4. Обща характеристика на релационните СУБД

Лекционен курс “Бази от данни”

Обща характеристика на РСУБД

Основата на модерните технологии за БД безспорно е релационният модел:

- Релационният модел (РМ) не е нещо статично;
- Разработен първоначално през 1970 год. от д-р Edgar Frank Codd;
- Развива се непрекъснато.

- Една от най-значимите разработки на релационния модел е System R, разработена от IBM в края на 70-те;
- System R е замислена като „доказателство на концепцията, показваща че РСУБД могат да бъдат създадени и да работят ефективно;
- Дава ход да множество разработки като Structured Query Language (SQL), който става стандарт за релационен език;
- През 1985 Код публикува списък от правила, превърнали се в стандартен начин за оценяване на релационна система. Тези правила представят релационния идеал.

1. Правило за информацията – цялата информация в дадена релационна база данни, включително имената на таблиците и колоните, се представят като явни стойности в таблиците;

2. Гарантиран достъп – всяка стойност в релационните бази данни задължително трябва да е достъпна чрез комбинацията от име на таблица, стойност на първичен ключ и име на колона;

3. Систематична поддръжка на нулеви стойности – СУБД трябва да осигури систематична поддръжка за нулевите стойности (непознати или неприложими данни), която е различна от стойностите по подразбиране и е независима от типовете данни (областите);

4. Активен онлайн релационен каталог – описанието на базата данни и нейното съдържание се представя на логическо ниво като таблици, достъпът до които се осъществява чрез езика на базата данни;

5. Пълен подезик за данните – поне един от поддържаните езици трябва да има добре дефиниран синтаксис и да бъде пълен. Той трябва да поддържа дефиниране на данни, манипулацията им, правила за интегритет на данните, упълномощаване и транзакции;

6. Правила за обновяване на изгледите – чрез всички изгледи, които са теоретично обновими, потребителите трябва да могат да обновяват данни в системата;

7. Вмъкване, обновяване и изтриване на ниво множество – СУБД трябва да поддържа не само извличането, но и вмъкването, обновяването и изтриването на информация на ниво множество;

8. Физическа независимост на данните – физическият достъп или структурите за съхранение на данните не влияят върху начина на достъп до тях (логиката на приложенията);

9. Логическа независимост на данните – бизнес-логиката на приложенията не трябва да влияе върху структурата на таблиците;

10. Независимост на цялостността на данните – езикът на базата данни трябва да може да дефинира правила за цялостност. Те трябва да бъдат записани в онлайн каталога и да не могат да бъдат заобикаляни;

11. Независимост на разпределението – приложенията и техните заявки не трябва да се влияят логически от това дали данните са централизирани или разпределени;

12. Невъзможност за намеса – ако СУБД поддържа релационен език от ниско ниво той не трябва да може да заобикаля правилата за цялостност, наложени с релационен език от по-високо ниво.

Zero Rule: Релационните СУБД трябва да могат да управляват напълно базите данни с техните релационни възможности.

Идеята на Код за релационните СУБД използва математическите концепции на релационната алгебра - за декомпозиция на данните на отделни множества и свързани с тях общи подмножества.

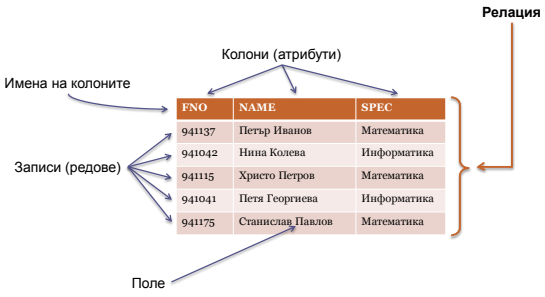
Понеже информацията може по естествен начин да бъде групирана в отделни множества – Код разработва теорията си около тази концепция.

Релационен модел на данните

- Релационният модел използва колекция от таблици, за да представя едновременно данните и взаимоотношенията между тях;
- Таблиците са логически структури, поддържани от мениджъра на базата данни.

- Релационният модел се състои от 3 части:
  - Структурна част (релационни обекти) – дефинира базата данни като колекция от релации;
  - Правила за цялостност на данните – интегритетът на базата данни се управлява с използване на първични и външни ключове;
  - Манипулативна част (релационни оператори за опериране с данните) – реализирана е на основата на релационната алгебра.

Основни понятия



FNO	NAME	SPEC
941137	Петър Иванов	Математика
941042	Нина Колева	Информатика
941115	Христо Петров	Математика
941041	Петя Георгиева	Информатика
941175	Станислав Павлов	Математика

Използвани термини

Релационен модел	Потребител
Подредено множество (tuple)	Ред (запис)
Атрибут	Колона
Релация	Таблица

За да бъде таблица релация трябва да отговаря на следните условия:

- Сечението на ред с колона трябва да съдържа атомарна стойност;
- Всички стойности в една колона трябва да са от един и същ тип;
- Всяка колона трябва да има уникално име;
- Да няма повтарящи се редове.

Основни характеристики на релационния модел

- Всеки ред в таблицата се нарича подредено множество;
- Всяка колона в таблицата се нарича атрибут;
- Сечението на ред с колона съдържа единична стойност от данни;
- Редовете са в произволна подредба;
- Атрибутите са в произволна подредба;
- Всички редове в релацията са уникални. Няма два реда с абсолютно еднакви стойности;

Основни характеристики на релационния модел

- Релацията трябва да има ключ;
- Ключовете са множества от атрибути;
- За всяка колона от таблицата има множество от допустими стойности, наречени област (домейн);
- Областта е множество от валидни стойности за даден атрибут;
- Степен на релацията е броят атрибути (колони) на тази релация;
- Кардиналност е броят подредени множества (редове) в релацията.

Забележки:

- Понятието 'ключ' (key) е едно от най-добре разработените в областта на БД - само в релационния модел различаваме следните ключове:
  - Кандидати (Candidate)
  - Първични (Primary)
  - Алтернативни (Alternate)
  - Уникални (Unique)
  - Външни (Foreign)

- В другите области на БД-технологията се срещат:
  - index keys
  - hash keys
  - search keys
  - secondary keys
  - ordering keys
  - и др.
- Ако се срещне понятието 'ключ' без допълнително пояснение, обаче, във всеки случай би трябвало да става въпрос за първичен ключ – тези ключове са много съществени;
- Не всички релационни системи поддържат всички аспекти на релационния модел.

Релационен модел на данните

Релационна БД (неформално определение) е система, в която:

- Данните се възприемат от потребителите като множество от таблици (и нищо друго освен таблици);
- Операторите, с които потребителят разполага, генерират нови таблици от съществуващите и тези оператори включват поне SELECT (RESTRICT), PROJECT и JOIN.

Операторът:

- SELECT извлича определени редове от таблица;
- PROJECT извлича определени колони от таблица;
- JOIN слива (съединява) две таблици на базата на общи стойности в общи колони.

Пример – база данни за отдели и служители

DEPT

DEPT_ID	DNAME	BUDGET
D1	Маркетинг	100000
D2	Разработки	120000
D3	Изследвания	70000

EMP

EMP_ID	ENAME	DEPT_ID	SALARY
E1	Петър	D1	8000
E2	Иван	D1	6000
E3	Мария	D2	10000
E4	Стефан	D2	5000

SELECT (RESTRICT)

DEPTs where BUDGET > 80000

DEPT_ID	DNAME	BUDGET
D1	Маркетинг	100000
D2	Разработки	120000

PROJECT

DEPTs over DEPT_ID, BUDGET

DEPT_ID	BUDGET
D1	100000
D2	120000
D3	70000

JOIN - пример

- Двете таблици имат обща колона – DEPT_ID, така че могат да бъдат съединени въз основа на общите стойности в тази колона
- По този начин даден ред от таблицата DEPT ще се слее с даден ред от таблицата EMP – създавайки нов ред, само ако в двата реда стойностите в общата колона DEPT_ID съвпадат

DEPT_ID	DNAME	BUDGET
D1	Маркетинг	100000

EMP_ID	ENAME	DEPT_ID	SALARY
E1	Петър	D1	8000

DEPT_ID	DNAME	BUDGET	EMP_ID	ENAME	SALARY
D1	Маркетинг	100000	E1	Петър	8000

- Множеството от всички възможни подобни съединени редове ще формира крайния резултат
- Важно е да се отбележи, че тъй като няма запис в EMP с DEPT_ID = D3, запис за D3 не се появява в резултата, въпреки че има запис за D3 в таблицата DEPT

JOIN

Двете таблици имат обща колона – DEPT_ID, така че могат да бъдат съединени въз основа на общите стойности в тази колона.

DEPTs and EMPs over DEPT_ID

DEPT_ID	DNAME	BUDGET	EMP_ID	ENAME	SALARY
D1	Маркетинг	100000	E1	Петър	8000
D1	Маркетинг	100000	E2	Иван	6000
D2	Разработки	120000	E3	Мария	10000
D2	Разработки	120000	E4	Стефан	5000

Изводи

- Ясно се вижда от примерите, че резултатът от всяка операция беше нова таблица;
- Това релационно свойство се нарича **затвореност** и е много съществено;
- Следователно, след като резултатът от операция е от същия вид като операндите ѝ, то той може да бъде операнд за друга операция.

Изводи

- Таблиците са логически структури, не физически – на физическо ниво системата е свободна да използва каквото и да е представяне;
- Цялата информация в базата данни е представена по един и същ начин – с явни стойности;
- Не съществуват указатели, свързващи отделните таблици;

Изводи

- Всички стойности са скаларни (атомарни, атомични), т.е. всяко поле съдържа точно една стойност, а не група от няколко.

1)

DEPT_ID	EMP_ID
D1	E1
D1	E2
...	...

2)

DEPT_ID	EMP_ID
D1	E1, E2
...	...

  - Съдържанието на колоната EMP_ID във втория вариант е пример за **повтарящи се групи** от данни – съдържа няколко стойности;
  - Релационните бази данни не позволяват повтарящи се групи.

Обобщение

- Всички стойности са атомарни;
- Цялата информация, съдържаща се в БД, е представена с явни стойности - това е друга характеристика на релационните БД;
- Не съществуват указатели, свързващи отделните таблици;
- Когато казваме, че една система е релационна приемаме, че тя поддържа релации на външно и концептуално ниво;

- На ниско ниво (вътрешно) системата е свободна да използва произволни структури, стига да е в състояние на горните две нива да ги представи като релации;
- Една релационна БД се възприема от потребителите като множество от таблици;
- Тя не е БД, в която данните се съхраняват като таблици;
- Релацията е математическо (абстрактно) понятие за една таблица.

Чрез математическите концепции за операции върху множества:

- Обединение
- Сечение

- От релационните бази данни могат бързо да се извлекат отделни елементи от различни множества (таблици);
- Извлечената информация може да бъде агрегирана и върната като резултат на потребителите на БД;
- Възможността за интегриране на информацията позволява на проектантите на бази данни да съхраняват множества от информация в различни таблици - по този начин може да се оптимизира излишеството на информацията.

Еволюция на някои от известните СУБД