UAlg FCT

UNIVERSIDADE DO ALGARVE
FACULDADE DE CIÊNCIAS E TECNOLOGIA

Bases de Dados

INTRODUÇÃO AO SQL

Adaptado de Silberschatz, Korth e Sudarschan

História

Linguagem IBM Sequel desenvolvida como parte do projeto System R no IBM San Jose Research Laboratory

Renomeada para *Structured Query Language* (SQL)

SQL como norma ANSI e ISO:

- SQL-86
- SQL-89
- SQL-92
- SQL:1999
- SQL:2003

Os sistemas comerciais oferecem a maioria, se não todos, os recursos do SQL-92, além de diversos conjuntos de recursos de normas posteriores e recursos proprietários especiais.

Componentes do SQL

DML *(Data Manipilation Language)* - fornece a capacidade de consultar, inserir, apagar e modificar registos numa base de dados

Esquema e integridade – o DDL *(Data Definition Language)* inclui comandos para especificar o esquema, incluindo as restrições de integridade.

Definição de vistas – O DDL inclui comandos para definir vistas.

Controlo de transações – inclui comandos para especificar o início e o fim das transações.

SQL *embedded* e SQL dinâmico – definem como as instruções SQL podem ser incorporadas em linguagens de programação de uso genérico.

Autorização – inclui comandos para especificar direitos de acesso a tabelas e vistas.

Data Definition Language (DDL)

A *Data Definition Language* do SQL (DDL) permite a especificação de informações sobre relações (tabelas), incluindo:

- O esquema para cada relação.
- O tipo de valores associados a cada atributo.
- As restrições de integridade
- O conjunto de índices para cada relação.
- Informações de segurança e autorização para cada relação.
- A estrutura de armazenamento físico de cada relação em disco.

Tipos de domínios em SQL

char(n) - Cadeia de caracteres de comprimento fixo, com comprimento *n.*

varchar(n) - Sequências de caracteres de comprimento variável, com comprimento máximo *n.*

int - Número inteiro (4 bytes).

smallint - Inteiro pequeno (2 bytes).

numeric(p,d) - Número com precisão de *p dígitos*, com *d* dígitos à direita do ponto decimal. (ex., **numeric** (3,1), permite que 44,5 seja armazenado exatamente, mas não 444,5 ou 0,32)

real e double - Números de vírgula flutuante e de precisão dupla, com precisão dependente da máquina (tipicamente, 4 bytes – 6 casas decimais de precisão e 8 bytes – 15 casas decimais de precisão).

float (n) - Número de vírgula flutuante, com precisão especificada de pelo menos *n* dígitos.

date - data (precisão 1 dia - 4 bytes).

time - horas num dia (precisão microsegundo - 4 bytes).

timestamp – data e hora (precisão microsegundo – 8 bytes)

Criar uma tabela

Uma relação SQL é definida usando o comando **create table**:

create table *r*

(A_1 D_1, A_2 D_1,..., A_n D_n,

(restrição de integridade$_1$),

...,
(restrição de integridade $_k$))

- *r* é o nome da relação
- cada A_i é um nome de atributo no esquema da relação *r*
- D_i é o tipo de dados dos valores no domínio do atributo A_i

Exemplo :

create table *instrutor* (
ID **char** (5),
name **varchar** (20) **,**
dept_name **varchar** (20),
salary **numeric** (8,2))

Restrições de integridade na criação de tabela

Tipos de restrições de integridade

- **primary key** (A_1, ..., A_n)
- **foreign key** (A_m, ..., A_n) **references** *r*
- **not null**
- **unique**
- **check**

SQL impede qualquer atualização na dase de dados que viole uma restrição de integridade.

Exemplo:

create table *instructor* (
ID **char**(5),
name **varchar**(20) **not null,**
dept_name **varchar**(20),
salary **numeric**(8,2),
primary key (*ID*),
foreign key *(dept_name*) **references** *department);*

E mais algumas definições de relação

create table *student* (
ID **varchar**(5),
name **varchar**(20) not null,
dept_name **varchar**(20),
tot_cred **numeric**(3,0),
primary key *(ID),*
foreign key *(dept_name*) **references** *department*);

create table *takes* (
ID **varchar**(5),
course_id **varchar**(8),
sec_id **varchar**(8),
semester **varchar**(6),
year **numeric**(4,0),
grade **varchar**(2),
primary key *(ID, course_id, sec_id, semester, year)* ,
foreign key (*ID*) **references** *student,*
foreign key (*course_id, sec_id, semester, year*) **references** *section*);

E mais ainda

```
create table course (
        course_id       varchar(8),
        title           varchar(50),
        dept_name       varchar(20),
        credits         numeric(2,0),
          primary key (course_id),
          foreign key (dept_name) references department);
```

Atualizações nas tabelas

Inserir

- **insert into** *instructor* **values** ('10211', 'Smith', 'Biology', 66000);

Excluir

- Remover todos os tuplos da relação *aluno*
  - **delete from** *aluno*

Eliminar tabela

- **drop table** *r*

Alterar

- **alter table** *r add A D*
  - onde *A* é o nome do atributo a ser adicionado à relação *r* e *D* é o domínio de *A.*
  - todos os tuplos existentes na relação recebem *null* como valor para o novo atributo.
- **alter table** *r* **drop** ***(column)*** *A*
  - onde *A* é o nome de um atributo da relação *r.*
  - A eliminação de atributos não é suportada por vários SGBD.

Estrutura básica de uma query

Uma query SQL típica tem o formato:

select $A_1, A_2, ..., A_n$
from $r_1, r_2, ..., r_m$
where P

- A_i representa um atributo
- R_i representa uma relação
- P é um predicado.

O resultado de uma query SQL é uma relação.

A cláusula select

A cláusula **select** lista os atributos desejados no resultado de uma consulta

- corresponde à operação de **projeção da álgebra relacional**

Exemplo: encontre os nomes de todos os instrutores:

select *name*
from *instructor*

NOTA: SQL não diferencia maiúsculas de minúsculas

- Por exemplo, *Name* ≡ *NAME* ≡ *name*

A cláusula de select (Cont.)

SQL permite duplicados nas relações e também nos resultados da consulta.

Para forçar a eliminação de duplicados, insira a palavra-chave **distinct** depois do select.

Encontre os nomes dos departamentos de todos os instrutores e remova os duplicados

select distinct *dept_name*
from *instructor*

A palavra-chave **all** especifica que os duplicados não devem ser removidos.

All é usado por omissão, por isso raramente se vê numa query

select all *dept_name*
from *instructor*

O mesmo que:

select *dept_name*
from *instructor*

dept_name
Comp. Sci.
Finance
Music
Physics
History
Physics
Comp. Sci.
History
Finance
Biology
Comp. Sci.
Elec. Eng.

A cláusula de select (Cont.)

Um asterísco na cláusula select indica "todos os atributos"

select *
from *instructor*

Um atributo pode ser um literal sem cláusula **from**

select ' 437 '

- Os resultados são uma tabela com uma coluna e uma única linha com valor "437"
- Pode dar um nome à coluna usando:

select '437' **as** *FOO*

Um atributo pode ser um literal com a cláusula **from**

select 'A'
from *instructor*

O resultado é uma tabela com uma coluna e *N* linhas (número de tuplos na tabela *instructor*), cada linha com valor "A"

A cláusula de select (Cont.)

A cláusula **select** pode conter expressões aritméticas envolvendo a operação +, –, * e /, operando em constantes e/ou atributos.

- A consulta:

  select *ID, name, salary/12*
  from *instructor*

  retornaria uma relação igual à relação *de instrutor*, exceto o valor do atributo *salário, que* é dividido por 12.

- Pode-se renomear "*salary/12*" usando a cláusula **as**:

  select *ID, name, salary/12* **as** *monthly_salary*

A cláusula where

- A cláusula **where** especifica as condições que o resultado deve satisfazer
  - Corresponde ao **predicado de seleção da álgebra relacional**.
- Para encontrar todos os instrutores no departamento de Comp. Sci.

```
select name
from instructor
where dept_name = 'Comp. Sci.'
```

- SQL permite o uso dos operadores lógicos **and, or,** e **not**
- Os operandos dos operadoress lógicos podem ser expressões envolvendo os operadores de comparação <, <=, >, >=, = e <>.
- Comparações podem ser aplicadas a resultados de expressões aritméticas
- Para encontrar os instrutores no departamento de Comp. Sci. com salário > 70000

```
select name
from instructor
where dept_name = 'Comp. Sci.' and salary > 70000
```

name
Katz
Brandt

A cláusula from

- A cláusula **from** lista as relações envolvidas na consulta
  - Corresponde à operação de **produto cartesiano** da álgebra relacional.
- Encontre o produto cartesiano de *instructor x teaches*

  select ∗
  from *instructor, teaches*
  - gera todos os pares instructor – teaches, com todos os atributos de ambas as relações
  - Para atributos comuns (por exemplo, *ID*), os atributos na tabela resultante são renomeados usando o nome da relação (por exemplo, *instrutor.ID*)
- Produto cartesiano não é muito útil diretamente, mas é útil combinado com a condição da cláusula where (operação de seleção em álgebra relacional).

Exemplos

Encontre os nomes de todos os instrutores que ministraram alguns cursos e o course_id

- **select** *name, course_id*
  from *instructor , teaches*
  where *instructor.ID = teaches.ID*

Encontre os nomes e o course_id de todos os instrutores do departamento de *Art* que ministraram alguns cursos

select *name, course_id*
from *instructor , teaches*
where *instructor.ID = teaches.ID*
and *instructor. dept_name* = 'Art'

name	*course_id*
Srinivasan	CS-101
Srinivasan	CS-315
Srinivasan	CS-347
Wu	FIN-201
Mozart	MU-199
Einstein	PHY-101
El Said	HIS-351
Katz	CS-101
Katz	CS-319
Crick	BIO-101
Crick	BIO-301
Brandt	CS-190
Brandt	CS-190
Brandt	CS-319
Kim	EE-181

A operação de renomeação

- O SQL permite renomear relações e atributos usando a cláusula **as** :

  nome_antigo **as** *novo_nome*

- Encontre os nomes de todos os instrutores que têm salários mais altos do que alguns instrutores em 'Comp. Sci.'
  - **select distinct** *T.name*
    from *instructor* **as** *T, instructor* **as** *S*
    where *T.salary > S.salary* **and** *S.dept_name = 'Comp. Sci.'*

A palavra-chave **as** é opcional e pode ser omitida

instrutor **as** *T* ≡ *instructor T*

Operações de *strings*

- SQL inclui um operador de correspondência de strings. O operador **like** usa padrões que são descritos com dois caracteres especiais:
  - (%). O carácter % corresponde a qualquer substring.
  - (_). O carácter _ corresponde a qualquer carácter.
- Encontre os nomes de todos os instrutores cujo nome inclui a substring “dar”.

```
select name
from instructor
where name like '%dar%'
```

- Se fosse a string "100%“?

like '100\% ' **escape** '\'

usamos backslash(\), por omissão, como caractér de *escape*.

Operações de *Strings* (Cont.)

- Os padrões de *strings* são *case sensitive*.
- Exemplos de correspondência de padrões:
  - 'Intro%' corresponde a qualquer string que comece com “Intro”.
  - '%Comp%' corresponde a qualquer string contendo “Comp” como substring.
  - '_ _ _' corresponde a qualquer sequência de exatamente três caracteres.
  - '_ _ _ %' corresponde a qualquer sequência de pelo menos três caracteres.
- SQL suporta a uma variedade de operações de strings, como
  - concatenação (usando “||”)
  - conversão de maiúsculas para minúsculas (e vice-versa)
  - encontrar o comprimento da string, extrair substrings, etc.

Ordenando a exibição dos tuplos

- Liste, por ordem alfabética, os nomes de todos os instrutores

```
select distinct name
from    instructor
order by name
```

- Podemos especificar **desc** para ordem decrescente ou **asc** para ordem crescente (por omissão), para cada atributo.
  - Exemplo: **order by** *name* **desc**

Pode-se ordenar em vários atributos

- Exemplo: **order by** *dept_name, name*