR می‌شود. به عبارت دیگر، یک درخواست دست‌یابی مفروض (یعنی هم شئ درخواستی، هم عملیات درخواستی و هم کاربر) قابل پذیرش است اگر و تنها اگر حداقل یکی از مجوزها (قواعد امنیتی)، این اجازه را بدهد. به هر حال توجه کنید که اگر برای مثال (الف) یک مجوز به کاربر Nancy اجازه دهد که رنگ قطعات را بازیابی کند و (ب) مجوز دیگری به او اجازه دهد که وزن قطعات را بازیابی کند. نمی‌توان نتیجه گرفت که او می‌تواند هم رنگ و هم وزن قطعات را بازیابی کند (برای این ترکیب، مجوز جداگانه دیگری نیاز است).

سرانجام، می‌توان این موضوع را مطرح نمود که کاربران تنها مجاز به انجام کاری هستند که صراحتاً برای آن‌ها تعریف شده است (با استفاده از قواعد امنیتی) و هر چیز دیگری که صراحتاً برای انها تعریف نشده غیر مجاز است.

تغییر درخواست[2]

به منظور تشریح بعضی از ایده‌هایی که تاکنون در این بخش مطرح شده است، مختصراً جنبه‌های امنیتی از یک نمونه[3] بنام University Ingres و زبان پرسش آن، یعنی QUEL را شرح می‌دهیم. زیرا رهیافت جالبی برای این مسئله می‌باشند. اساساً هر درخواست ارائه شده به QUEL قبل از اجرا به طور خودکار تغییر خواهد کرد به

---

[1] Built-in
[2] Request Modification
[3] prototype

طوری که دیگر نمی‌تواند محدودیت و قواعد امنیتی را نقض کند. برای مثال، فرض کنید که کاربر U تنها اجازه دارد که قطعات انبار شده در لندن را بازیابی کند:

```
DEFINE PERMIT RETRIEVE ON P TO U
        WHERE P.CITY = "LONDON"
```

(بعداً جزئیات عملگر DEFINE PERMIT را خواهید دید) حال فرض کنید که کاربر U درخواست QUEL را صادر می‌کند.

```
RETRIEVE {P.P#, P.WEIGHT}
WHERE P.COLOR = "Red"
```

با استفاده از اجازه مشخص شده که برای کاربر U و حیطه عمل P در کاتالوگ سیستم ذخیره شده، سیستم به طور خودکار این درخواست را تغییر می‌دهد، به طوری که شبیه درخواست زیر می‌گردد:

```
RETRIEVE {P.P#, P.WEIGHT}
WHERE P.COLOR = "Red"
AND P.CITY = "LONDON"
```

و این درخواست تغییر یافته احتمالاً دیگر هیچ محدودیت امنیتی را نقض نخواهد کرد. در ضمن توجه کنید که فرایند تغییر مخفی[1] است: کاربر U نمی‌داند که در واقع سیستم دستوری را اجرا می‌کند که کمی متفاوت از درخواست اصلی است، زیرا این واقعیت می‌تواند کاربر را حساس کند (کاربر U ممکن است اجازه نداشته باشد که از قطعات انبار شده در سایر شهرها اطلاع داشته باشد).

طرح کلی تغییر درخواست در واقع همان تکنیک استفاده شده برای پیاده سازی دیدها است و همچنین (به ویژه در مورد نمونه Ingres) همان محدودیت جامعیتی[2] است. بنابراین یک مزیت این طرح این است که پیاده سازی آن بسیار ساده است (اکثر کدهای مورد نیاز در سیستم مدیریت پایگاه داده موجود است). مزیت دیگر این طرح این است که کارآمد است، زیرا حداقل بخشی از سربار اعمال محدودیت‌های امنیتی بجای زمان اجرا در زمان ترجمه صورت می‌گیرد. مزیت دیگر این است که در

---

[1] Silent
[2] Integrity constraints

روش‌های قبلی، وقتی کاربر برای بخش‌های مختلف یک متغیر رابطه‌ای به مجوزهای متفاوتی نیاز داشت، مشکلاتی به وجود می‌آمد، اما در این روش این‌طور نیست (بخش ۱۷,٦ بیشتر به این موضوع پرداخته است).

یک عیب این روش ساده این است که تمام محدودیت‌های امنیتی نمی‌تواند مدیریت شود. به عنوان یک مثال نقض ساده، فرض کنید که کاربر U به هیچ وجه اجازه ندارد به متغیر رابطه‌ای P دسترسی داشته باشد. در این صورت هیچ شکل تغییر یافته از دستور بازیابی اصلی (RETRIEVE) نباید این توهم را ایجاد کند که متغیر رابطه‌ای P وجود ندارد. در عوض لازم است پیام خطای صریحی مبنی بر «شما اجازه ندارید به این متغیر رابطه‌ای دسترسی داشته باشد» صادر گردد. یا سیستم می‌توانست به دروغ بگوید «چنین متغیر رابطه‌ای وجود ندارد» و یا بهتر آن که بگوید «چنین متغیر رابطه‌ای وجود ندارد یا شما اجازه دست‌یابی به آن را ندارید».

در اینجا نحو DEFINE PERMIT را تشریح می‌کنیم:

```
DEFINE PERMIT <operation name comma list>
   ON <relvar name> [{<attribute name comma list>}]
        TO <user ID>
      [AT <terminal ID comma list>]
      [FROM <time> TO <time>]
      [ON <day> TO <day>]
      [WHERE <bool exp>]
```

این دستور از نظر مفهومی شبیه دستور AUTORITY است، با این تفاوت که در این دستور از (فراکرد) WHERE پشتیبانی می‌شود (در ضمن) فراکرد WHEN شامل کلاس‌های AT، FROM و ON می‌باشد مثال زیر را در نظر بگیرید:

```
DEFINE PERMIT RETRIEVE, APPEND, REPLACE
        ON S (S#, CITY)
        TO Joe
        AT TTA4
        FROM 9:00 TO 17:00
        ON Sat TO Sun
       WHERE S.STATUS < 50
       AND S.S# = SP.P#
       AND SP.P# = P.P#
       AND P.COLOR = "Red"
```

نکته: APPEND و REPLACE در زبان QUEL به ترتیب معادل INSERT و UPDATE می‌باشند.

فایل‌های ردگیری

این مهم است که فرض کنیم که سیستم ایمنی کامل است. نفوذ گری که کاملاً مصمم است، معمولاً راهی برای درهم شکستن کنترل‌ها پیدا می‌کند، بـه خصـوص اگـر هزینه انجام دادن این کار زیاد باشد. بنابراین، در مواقعی که داده‌ها بسیار حساس هستند و یا پردازش بر روی داده‌ها از حساسیت خاصی برخوردار اسـت، فایـل ردگیـری لازم خواهد بود. به عنوان مثال، اگر از روی اختلاف داده‌ها، شک کنیم که آیا پایگـاه داده‌هـا مورد تعرض قرار گرفته است یا خیر؟ می‌توان فایل ردگیری را بررسی کنیم و ببینیم کـه چه کارهای صورت پذیرفته و آیا همه چیز تحت کنترل هست یا خیر.

یک فایل ردگیری، اساساً یک فایـل یـا پایگـاه داده ویـژه‌ای اسـت کـه در آن سیستم به طور خودکار تمام عملیاتی را که توسط کاربر بر روی داده‌ها انجام گرفتـه، در آن ذخیره می‌کند. در برخی از سیستم‌های مدیریت پایگاه داده، فایل ردگیـری بـه طـور فیزیکی با فایل ترمیم، مجتمع شده است. ولی در برخـی از سیسـتم‌هـای دیگـر ممکـن است این دو فایل از یکدیگر جدا باشند. به هر حال، کاربران باید قادر باشـند کـه فایـل ردگیری را با استفاده از زبان پرسش مناسب، بررسی کنند. (البته به شرطی که مجـاز بـه این کار باشند!). نمونه‌ای از رکورد فایل ردگیری باید شامل موارد زیر باشد:

- متن اصلی درخواست
- پایانه‌ای که عملیات از سوی آن فراخوانی شده است.
- کاربری که عملیات را فراخوانی کرده است.
- تاریخ و زمان انجام عملیات.
- متغیر (های) رابطه‌ای، تاپل (ها)، صفت (صفات) که تحت تأثیر قـرار گرفته‌اند.
- پیش تصویر (مقادیری قبلی).

- پس تصویر (مقادیر جدید).

همان‌طور که گفته شد، حقیقت نگهداری فایل ثبت این است که به وسیله آن، در بعضی موارد می‌توان از نفوذهای آینده جلوگیری نمود.

۳,۱۷ کنترل دسترسی اجباری (روش اجباری)

کنترل‌ها اجباری بر روی پایگاه داده‌هایی قابل اعمال خواهد بود که داده‌ها آن دارای ساختار رده بندی ایستا و دقیق (انعطاف ناپذیر) باشند؛ و در محیط‌های دولتی و نظامی چنین است. همان‌طور که مختصراً در بخش ۱۷,۱ گفته شد، هر شئ داده دارای سطح رده بندی است (مثل: خیلی سری[1] (TS)، سری[2] (S)، محرمانه[3] (C)، رده بندی نشده[4] (U)))و هر کاربر دارای یک سطح مجوز[5] (با همان موارد سطح رده بندی) است. ترتیب سطوح محرمانه بدین صورت است: TS > S > C > U . دقت داشته باشید که اگر سطح محرمانه داده A از سطح محرمانه داده B بالاتر باشد داریم A > B. حال قوانین ساده زیر که مربوط به بل ولاپادولا[6] است قابل اجرا هستند:

۱. کاربر i تنها در صورتی می‌تواند شئ داده j را بازیابی کند که سطح مجوز کاربر i بزرگ‌تر یا مساوی سطح رده بندی شئ داده j باشد (خاصیت ایمنی ساده).
۲. کاربر i تنها در صورتی می‌تواند شئ داده j را به هنگام کند که سطح مجوز کاربر i برابر با سطح رده بندی شئ داده j باشد (خاصیت ستاره دار).

---

[1] Top secret
[2] secret
[3] confidential
[4] unclassified
[5] Clearance level
[6] Bell and La Padula

قاعده[1] اول کاملاً معلوم و مشخص است، اما قاعده دوم احتیاج به کمی توضیح دارد. ابتدا توجه داشته باشید که قاعده دوم را می‌توان به صورت دیگری بیان نمود: طبق تعریف، هر چیزی (شئ داده‌ای) که توسط کاربر i نوشته می‌شود، به طور خودکار سطح رده بندی آن شی داده برابر با سطح مجوز کاربر i قرار داده می‌شود. چنین قاعده‌ای از این جهت لازم است که مثلاً کاربری با سطح دسترسی (مجوز) سری نتواند داده‌ها را در فایلی با سطح رده بندی پایین‌تر کپی کند و در این صورت داده‌هایی با سطح رده بندی سری به راحتی در اختیار سایر کاربرانی که نباید اجازه دسترسی به این داده‌ها را داشته باشند، فراهم شده است.

کنترل‌های اجباری در اوایل دهه ۱۹۹۰ در دنیای پایگاه داده‌ها بسیار مورد توجه قرار گرفتند، زیرا در آن زمان وزارت دفاع امریکا (DOD) می‌خواست هر سیستمی که می‌خرد از چنین کنترل‌هایی پشتیبانی کند، از این رو فروشندگان سیستم‌های مدیریت پایگاه داده (DBMS)، این کنترل‌ها را پیاده سازی کردند. کنترل‌های مورد نظر در دو کتاب به نام‌های Orange book و Lavender Book مستند شدند. در Orange book یک مجموعه از نیازمندی‌های امنیتی برای هر TCB تعریف می‌کند و کتاب Lavender Book تفسیری از نیازمندی‌های TCB برای سیستم‌های پایگاه داده تعریف می‌کند.

کنترل‌های اجباری که در کتاب‌های Orange و Lavender تعریف شدند، در واقع بخشی از طرح رده بندی امنیتی کلی را تشکیل می‌دهند که در اینجا به طور خلاصه بیان می‌شود. اول از همه، چهار کلاس (رده) امنیتی تعریف می‌شود، کلاس D کم‌ترین سطح امنیتی را دارد و کلاس C در رده بعدی قرار دارد، در واقع رده بندی کلاس‌های امنیتی بدین ترتیب است A>B>C>D. می‌گوییم کلاس D کم‌ترین امنیت را دارد (حفاظت نشده)، کلاس C حفاظت اختیاری دارد، کلاس B حفاظت اجباری دارد،

---

[1] می‌توان از کلمه محدودیت نیز استفاده نمود.

و کلاس A حفاظت بازبینی شده[1] دارد. حال به طور خلاصه به تشریح کلاس‌های A، B و C می‌پردازیم.

- **حفاظت اختیاری**:کلاس C به دو زیر کلاس C_1 و C_2 تقسیم می‌شود (به طوری که امنیت C_1 کمتر از C_2 است). هر یک از کنترل‌های اختیاری پشتیبانی می‌کنند، یعنی دست‌یابی موضوعی است که در اختیار مالک داده قرار دارد (در بخش ۱۷,۲ توضیح داده شد). در مجموع:

۱. کلاس C_1 بین مالکیت داده و دست‌یابی به داده تفاوت قائل است–یعنی از مفهوم داده مشترک پشتیبانی می‌کند، در حالی که به کاربران نیز اجازه می‌دهد داده‌های اختصاصی خود را داشته باشند.

۲. کلاس C_2 علاوه بر این، نیازمند پشتیبانی از نگهداری سوابق از طریق روال‌های ثبت ورود، ردگیری و جدایی منابع می‌باشد.

- **حفاظت اجباری**: کلاس B، کلاسی است که با کنترل‌های اجباری سروکار دارد. این کلاس به سه زیر کلاس B_1، B_2و B_3 تقسیم می‌شود (به طوری که B_1 کم‌ترین ایمنی و B_3 بیشترین ایمنی را دارد) :

۱. در کلاس B_1 اشیاء داده نیاز به برچسب ایمنی دارند (یعنی هر شئ داده با یک سطح رده بندی برچسب زده می‌شود مثلاً سری، محرمانه و غیره). همچنین این کلاس نیازمند یک دستور غیر رسمی از سیاست ایمنی است.

۲. کلاس B_2 علاوه بر آن چه در کلاس B_1 است نیازمند یک دستور رسمی از همان چیز نیاز دارد، همچنین لازم است که

---

[1] Verified protection

کانال‌های مخفی[1] مشخص شده و حذف شوند. مثال‌های از کانال‌های مخفی (الف) احتمال پی بردن به اینکه کدام پرسش مجاز است و کدام پرسش غیر مجاز (بخش ۱۷,٤ را ببینید). یا (ب) احتمال پی بردن به اطلاعات حساس از طریق محاسبه زمان (مدت زمانی که طول می‌کشد یک محاسبه مجاز انجام پذیرد).

۳. کلاس B_3 علاوه بر این نیازمند پشتیبانی از ترمیم، ردگیری و مدیر امنیت می‌باشد.

- **محافظت بازبینی شده**: کلاس A که بیشترین امنیت را دارد، لازم است که یک اثبات ریاضی انجام گیرد که مشخص کند یک راهکار ایمنی سازگار است و برای پشتیبانی از سیاست امنیتی کافی است (!).

چنین مدیریت پایگاه داده‌های تجاری از کنترل‌های اجباری سطح B_1 و همچنین کنترل‌های اختیاری سطح C_2 پشتیبانی می‌کنند. اصطلاحات: مدیریت‌های پایگاه داده‌ها که از کنترل‌های اجباری پشتیبانی می‌کنند، سیستم‌های امن چند سطحی[2] نامیده می‌شوند.

ایمنی چند سطحی

فرض کنید می‌خواهیم ایده‌های کنترل دسترسی اجباری را به متغیر رابطه‌ای S (عرضه کنندگان) اعمال کنیم. برای قطعیت و همچنین سهولت، فرض می‌کنیم که واحد داده‌ای که می‌خواهیم دست‌یابی به آن را کنترل کنیم، یک تاپل در این متغیر رابطه‌ای است (یک تاپل از رابطه S است). در ضمن به جهت استفاده از روش کنترل دسترسی

---

[1] Covert channel
[2] Multi-level secure systems

اجباری لازم است هر تاپل یک سطح رده بندی داشته باشد که می‌تواند طبق آنچه که در شکل ۱۷,۱ آمده است باشد (خیلی سری = ٤، سری = ۳، محرمانه = ۲ و غیره).

حال فرض کنید که دو کاربر داریم که سطح مجوز آن‌ها بدین صورت است U_3=3(سری) و U_2=2(محرمانه)، این دو کاربر حیطه عمل S(رابطه S) را به صورت متفاوتی می‌بینند. اگر درخواست بازیابی تمام عرضه کنندگان توسط کاربر U_3 صادر شود، چهار تاپل را برمی گرداند (تاپل های s_1,s_2,s_3,s_5) در حالی که اگر این درخواست توسط U_2 صورت پذیرد تنها دو تاپل (s_1,s_3) را باز می‌گرداند، در ضمن هیچ یک از این دو کاربر تاپل s_4 را نمی‌بینند.

S

S#	SNAME	STATUS	CITY	LEVEL
S1	Smith	20	London	2
S2	Jones	10	Paris	3
S3	Blake	30	Paris	2
S4	Clark	20	London	4
S5	Adams	30	Athens	3

شکل ۱۷,۱ حیطه عمل (رابطه) S با سطوح رده بندی (مثال)

یک روش پرداختن به این موضوع، تغییر دوباره درخواست است. این پرسش را در نظر بگیرید. «عرضه کنندگان در لندن کدامند؟"

S WHERE CITY = 'London'

سیستم مدیریت پایگاه داده این درخواست را بدین صورت تغییر می‌دهد:

S WHERE CITY = 'London' AND LEVEL ≤ user clearance

همین ملاحظات در مورد عملیات به هنگام سازی نیز وجود دارد. برای مثال، کاربر U_3 از وجود تاپل s_4 آگاه نیست از این رو، از نظر کاربر U_3 دستور درج زیر مشکلی ندارد و باید توسط سیستم صورت پذیرد:

```
INSERT S RELATION {TUPLE {S# S# ('s4'),
                          SNAME NAME
('Saker'),
                          STATUS 25,
                          CITY 'Rome'}};
```

سیستم نباید این دستور درج را رد کند، زیرا این کار در واقع به کاربر خواهد گفت تاپل S_4 وجود دارد (در واقع در صورت انجام این عمل کاربر به طور غیر مستقیم به اطلاعاتی دست یافته که نباید دست می‌یافت). پس سیستم این درخواست را می‌پذیرد و به صورت زیر تغییر می‌دهد:

```
INSERT S RELATION {TUPLE {S# S# ('s4'),
                          SNAME NAME ('Saker'),
                          STATUS 25,
                          CITY 'Rome',
                          LEVEL 3}};
```

بنابراین، ملاحظه می‌کنید که کلید اصلی برای رابطه S(عرضه کنندگان) تنها {S#} نیست، بلکه ترکیبی از {S#,LEVEL} است. توجه: در اینجا برای سادگی فرض شده است که در رابطه، یک کلید کاندید وجود دارد که می‌توان آن را کلید اصلی در نظر گرفت.

اصطلاحات بیشتر: متغیر رابطه‌ای عرضه کنندگان یک مثال از متغیر رابطه‌ای چند سطحی است. این حقیقت که یک داده از نظر کاربران گوناگون، متفاوت به نظر رسد، چند نمونه سازی[1] نام دارد. برای مثال جواب یک پرسش بازیابی مثال قبل برای کاربران مختلف، متفاوت است. پرسش »عرضه کننده s_4 کدام است؟« یک جواب برای کاربر U_4 با سطح مجوز خیلی سری بازمی گرداند، و جواب دیگری برای کاربر U_3 با سطح مجوز سری، حتی برای کاربر U_2 با سطح مجوز محرمانه چیزی دیگر را برمی گرداند.

با حذف و به هنگام سازی نیز به همین صورت رفتار می‌شود و از این رو به جزئیات نمی‌پردازیم. در اینجا این سوال مطرح می‌گردد که آیا ایده بحث شده در این بخش اصل اطلاعات را نقض می‌کند؟ جواب خود را توجیه کنید.

---

[1] polyinstantiation

٤,١٧ پایگاه داده‌های آماری

در این متن، پایگاه داده آماری، یک پایگاه داده‌ای است که در آن می‌توان پرسش‌هایی که نیاز به اطلاعات جمعی (گروهی[1]) دارند (مثل مجموع و میانگین) را پرسید ولی پرسش‌هایی که نیاز به اطلاعات انفرادی دارند، رد می‌شود. برای مثال این پرسش را می‌توان مطرح نمود «میانگین حقوق کارمندان چقدر است؟» اما پرسشی مثل «حقوق شیرین (یک شخص خاص) چقدر است؟» پاسخ داده نخواهد شد.

مشکل این پایگاه داده این است که بعضی مواقع می‌توان از طریق پرسیدن یکسری سوال مجاز یک جواب غیر مجاز (جوابی که نباید پاسخ داده شود) را استنباط کرد. چون در سوالات جمعی بقایای اطلاعات اصلی وجود دارد پس یک نفر با پرسیدن هدفمند یک سری سوال جمعی و پردازش جواب آن‌ها، ممکن است قادر باشد اطلاعات اصلی را بازیابی کند. به این کار اصطلاحاً «حدس زدن اطلاعات محرمانه از طریق استنتاج» می‌گویند. یادآوری می‌کنیم که این مشکل، زمانی که از مخزن داده[2] استفاده می‌کنیم جدی‌تر خواهد شد (فصل ۲۲ را ببینید).

مثال: فرض کنید که پایگاه داده تنها شامل یک حیطه عمل STATS است (شکل ۱۷,۲). برای سادگی فرض کنید که نوع داده همه صفات، رشته کاراکتری یا عدد است. همچنین فرض کنید که کاربر U (تنها) مجاز است که پرسش‌های آماری را مطرح سازد ولی قصد کاربر U، بدست آوردن میزان حقوق دریافتی شخص Alf است. در آخر فرض کنید که کاربر U می‌داند که Alf مرد است و شغلش برنامه نویسی است. حال پرسش‌های زیر را در نظر بگیرید:

```
1. WITH (STATS WHERE SEX = 'M' AND)
            OCCUPATION =
'Programmer' AS X: COUNT(X)
                                    Result: 1;

2. WITH (STATS WHERE SEX = 'M' AND)
```

---

[1] Aggregated information
[2] Data warehouses

OCCUPATION =
'Programmer' AS X: SUM(X, SALARY)

Result: 50K;

NAME	SEX	CHILDREN	OCCUPATION	SALARY	TAX	AUDITS
Alf	M	3	Programmer	50K	10K	3
Bea	F	2	Physician	130K	10K	0
Cyn	F	0	Programmer	56K	18K	1
Dee	F	2	Builder	60K	12K	1
Ern	M	2	Clerk	44K	4K	0
Fay	F	1	Artist	30K	0K	0
Guy	M	0	Lawyer	190K	0K	0
Bal	M	3	Homemaker	44K	2K	0
Ivy	F	4	Programmer	64K	10K	1
Joy	F	1	Programmer	60K	20K	1

شکل ۱۷,۲ حیطه عمل STATS

بدیهی است که ایمنی پایگاه داده به خطر افتاده است. اگر چه کاربر U تنها پرسش‌های آماری قانونی را پرسیده است، اما همان‌طور که در این مثال می‌بینید، اگر کاربر یک عبارت منطقی بیابد که یک فرد مشخص را معلوم می‌کند، آنگاه اطلاعات مربوط به آن فرد مشخص، دیگر امن نیست. این حقیقت می‌گوید که سیستم نباید به پرسش‌هایی که کاردینالتی مجموعه‌ای، که باید تجمیع شود (در یک گروه قرار گیرد) از یک کران پایین به فرض b کمتر باشد. همچنین سیستم نباید به پرسش‌هایی که کاردینالتی آن‌ها بیشتر از کران n-b است پاسخ دهد (n کاردینالتی رابطه دربرگیرنده است) زیرا نتیجه قبلی می‌تواند به این صورت بدست آید:

3. COUNT (STATS)
Result: 12.
4. WITH (STATS WHERE NOT) SEX = 'M' AND
OCCUPATION =
'Programmer' AS X: COUNT(X)

Result: 11; 12 – 11 = 1.

5. SUM (STATS, SALARY)

Result: 728K;

6. WITH (STATS WHERE NOT) SEX = 'M' AND

OCCUPATION =
'Programmer' AS X SUM(X, SALARY)
Result: 678K; 728K- 678K = 50K.

متأسفانه به راحتی می‌توان نشان داد که محدود کردن پرسش‌ها، به پرسش‌هایی که کاردینالتی گروه‌هایی که در آن‌ها تجمیع صورت می‌پذیرد به c بطوریکه $b \le c \le n-b$ نیز، برای اجتناب از به خطر افتادن پایگاه داده‌ها کافی نیست. بار دیگر با توجه به جدول شکل ۱۷,۲، فرض کنید که b=2 باشد. در این صورت پرسش‌هایی پاسخ داده خواهند شد که c در بازه $2 \le c \le 8$ باشد. عبارت منطقی:

SEX = 'M' AND OCCUPATION = 'Programmer'

دیگر قابل قبول نیست، اما پرسش زیر را در نظر بگیرید:

7. WITH (STATS WHERE SEX = ‘M’) AS X;
COUNT(X)
Result: 4.

8. WITH (STATS WHERE NOT) SEX = 'M' AND NOT
OCCUPATION =
'Programmer' AS X COUNT(X)
Result: 3.

از نتیجه پرسش‌های ۷ و ۸ کاربر U می‌تواند این نتیجه را بگیرد که تنها یک مرد برنامه نویس وجود دارد، بنابراین این شخص Alf است (چون کاربر U از قبل می‌داند که این توصیف مربوط به Alf است). حال میزان حقوق Alf می‌تواند بدین صورت بدست آید:

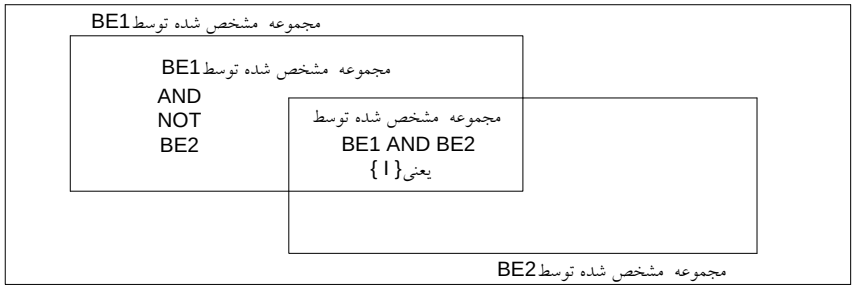
9. WITH (STATS WHERE SEX = ‘M’) AS X;
SUM(X, SALARY)
Result: 328K;

6. WITH (STATS WHERE) SEX = 'M' AND NOT
(OCCUPATION =
‘Programmer’); SUM(X, SALARY)
Result: 278K; 328K- 278K = 50K.

عبـارت منطقـی OCCUPATION = 'Programmer' SEX = 'M' AND یک ردیاب اختصاصی[1] برای Alf نامیده می‌شود، زیرا به کـاربر ایـن امکـان را می‌دهد که اطلاعات مربوط به شخص Alf را پیدا کند. به طور کلی می‌توان گفت، اگـر کاربر یک عبارت منطقی BE را بداند که با توجه به آن می‌تـوان شـخص i را مشـخص نمود، و اگر BE بتواند به شکل BE1 and BE2 بیان شود، پـس عبـارت منطقـی BE1 and not BE2 یک ردیاب برای شخص i است (به شرط آن که BE1 و BE1 and not BE2 هر دو قابل قبول باشند- یعنی در هر دو، کاردینالتی مجموعه‌های جـواب برابـر c است که c در بازه $b \le c \le n-b$ قرار دارد). علت به این خاطر است که مجموعه مشخص شده توسط BE برابر اسـت بـا تفاضـل بـین مجموعـه مشـخص شـده توسـط BE1 و مجموعه مشخص شده توسط BE1 and not BE2 است:

{X: BE} ≡ {X: BE1 AND BE2}
≡ {X: BE} MINUS {X: BE1 AND NOT BE2}

شکل ۱۷,۳ را ببینید.



شکل ۱۷,۳ ردیاب اختصاصی BE1 AND NOT BE2

ایده مذکور را تعمیم داده و نشان می‌دهید که تقریباً همیشه می‌توان بـرای هـر پایگاه داده آماری یک ردیاب عمومی[2] (بر عکس مجموعه‌ای از ردیـاب‌هـای شخصـی) پیدا کرد. ردیاب عمومی، یک عبارت منطقی است که از آن، بـرای پیـدا کـردن جـواب

---

[1] INDIVITUAL TRAKER
[2] General tracker

برای هر پرسش غیر مجازمی توان استفاده نمود (یعنی هر پرسشی که یک عبارت غیر مجاز دارد) در مقایسه با ردیاب اختصاصی که تنها برخی، عبارات غیر مجاز خاص آن‌را استفاده می‌کنند. در حقیقت، هر عبارتی که کاردینالتی حاصل از اجرای پرسش حاوی آن، c که c در بازه $2b \leq c \leq n-2b$ باشد، یک ردیاب عمومی است (دقت داشته باشید که b باید کمتر از $n/4$ باشد) بدیهی است. یک‌بار که چنین ردیابی (کلی) پیدا شد، پرسشی که حاوی عبارت غیر مجاز E است، همان‌طور که در مثال بعدی می‌بینید می‌تواند پاسخ داده شود (برای قطعیت، حالتی را در نظر می‌گیریم که کاردینالتی مجموعه حاصل از اجرای پرسش حاوی BE کمتر از b باشد. حالتی که کاردینالتی بیشتر از $n - b$ است به این ترتیب مدیریت می‌شود). توجه داشته باشید که بر طبق تعریف، T یک ردیاب عمومی است اگر و فقط اگر نقیض T یعنی (NOT T) نیز یک ردیاب عمومی باشد.

مثال: دوباره فرض کنید که $b = 2$است، پس هر عبارتی که کاردینالتی (c) مجموعه حاصل از اجرای پرسش حاوی آن در بازه $4 \leq c \leq 6$ باشد، یک ردیاب عمومی است. دوباره فرض کنید که کاربر U می‌داند که Alf یک برنامه نویس مرد است– یعنی همانند قبل عبارت منطقی غیر مجاز BE برابر است با:

```
SEX = 'M' AND OCCUPATION = 'Programmer'
```

همچنین مانند قبل فرض کنید که کاربر U قصد دارد که بفهمد حقوق Alf چقدر است. از ردیاب عمومی دو بار استفاده می‌کنیم، اول برای اینکه واقعاً مطمئن شویم که عبارت منطقی BE، شخص Alf را به تنهایی مشخص می‌کند (گام‌های ۲ تا ٤)، سپس حقوق Alf را مشخص می‌کنیم (گام‌های ٥ تا ۷).

گام ۱: یک ردیاب عمومی (T) حدس بزنید. حدس می‌زنیم Tاین عبارت باشد:

```
AUDITS = 0
```

گام ۲ : با استفاده از عبارت T و نقیض T یعنی (NOT T)، تعداد کـل افـراد در پایگاه داده را بدست آورید :

```
 WITH ( STATS WHERE AUDITS = 0 ) AS X;
COUNT( X )
Result : 5;
WITH ( STATS WHERE NOT )AUDITS = 0 AS X;
COUNT( X )
                                    Result : 5; 5+5 = 10.
```

حال به راحتی می‌توان دید که حدسـمان (عبـارت T) در واقـع یـک ردیـاب عمومی است.

گام ۳: حاصل جمع (الف) تعداد اشخاص در پایگاه داده به اضافه (ب) تعـداد (اشخاصی) که در عبارت غیر مجاز BE صدق می‌کنند، با استفاده از عبارات BE OR T و BE OR NOT T (دقت داشته باشید که BE همان اطلاعات اولیه هست که در مـورد شخص) اشخاص داریم، در اینجا Alf یک برنامه نویس مرد است:

```
WITH ( STATS WHERE ) SEX = 'M' AND
                        OCCUPATION = 'Programmer'
                  OR AUDITS = 0 AS X;
COUNT ( X )
Result : 6.
WITH ( STATS WHERE ) SEX = 'M' AND
                        OCCUPATION = 'Programmer'
                  OR NOT( AUDITS = 0) AS X;
COUNT ( X )
                                    Result : 5: 6 + 5 = 11.
```

گام ٤ : با توجه به نتایجی که تا کنون بدست آمده، تعداد افراد صدق کننده در شرط BE برابر یک است (نتیجه گام ۳ من‌های نتیجه گام ۲) : این بدین معنی است کـه شرط BE شخص Alf را به تنهایی مشخص می‌نماید.

حال در گام ٥ و ٦ همان پرسش‌های گام‌های ۲ و ۳ را می‌پرسیم امـا ایـن بـار بجای COUNT از SUM استفاده می‌کنیم.

گام ۵ : با استفاده از عبـارت T و نقـیض T (NOT T)، کـل حقـوق افـراد در پایگاه داده را بدست آورید :

```
    WITH (STATS WHERE AUDITS = 0) AS X;
    SUM(X, SALARY)
Result: 438K;
    WITH (STATS WHERE NOT) AUDITS = 0 AS X;
    SUM(X, SALARY)
                                    Result: 290; 438K + 290K = 728K.
```

گام ٦ : با استفاده از عبارات BE OR T و BE OR NOT T، جمع حقوق شخص Alf و کل افراد را بدست آورید.

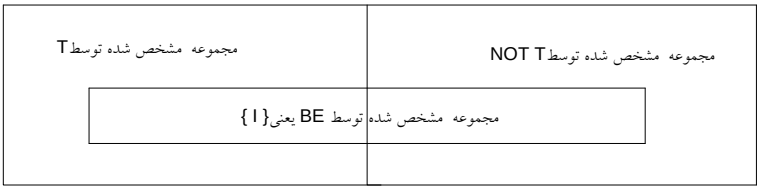
```
  WITH (STATS WHERE) SEX = 'M' AND
                              OCCUPATION = 'Programmer'
                         OR AUDITS = 0 AS X;
    SUM(X, SALARY)
Result: 488K.
  WITH (STATS WHERE) SEX = 'M' AND
                              OCCUPATION = 'Programmer'
                         OR NOT (AUDITS = 0) AS X;
   SUM(X, SALARY)
                                  Result: 290K: 488K + 290K = 778K.
```

گام ۷ : با کم کردن کل حقوق (که در گام ۵ بدسـت آمـد) از حاصـل بدسـت آمده از گام ٦ حقوق Alf بدست می‌آید.

Result: 50K.

شکل ۱۷,٤ ردیاب عمومی زیر را نشان می‌دهد:

```
{X: BE} ≡ ({X: BE OR T} UNION {X: BE OR NOT T}})
                              MINUS {X: T OR NOT T}
```



شکل ۱۷,٤ یک ردیاب عمومی T

اگر حدس اولیه اشتباه باشد (یعنی: T یک ردیاب عمومی نباشد)، پس یکی یا هر دو عبارات (BE OR T) و (BE OR NOT T) ممکن است غیر مجاز باشند. برای مثال، اگر کاردینالتی مجموعه‌های حاصل از تصدیق شرط BE و T به ترتیب برابر p و q باشد، به طوری که $p< b$ و $b \leq q < 2b$ پس این امکان وجود دارد که کاردینالتی مجموعه حاصل از تصدیق شرط (BE OR NOT T) بیشتر از n-b است. در چنین مواردی لازم است که حدس دیگری برای ردیاب عمومی زده شود و مراحل گفته شده دوباره انجام گیرد. در مرجع [8] یک فرایند برای پیدا کردن ردیاب عمومی پیشنهاد کرده که در عمل مشکل نیست. در مثال که گفته شد، حدس اولیه یک ردیاب عمومی است (کاردینالتی مجموعه حاصل از آن برابر ۵ است) و پرسش‌ها در گام سوم مجاز است.

به طور خلاصه: «تقریباً همیشه» یک ردیاب عمومی وجود دارد و معمولاً یافتن و استفاده از آن آسان است. در حقیقت، با حدس زدن سریع می‌توان یک ردیاب عمومی را بدست آورد. حتی در مواردی که ردیاب کلی وجود ندارد، منبع [8] نشان می‌دهد که یک سری ردیاب‌های ویژه برای پرسش‌های ویژه می‌توان یافت. از این نتیجه گیری که ایمنی در پایگاه داده‌های آماری یک مشکل ذاتی است، نمی‌توان فرار کرد.

پس چه باید کرد؟ چندین پیشنهاد در متون آمده است، اما هیچ‌کدام از آن‌ها کاملاً رضایت بخش نیستند. برای مثال یک راه حل «تعویض داده[1]» است- یعنی تعویض مقادیر صفات بین تاپل ها، به طوری که صحت آمار کلی حفظ گردد، بنابراین اگر یک مقدار خاص (مثلاً حقوق خاصی) مشخص گردد، نمی‌توان مشخص کرد که این مبلغ متعلق به چه کسی است. مشکل این روش اینجاست که، مجموعه‌ای از فقره داده‌ها (رکوردها) داریم که مقادیرشان با یکدیگر تعویض شده است. چنین محدودیتی در اغلب روش‌های پیشنهادی وجود دارد، بنابراین، در حال حاضر به سختی می‌توان با

---

[1] Data swapping

نتایج منبع [8] مخالفت نمود. روش دیگر جلوگیری از اجـرای دنبالـه‌ای از پرسـش‌هـا است که مکرراً به جمعیت ثابتی از یک رابطه دست‌یابی دارند. همچنین روش دیگر این است که عمداً به نتایج پرسش‌های آماری یک سری اطلاعات غیر واقعی اضافه کنیم که کار نتیجه گیری از این اطلاعات سخت گردد.

۵,۱۷ رمزگذاری داده‌ها

تاکنون در این فصل فرض کرده‌ایم که هر رخنه گری که در آینده بخواهـد بـه پایگاه داده‌ها دست‌یابی داشته باشد، از امکانات معمول سیستم اسـتفاده مـی‌کنـد. حـال توجهمان را به موردی جلب می‌کنیم که کـاربر تـلاش دارد از طریـق راه‌هـای جـانبی[1] سیستم را دور بزند (برای مثال دیسک را به طور فیزیکـی بـردارد یـا در خـط ارتبـاطی رخنه کند). معمول‌ترین اقدام موثر برای مقابله با این کار، رمزگذاری داده[2] است. یعنـی اینکه داده‌های حساس را به شکل کد شده، ذخیره و انتقال دهیم.

برای آنکه در مورد برخی مفاهیم رمزگذاری داده بحث کنیم، لازم اسـت ابتـدا چند اصطلاح را معرفی کنیم. داده‌های اصلی (نهان نشده) متن سـاده[3] نامیـده مـی‌شـود. متن ساده توسط یک الگوریتم رمزگذاری، کد گذاری مـی‌شـود، کـه ورودی‌هـای ایـن الگوریتم متن ساده و کلید رمزگذاری است. خروجی این الگوریتم شکل کد شـده مـتن ساده است که متن کد شده[4] نامیده می‌شود. جزئیات الگوریتم رمزگذاری عمومی است یا حداقل مخفی نیست اما کلید رمزگذاری به طور محرمانه نگهداری می‌شود. مـتن کـد شده برای تمام کسانی که کلید رمزگذاری را ندارنـد، چـه در پایگـاه داده ذخیـره شـده باشد یا از طریق خط ارتباطی منتقل شود، نامفهوم است.

مثال: متن ساده زیر را در نظر بگیرید.

AS KINGFISHERS CATCH FIRE

---

[1] bypass
[2] Data encryption
[3] Plaintext
[4] cipher text

(برای سادگی فرض می‌کنیم که فقط با داده‌های کاراکتری حروف بزرگ و فاصله) فضای خالی سروکار داریم. کلید رمزگذاری را رشته زیر در نظر می‌گیریم.

ELIOT

همچنین الگوریتم رمزگذاری را بدین صورت در نظر می‌گیریم.

۱. متن ساده را به بلوک‌های (قسمت‌های) مساوی با طول کلید (در این مثال ۵) تقسیم می‌کنیم.

A S + K I N G P I S H E R S + C A T C H + F I R E

فضاهای خالی (فاصله‌ها) با + مشخص شده‌اند.

۲. هر کاراکتر در متن ساده را با یک عدد صحیح در بازه ۰۰–۲۶ جایگزین می‌کنیم، فضای خالی=00، A=01,…,Z=26

0301200308 0805181900 1407060919 0119001109
0006091805

۳. همین کار که در گام ۲ توضیح داده شد را برای کلید رمزگذاری انجام می‌دهیم.

0512091520

٤. برای هر بلوک متن ساده، بجای هر کاراکتر، مجموع عدد صحیح متن ساده (که در گام ۲ بدست آمد) و عدد صحیح کلید رمزگذاری را در پیمانه ۲۷(حاصل مجموع همیشه در محدوده ۰۰–۲۶ قرار بگیرد) محاسبه کنید. دقت داشته باشید اعداد دو رقم، دو رقم جمع گردد یعنی XX + YY MOD 27 = ZZ.

0301200308 0006091805	0805181900	1407060919	0119001109
0512091520 0512091520	0512091520	0512091520	0512091520
_____ _____	_____	_____	_____
0813021801 0518180625	1317000720	1919152412	0604092602

۵. حرف معادل عدد صحیح حاصل از گام ٤ الگوریتم را بجای آن جایگزین کنید.

F O I Z B S S O X L M Q + G T H M B R A E R R F Y

رویه عیان سازی[1] این مثال، با فرض معلوم بودن کلید، ساده است. (تمرین: متن کد شده فوق را عیان سازی کنید) پرسش اینجا است که پیدا کردن کلید برای رخنه گر تنها با مقایسه متن ساده و متن کد شده و بدون داشتن دانش قبلی، چقدر مشکل است؟ در مثال ساده خودمان، پاسخ به این سوال مشخص است البته نه خیلی واضح، اما بدیهی است که طرح‌های پیچیده‌تری می‌تواند طراحی گردد. طرح ایده آل بکارگرفته شده باید طوری باشد که کارهای لازم برای شکستن آن خیلی پرهزینه‌تر از امتیازات بالقوه‌ای باشد که موفقیت این کار (شکستن طرح) برای رخنه گر خواهد داشت. هدف نهایی توافق شده برای چنین طرحی این است که (حتی) مخترع طرح، با مقایسه متن ساده و متن کد شده، قادر نباشد کلید را مشخص کند و در نتیجه نتواند قطعه دیگری از متن کد شده را عیان سازی کند.

استاندارد رمزگذاری داده

مثال قبلی از یک رویه جانشینی[2] استفاده کرد. به ازای هر حرف (کاراکتر) در متن ساده یک حرف در متن کد شده جایگزین می‌شود که این حرف با استفاده از کلید رمزگذاری مشخص می‌گردد. جانشینی یکی از دو رهیافت اصلی رمزگذاری است، روش دیگر جایگشت[3] است که در این روش حروف متن ساده به ترتیب دیگری چیده می‌شوند. هیچ‌کدام از این دو رهیافت فی‌نفسه ایمن نیستند. اما الگوریتم‌هایی که هر دو رهیافت را با هم ترکیب می‌کنند همگی یک درجه بالاتری از ایمنی را فراهم می‌کنند.

---

[1] Decryption
[2] substitution
[3] permutation

یکی از این الگوریتم‌ها، استاندارد رمزگذاری داده‌ها (DES[1]) است، که توسط IBM توسعه یافت و در سال ۱۹۷۷ به عنوان یک استاندارد فدرال امریکا پذیرفته شد.

برای استفاده از DESمتن ساده به بلوک‌های ٦٤ بیتی تقسیم شده و هر بلوک با استفاده از یک کلید ٦٤ بیتی (در واقع کلید شامل ٥٦ بیت داده به اضافه ٨ بیت توازن) بخش است، بنابراین تعداد کلیدهای ممکن برابر است با 2^{56} نه 2^{64} رمزگذاری می‌گردد. یک بلوک با اعمال یک جایگشت اولیه بر روی آن، رمزگذاری شده و سپس بلوک جایگشت شده را در معرض ١٦ مرحله جانشینی پیچیده قرار داده و دست آخر یک جانشینی دیگر که برعکس جانشینی اولیه است، را بر روی بلوک حاصل از نتیجه این مراحل، اعمال می‌کنیم. جانشینی مرحله i ام به طور مستقیم با کلید رمزگذاری k کنترل نمی‌شود اما توسط یک کلید Ki که از مقادیر K و I محاسبه می‌شود، کنترل می‌گردد. برای جزئیات بیشتر منبع [20] را ببینید.

خاصیت DES این است که الگوریتم رمزگذاری معادل الگوریتم عیان سازی است، با این تفاوت که کلیدهای Ki به ترتیب وارونه اعمال شوند.

اما با افزایش سرعت و ظرفیت کامپیوترها، DES به خاطر وابستگی به کلیدهای کوچک (٥٦ بیتی) به طور فزاینده‌ای مورد نکوهش قرار گرفته‌اند. بنابراین در سال ۲۰۰۰ دولت فدرال امریکا با یک استاندارد جدید به نام استاندارد رمزگذاری پیشرفته (AES)توافق کرد که مبتنی بر الگوریتم Rijndael است و از کلیدهای ۱۲۸، ۱۹۲ یا ٢٥٦ بیتی استفاده می‌کند. حتی کلیدهای ۱۲۸ بیتی بیشتر از نمونه قبلی ایمن هستند. بر طبق منبع [34] اگر کامپیوتر این قدر سریع باشد که DES را در یک ثانیه شکست دهد[2]، همان کامپیوتر ١٤٩ تریلیون سال طول می‌کشد تا بتواند AES را شکست دهد.

---

[1] Data encryption standard
[2] crack

رمزگذاری کلید عمومی

نشان دادیم که DES ممکن است به خوبی ایمن نباشد. AES بهتر است، اما با این وجود بسیاری از مردم احساس می‌کنند که این طرح‌ها اگر با ابزار بسیار هوشمند شکست نخورند با تلاش طاقت فرسا شکست بخورد. همچنین بسیاری از مردم احساس می‌کنند که طرح‌های رمزگذاری کلید عمومی[1] این رهیافت‌ها را از نظر فناوری از دور خارج می‌کند. در یک طرح کلید عمومی هم الگوریتم رمزگذاری و هم کلید رمزگذاری در اختیار همه قرار می‌گیرد، بنابراین هر شخصی می‌تواند متن ساده را به متن کد شده تبدیل کند. اما کلید عیان سازی این متن کد شده محرمانه نگهداری می‌شود (طرح‌های کلید عمومی دو کلید دارند، یکی برای رمزگذاری و دیگری برای عیان سازی). به علاوه کلید عیان سازی نمی‌تواند توسط کلید نهان سازی حدس زده شود، بنابراین حتی اگر شخصی رمزگذاری اصلی را انجام دهد (متن را با کلید رمزگذاری درست کد کند) نمی‌تواند عیان سازی نظیر آن را انجام دهد البته اگر مجاز نباشد.

ایده اصلی رمزگذاری کلید عمومی توسط Diffie و Hellman مطرح گردید. در اینجا معروف‌ترین رهیافت را که توسط Revest، Shamir و Adleman مطرح گردید را شرح داده و نشان می‌دهیم که در عمل چگونه کار می‌کند. رهیافت آن‌ها (اکنون طرح RSA خوانده می‌شود که برگرفته از اول نام آن‌هاست) بر اساس دو اصل زیر است:

۱. برای آنکه مشخص کنیم یک عدد مفروض اول است یا خیر یک الگوریتم سریع و شناخته شده وجود دارد.
۲. هر عدد مرکب (عددی که اول نباشد) را می‌توان با استفاده از اعداد اول (عامل اول) تولید کرد، الگوریتم سریع و شناخته شده‌ای برای انجام این کار وجود ندارد.

---

[1] Public key

مرجع [17.12] مثالی را ارائه کرده که در آن مشخص کردن اینکه آیا یک عدد ۱۳۰ رقمی مفروض، اول هست یا خیر در حدود ۷ دقیقه زمان برده است (روی یک ماشین معمولی). در حالی که پیدا کردن دو عامل اول از یک عدد (مرکب) که از ضرب دو عدد اول ٦٣ رقمی بدست می‌آید در حدود ٤٠ کادریلیون سال طول می‌کشد (روی همان ماشین). هر کاردبلیون برابر ۱,۰۰۰,۰۰۰,۰۰۰,۰۰۰,۰۰۰ است.

طرح RSA بدین صورت کار می‌کند:

۱. به طور تصادفی دو عدد بزرگ اول p و q را انتخاب کرده و حاصل ضرب آن دو یعنی r را محاسبه کنید r = p * q.

۲. به طور تصادفی یک عدد صحیح و بزرگ e را که نسبت به حاصل‌ضرب (p-1)* (q-1) اول هست را انتخاب کنید (یعنی هیچ عامل مشترکی ندارند). عدد صحیح e کلید نهان سازی است. توجه: انتخاب e آسان است، برای مثال، هر عدد اول بزرگ‌تر از p و q می‌تواند انتخاب شود.

۳. کلید عیان سازی، d را بدین صورت محاسبه کنید که به نوبت d را افزایش داده تا در رابطه زیر صدق کند. مقدار محاسبه شده برای d یکتا است.

d * e = 1 module (p-1) * (q-1)

٤. اعداد صحیح r و e را انتشار داده اما d را مخفی نگاه دارید.

۵. برای رمزگذاری بخشی از متن ساده P(که برای سادگی فرض می‌کنیم که یک عدد صحیح کوچک‌تر از r است) را با متن کد شده C که از رابطه زیر بدست می‌آید، جایگزین می‌کنیم.

C = Modulo r p^e

٦. برای عیان سازی، بخشی از متن کد شده C با متن ساده pکه به صورت زیر محاسبه می‌شود، جایگزین می‌کنیم.

P = Modulo r C^d

مرجع [17] اثبات می‌کند که این طرح کار می‌کند- از عیان سازی C با استفاده از d، متن اصلی P حاصل می‌گردد. همان‌طور که قبلاً ادعا شد، محاسبه d فقط با داشتن r و e(و نه p یا q) غیر ممکن است. از این رو هرکسی می‌تواند متن ساده را رمزگذاری کند، اما تنها کاربران مجاز (کسانی که d را می‌دانند) می‌توانند متن کد شده را عیان سازی کنند.

مثال ساده‌ای برای نشان دادن رویه مذکور ارائه می‌دهیم. برای سادگی تنها از اعداد بسیار کوچک (در مقایسه با آن چه در عمل استفاده می‌شود) استفاده می‌کنیم.

مثال: فرض کنید p=3، q=5 است پس r =15 می‌باشد و حاصل‌ضرب (p-1)*(q-1) = 8 است. فرض کنید e=11(یک عدد اول بزرگ‌تر p و q). برای محاسبه d داریم:

d * 11 = 1 module 8

از این رو d=3 است.

حال فرض کنید که متن ساده P شامل عدد صحیح ۱۳ است. پس متن کد شده C بدین صورت بدست می‌آید:

$$C = \text{Modulo } r \quad p^{e}$$

$$\text{Modulo } 13^{11} =$$

$$= 1{,}792{,}160{,}394{,}037 \text{ modulo } 15$$

$$= 7$$

حال متن ساده اصلی Pبدین صورت بدست می‌آید:

$$P = \text{Modulo } r \quad C^{d}$$

$$\text{Modulo } 15 \quad 7^{3} =$$

$$= 343 \text{ modulo } 15$$

$$= 13$$

چون e و d معکوس یکدیگرند، طرح کلید عمومی نیز اجازه می‌دهد که پیام رمزگذاری شده به نحوی **امضا**[1] شود که گیرنده پیام بتواند مطمئن شود اصل پیام مربوط به شخص مورد نظر است (یعنی امضاها جعل نمی‌شوند). فرض کنید که A و

---

[1] signed

B دو کاربر هستند که می‌خواهند با استفاده از طرح کلید عمومی با یکدیگر ارتباط برقرار کند. پس A و B هر یک، یک الگوریتم نهان سازی و کلید نهان سازی متناظر آن‌را، انتشار می‌دهند (اعلام عمومی می‌کنند)، اما الگوریتم عیان سازی و کلید متناظر آن‌را از یکدیگر محرمانه نگاه می‌دارند. فرض کنید که الگوریتم‌های رمزگذاری هر یک به ترتیب، ECA و ECB است، و الگوریتم عیان سازی هر یک به ترتیب DCA و DCB است. ECA و DCA همانند ECB و DCB عکس یکدیگرند.

حال فرض کنید که A قصد دارد تکه‌ای از متن ساده Pرا برای B بفرستد. بجای محاسبه ساده ECB(P) و انتقال نتیجه آن، ابتدا A الگوریتم عیان سازی DCA را بر روی P اعمال کرده، سپس نتیجه را رمزگذاری کرده و به عنوان متن کد شده C به کاربر B انتقال می‌دهد:

ECB (DCA) P C =

به محض دریافت C، کاربر B الگوریتم DCB و سپس الگوریتم رمزگذاری ECAرا بر روی آن اعمال کرده و نتیجه نهایی متن ساده Pتولید خواهد کرد:

```
ECA (DCB) c
    = ECA (DCB) ECB (DCA) P
    = ECA (DCA) P /* because DCB and ECB cancel */
= P          /*because ECA and DCA cancel */
```

حال B می‌داند که پیام از طرف A آمده است، زیرا ECA در صورتی P را تولید می‌کند که الگوریتم DCA در فرایند رمزگذاری مورد استفاده قرار گرفته باشد، و آن الگوریتم را تنها A می‌داند، هیچ کس حتی کاربر B نمی‌تواند امضای A را جعل کند.

٦,١٧ امکانات SQL

SQL تنها از کنترل دست‌یابی محتاطانه پشتیبانی می‌کند. دو ویژگی SQL با استقلال بیشتر یا کمتر وجود دارند، راهکار دید که می‌تواند برای مخفی کردن داده‌های

حساس از کاربران غیر مجاز به کار می‌رود، و خود **زیر سیستم مجاز شماری**[1] که به کاربرانی با امتیاز ویژه این اجازه را می‌دهد که به طور انتخابی و پویا این امتیازات را به دیگر کاربران اعطا نمایند و متعاقباً اگر خواستند، این امتیازات را پس بگیرند. هر دو ویژگی در ادامه بحث می‌شود.

دیدها و ایمنی

برای نشان دادن استفاده دیدها به منظور اهداف امنیتی در SQL، شبیه SQL های مثال‌های دید (مثال‌های ۲ تا ٤) از بخش ۱۷,۲ را بررسی می‌کنیم.

```
2. CREATE VIEW LS AS
        SELECT C.S#, S.SNAME, S, STATUS,
S.CITY
        FROM S
        WHERE S.CITY = 'London';
```

این دید داده‌هایی را تعریف می‌کند که باید مجاز شماری روی آن تعیین شود. این کار توسط دستور GRANT انجام می‌شود، برای مثال:

```
GRANT SELECT, DELETE, UPDATE (SNAME,
 STATUS)
 ON LS
  TO Dan, Misha;
```

توجه کنید که، گویا چون آن‌ها توسط دستور GRANT خاصی تعریف شده‌اند و توسط دستور فرضی CREATE AUTHORITY تعریف نشده اند، در SQL مجوزها بدون نام هستند (در مقایسه محدودیت‌های جامعیتی که دارای نام بودند).

```
3. CREATE VIEW SSPPO AS
        SELECT S.S\#, S.SNAME, S.STATUS, S.CITY
        FROM S
        WHERE EXISTS
          (SELECT * FROM SP)
            WHERE EXISTS
```

---

[1] Authorization subsystem

```
        (SELECT * FROM P)
    WHERE S.S# = SP.S#
    AND SP.P# = P.P#
    AND P.CITY = 'Oslo'
```

GRANT متناظر:

```
GRANT SELECT ON SSPPO TO Lars;
4. CREATE VIEW SSO AS
        SELECT S.S#, (SELECT SUM) SP.QTY
                FROM SP
                WHERE SP/S#
= S.S# AS SQ
        FROM S;
```

GRANT متناظر:

```
GRANT SELECT ON SSQ TO Fidel
```

مثال ۵ از بخش ۱۷,۲ شامل مجوز وابسته به متن است. SQL از چندین عملگر توکار بدون عمل وند (تابع بدون آرگومان) استفاده می‌کند، مثل CURRENY_USER، CURRENT_DATE، CURRENT_TIME و امثال این‌ها. که می‌توان از آن‌ها در تعریف دیدهای وابسته به متن استفاده نمود. SQL از عمل گری شبیه عملگر (تابع) () DAY در مثال ۵ بکار رفته پشتیبانی نمی‌کند. بنابراین آن مثال را ساده‌تر می‌کنیم:

```
CREATE VIEW S_NINE_TO_FIVE AS
        SELECT S.S#, S.SNAME, S.STATUS, S.CITY
        FROM S
        WHERE CURRENT_TIME ≥ '09:00:00'
        AND CURRENT_TIME ≥ '17:00:00'
```

GRANT متناظر:

```
GRANT SELECT, UPDATE (STATUS)
ON S_NINE_TO_FIVE
TO ACCOUNTING
```

به هر حال توجه کنید که S_NINE_TO_FIVE بیشتر دیدی از نوع فرد است! مقدارش در طی زمان تغییر می‌کند حتی اگر داده‌های اصلی آن تغییر نکند. (گزاره متناظر آن چیست؟). به علاوه، دیدی در تعریف آن از عملگر (تابع) توکار CURRENT_USER استفاده شده است ممکن است مقادیر مختلفی را برای کاربران گوناگون داشته باشد. چنین دیدهایی با دیدهای معمولی که می‌شناسیم، فرق دارند– در حقیقت آن‌ها پارامتریک هستند.

مثال‌های که گفته شد این نکته را نشان دادند که مکانیسم دید به میزان زیادی ایمنی پایگاه داده را به طور رایگان افزایش می‌دهد (رایگان به این دلیل که این مکانیسم برای اهداف دیگری در سیستم قرار داده شده است). بسیاری از مجاز شماری، حتی آن‌هایی که وابسته به مقدار می‌باشند، بجای آنکه در زمان اجرا بررسی شوند در زمان ترجمه انجام می‌پذیرند، که این خود باعث افزایش کارایی می‌گردد. هر چند، رهیافت مبتنی بر دید نسبت به ایمنی در مواقعی از برخی ناتوانی‌های خفیف رنج می‌برد– به ویژه، اگر کاربران مختلف روی زیر مجموعه‌های گوناگون از یک جدول در یک زمان نیاز به امتیازات متفاوتی داشته باشند. برای مثال ساختار یک برنامه کاربردی را که هم اجازه می‌دهد تمام قطعات لندن پیمایش و نمایش داده شود و هم اجازه می‌دهد که برخی از آن‌ها (برای مثال، تنها قطعات قرمز رنگ) در حین پیمایش، به هنگام شوند را در نظر بگیرید.

اعطا (GRANT) و سلب (REVOKE)

مکانیسم دید به پایگاه داده این اجازه را می‌دهد که به طور مفهومی به روش‌های گوناگون به قسمت‌های تقسیم شود، بنابراین می‌توان اطلاعات حساس را از کاربران غیرمجاز مخفی نگاه داشت. اما این مکانیسم این اجازه را نمی‌دهد که مشخصاتی از عملگرهای که کاربر مجاز اجازه اجرای آن‌ها را دارد، بر روی این قسمت‌ها اجرا شوند. در عوض، همان‌طور که در مثال‌های بخش قبلی دیدیم، این کار

توسط دستور GRANT انجام می‌شود، که اکنون آن‌را با جزئیات بیشتری بررسی می‌کنیم.

اول از همه، ایجاد کننده هر شئ به طور خودکار تمام امتیازاتی که برای یک شئ تصور می‌شود را به آن اعطا می‌کند. برای مثال، ایجاد کننده جدول پایه T به طور خودکار امتیازات SELECT، INSER، DELETE،UPDATE ، REFERENCES و TRIGGER را به T اعطا می‌کند (این امتیازات را در ادامه شرح می‌دهیم). به علاوه، در هر مورد این امتیازات «با مجوز اعطا» اعطا می‌گردد، که بدین معنا است که کاربری که این امتیاز را دارد می‌تواند آن‌را به دیگر کاربران اعطا کند.

نحو دستور GRANT به این صورت است:

```
GRANT <privilege comma list>
        ON <object>
        TO < user ID comma list>
      [WITH GRANT OPTION];
```

۱. امتیاز (ات) s<privilege> مجاز USAGE، UNDER، SELECT، INSERT، DELETE، UPDATE، REFERENCES، TRIGGER و EXECUTE هستند. امتیازات SELECT، INSERT، UPDATE و REFERENCES همگی می‌توانند وابسته به ستون[1] (یعنی می‌توان ستون یک شئ را نیز مشخص نمود) باشند. توجه : ALL PRIVILEGES را نیز می‌توان مشخص کرد اما معنای آن روشن نیست.

- USAGE برای استفاده از نوع تعریف شده توسط کاربر به کارمی رود.

[1] Column specific

- UNDER الف برای ایجاد یک زیر نوع از نوع تعریف شده توسط کاربر است ب برای ایجاد یک زیر جدول از یک جدول
- SELECT، INSERT و UPDATE که نیاز به توضیح ندارند.
- REFERENCES بر روی جدولی تعریف می‌گردد که به آن جدول در یک محدودیت جامعیتی مراجعه می‌شود (هر محدودیتی، نه لزوماً محدودیت ارجاعی)
- TRIGGER این امتیاز بر روی جدول پایه که قصد داریم یک رهانا[1] بر روی آن تعریف کنیم.
- EXECUTE یک امتیاز روی یک روال SQL است که قصد داریم آن‌را فراخوانی کنیم.

۲. شئ (اشیاء) s<object> معتبر عبارتند از: نوع داده TYPE <type name>، جدول TABLE ,table name> و (برای اجرا) هر چیزی که یک روال مخصوص را فراخوانی کند که جزئیات آن خارج از بحث این کتاب است. (کلمه TABLE علاوه بر جداول پایه دیدها را هم شامل می‌شود)

۳. لیست شناسه کاربران <user ID comma list> که می‌توان بجای آن از کلمه کلیدی public استفاده نمود که معنای آن تمام کاربران شناخته شده سیستم است. نکته: SQL از نقش‌های تعریف شده توسط کاربر نیز پشتیبانی می‌کند، برای مثال ACCOUNTING که به معنای هر شخصی که در بخش حسابداری است. . وقتی که یک نقش ایجاد شد، می‌توان امتیازاتی را همانند یک شناسه کاربر عادی،

---

[1] trigger

به آن اعطا کرد. به علاوه، خود نقش‌ها می‌توانند همانند امتیازات، بـه یک شناسه کاربر یا نقش دیگر اعطا شود. به عبارت دیگـر، نقـش‌هـا مکانیسم SQL برای پشتیبانی از گروه‌های کاربری است.

٤. اگر WITH GRANT OPTION مشخص شود، معنای آن این اسـت که کاربران مشخص شده می‌توانند، امتیازات معینـی را بـر روی شـئ مشخصـی **با مجوز اعطـا**[1] بدسـت آورنـد- یعنـی آن‌هـا مـی‌تواننـد امتیازات روی آن شئ را به دیگر کاربران اعطـا کننـد. البتـه WITH GRANT OPTION تنها در صورتی می‌تواننـد مشـخص شـود، کـه کاربر صادر کنند دستور اعطا خود دارای مجوز اعطا باشد.

سپس، اگر کاربر A امتیـازاتی را بـه کـاربر B اعطـا نمـود، متعاقبـاً کـاربر A می‌تواند همان امتیازات را از کاربر B سلب کند (پس بگیرد). سلب امتیـاز بـه وسیله دستور REVOKE انجام می‌پذیرد که نحو آن بدین صورت است:

REVOKE [GRANT OPTION FOR] <privilege comma list>
ON <object>
FROM <user ID comma list>
<Behavior>;

در اینجا، GRANT OPTION FOR به معنای آن است که (تنهـا) مجـوز اعطـا سـلب شـود؛ (ب)<object> ، <privilege comma list> و <user ID comma list> همانند دسـتور GRANT اسـت، و (ج) <behavior> یـا محـدود (RESTRICT)است و یا آبشاری (CASCADE) است. برای مثال:

1. REVOKE SELECT ON S FROM Jacques, Anne, and Charley RESTRICT;
2. REVOKE SELECT, DELETE, UPDATE (SNAME, STATUS)
ON LS FROM Dan, Misha CASCADE;
3. REVOKE SELECT ON SSPPO FROM Lars RESTRICT;

---

[1] With grant authority

4. REVOKE SELECT ON SSQ FROM Fidel RESTRICT;

اکنون RESTRICT را در برابر CASCADE بررسی می‌کنیم. فرض کنید که p امتیازی روی یک شئ باشد، اگر کاربر A امتیاز p را به کاربر B اعطا کند، کـه او (کاربر B) آن‌را به کاربر C اعطا کند. اگر کاربر A این امتیاز p را کاربر B پس بگیـرد، چه اتفاقی می‌افتد؟ لحظه‌ای را در نظر بگیرید که این سلب امتیاز صورت گرفتـه اسـت، پس امتیاز p که در اختیار کاربر C است"متروکه[1] «شده است–این امتیاز از طرف کـاربر B اعطا شده که در حال حاضر فاقد آن است. هدف گزینـه RESTRICT در برابـر CASCADE، جلوگیری از امتیاز متروکه است. اگر از RESTRICT استفاده شـود این امتیاز متروکه باقی می‌ماند اما اگر از CASCADE اسـتفاده شـود امتیـاز p نیـز از کاربر C نیز گرفته می‌شود، پس امتیاز متروکه ایجاد نمی‌شود.»

سرانجام، حذف یک نوع، جدول، ستون یا یک روال بـه طـور خودکـار تمـام امتیازات بر روی آن شئ حذف شده را از تمام کاربران سلب می‌کند.

---

[1] abandoned