



# طراحی امنیت پایگاه داده ها

رسول جلیلی

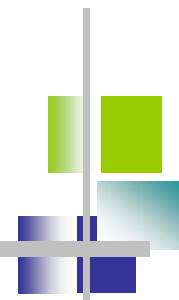
[jalili@sharif.edu](mailto:jalili@sharif.edu)





## مقدمه

- تمرکز ما در مطالب قبل بروی مفاهیم امنیت پایگاه داده های منطقی (Logical DB Sec.) بوده است.
- از این پس بروی طراحی معیارهای امنیتی پایگاه داده های منطقی متمرکز میشویم.
- داشتن یک پایگاه داده امن، باید از اولین مراحل طراحی پایگاه داده در نظر گرفته شود.
- متدولوژی طراحی مورد نیاز است.

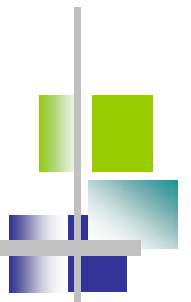




## مقدمه - ۲

• باید در تمام فازهای طراحی پایگاه داده نکات امنیتی در نظر گرفته شوند:

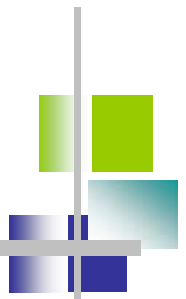
- تحلیل (Analysis)
- طراحی مفهومی (Conceptual design)
- طراحی جزئی (Detailed design)
- پیاده سازی (Implementation)
- تست (Test)
- نگهداری (Maintenance)





# طراحی DBMS امن-۱

- برای داشتن امنیت پایگاه داده باید مکانیزمها در هر دو سطح **DBMS** و **OS** اعمال گردند.
- نمی توان به امنیت **DBMS** به عنوان یک توسعه ساده بروی توابع **OS** زیرین نگاه کرد.
- تفاوت‌های میان **OS** و **DBMS** را از منظر نگرانیهای امنیتی می توان به صورت زیر لیست نمود:
  - دانه بندی اشیا:
    - درجه دانه بندی در **OS** در سطح **فایل** است
    - درجه دانه بندی **ریزتر** در **DBMS** در سطح **رابطه، سطر، فیلد** و ... است
  - وجود روابط معنایی میان داده ها در **DBMS**
  - فراداده های موجود در **DBMS**
    - فراداده ها بسیار حساس هستند و باید به شدت محافظت شوند.
    - در **OS** معمولاً هیچ نوع فراداده وجود ندارد
  - اشیا در **OS** و **DBMS**
    - در **OS** اشیا **فیزیکی** مثل فایل (**file**)
    - در **DBMS** اشیا **منطقی** مثل دید (**view**)





## طراحی DBMS امن-۲

- ادامه تفاوت های امنیتی میان OS و DBMS
  - چندین انواع داده ای در DBMS
    - در DBMS چندین نوع داده ای و بالتبع چندین نوع مد دسترسی مانند مد مدیریتی و مد آماری
    - در سطح OS برای دسترسی فیزیکی سه مد دسترسی **R**، **W** و **X** وجود دارد
  - اشیای ایستا و پویا
    - اشیای فیزیکی در OS در مقابل دیدها در DBMS
    - نیاز به محافظت بیشتر برای اشیای پویا
  - تراکنشهای چند سطحی
    - تراکنش ها در DBMS داده ها در سطوح مختلف امنیتی را درگیر می کنند که باید به صورت امن اجرا گردند.
    - در سطح OS تنها بعضی عملیات پایه ای مانند **write**، **read** و **execute** اجرا میشوند که همه در یک سطح امنیتی قرار دارند.
  - چرخه حیات داده (Data Life cycle)
    - داده ها در پایگاه داده ها یک چرخه حیات بلندی دارند و DBMS باید حفاظت از آنها را در طول حیات آن داده تضمین نماید.



# مکانیزم های امنیتی در DBMS-۱

• مکانیزم های امنیتی که DBMS باید تامین کند:

– درجات مختلف دانه بندی برای دسترسی

• برای پایگاه داده های رابطه ای: رابطه ها، تاپلها، پایگاه داده ها

– حالات دسترسی مختلف

• کنترل دسترسیهای مختلف با توجه به نوع عمل باید اعمال گردند

– مثلاً میان خواندن (read) و نوشتن (write) باید تمایز قائل شد.

– انواع مختلف کنترل دسترسیها برای یک درخواست دسترسی

• وابسته به اسم (Name-dependant): کنترل دسترسی به اسم اشیا وابسته است

• وابسته به داده (Data-Dependant): کنترل دسترسی به محتوای اشیا وابسته است

• وابسته به زمینه (Context-Dependant): وابسته به زمان، مکان و ...

• وابسته به تاریخچه (History-Dependant)

• وابسته به نتیجه (Result-Dependant): وابسته به نتایج پرس و جوها





## مکانیزم های امنیتی در DBMS-۲

### – مجازشماری پویا

- DBMS باید تغییرات مجازشماریهای کاربران در حین اینکه پایگاه داده ها عملیاتی است، پشتیبانی نماید.

### – حفاظت چند سطحی

- در نظر گرفتن برچسبهای امنیتی و تخصیص آنها به اشیا و عاملها
- در هر شی، می تواند به داده های مختلف برچسبهای مختلف تخصیص بدهد.

### – عاری از پنهان (Covert Channel):

- به این معنا که کاربران نباید امکان بدست آوردن اطلاعات غیر مجاز با استفاده از متدهای ارتباطی غیر مستقیم را پیدا کنند.

### – کنترل استنتاج (Inference Control)

- عموماً در بکارگیری توابع aggregation که می توان به کمک این توابع به استنتاجهای دیگری از پایگاه داده دست پیدا کرد.





## مکانیزم های امنیتی در DBMS-۳

### چند نمونه سازی (Polyinstantiation)

- چندین نمونه از یک قلم داده هر یک سطح رده آنها متفاوت باشد

### بازنگری (Auditing)

- اطلاعات مربوط به بازنگری برای کنترل استنتاج مفید می باشند.

### نبود هر نوع back door

- تنها دسترسی از طریق DBMS امکان پذیر باشد.

### یکنواختی مکانیزمها (Uniformity of Mechanisms)

- برای پشتیبانی خط مشی های مختلف و تمام کنترل های مرتبط با امنیت بعضی مکانیزمهای کلی و عمومی باید مورد استفاده قرار گیرد.

### کارایی مناسب

- کنترل های امنیتی باید کارایی سیستم را خیلی تحت شعاع قرار ندهند. هرچند به هر حال افت کارایی غیر قابل اجتناب است.







# اصول پایه ای صحت اطلاعات-۱

- تراکنش های خوش تعریف (**Well-formed Transactions**)
  - بجای رویه های اختیاری (**Arbitrary Procedures**)
- کاربران تصدیق هویت شده (**Authenticated Users**)
  - تغییرات تنها باید بوسیله کاربران مجاز انجام شود.
- حداقل مجوز (**Least Privilege**)
  - یک اصل برای محدودسازی کاربران برای کارکردن با مجموعه کمینه مجوزها و منابع مورد نیاز برای انجام کارهایشان
- تفکیک وظایف (**Separation of duties**)
- ادامه پذیری یک عمل (**Continuity of Operation**)
  - با کمک مفاهیم replication
- دوباره سازی وقایع (**Reconstruction of Events**)
  - رفتار اشتباه باید کشف و حق داده شده بلاسفاده شود.
  - با کمک مفاهیم auditing و after-Image و before-image



## اصول پایه ای صحت اطلاعات-۲

- چک کردن با واقعیت
  - چک کردن به صورت پریودیکی با موجودیت های دنیای واقع برای نگهداری از مقادیر درست داده ها در سیستم
- سادگی استفاده امن
  - ساده ترین راه استفاده از سیستم باید همزمان امنترین هم باشد.
- تفویض مدیریت (Delegation of Authority)
  - انجام تفویض باید منعطف و در عین حال به نحوی باشد که امنیت دور زده نشود.



# مدل مجاز شماری System R-۱

- از نخستین DBMS های IBM
- **جداول** به عنوان **اشیایی** هستند که مورد حفاظت قرار می گیرند
  - این جداول می توانند **جدولهای پایه** و یا **دیدها** باشند
- عملها همان کاربران سیستم هستند که به سیستم DB دسترسی پیدا میکنند.
- حالات دسترسی زیر را می تواند در نظر گرفت:
  - Read: برای خواندن تاپلها از یک جدول
  - Insert: برای اضافه کردن تاپلها به جدول
  - Delete: برای حذف بعضی تاپلها از یک جدول
  - Update: برای تغییر بعضی تاپلهای موجود در یک جدول
  - Drop: برای حذف کامل یک جدول از پایگاه داده ها
- تمام حالات دسترسی به یک جدول به صورت کلی نگاه می کنند مگر update که به یک ستون هم می تواند ارجاع داده شود.



## مدل مجازشماری System R-۲

- مدل، مدیریت مجازشماری ها را به صورت **توزیع شده** (Decentralized) پشتیبانی میکند.
- **هر کاربر** پایگاه داده می تواند این امکان را داشته باشد که یک **جدول جدید** ایجاد نماید.
- اگر یک کاربر جدولی را ایجاد نماید می تواند به صورت کاملاً مجاز **هر نوع** مجوزی را بروی آن جدول داشته باشد.
  - مگر در شرایطی که این جدول یک **دید** باشد
- **مالک** جدول (ایجاد کننده آن) این حق را دارد که به دیگر کاربران بعضی مجوزها را **اعطا** نماید.
  - با اعطای یک حق به کاربر دیگر به **مجموعه مجازشماریهایی** که در سیستم نگهداری می شود یک مجازشماری اضافه می شود.
  - اعطای یک مجازشماری می تواند به همراه **grant option** برای دادن **حق اعطای مجوز** به دیگر کاربران همراه باشد.



## مدل مجازشماری System R-۳

- هر مجازشماری میتواند به صورت یک چندتایی معرفی گردد: (s, p, t, ts, g, go)
  - s: عامل یا کسی که به او یک مجوز اعطا می شود
  - p: همان حق و یا مجوزی است که اعطا می شود.
  - t: جدولی است که مجازشماری به آن ارجاع داده می شود
  - ts: زمانی است که در آن زمان عمل اعطا انجام شده است.
  - g: عاملی است که مجوز را اعطا کرده است.
  - go: همان grant option است که یکی از مقادیر {yes,no} می باشد

• مثال:

<B, select, T, 10, A, yes>

<C, select, T, 20, B, no>

- اگر عمل مجازشماری update باشد، ستون مربوطه نیز باید مشخص باشد.  
زمان را به این دلیل نگه می داریم که بتوانیم عمل revoke را انجام دهیم (بعداً توضیح می دهیم).



## مدل مجازشماری System R-۴

- دستور grant در SQL به صورت زیر خواهد بود:

$$\text{GRANT} \begin{cases} \text{ALL RIGHTS} \\ \langle \text{Privileges} \rangle \\ \text{ALL BUT} \langle \text{Privileges} \rangle \end{cases} \text{ ON } \langle \text{table} \rangle \text{ TO } \langle \text{user-list} \rangle [\text{WITH GRANT OPTION}]$$

- عامل اعطا کننده می تواند به جای **user-list** می تواند از **PUBLIC** استفاده نماید که در این صورت به تمام کاربران پایگاه داده ها این حق بروی جدول مربوطه اعطا خواهد شد.
- هر کاربر که حق اعطا داشته باشد حق بازپس گیری را هم دارد. هرچند هر عامل تنها حق دارد مجازشماری هایی را که توسط او اعطا شده اند، بازپس بگیرد.
- دستور revoke در SQL به صورت زیر خواهد بود

$$\text{REVOKE} \begin{cases} \text{ALL RIGHTS} \\ \langle \text{Privileges} \rangle \end{cases} \text{ ON } \langle \text{table} \rangle \text{ FROM } \langle \text{user-list} \rangle$$




## مدل مجازشماری System R-۵

- مکانیزم عمل revoke

– System R عمل بازپس گیری را به صورت بازگشتی (Recursive) و یا آبشاری (Cascading) انجام می دهد.

– باز پس گیری **p** بروی **t** از کاربر **y** بوسیله کاربر **x** به این ترتیب تعریف می شود که تمام مجازشماریهای **p** بروی **t** اعطا شده توسط **x** به **y** هرگز اعطا نشده است.

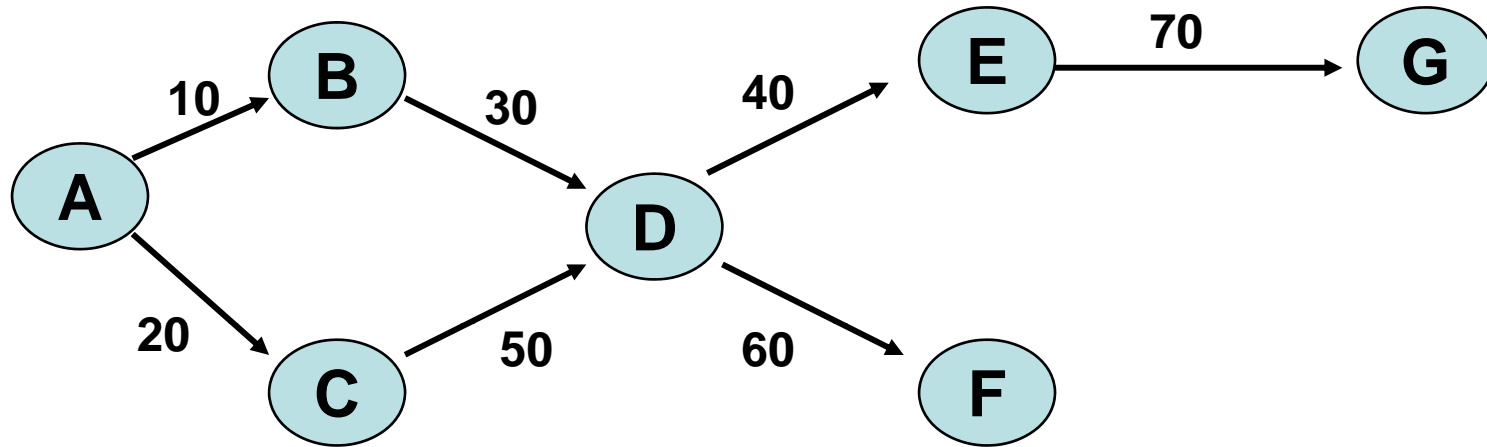
- تمام تاثیر این اعطا باید از بین می رود.

- مثال در اسلاید بعدی

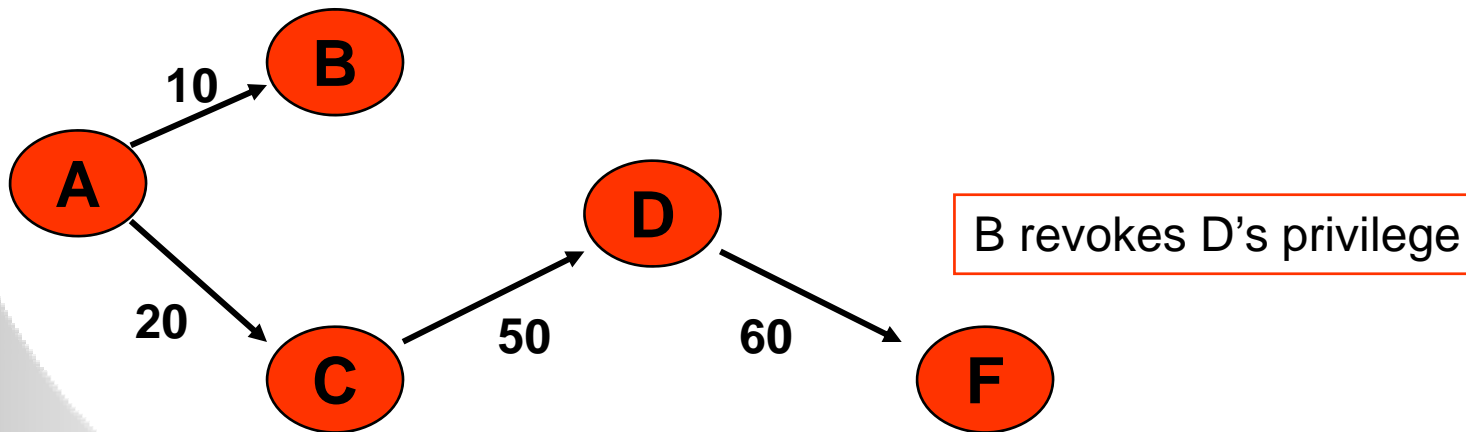




## مدل مجاز شماری System R-۶



B has granted a privilege to D, who has passed it to E, who has passed it to G







# مدل مجازشماری $\nu$ -System R

## • دید

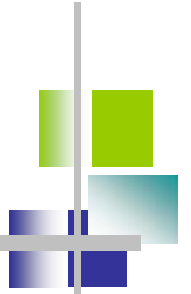
- System R این امکان را به تمام کاربران می دهد که بعضی دید ها را بروی **جداول پایه** و یا **دیگر دیدها** تعریف نمایند.
- این یک روش ساده و کارا برای پشتیبانی مجازشماری **وابسته به محتوا** میباشد.
- برای مثال:
  - کاربر B جدول B را می سازد و می خواهد به کاربر A تنها مجوز دسترسی به تاپلهایی را بدهد که  $salary > 1000$
- در این صورت می تواند یک دید بروی آن بسازد و بروی آن دید مجازشماری به A را اعطا نماید





## مدل مجاز شماری System R-۸

- مقایسه میان حقوق روی جداول پایه و دیدها
  - وابسته به این که معنای دید (View Semantics) چیست، یک کاربر مالک دید ممکن است مجوزهای محدودتری نسبت به جداول پایه داشته باشد.
  - داشتن یک حق بروی یک دید کاملاً به این که آیا این حق بروی تمام جداول پایه وجود دارد یا نه بستگی دارد.
  - اگر کاربر بروی تمام جداول پایه مجوزی را با **grant option** داشته باشد، این مجوز بروی دید مربوطه را هم با **grant option** خواهد داشت.
  - یک برچسب زمانی تعریف می شود (Time Stamp) که در زمان تعریف دید است.
  - حقوق مالک دید در زمان برچسب زمانی مربوطه تعریف می گردد.





# مدل مجازشماری System R-۹

## • پیاده سازی مدل-۱

– اطلاعات مجازشماری مربوط به کاربران برای دسترسی به جداول پایگاه داده در دو رابطه به نام‌های **SYSAUTH** و **SYSCOLAUTH** ذخیره می شود.

– **SYSAUTH** صفات زیر را دارد:

- **UserId**: کاربری که مجازشماری ها در مورد او می باشد
- **Tname**: جداولی که مجازشماری ها بروی آن انجام می شود.
- **Type**: نوع جدول **Tname** را مشخص می کند. (R=جدول پایه و V=دید)
- **Grantor**: کاربری که به **userid** حق را اعطا کرده است.
- **Read**: نشاندهنده زمانی است که مجوز **read** به کاربر داده شده است(مقدار پیش فرض ۰ است)
- **Insert**: نشاندهنده زمانی است که مجوز **insert** به کاربر داده شده است(مقدار پیش فرض ۰ است)
- **Delete**: نشاندهنده زمانی است که مجوز **delete** به کاربر داده شده است(مقدار پیش فرض ۰ است)
- **Update**: نشاندهنده ستونهایی است که مجوز **update** به کاربر در آن ستون ها داده شده است
- **Grantopt**: نشاندهنده آن است که آیا مجوز داده شده با **grant option** هست یا خیر



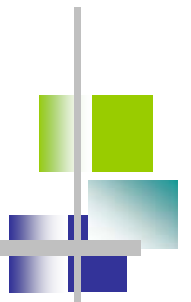
# مدل مجازشماری System R-۱۰

## • پیاده سازی مدل-۲

- جدول SYSCOLAUTH زمانی مورد نیاز است که در جدول SYSAUTH در یک تاپل در **ستون Update** کلمه **some** آمده باشد. در این صورت در این جدول تاپلهایی ظاهر میشود.

- SYSCOLAUTH صفات زیر را دارد:

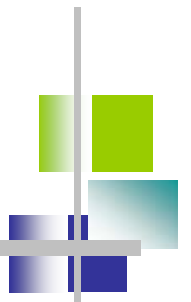
- **UserId**: کاربری که مجازشماری **update** در مورد او می باشد
- **Table**: نشاندهنده جدولی که حق **update** بروی آن تعریف شده است.
- **Column**: نام ستونی که حق **update** بروی آن تعریف شده است.
- **Grantor**: نام کاربری که این حق را اعطا کرده است.
- **Grantopt**: نشاندهنده اینکه آیا این مجوز با **grant option** داده شده است یا خیر





# مدل مجازشماری System R-۱۱

- گسترش بر مدل-۱
  - در سال ۱۹۸۲ ویلمز و لینزدی سعی کردند **مدیریت گروه‌های کاربران** را در نظر بگیرند.
  - **مجموعه ای کاربران** را یک **گروه** در نظر می‌گرفت
  - گروه‌ها می‌توانستند **همپوشانی** داشته باشند.
  - در مکانیزم‌های مجازشماری هم گسترشی دادند که یک **DBMS** برای **System R\*** به صورت **توزیع** شده داشته باشد





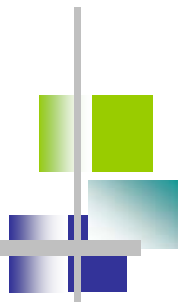
## مدل مجازشماری System R-۱۲

- گسترش بر مدل-۲

- در سال ۱۹۹۳ توسط برتینو و دیگران

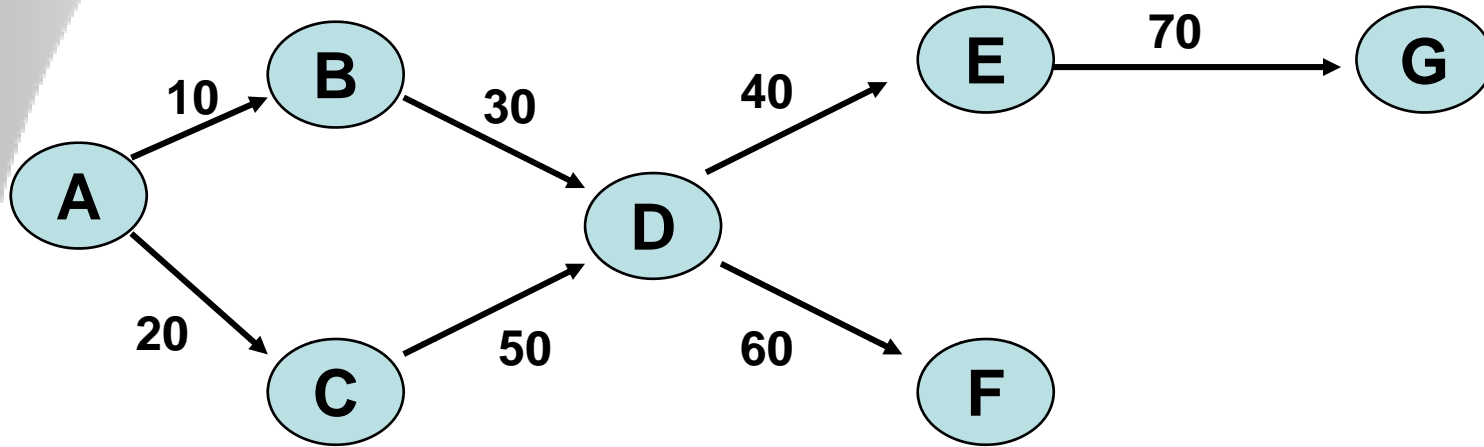
- داشتن **cascadable revoke** و یا **non-cascadable revoke**

- نمونه ای از **non-cascadable revoke** در اسلاید بعدی

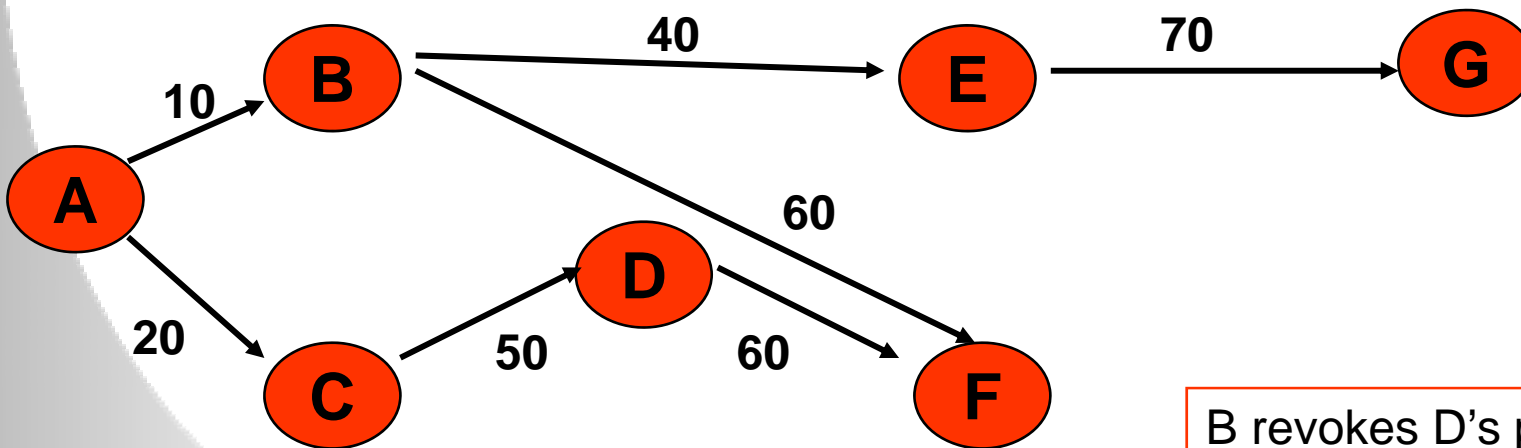




# مدل مجاز شماری - System R-۱۳



B has granted a privilege to D, who has passed it to E, who has passed it to G



B revokes D's privilege



# ویژگیهای اصلی معماری پایگاه داده های امن

• SDBMSها با دو مد اصلی می توانند کار میکنند:

## – System-High

- در این روش به تمام کاربران بالاترین سطح امنیتی ممکن داده می شود.
- در هنگام آشکارسازی داده یک گارد مسئول مرور داده ها است که به درستی آشکار شوند.
- ریسک امنیتی بالایی در این مد وجود دارد
- استفاده از DBMSهای موجود امکان پذیر خواهد بود البته با پرداخت کمی هزینه

## – Multilevel

- براساس استفاده از DBMSهای مورد اعتماد (Trusted) و یا غیر قابل اعتماد (Untrusted)

## – Trusted Subject Arch.

» با استفاده از یک DBMS و OS مورد اعتماد

## – Woods Hole Arch.

» استفاده از یک DBMS غیر قابل اعتماد با فیلترهای مورد اعتماد اضافی

» بر سه نوع است: Integrity Lock-kernelized-Replicated







# انواع معماری های امن

Architecture	Research prototype	Commercial DBMS
Integrity Lock	Mitre	TRUDATA
Kernelized	SeaView	Oracle
Replicated	NRL	
Trusted Subject	A1 Secure DBMS	Sybase Informix Ingres Oracle DEC



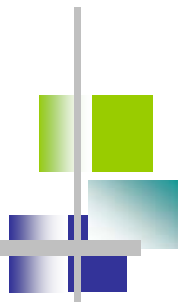
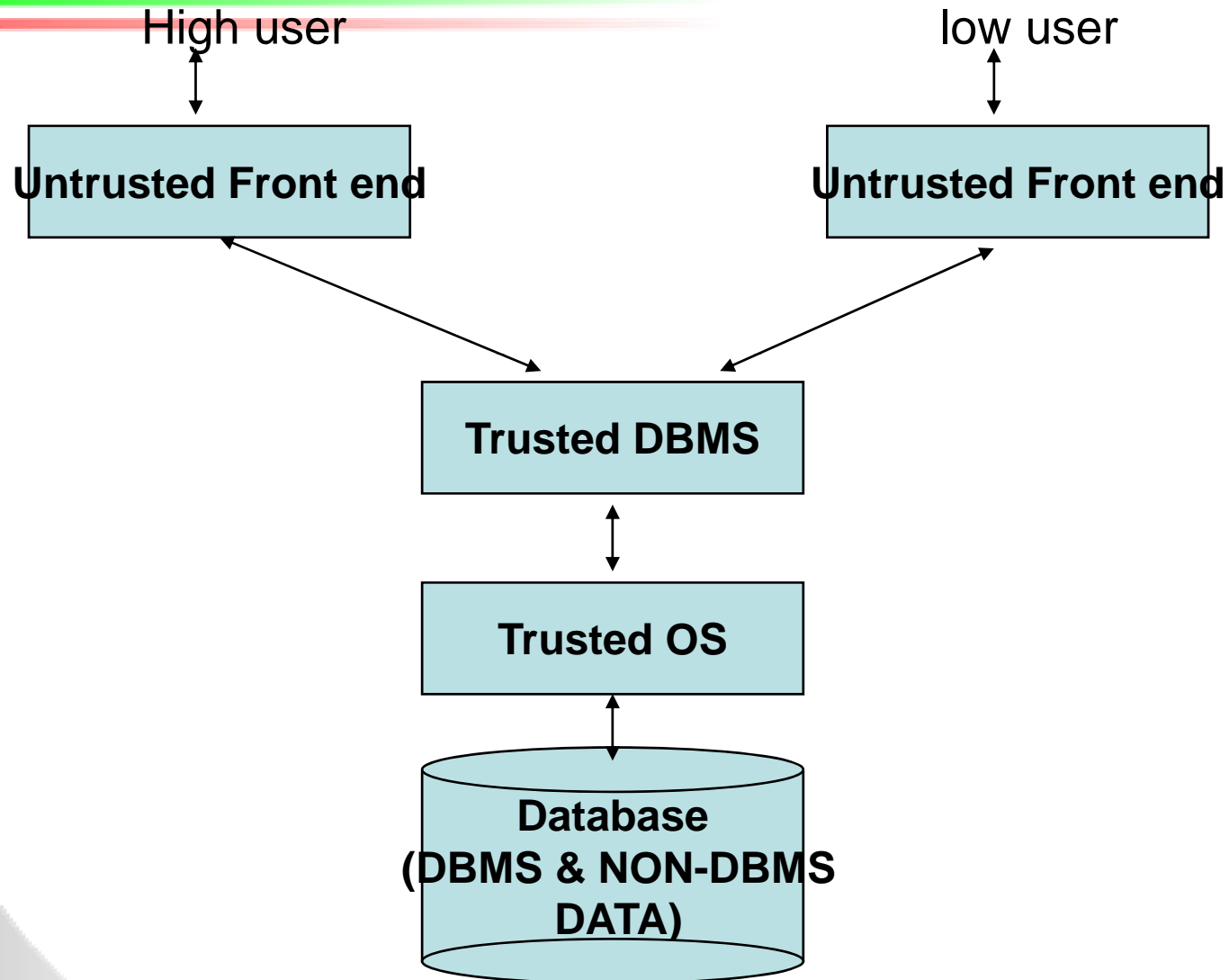
# Trusted Subject Arch

- مجموعه ای از front-end های غیر قابل اعتماد برای واسط قرار دادن آن را سطوح امنیتی متفاوت دارد (high و low)
- TOS و TDBMS یک TCB (Trusted Computer Base) را تشکیل می دهند.
- DBMS مسئول حفاظت از اشیای چند سطحی پایگاه داده است
- سطح بالا بر سطح پایین تفوق دارد
- Sybase از این راه حل با بکارگیری برچسبهای سطح تاپل استفاده می کند.





## ۲- Trusted Subject Arch





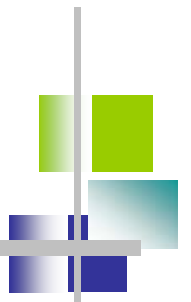
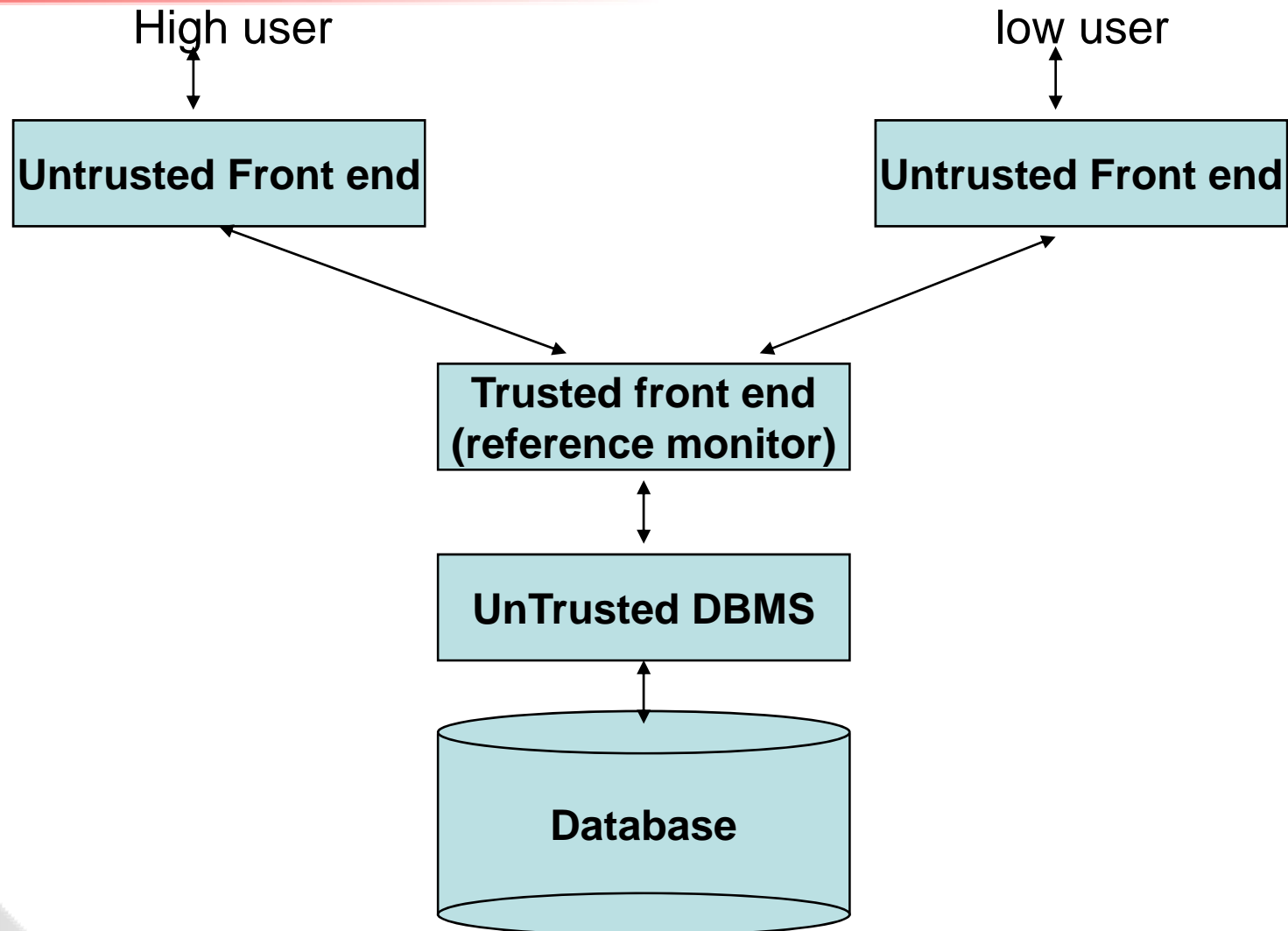
# ۱- Woods Hole Archs

- مجموعه ای از **front-end** های غیر قابل اعتماد وجود دارد
- پس از این مجموعه یک واسط مورد اعتماد وجود دارد که عمل نظارت را انجام داده و نمیتوان آن را دور زد.
- این ناظر سپس با یک **back-end** غیر قابل اعتماد در رابطه است.





## ۲-Woods Hole Archs





# ۱- Integrity Lock Arch

- به یک UFE (Untrusted Front-end) متصل است که اعمال پیش و پس پردازشی را بروی پرس و جو انجام میدهد.
- یک TFE (Trusted Front-end) نیز دارد که میان UFE و DBMS غیر قابل اعتماد قرار دارد. این لایه به عنوان یک فیلتر مورد اعتماد (Trusted Filter) عمل می کند.
- stamp یک فیلد مخصوص برای یک شیء است که اطلاعات مربوط به برچسب امنیتی و دیگر داده های کنترلی را در یک فرمت رمز شده، نگه می دارد.
- stamp بوسیله TFE تولید و واریسی میکند.
- مشکل اصلی نشت اطلاعات غیر مجاز و همین طور استنتاج می باشد.
  - به علت استفاده از selection و projection
  - حل این مسائل در TFE و یا UFE است نه در DBMS



High user

low user

Untrusted Front end

Untrusted Front end

Trusted Filter

Cryptographic Unit

Append Stamp

Check Stamp

Query

Store

Response

UnTrusted DBMS

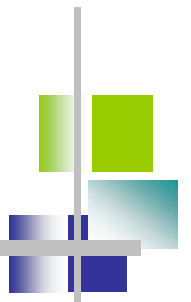
Database



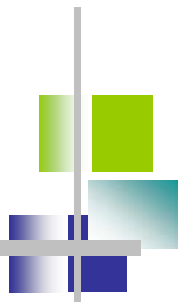
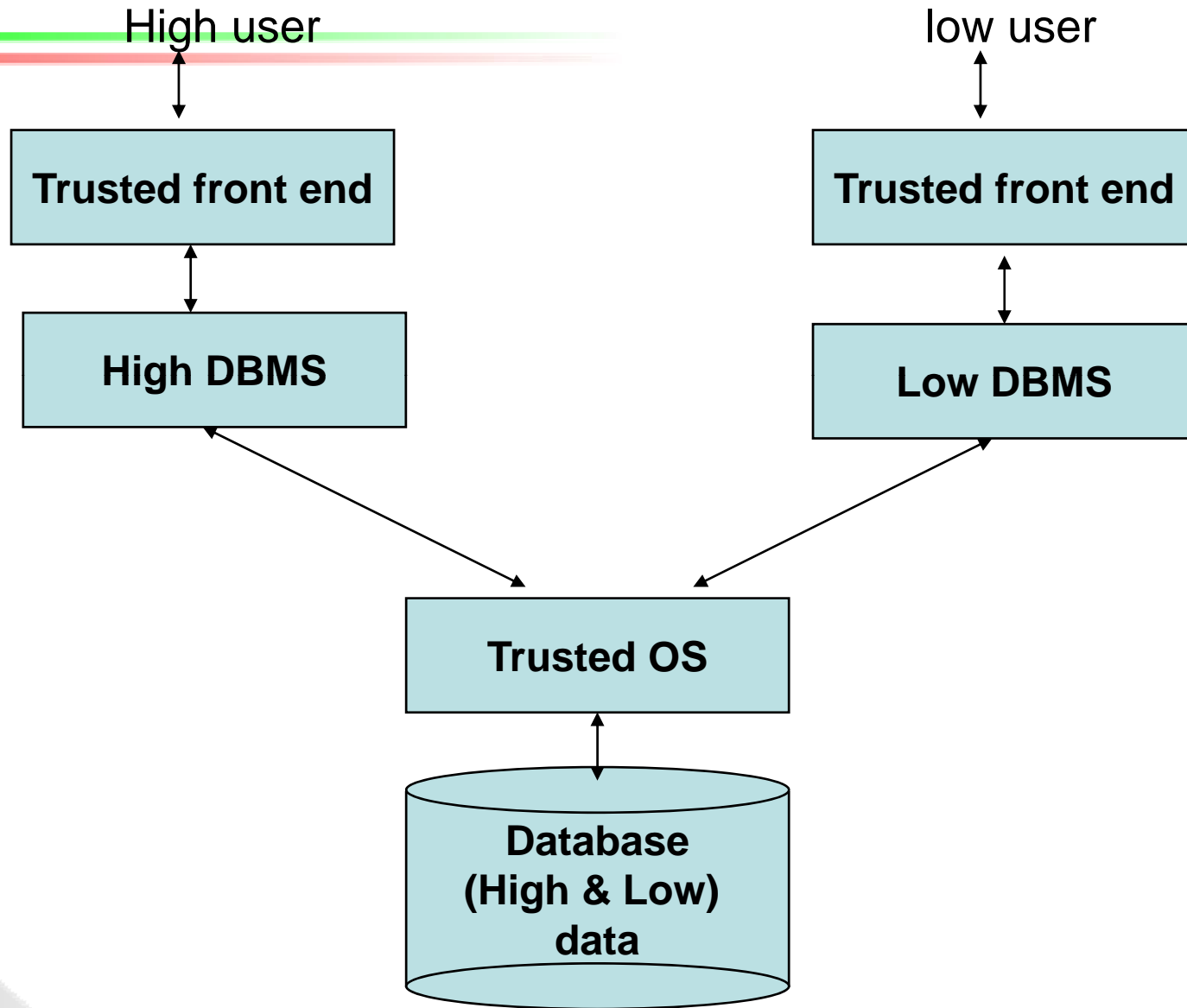


# Kernelized Arch.

- یک OS مورد اعتماد مورد استفاده قرار می گیرد که مسئول دسترسی فیزیکی در DB و اعمال کنترل دسترسی اجباری است
- اشیای DB دارای برچسبهای امنیتی مشابه ذخیره شده در OS مورد اعتماد می باشند.
- در اینجا دسترسی چند سطحی را به دسترسی های تک سطحی تبدیل می کند.









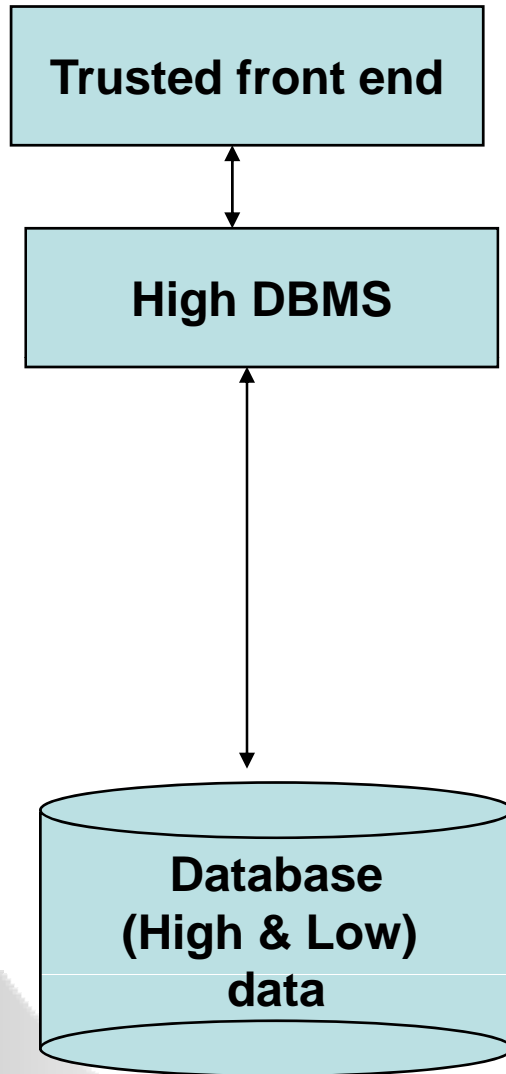
# Replicated Arch.

- این معماری بسیار گران است
- در محیط واقعی تاکنون پیاده سازی نشده است
- شکل در اسلاید بعدی

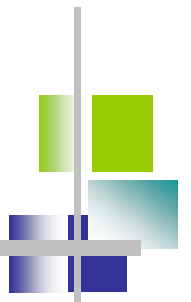
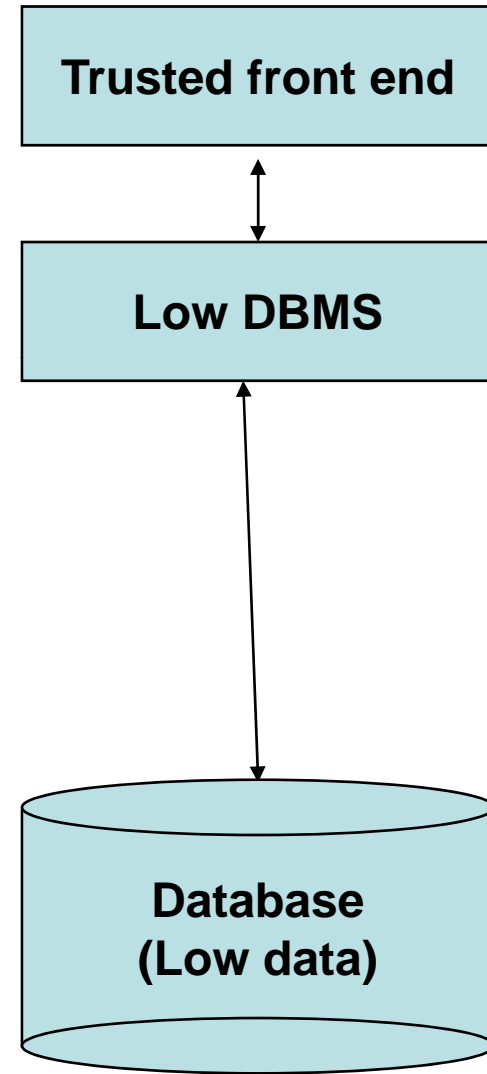




High user



low user





# با تشکر

مرکز امنیت شبکه شریف

<http://nsc.sharif.edu>

