Views

- یک متغیر رابطه‌ای مشتق شده از متغیرهای رابطه‌ای پایه

- یک پرس و جوی نامگذاری شده

- یک جدول مجازی

مثال: تامین کنندگانی که در شهر تهران نباشند

```
Create view S1 as ( select S#, STATUS, CITY
                    from Supplier
                    where CITY <> 'تهران')
```

مثال: تامین کنندگان غیر تهرانی که وضعیت آنها بیشتر از ۱۵ است.

```
select S#, STATUS, CITY                 select S#, STATUS, CITY
from S1                        =>       from S
where S1.STATUS > 15                    where S1.STATUS > 15 and S.city
                               |
                               v
```

جانشینی (به ازای هر وقوع از S_1 عبارت معادل را قرار می‌دهیم.)

کاربردهای دیدگاه‌ها

۱- ساده سازی فرمولبندی پرس و جوها

۲- دادن حق دسترسی و دستیابی به هر بخش از پایگاه داده (ایجاد امنیت برای داده‌های پنهان)

۳- بخشی از جدول خارجی (به داده یکسان دیدهای مختلف است)

۴- پنجره‌ای منطقی روی یک یا چند جدول (استقلال منطقی)

رشد → ستون جدید یا جدول جدید اضافه می‌کنیم

سازماندهی مجدد ← مجبور هستیم یک جدول را به بیش از یک جدول تجزیه کنیم.

بدون Viewها

اشکال پدید میشه

جامعیت ارجاع

Integrity Constraints اسم کیفیت کلی و صحت داده ها را ارزیابی

* I.C یک مانع در مقابل آسیب های احتمالی به پایگاه داده است

* I.C با تشخیص تغییراتی که ممکن است تاثیرات منفی روی ثبات پایگاه داده ها داشته باشد.

I.C

- Domain Constraints
- Referential Integrity
- Assertions
- Triggers
- Functional Dependency

Domain Constraints

- تعریف مقادیر معتبر برای صفات

- ابتدایی ترین روش I.C

- با استفاده از check می توانیم در SQL این نوع محدودیت ها را ایجاد کنیم

مثال

```
Create Domain hourly-wage numeric (5, 2)            محدودیت
       Constraint value-test Check (Value >= 4)     True / false
                  نام محدودیت
```

مثال

```
Create Domain AccountType Char(10)
       Constraint ATTest
              check ( value in ( 'checking', 'saving'))
                                     ↓
                                   select
```

Referential Integrity

مقادیر جهت خارجی با مقادیر آن در جدول اصلی مطابقت داشته باشند

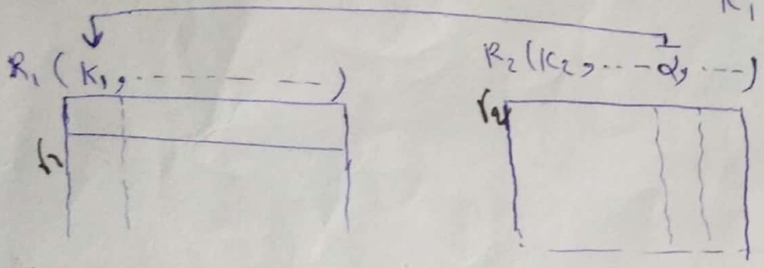
ACCOUNT (account-no, branch-name, ~~balance~~) balanc, cus-id)

branch (branch-name, branch-city)

customer (cus-id, name, ...)

loan (l-id, branch-name, cus-id, amount)

جبر رابطه‌ای $\longrightarrow$ $\pi_{\alpha}(r_2) \subseteq \pi_{k_1}(r_1)$



Insertion *

چنانچه چندتایی t_2 به r_2 اضافه شود، باید در r_1، چندتایی باشد t_1 وجود داشته باشد

که $t_1[k]=t_2[\alpha]$

یا

$$t_2[\alpha] \in \Pi_k(r_1)$$

Deletation *

اگر t_1 از r_1 حذف شود، پایگاه داده، تمام چندتایی که در r_2 موجود است که مقدار کلید خارجی آنها $t_1[k]$ است را باید جمع آوری کند.

$$\sigma_{\alpha=t_1[k]}(r_2)$$

Update: *

* اگر t_2 در r_2 به روز شود و مقدار $t_2[\alpha]$ باید در r_1 موجود باشد (مانند Insert) $(t'_2[\alpha] \in \Pi_k(r_1))$

* اگر t_1 در r_1 به روز شود. باید تمام مقادیر t_1 مقادیر در r_2 جمع آوری شود (مانند Delete)

$$\sigma_{t_1[k]}(r_2)$$

Assertions

یک گزاره شرطی است که پایگاه داده همیشه باید آن شرط را برآورده کند

Create Assertion <ن> Check < گزاره >

* سربار اضافی زیادی روی سیستم دارد و باید به دقت استفاده کرد

شعبه بانک — جدول وام ها — جدول حساب ها

```
Create assertion   sum-cons   check(
    not exists (select * from  branch
        where  ( select  sum (amount)  from  loan
                 where  loan.branch-name = branch.branch-name)
                 >=
                 ( select sum (~~amount~~ balance) from  account
                   where ~~loan~~ :number-name = branch.branch-name
                                                 account
```

مثال: حجم وام داده شده کمتر از مجموع سپرده باشد.

Branch(Branch_name,

loan(loan_id, Branch-name, amount)

account(account-id, balance, Branch-name)

مثال: کسی می‌تواند وام دریافت کند که حساب هایش 1000$ مبلغ داشته باشد.

```
Create assertion   balance-cons   Check (
    not Exists ( select * from  loan
        where  not exists (
            select  *  from  customer, account
            where  customer.id = account.cust-id and
                   customer.id = loan.cust-id  and
                   account.balance >= 1000)
)
```

می‌توان این جدول را ننویسیم

account.cust-id = loan.cust-id

مثال: کسی می‌تواند وام دریافت کند که حداقل یک حساب با بیش از 1000$ مبلغ داشته باشد.

Triggers

زمانی اجرا می‌شوند که اتفاقی در سیستم رخ دهد

برای ایجاد Triggre ها

۱- تعریف شرایطی که Triggre ها ایجاد شوند

۲- عملی که باید در آن شرایط انجام شود.

در SQL92 نیست

مثال) وضعیت STATUS هر تامین کننده حداقل برابر 5 است

```
Create ASSERTION  ICB  Check
        (( select  min (S.Status) From S)) >4)
```

مثال) قطعات قرمز باید در انبار london باشند

```
Create  assertion  IC97  Check(
       not  exists  (selecT *  from P  where  p.color = "red"  P.CiTy <> 'london')
```

مثال ۲) وزن هیچ ارسالی (وزن قطعه * تعداد ارسالی) نباید بیشتر از ۲۰۰۰۰ باشد

```
Create  assertion  IC 64  Check
    (noT  exists (SelecT * from P, SPj  where  p.p# =SP.j.p# and
                 (p.weighT * SPj.QTy) > 20000))
```

مثال: هیچ تامین کننده‌ای با وضعیت کمتر از ۲۰ نمی‌تواند قطعه‌ای به میزان بزرگتر از ۲۰۰ ارائه کند.

```
Create  assertion  IC65  Check  (NoT  exists (
  selecT  *  from S, SPj
  where    S.StaTuse <20
  And      S.S# = SPj.S#
  And      SPj.QTY > ~~500~~ 200)
```