Uvod v podatkovne shrambe

1. Teme

- Namen po podatkovnih zbirk
- Pogled na podatke
- Definicijski jezik podatkov (DDL – data definition language)
- Jezik za manipulacijo podatkov (DML – data manipulation language)

2. DBMS – Database Management System

- Podatkovna baza je zbirka povezanih podatkov
- Vsebuje tudi programe za dostop do podatkov,
- DBMS vsebuje informacije o specifičnem produktu
- Aplikacije:
    - Bančništvo: transakcije denarja
    - Letalstvo: rezervacije, urniki,
    - Fakultete: prijave, ocene,
    - Prodaja: potrošniki, produkti, nakupi,
    - Industrija: produkcija, zaloge, naročila, dobavne verige

a. Namen

Na začetku so bile podatkovne zbirke narejene kot dodatek datotečnim sistemom.

Slabosti datotečnih sistemov, kot podatkovne shrambe:

- Nekonsistentnost in odvečnost podatkov
    - Veliko datotečnih formatov, podvajanje podatkov po različnih datotekah
- Težavnost dostopa do podatkov
    - Potreben je nov program za vsako novo operacijo, kar je neučinkovito
- Izolacija podatkov
    - Veliko formatov datotek
- Problemi integritete
    - Izračun stanja na računu > 0, postane kar naenkrat del programske kode
    - Težko dodajati nove omejitve na podatke in spreminjat trenutne.
- Atomičnost posodobitev:
    - Če se zgodi napaka, lahko operacija pusti bazo v nekonsistentem stanju z drugimi deli baze
    - Prenos sredstev iz enega na drug račun, se mora zgoditi v celoti ali pa ne, ne sme se zgoditi, da bi operacija v vmesnem stanju pustila podatke.
- Hkratni dostop do baze večih uporabnikov
    - Hkratni dostop zaradi zmogljivosti sistema.
    - Nekontroliran dostop večih uporabnikov do iste zbirke podatkov
        - Dva uporabnika bereta stanje na računu uporabnika in ga posodabljata ob istem času? Lahko pride do velike napake
- Varnosti problemi

DBMS sistemi ponujajo rešitev za navedene probleme datotečnih sistemov.

b. Nivoji abstrakcije podatkov

- Fizični nivo – pove kako je določen zapis v bazi shranjen

- Logični nivo – opiše podatke shranjene v podatkovni zbirki in pove njihove relacije

```
type customer = record
                    name : string;
                    street : string;
                    city : integer;
                end;
```

- Predstavitveni nivo – aplikacija, ki lahko določene podatke prikaže ali pa ne.

3. Instance in sheme

Podobno kot tipi in spremenljivke v programskih jezikih

SHEMA – Logična struktura podatkovne baze

- Npr. podatkovna baza vsebuje informacije o uporabnikih in njihovih računih ter ralacij med njimi.
- Fizična shema: načrtovanje baze na fizičnem nivoju (nivo bitov in vezji)
- Logična shema: načrtovanje logičnih povezav podatkov
- Analogija s tipov vrednosti v programu

INSTANCA – dejanska vsebina podatkovne baze v določenem trenutku

- Analogija s konkretno vrednostjo spremenljivke v programu

NEODVISNOST FIZIČNIH PODATKOV – lahko spreminjamo fizično shema, brez da bi spreminjali logično shemo.

- Aplikacije se odvisne od logičnih povezav med podatki,
- Povezava med različnimi nivoji abstrakcije je skoraj neodvisna, fizični, logični, aplikacijski nivo. Če spreminjamo en nivo, ni potrebe po spreminjanju drugih nivojev, če so relacije dobro definirane.

4. Podatkovni modeli

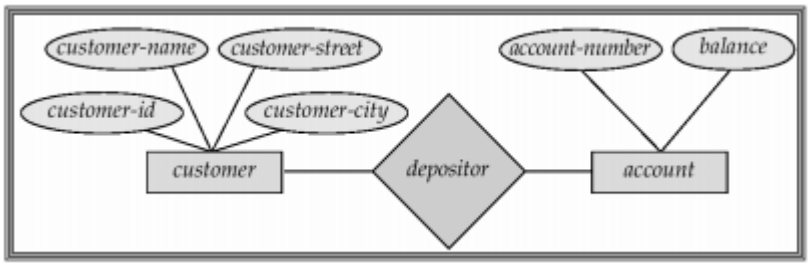
Zbirka orodji, ki nam opišejo:

- Podatke,
- Relacije med podatki,
- Semantiko podatkov,
- Omejitve posameznih podatkov

a. Model entitetnega odnosa (»Entity-Relationship model«)
b. Ralacijski model (Relational model)
c. Ostali modeli
    i. Objektno orientiran model
    ii. Pol-strukturiran podatkovni model
    iii. Omrežni model in hierarhični model

5. Entity-Relationship Model

Primer sheme:



E-R model v resničnem svetu:

- Entitete (objekt):
  - Stranke, računi, banka
- Ralacije med entitetami:
  - Npr. Račun A-101 pripada stranki po imenu Janez
  - Npr. Povezovanje strank z računi, preko depozitov na bankah
- E-R model načrtovanje običajno konvergira v relacijski model, ki se uporablja za shranjevanje in procesiranje podatkov

customer-id	customer-name	customer-street	customer-city	account-number
192-83-7465	Johnson	Alma	Palo Alto	A-101
019-28-3746	Smith	North	Rye	A-215
192-83-7465	Johnson	Alma	Palo Alto	A-201
321-12-3123	Jones	Main	Harrison	A-217
019-28-3746	Smith	North	Rye	A-201

Prva vrstica predstavlja atribute, druga pa dejanske podatke. Primer ralacijskega modela predstavljen s tabelo

customer-id	customer-name	customer-street	customer-city
192-83-7465	Johnson	12 Alma St.	Palo Alto
019-28-3746	Smith	4 North St.	Rye
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsay	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

(a) The *customer* table

account-number	balance
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

(b) The *account* table

customer-id	account-number
192-83-7465	A-101
192-83-7465	A-201
019-28-3746	A-215
677-89-9011	A-102
182-73-6091	A-305
321-12-3123	A-217
336-66-9999	A-222
019-28-3746	A-201

(c) The *depositor* table

Primer ralacijske baze, kjer so stranke, računi in njihovo stanje povezani preko, customer-id in account-number

i. DDL – data definition language

Jezik, ki opiše shemo baze

```
create table account (
        account-number   char(10),
        balance          integer)
```

Prevajalnik DDL generira ustrezne tabele, ki se shranijo v slovarju baze. Slovar baze vsebuje:

- Shema baze
- Podatkovno shrambo in definicijski jezik
    - Jezik, za poizvedovanje in shranjevanje podatkov
    - Ponavadi je to razširitev DDL

ii. DML – data manipulation language

Jezik, ki ponuja dostop in spreminjanje podatkov v podatkovni zbirki, poznan tudi kot poizvedbeni jezik (»query language«)

- Procedruralen --- uporabnik določi katere podatke hoče in na kakšen način jih želi dobiti
- Neproceduralen – uporabnik določi katere podatke želi, vendar ne pove na kakšen način želi te podatke

Najbolj poznan poizvedbeni jezik je SQL -- podrobneje si ga bomo ogledali v prihodnjem tednu, ko bomo delali osnovne praktične primere.

SQL

- Neproceduralni poizvedbeni jezik
- Npr: Poišči ime potrošnika, ki ima customer-id 192-83-7465

```
select  customer.customer-name
from    customer
where   customer.customer-id = '192-83-7465'
```

- Npr. Poišči stanje na vseh računih stranke z id: 192-83-7465

```
select  account.balance
from    depositor, account
where   depositor.customer-id = '192-83-7465' and
        depositor.account-number = account.account-number
```

Aplikacije do podatkovnih zbirk dostopajo preko specifičnih programskih razširitev, ki omogočajo SQL poizvedbe ali pa preko aplikacijskih vmesnikov (interfaces), ki omogočajo, da se SQL poizvedba pošlje do baze. Splošni vmesnik je ODBC, za programski jezik Java pa je poznan JDBC vmesnik.

6. Relacijski podatkovni model

account-number	*branch-name*	*balance*
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

Formalno: Če imamo množice D1, D2, … Dn, je ralcija r podmnožica množic D1 x D2 x … x Dn. Torej ralacija je množica n-množic (»n-tuple«) (a1, a2, …, an), kjer je ai elemnt Di

Npr.

customer-name = {Jones, Smith, Curry, Lindsay}
customer-street = {Main, North, Park}
customer-city = {Harrison, Rye, Pittsfield}
Then *r* = { (Jones, Main, Harrison),
(Smith, North, Rye),
(Curry, North, Rye),
(Lindsay, Park, Pittsfield)}

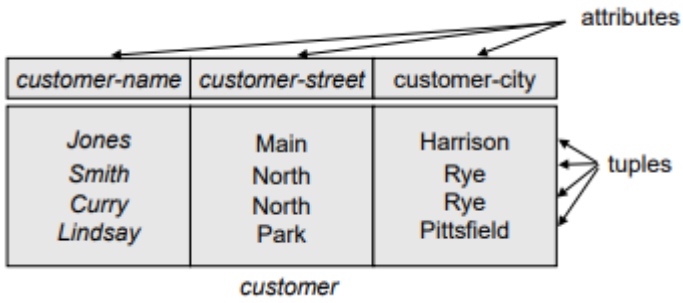
r = consumer-name x consumer-street x consumer-city

ATRIBUTI
Vsak atribut relacije ime ime. Domena atributa je množica dovoljenih vrednosti za ta atribut.

A1, A2, …. An so atributi

R = (A1, A2, …. An) je relacijska shema. Customer-schema = (customer-name, customer-stree, customer-city)

r(R) – je ralacija na relacijski shemi R. → customer(Customer-schema)



1 vrstica v tabeli predstavlja 1 konkreten zapis v bazi. (»tuples«)

Vrstni red zapisov v tabeli ni pomemben.

7. Podatkovna baza

Podatkovna shramba o enem podjetju (npr. banka), je razdeljeno v več ralacijskih tabel. Vsaka vsebuje del informacije. Npr.

- Account: tabela shranjuje informacije o računih
- Depositor: shranjuje informacije, kateri uporabnik ima kater račun
- Customer: shranjuje informacije o uporabnikih

Če bi vse informacije shranjevali v eni tabeli bi prišli do:

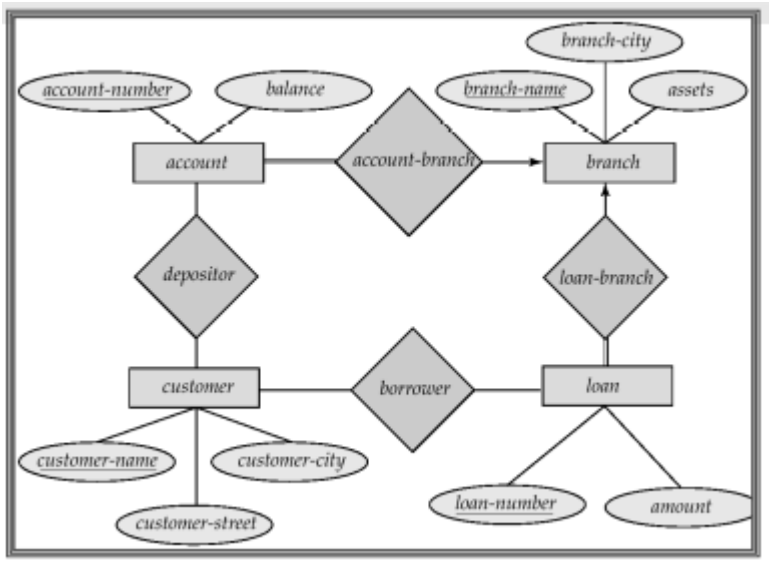
- Ponavljanja informacij (dva uporabnika, bi lahko imala isti račun)
- Null vrednosti (kako predstavit uporabnika brez računa?)

Za pravilno načrtovanje ralacij v bazi se ukvarja normalizacijska teorija v ketere podrobnosti se ne bomo spuščali.

Primeri:

customer-name	*customer-street*	*customer-city*
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Green	Walnut	Stamford
Hayes	Main	Harrison
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Putnam	Stamford
Williams	Nassau	Princeton

customer-name	*account-number*
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

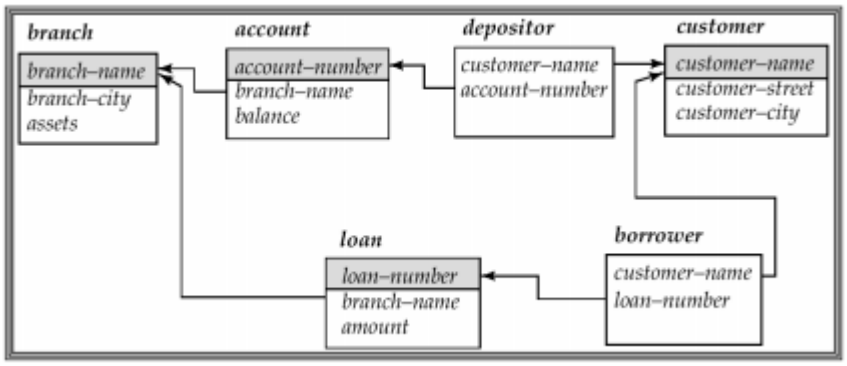


Ključi:

K pomnožica R

»Indetifikacijski ključ« -- enolično določi vsak zapis v določeni relacijski tabeli

- »Strong entity set« -- primarni ključ relacije. Primarni ključ enolično določa zapis v tabeli. Torej en atribut je primarni ključ relacije.
- »Weak entity set« -- primarni ključ ralacije je unija večih primarnih ključev. Torej več atributov skupaj določa primarni kluč.
- »Realationship set« -- povezave med ralacijami ključev tabel:
    - »Many-to-one« relacija: primarni ključ iz ene tabele postane »foreign-key« v drugi tabeli in se lahko pojavi večrkrat kot »foreign key«
    - »One-to-One«
    - »Mani-to-Mani«



Primeri relacij so vidni na zgornji sliki. Primari key v tabelah je označen s sivo.

8. Poizvedbeni jeziki

Poizvedbeni jeziki nad podatki bazah vsebujejo sledeče elemente:

- Relacijska algebra (beri nakaj matematike)

- Matematika množic
- Relacijska matematika

a. Relacijska algebra

- Select,
- Preslikave,
- Unije,
- Razlika množic
- Kartezijski produkt
- Preimenovanja

Operacije sprejmejo dve ali vel ralacije in izvržejo novo relacijo kot rezultat.

Primeri v angleščini za boljše razumevanje (Večina literature na to temo je vendarle v angleščini.)

Select Operation – Example

- Relation *r*

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

- $\sigma_{A=B \wedge D > 5}(r)$

A	B	C	D
α	α	1	7
β	β	23	10

- Notation: $\sigma_p(r)$
- *p* is called the selection predicate
- Defined as:

$$\sigma_p(r) = \{t \mid t \in r \text{ and } p(t)\}$$

Where *p* is a formula in propositional calculus consisting of terms connected by : $\wedge$ (**and**), $\vee$ (**or**), $\neg$ (**not**)
Each term is one of:

<attribute> *op* <attribute> or <constant>

where *op* is one of: =, ≠, >, ≥. <. ≤

- Example of selection:

$\sigma_{branch\text{-}name=\text{“}Perryridge\text{”}}(account)$

Project Operation – Example

- Relation *r*:

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

- $\Pi_{A,C}$ (*r*)

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

Project Operation

- Notation:

$$\Pi_{A1, A2, \ldots, Ak}(r)$$

where A_1, A_2 are attribute names and *r* is a relation name.

- The result is defined as the relation of *k* columns obtained by erasing the columns that are not listed
- Duplicate rows removed from result, since relations are sets
- E.g. To eliminate the *branch-name* attribute of *account*

$$\Pi_{account\text{-}number,\ balance}(account)$$

Union Operation – Example

- Relations *r*, *s*:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

r ∪ s:

A	B
α	1
α	2
β	1
β	3

Union Operation

- Notation: $r \cup s$
- Defined as:

$$r \cup s = \{t \mid t \in r \text{ or } t \in s\}$$

- For $r \cup s$ to be valid:
  1. r, s must have the *same arity* (same number of attributes)
  2. The attribute domains must be *compatible* (e.g., 2nd column of r deals with the same type of values as does the 2nd column of s)
- E.g., to find all customers with either an account or a loan

$$\Pi_{customer\text{-}name}(depositor) \cup \Pi_{customer\text{-}name}(borrower)$$

Set Difference Operation – Example

- Relations r, s:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

$r - s$:

A	B
α	1
β	1

Set Difference Operation

- Notation $r - s$
- Defined as:

$$r - s = \{t \mid t \in r \textbf{ and } t \notin s\}$$

- Set differences must be taken between *compatible* relations.
  - ★ r and s must have the *same arity*
  - ★ attribute domains of r and s must be compatible

Cartesian-Product Operation-Example

Relations *r*, *s*:

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

r x *s*:

A	B	C	D	E
α	1	α	10	a
α	1	β	19	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Cartesian-Product Operation

- Notation *r* x *s*
- Defined as:

$$r \times s = \{t\, q \mid t \in r \textbf{ and } q \in s\}$$

- Assume that attributes of r(R) and s(S) are disjoint. (That is, $R \cap S = \varnothing$).
- If attributes of *r(R)* and *s(S)* are not disjoint, then renaming must be used.

Composition of Operations

- Can build expressions using multiple operations
- Example: $\sigma_{A=C}(r \times s)$
- *r* x *s*

A	B	C	D	E
α	1	α	10	a
α	1	β	19	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

- $\sigma_{A=C}(r \times s)$

A	B	C	D	E
α	1	α	10	a
β	2	β	20	a
β	2	β	20	b

Rename Operation

- Allows us to name, and therefore to refer to, the results of relational-algebra expressions.
- Allows us to refer to a relation by more than one name.

Example:

$$\rho_X(E)$$

returns the expression E under the name X

If a relational-algebra expression E has arity n, then

$$\rho_{X(A1, A2, \ldots, An)}(E)$$

returns the result of expression E under the name X, and with the attributes renamed to $A1, A2, \ldots, An$.

b. Dodatne operacije, ki poenostavijo poizvedbe

Set-Intersection Operation

- Notation: $r \cap s$
- Defined as:
- $r \cap s = \{ t \mid t \in r \text{ and } t \in s \}$
- Assume:
  - ★ r, s have the *same arity*
  - ★ attributes of r and s are compatible
- Note: $r \cap s = r - (r - s)$

Set-Intersection Operation - Example

- Relation r, s:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- r ∩ s

A	B
α	2

Natural-Join Operation

- Notation: r ⋈ s
- Let *r* and *s* be relations on schemas *R* and *S* respectively. The result is a relation on schema $R \cup S$ which is obtained by considering each pair of tuples t_r from *r* and t_s from *s*.
- If t_r and t_s have the same value on each of the attributes in $R \cap S$, a tuple *t* is added to the result, where
  - ★ *t* has the same value as t_r on *r*
  - ★ *t* has the same value as t_s on *s*
- Example:

  $R = (A, B, C, D)$

  $S = (E, B, D)$
- Result schema = (*A*, *B*, *C*, *D*, *E*)
- *r* ⋈ *s* is defined as:

$$\Pi_{r.A,\ r.B,\ r.C,\ r.D,\ s.E}\ (\sigma_{r.B = s.B\ \ r.D = s.D}\ (r \times s))$$

Natural Join Operation – Example

- Relations r, s:

A	*B*	*C*	*D*
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

r

B	*D*	*E*
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	∈

s

r ⋈ *s*

A	*B*	*C*	*D*	*E*
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

Division Operation

$$r \div s$$

- Suited to queries that include the phrase "for all".
- Let *r* and *s* be relations on schemas R and S respectively where
  - ★ $R = (A_1, \ldots, A_m, B_1, \ldots, B_n)$
  - ★ $S = (B_1, \ldots, B_n)$

  The result of r ÷ s is a relation on schema

  $R - S = (A_1, \ldots, A_m)$

$$r \div s = \{ t \mid t \in \prod_{R-S}(r) \wedge \forall u \in s \, (tu \in r) \}$$

Division Operation – Example

Relations *r*, *s*:

A	B
α	1
α	2
α	3
β	1
γ	1
δ	1
δ	3
δ	4
∈	6
∈	1
β	2

r

B
1
2

s

r ÷ *s*:

A
α
β

Assignment Operation

- The assignment operation (←) provides a convenient way to express complex queries, write query as a sequential program consisting of a series of assignments followed by an expression whose value is displayed as a result of the query.
- Assignment must always be made to a temporary relation variable.
- Example: Write $r \div s$ as

$$temp1 \leftarrow \Pi_{R-S}(r)$$
$$temp2 \leftarrow \Pi_{R-S}((temp1 \times s) - \Pi_{R-S,S}(r))$$
$$result = temp1 - temp2$$

  - ★ The result to the right of the ← is assigned to the relation variable on the left of the ←.
  - ★ May use variable in subsequent expressions.

c. Agregacijske funckije

Aggregate Functions and Operations

- **Aggregation function** takes a collection of values and returns a single value as a result.

  avg: average value
  min: minimum value
  max: maximum value
  sum: sum of values
  count: number of values

- **Aggregate operation** in relational algebra

$$_{G1, G2, \ldots, Gn}\mathcal{G}_{F1(A1), F2(A2), \ldots, Fn(An)}(E)$$

  - ★ E is any relational-algebra expression
  - ★ $G_1, G_2 \ldots, G_n$ is a list of attributes on which to group (can be empty)
  - ★ Each F_i is an aggregate function
  - ★ Each A_i is an attribute name

Aggregate Operation – Example

- Relation *r*:

A	B	C
α	α	7
α	β	7
β	β	3
β	β	10

$\mathcal{G}_{\mathbf{sum(c)}}(\mathrm{r})$

sum-C
27

Aggregate Operation – Example

- Relation *account* grouped by *branch-name*:

branch-name	account-number	balance
Perryridge	A-102	400
Perryridge	A-201	900
Brighton	A-217	750
Brighton	A-215	750
Redwood	A-222	700

$branch\text{-}name\ \mathcal{G}_{sum(balance)}\ (account)$

branch-name	balance
Perryridge	1300
Brighton	1500
Redwood	700

Aggregate Functions (Cont.)

- Result of aggregation does not have a name
  - ★ Can use rename operation to give it a name
  - ★ For convenience, we permit renaming as part of aggregate operation

$branch\text{-}name\ \mathcal{G}\ \mathbf{sum}(balance)\ \mathbf{as}\ sum\text{-}balance\ (account)$

d. Spreminjanje podatkov v podatkovni zbirki

- Brisanje
- Vstavljanje
- Posodabljanje

Deletion

- A delete request is expressed similarly to a query, except instead of displaying tuples to the user, the selected tuples are removed from the database.
- Can delete only whole tuples; cannot delete values on only particular attributes
- A deletion is expressed in relational algebra by:

$$r \leftarrow r - E$$

where r is a relation and E is a relational algebra query.

Insertion

- To insert data into a relation, we either:
  - ★ specify a tuple to be inserted
  - ★ write a query whose result is a set of tuples to be inserted
- in relational algebra, an insertion is expressed by:

$$r \leftarrow r \cup E$$

where r is a relation and E is a relational algebra expression.

- The insertion of a single tuple is expressed by letting E be a constant relation containing one tuple.

Updating

- A mechanism to change a value in a tuple without charging *all* values in the tuple
- Use the generalized projection operator to do this task

$$r \leftarrow \prod_{F1, F2, \ldots, Fi,} (r)$$

- Each F, is either the ith attribute of r, if the ith attribute is not updated, or, if the attribute is to be updated
- F_i is an expression, involving only constants and the attributes of r, which gives the new value for the attribute