Relacijski podatkovni model

Iztok Savnik
FAMNIT

Viri

- *Prosojnice: „Cow Book“: R.Ramakrishnan, http://pages.cs.wisc.edu/~dbbook/*
- *Raghu Ramakrishnan, Johannes Gehrke, Database Management Systems, McGraw-Hill, 3rd ed., 2007.*

Zakaj študij relacijskega modela?

- Najbolj široko uporabljen podatkovni model
  - Relacije imajo močne matematične osnove
  - Firme: IBM, Informix, Microsoft, Oracle, Sybase, etc.
  - 80% vseh SUPB je relacijskih
- 1980-1995: vzpon objektno-usmerjenih sistemov
  - ObjectStore, ObjectDatabase++, GemStone/S, Versant, Ontos, ZODB, Wakanda, ObjectDB, …
  - Samo peščica OO SUPB je na trgu v letu 2000
- 1995-2000: Objektno-Relacijski Model
  - Relacije objektov so preslikane v ravninski relacijski model
  - Implementacije: Oracle, DB2, Sybase, etc.
  - OR model še vedno ni v širši uporabi!

Zakaj študij relacijskega modela?

- 2000-2010: NoSQL gibanje
  - Kaj se je zgodilo?
    - Masovni podatki, internetni inform. sistemi, nestrukturirani podatki, semi-strukturirani podatki, razsikovalni podatki, multimedia, slike, video, itd.
    - Obstoječi relacijski SUPB nimajo primerne skalabilnosti in dostopnosti potrebne za delo z masovnimi podatki.
  - Istočasno:
    - Nove tehnologije: obsežen RAM, SSD diski, masovni trdi diski, vzporedne in porazdeljene arhitekture, več-procesorski sistemi, več-jedrni procesorji, »samostojni« sistemi (angl. shared-nothing systems), itd.
  - Rezultat:
    - Vzpon No-SQL sistemov!
    - Produkti: zbirke ključ/vrednost, zbirke dokumentov, zbirke stolpcev, grafovski SUPB, SUPB v glavnem spominu, itd.

Zakaj študij relacijskega modela?

- Obstoječi NoSQL sistemi:
  - MongoDB, CouchDB, Berkeley DB, Dynamo, Hbase, Bigtable, Hypertable, Cassandra, Sybase IQ, Vertica, ArangoDB, OrientDB, Neo4j, GraphDB, Dgraph, Virtuoso, ...
- 2010-2020: NewSQL sistemi
  - Razvita tehnologija se uporabi v novih relacijskih SUPB!
  - Na novo razviti relacijski SUPB:
    - Google: Megastore (2011), Spaner (2012), F1 (2013)
    - Amazon: RDS
  - Obstoječa podjetja vključujejo nove tehnologije v svoje produkte
    - Skalabilnost in dostopnost je izboljšana
    - Oracle, DB2 (IBM), Sybase, itd.

Zakaj študij relacijskega modela?

- Novi SUPB predstavljeni pri predmetu:
  - ”Podatkovne baze za masovne podatke”, 2. stopnja, Podatkovne znanosti, FAMNIT
- Relacijski sistemi imajo največji trg
  - ... tudi po "revoluciji".
  - Po začetnem navdušenju se je celoten tok umiril.
  - Raziskovalno področje ima dobro organizirano znanje.
  - Značilnosti Relacijskih SUPB:
    - konsistentnost, transakcije, optimizator, vzporednost, zanesljivost, okrevanje po zrušitvi, porazdeljeljenost, skalabilnost in dostopnost.

Relacijska podatkovna baza

- *Relacijska PB:* *množica* *relacij.*
- *Relacija:* dva dela:
  - *Instanca* : *tabela*, ki ima vrstice in stolpce.
    #vrstic = *kardinalnost*, #stolpcev = *stopnja.*
  - *Shema* : določa ime relacije ter imena in tipe vseh stolpcev.
    - Studenti(*sid*: string, ime: string, *login*: string,
      starost: integer, po: real).
- Relacijo si lahko predstavljamo kot množico n-teric ali vrstic.

Primer relacije *študent*

sid	ime	enaslov	star.	po
53666	Novak	novak@pef	18	7.2
53688	Kranjc	kranjc@ma	18	6.8
53650	Kranjc	kranjc@fi	19	9.3

- Kardinalnost = 3, stopnja = 5, vse vrstice različne
- Ali morajo biti vsi stolpci v tabeli različni?

Relacijski povpraševalni jeziki

- Največja moč relacijskega modela:
  - SQL - enostaven povpraševalni jezik z veliko izrazno močjo.
- Vprašanja se lahko pišejo intuitivno, SUPB je odgovorna za učinkovito evaluacijo.
  - Razlog: natančna semantika relacijskih poizvedb.
  - Optimizator pogosto temeljito preuredi operacije poizvedbe. Dobimo logično ekvivalentno poizvedbo, ki vrne isti rezultat.

Povpraševalni jezik SQL

- Razvit na IBM (System R) leta 1970
- Potreba po standardu, ker ga uporablja veliko število proizvajalcev.
- Standardi:
  - SQL1: SQL-86
  - SQL2: SQL-89 (majhne spremembe)
  - SQL2: SQL-92 (večje spremembe)
  - SQL3: SQL-99 (večje spremembe)
  - SQL3: 2003, 2006, 2008, 2011, 2017, 2019 (manjše spremembe)

Povpraševalni jezik SQL

- Vprašanje*: Poišči vse študente, ki so stari 18 let.*

```
SELECT *
FROM Studenti S
WHERE S.starost=18
```

sid	ime	enaslov	star	po
53666	Novak	novak@pef	18	3.4
53688	Kranjc	kranjc@ma	18	3.2

- Vprašanje: *poišči imena in e-naslov študentov*.

```
SELECT S.ime, S.enaslov
FROM Studenti S
```

Povpraševanje nad večimi relacijami

- Kaj izračuna naslednje vprašanje?

```
SELECT S.ime, V.pid
FROM   Studenti S, Vpis V
WHERE  S.sid=V.sid AND V.ocena="10"
```

Relaciji Studenti in Vpis:

sid	name	login	Star	po
53666	Novak	novak@pef	18	7.2
53688	Kranjc	kranjc@ma	18	6.8
53650	Kranjc	kranjc@fi	19	9.3

sid	pid	ocena
53831	Baze	5
53831	Matematika	7
53650	Geometrija	10
53666	Zgodovina	7

Dobimo:

S.ime	V.pid
Kranjc	Geometrija

Kreiranje relacij v SQL

- Kreiranje relacije *Studenti*. Tip se specificira za vsako polje. Pri vpisu podatkov v bazo, SPUB zahteva definiran tip podatka.
- Tabela *Vpis* vsebuje podatke o predmetih, ki so jih vpisali študenti.

```
CREATE TABLE Studenti
        (sid: CHAR(20),
         ime: CHAR(20),
         login: CHAR(10),
         star: INTEGER,
         po: REAL)
```

```
CREATE TABLE Vpis
        (sid: CHAR(20),
         pid: CHAR(20),
         ocena: CHAR(2))
```

Brisanje in spreminjanje relacij

```
DROP TABLE Studenti
```

- Zgornji ukaz izbriše relacijo Studenti -- shemo in zapise.

```
ALTER TABLE Studenti
        ADD COLUMN kraj: varchar(25)
```

- Shemo študenti spremenimo tako, da dodamo nov atribut `kraj` s katerim opišemo kraj stanovanja študenta.

Dodajanje in brisanje zapisov

- Vstavljanje enega zapisa:

```
INSERT INTO Studenti(sid, ime, login, star, po)
VALUES  (53688, ‘Novak’, ‘novak@pef’, 18, 3.2)
```

- Izbrišemo vse zapise, ki zadoščajo določenem pogoju, npr., ime=“Novak”:

```
DELETE
FROM Studenti S
WHERE S.ime = ‘Novak’
```

☛ *Možne so variante ukazov z večjo izrazno močjo; več kasneje!*

Integritetne omejitve (IO)

- IO: pogoj, ki mora biti izpolnjen za vsako n-terico relacije.
  - IO so določene pri definiciji sheme.
  - IO se preverjajo ob spreminjanju relacij.
- Legalen primerek relacije zadovoljuje vse specificirane IO.
  - SPUB ne dopušča nelegalne relacije.
- Če SPUB preverja IO, potem so shranjeni podatki bližje pomenu v realnem svetu.
  - Izogibanje napakam pri vnosu!

Primarni ključ

- Množica atributov je ključ relacije če:
  1. Ne obstajata dva enaka zapisa z isto vrednostjo ključa.
  2. To ne velja za nobeno podmnožico ključa.
- (2) ne velja. Imamo superključ.
- Če obstaja več kot en ključ za relacijo potem izberemo enega, ki ga imenujemo primarni ključ.
- Npr. *sid* je ključ relacije *Studenti.* (*ime*?)
- Množica {*sid, po*} je superključ.

Primarni in kandidatni ključ v SQL

- Običajno je na voljo več kandidatnih ključev (definiramo kot UNIQUE) izmed katerih izberemo en primarni ključ.

- "Za danega študenta in predmet imamo eno samo oceno." ALI "Študenti lahko vpišejo en sam predmet in dobijo eno samo oceno za ta predmet; ne smeta obstajati dva študenta, ki imata isto oceno."

- Če uporabimo IO nepravilno lahko onemogočimo vnos dejanskih podatkov iz realnega sveta!

```
CREATE TABLE Vpis
  (sid CHAR(20)
   pid  CHAR(20),
   ocena CHAR(2),
   PRIMARY KEY  (sid,pid) )
CREATE TABLE Vpis
  (sid CHAR(20)
   pid  CHAR(20),
   ocena CHAR(2),
   PRIMARY KEY  (sid),
   UNIQUE (pid, ocena) )
```

Tuji ključi in referenčna integriteta

- *Tuj ključ*: Množica atributov neke relacije, ki referencira zapise druge relacije. Izbrana množica atributov mora ustrezati primarnem ključu druge relacije. Neke vrste “logični kazalec”.
- Primer: *sid* je tuj ključ (ključ relacije Studenti):
  - Vpis(*sid*: string, *pid*: string, *ocena*: string)
  - Če so v PB upoštevane integritetne omejitve tujih ključev, potem pravimo, da dosežemo referenčno integriteto; ni “visečih referenc”.
- Naštej imena podatkovnih modelov z / brez referenčne integritete?
  - Link & HTML?

Tuji ključi v SQL

- Samo študenti, ki so v tabeli *Studenti* lahko nastopajo v relaciji *Vpis*.

```
CREATE TABLE Vpis
  (sid CHAR(20),  pid CHAR(20),  ocena CHAR(2),
    PRIMARY KEY  (sid,pid),
    FOREIGN KEY (sid) REFERENCES Studenti )
```

Vpis

sid	pid	grade
53666	Baze	5
53666	Matematika	7
53650	Geometrija	10
53666	Zgodovina	7

Studenti

sid	ime	login	Star	po
53666	Novak	novak@cs	18	7.2
53688	Kranjc	kranjc@ma	18	6.8
53650	Kranjc	kranjc@fi	19	9.3

Zagotavljanje referenče integritete

- Poglejmo si tabeli Studenti in Vpis; *sid* v tabeli Vpis je tuj ključ, ki referencira tabelo Studenti.
- Kaj naj se zgodi, če poskušamo dodati zapis, ki vsebuje neobstoječ sid v tabelo Vpis? *Zavrnitev zapisa!*
- Kaj naj se zgodi, če izbrišemo zapis tabele Studenti?
  - Pobrišejo se tudi pripadajoči zapisi v Vpis.
  - Ne dovoli se brisanje zapisa Studenti na katerega se referencirajo zapisi iz tabele Vpis.
  - Vrednosti zbrisanih *sid* se v tabeli Vpis postavi na privzeto vrednost.
  - SQL: Postavi vrednosti izbrisanih *sid* v Vpis na null vrednost, ki pomeni *neznano* ali *nenavedeno*.
- Podobno v primeru, da se ključ sid v tabeli Studenti spremeni.

Referenčna integriteta v SQL

- SQL/92 in SQL3 podpirata vse 4 možnosti pri brisanju in popravljanju.
  - Privzeto je NO ACTION (*delete/update je zavrnjen*)
  - CASCADE (zbriši vse zapise, ki se sklicujejo na izbrisani zapis)
  - SET NULL / SET DEFAULT (postavi tuj ključ v zapisih, ki se referencirajo na izbrisani zapis)

```
CREATE TABLE Vpis
  (sid CHAR(20),
   pid CHAR(20),
   ocena CHAR(2),
   PRIMARY KEY (sid,pid),
   FOREIGN KEY (sid)
     REFERENCES Studenti
          ON DELETE CASCADE
          ON UPDATE SET DEFAULT )
```

Od kje so prišle IC omejitve?

- IC so osnovane na pomenu okolja, ki ga modeliramo z relacijskim modelom.
- Lahko preverimo podatkovno bazo ali je v skladu z integritetnimi omejitvami . Ne moremo pa NIKOLI sklepati, da je omejitev pravilna z opazovanjem modela (instance).
  - IC je izjava o vseh možnih instancah!
  - Iz primera vidimo, da ime ni ključ. Določitev, da sid je ključ je prepuščeno nam.
- Ključ in tuj ključ so najbolj pogoste IC; bolj splošne IC tudi obstajajo.

Relacijski model: pregled

- Tabularna predstavitev podatkov.
- Enostavno in intuitivno, trenutno najbolj pogosto uporabljan model.
- Integritetne omejitve lahko vnesemo v podatkovno bazo na osnovi lastnosti modeliranega podatkovnega okolja. SUPB preveri veljavnost omejitev.
  - Pomembni omejitvi: primarni in tuj ključ.
  - Vedno so definirane domene za atribute.
- Na voljo je močan in intuitiven povpraševalni jezik.