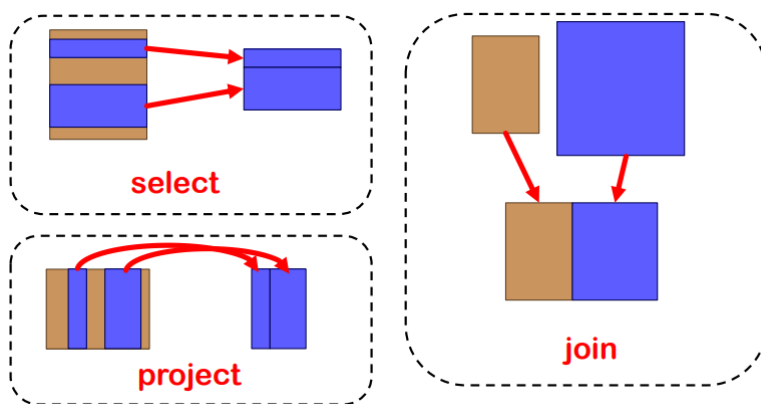


Lecture 1

1. Introduction to Databases

- Rows are also called records or tuples
- Columns are also called attributes

Operations



2. Key

Duplicates are forbidden in relational tables

3. Normalization

You need to standardize your data, a process also known as normalization

From a raw spreadsheet to a relatively clean database model allowing us to handle weird cases

Every non key attribute must provide a fact about the key, the whole key, and nothing but the key

[Tutorial3 Database Design.pdf](#)

其实三大范式的设计遵循的就是尽可能的减少不必要的重复

1NF

Simple attributes

Director	
Welles, Orson	
FirstName	Surname
Orson	Welles

- 原子性、不能再分
- each column should only contain ONE piece of information
- We can also add a numerical id, and make it the primary key
- Simple attributes
- A relation is in first normal form if and only if the domain of each attribute contains only atomic (indivisible) values, and the value of each attribute contains only a single value from that domain.

2NF

Title	Year	Country	Continent
Good Bye, Lenin!	2003	Germany	Europe

- 表中每一个非主键列与主键列直接相关
直接相关：主键一改，其它列都会跟着改
- 不会有与主键不相关的列
- attributes depend on the full key
- Wrong. I don't want to repeat it for every German film. It should be said once in a table of countries that Germany is in Europe

3NF

Country	Name	Continent	Currency	Symbol
de	Germany	Europe	Euro	€

- 非主键列不会互相相关
- non-key attributes not depend on each other
- Wrong. I don't want to repeat it for every country in the Euro zone. I should have it in a currency table, the key of which would be the currency

Lecture 2

1. Introduction to SQL

DDL

The Data Definition Language deals with tables

- CREATE: creating a new table
- ALTER: changing its structure
- DROP: deleting a table

DML

The Data Manipulation Language deals with rows

- INSERT
- UPDATE
- DELETE
- SELECT

DBMS

There is an official SQL standard that no product fully implements

- PostgreSQL
- ORACLE
- MySQL
- DB2
- SQL Server

SQL wasn't designed as a "relationally correct" language, but as an "easy language" that anybody could use

2. Create Tables

Creating tables requires

- Proper modelling
- Defining keys
- Determining correct data types
- Defining constraints

Data Type

Text

- CHAR
- VARCHAR
- VARCHAR2
- CLOB

Number

- INT
- FLOAT
- NUMERIC
- DECIMAL

Date

- DATE
- DATETIME
- TIMESTAMP

Binary

- RAW
- VARBINARY
- BLOB
- BYTEA

Comment

如果你想对数据库的表的列写备注，有两种写法

写入数据库内，可以读出来

```
1 COMMENT ON COLUMN <table name>.<column name> IS 'comment~~~'
```

单纯的注释

```
1 -- comment
```

3. Constraints

Constraints are declarative rules that the DBMS will check every time

- new data will be added
- when data is changed
- when data is deleted

In order to prevent any inconsistency. Any operation that violates a constraint fails and returns an error

	可否为空	是否唯一
NOT NULL	单一列：不可以为空	不做要求
PRIMARY KEY	单一列：不可以为空 组合列：组合列任何列不可以为空	单一列：唯一 组合列：组合后唯一
UNIQUE	单一列：可以为空 组合列：任何列均可以为空	单一列：唯一 组合列：如果某一列中有 NULL 存在，则不受到 UNIQUE 约束，如果组合列没有任何一列有 NULL，则整个组合列受到 UNIQUE 约束
FOREIGN KEY	不做要求	不做要求

4. Insert

```
1 INSERT INTO <table name>
2 (list of columns)
3 VALUES (list of values)
```

注意，list of columns 不是强制的，你可以选择不添加

- 如果不添加，则按照表的原有顺序插入数据
 - 如果表中没有非空约束，且数据类型正确，则添加成功
 - 如果表中有非空约束/数据类型不正确，则添加失败
- 如果添加，则按照添加的顺序插入数据

Note that strings are given between SINGLE quotes

Some products (MySQL, SQLite) also accept double quotes but SINGLE quotes is the standard, you can use them everywhere

escape character

The standard SQL way to escape a quote is to double it

```
insert into movies(movieid,
                    title,
                    country,
                    year_released)
values (123,
        'L''Avventura',
        'it',
        1960)
```

Lecture 3

1. SELECT

SELECT *

```
1 SELECT * FROM <table name>
```

You should not use it in programs !!

WHERE

For using WHERE, you can compare

- number
- string
- other column

Beware that string constants must be quoted between single quotes. If they aren't quoted, they will be interpreted as column names

Column doesn't exist!



where surname = Han

where surname = 'Han'

```
filmdb=# select * from people where surname=Han;  
ERROR: column "han" does not exist  
LINE 1: select * from people where surname=Han;
```

AND OR

AND > OR !!!

One thing to remember is that, like numerical operators, all logical operators haven't the same precedence and, and is "stronger" than or

where country = 'us'
or country = 'gb'
and year_released between 1940
and 1949

(the condition will select British films of the 1940s) AND (all American films irrespective of the date)

If you wanted films from the 1940s that are either American or British, your result will be wrong

Compare

如果比较符左右有数字类型的，则按照都数字类型运算

```
1 SELECT 2 < 10; -- True
2 SELECT '2' < '10'; -- False
3 SELECT '2' < 10; -- True
4 -----
5 SELECT 'A' > 10; -- 无效
6 SELECT 'A' > 'B'; -- False
7 -----
8 SELECT '1' + 1; -- 2
9 SELECT 'a' + 1; -- 无效
10 SELECT 'a' + 'b'; -- 需要添加类型转换
11 -----
12 SELECT '1' = 1; -- True
13 SELECT '1' <> 1; -- False
```

2. Date

Whenever you are comparing data of slightly different types, you should use functions that "cast" data types

It will avoid bad surprises

```
select * from forum_posts where post_date>= '2018-03-12';
select * from forum_posts where post_date>=date('2018-03-12');
select * from forum_posts where post_date>=date('12 March, 2018');
```

Default formats vary by product, and can often be changed at the DBMS level

You can format something that is internally stored as a date and turn it into a character string that has almost any format you want

Another frequent mistake is with datetime values. If you compare them to a date (without any time component)

- March 02, 2020
14:23:52

Datetime type

↓

where issued >= <March 12> ✓
 <March 12 00:00:00>

and issued <= <March 16>
 <March 16 00:00:00> ✗

- A correct condition would select everything from March 12th at 00:00:00 to March 16th at 23:59:59 included, or March 17th at 00:00:00 excluded.

你可以使用

```
1 WHERE <column_name> IS NULL
2 WHERE <column_name> IS NOT NULL
```

4. Functions

```
select title
      || ' was released in '
      || year_released movie_release
from movies
where country = 'us'
```

Although YEAR_RELEASED is actually a number, it's implicitly turned into a string by the DBMS

In that case it's not a big issue but it would be better to use a function to convert explicitly

```
select title
      || ' was released in '
      || cast(year_released as varchar) movie_release
from movies
where country = 'us'
```

上述情况下，隐式转换和显式转换都可以

For performance reasons that will be explained later, there is no issue applying a function (or transformation) to data that is returned or constants that were input

```
select f(column1), ...
from some_table
where f(some_column) = f(some_user_input)
```

一些有用的函数

```
round(3.141592, 3)  3.142
trunc(3.141592, 3)  3.141
```

```
upper(), lower()

substr('Citizen Kane', 5, 3) 'zen'
  ↑
  1

trim(' Oops ') 'Oops'

replace('Sheep', 'ee', 'i') 'Ship'
```

5. Case

```
case upper(color)
✓ when 'Y' then 'Color'
  when 'N' then 'B&W'
  else '?'
end as color,
...
```

CASE 后也可以允许函数调用

如果你在使用 CASE 的时候遇到了 NULL 会怎么样

- NULL cannot be tested in a WHEN branch

```
case column_name
  when null then
  else ...
end
```

- An unset value MIGHT be caught by an ELSE branch
-

CASE 还有第二种形式

- So there is a second form for CASE, in which CASE isn't followed by the name of the column to test, but each WHEN branch explicitly tests data

```
case
  when died is null then
    'alive and kicking'
  else 'passed away'
end as status
```

Lecture 4

1. Distinct

如果 DISTINCT 修饰多个列，它将保证多个列去重

```
select distinct country, year_released
from movies
where year_released in (2000,2001)
```

- The selected combination (country, year_released) will be identical

2. Aggregate Function

聚合函数使用关键词 GROUP BY 提示

这个时候 SELECT 的项只能有

- GROUP BY 后面跟着的约束列
- 聚合函数

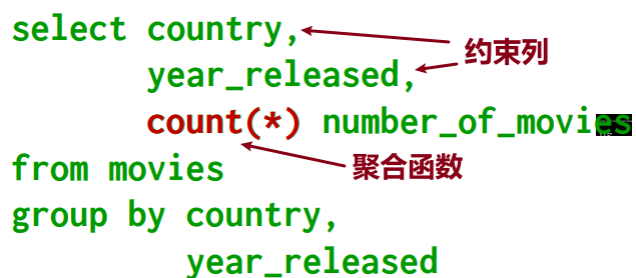
如下所示



```
select country, count(*) number_of_movies
from movies
group by country
```

Diagram annotations for the first query:

- Red arrow pointing to `country`: 约束列 (Constraint Column)
- Red arrow pointing to `count(*)`: 聚合函数 (Aggregate Function)
- Red circle around `group by country`



```
select country, year_released, count(*) number_of_movies
from movies
group by country, year_released
```

Diagram annotations for the second query:

- Red arrow pointing to `country`: 约束列 (Constraint Column)
- Red arrow pointing to `year_released`: 约束列 (Constraint Column)
- Red arrow pointing to `count(*)`: 聚合函数 (Aggregate Function)

聚合函数需要进行分组再返回 GROUP BY 的值，会更慢，相比而言 WHERE 匹配就返回

- When you apply a simple WHERE filter, you can start returning rows as soon as you have found a match
- With a GROUP BY, you must regroup rows before you can aggregate them and return results

现在，请记住，聚合行需要以某种方式对它们进行排序，而且排序总是代价高昂的操作，伸缩性不好(成本的增长速度快于已排序的行数)

使用 WHERE 和 HAVING 哪个好？

```
select country,  
       min(year_released) oldest_movie  
from movies
```

```
group by country  
having country = 'us'  
or  
where country = 'us'  
group by country
```

WHERE 肯定更好
因为 WHERE 先进行了筛选



All database management systems have an important component that is called the "query optimizer"

Aggregate functions ignore NULL !!!!

```
select max(died) most_recent_death  
from people  
where died is not null
```

In this query, the WHERE condition changes nothing to the result (perhaps it makes more obvious that we are dealing with dead people only, but for the SQL engine it's implicit)

聚合函数依据列的不同返回不同

- COUNT(*): 返回行数，因为一行中肯定有值
- COUNT(col): 不会计算该列为 NULL 的行

3. Retrieving Data from Multiple Tables

JOIN = INNER JOIN

```
select title,  
       country_name,  
       year_released  
from movies  
join countries  
on country_code = country
```

movies join countries

1	Casablanca	us	1942	us	United States	AMERICA
2	Goodfellas	us	1990	us	United States	AMERICA
3	Bronenosets Potyomkin	ru	1925	ru	Russia	EUROPE
4	Blade Runner	us	1982	us	United States	AMERICA
5	Annie Hall	us	1977	us	United States	AMERICA
6	Da Nao Tian Gong	cn	1965	cn	China	ASIA
7	Sholay	in	1975	in	India	ASIA
8	On The Waterfront	us	1954	us	United States	AMERICA
9	Lawrence Of Arabia	gb	1962	gb	United Kingdom	EUROPE
10	The Third Man	gb	1949	gb	United Kingdom	EUROPE
11	Ladri di biciclette	it	1948	it	Italy	EUROPE

We use on country_code=country to filter out unrelated rows to make a much smaller virtual talbe

4. More about Join

如果多表关联，JOIN 的顺序不会影响结果

如果在 JOIN 的时候，关联表中的某一张表中没有记录相关信息（为 NULL），那么这条数据将不会被显示

```
select m.year_released, m.title,
       p.first_name, p.surname
from movies m
      join credits c
        on c.movieid = m.movieid
      join people p
        on p.peopleid = c.peopleid
where c.credited_as = 'D'
      and m.country = 'us'
      and year_released = 2018
```

year_released	title	first_name	surname
2018	Red Sparrow	Francis	Lawrence
2018	Ready Player One	Steven	Spielberg
2018	A Star Is Born	Bradley	Cooper
2018	Mary Queen of Scots	Josie	Rourke

```
select m.movieid, m.year_released, m.title
from movies m
where m.country = 'us'
      and year_released = 2018
```

movieid	year_released	title
8987	2018	Red Sparrow
8988	2018	Ready Player One
8990	2018	A Star Is Born
8992	2018	Mary Queen of Scots
9202	2018	Black Panther
9203	2018	A Wrinkle in Time

上面的情况发生的原因是，people 表中没有与 movie 表中匹配的 peopleid，查询结果为 NULL，该信息被省略

Lecture 5

1. Outer join

An outer join tells to complete the result set with NULLs if we can't find a match in the outer joined table

✓					
✓					
✓					
✓					

LEFT OUTER JOIN 中右边表没有找到值的情况下是以 NULL 的形式返回

country_name	number_of_movies
Algeria	2
Angola	
Benin	
Botswana	
Burkina Faso	2
Burundi	
Cameroon	
Central African Republic	
Chad	
Comoros	
Congo Brazzaville	
Congo Kinshasa	
Cote d'Ivoire	
Djibouti	
Egypt	11
Equatorial Guinea	
Eritrea	
Ethiopia	
Gabon	
Gambia	
Ghana	1
Guinea	
Guinea-Bissau	1
Kenya	1
Lesotho	
Liberia	
Libya	2

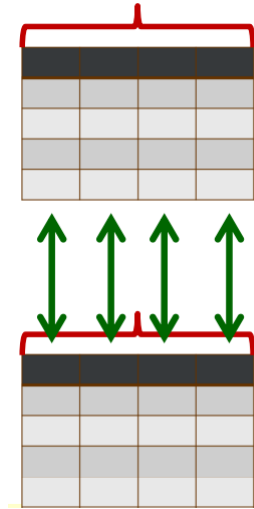
当然你可以选择用 CASE 来在 NULL 的情况下返回 0

```
select c.country_name,
       case
         when x.number_of_movies is null then 0
         else x.number_of_movies
       end number_of_movies
from countries c
left outer join
(select country as country_code,
  count(*) as number_of_movies
 from movies
 group by country) x
on x.country_code = c.country_code
```

2. Set Operators

UNION

UNION requires two commonsensical conditions: to combine the results of two queries



1. They must return the same number of columns
2. The data types of corresponding columns must match

UNION eliminates duplicates because when you put two duplicate-free relations together

UNION ALL 不会去重，但是这不意味着你用它就因为它不去重

UNION ALL 的意思是告诉数据库，我合并的这两个东西没有重复，你不要像 UNION 一样浪费时间去查找重复了

3. Subqueries

在之前的子查询中都是在 FROM 里面

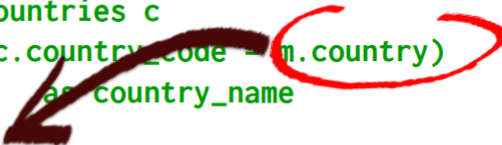
事实上子查询还可以在 SELECT 里面，并且，它可以调用主查询的值

我们说它 CORRELATED with the outer (main) query


```

select m.title, m.year_released,
       (select c.country_name
        from countries c
        where c.country_code = m.country)
       as country_name
from movies m
where m.country <> 'us'

```



- 首先它执行外层查询，WHERE m.country <> 'us'
- 一旦条件匹配，它开始执行内部的子查询，并进行显示
- What happens if we don't find the country name?
- The subquery would return nothing, also known as NULL
- So, strictly speaking, a subquery after the SELECT is more equivalent to a LEFT OUTER JOIN

不同的子查询位置与相关性

a subquery in the FROM
cannot be correlated.
from clause

uncorrelated

a subquery after SELECT
usually is
select list

correlated

where clause
a subquery after WHERE
usually is.

uncorrelated
or correlated

- Correlated queries in the WHERE clause are used with the (NOT) EXISTS construct
- NEVER try to correlate an IN()

子查询可以放在 IN() 里面

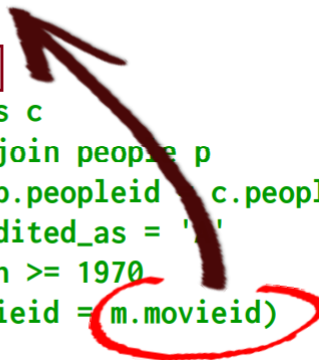
- IN () means an implicit DISTINCT !!!

我们看下 EXIST

```

select distinct m.title
from movies m
where exists
  (select null
   from credits c
    inner join people p
      on p.peopleid = c.peopleid
   where c.credited_as = 'A'
    and p.born >= 1970
    and c.movieid = m.movieid)

```



- 这里我们只是想看 movie 是否满足 exist 中的条件
- 不想返回任何值
- 所以子查询 SELECT NULL 表示：我就是查一下是否存在，我不需要你里面的值

你就是把它换成 hello 也可以

```

select m.title
from movies m
where exists
  (select 'hello'
   from credits c
    inner join people p
      on p.peopleid = c.peopleid
   where c.credited_as = 'A'
    and p.born >= 1970
    and c.movieid = m.movieid)

```

So to summarize we can switch correlated and uncorrelated subqueries, and turn them into joins

Lecture 6

1. NULL and Logical Operators

Arithmetic Operators

- ALL NULL

col+NULL -> NULL

col-NULL -> NULL

col*NULL -> NULL

col/NULL -> NULL

Logical operators

- Compare: ALL NULL

(col > NULL) -> NULL

(col = NULL) -> NULL

- col is NULL -> True or False
- TRUE and NULL -> NULL
- FALSE and NULL -> FALSE
- TRUE or NULL -> TRUE
- FALSE or NULL -> NULL

IN

- col in ('a', 'b', null)
 - If col is 'a', the result is: TRUE or FALSE or NULL -> TRUE
 - If col is 'c', the result is: FALSE or FALSE or NULL -> NULL
 - If col is NULL, the result is : NULL or NULL or NULL -> NULL
- col not in ('a', 'b', null)
 - If col is 'a', the result is: FALSE and TRUE and NULL -> FALSE
 - If col is 'c', the result is: TRUE and TRUE and NULL -> NULL
 - If col is NULL, the result is : NULL and NULL and NULL -> NULL

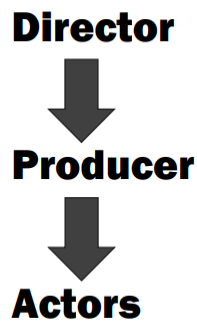
2. Ordering

Sorts the result of the query table unchanged

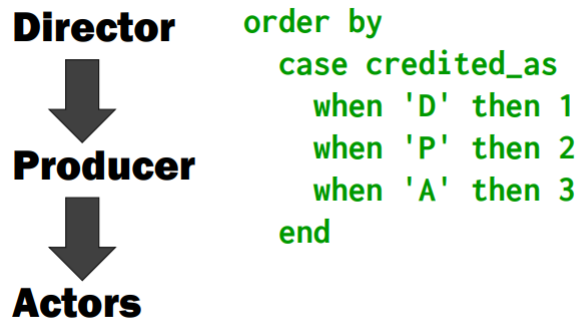
表是不会变的！

不同的 DBMS 有的会把 NULL 放在前面，有的会放在后面

如果你不想按照字母表/规定好的排序方案排序



例如上面，你想按照 Director - Producer - Actors 排序



可以使用 CASE

3. Window Functions

Aggregate window function

With DBMS products that support window functions, every aggregate function can be used as a window function

Instead of specifying with GROUP BY the subset on which the result is computed, you say OVER (PARTITION BY ...)

```
min(year_released)
  over (partition by country)
```

One related characteristics is that they can only appear after the SELECT, not in the WHERE clause

```
min(year_released) over()
```

对于整个表的聚合函数

In the same way that you can have an aggregate function without a GROUP BY when you want ONE result for the whole table, you can have an empty OVER clause to indicate that you want the result computed over all rows selected

Ranking window function

There are also some window functions that provide ranking capabilities

These functions are original functions and unrelated to either aggregate functions or scalar functions

When we talk about "ranking", of course, we implicitly talk about "ordering"

With a ranking window function you MUST have an ORDER BY clause in the OVER()

over (order by ...)

**over (partition by ...
order by ...)**

```
select title,  
       country,  
       year_released,  
       row_number()  
       over ( partition by country  
              order by year_released desc) rn  
from movies
```

id	title	country	year_released	
1	Casablanca	us	1942	5
2	Blade Runner	us	1982	2
3	On The Waterfront	us	1954	4
4	Lawrence Of Arabia	gb	1962	1
5	Annie Hall	us	1977	3
6	Goodfellas	us	1990	1
7	The Third Man	gb	1949	2
8	Citizen Kane	us	1941	6
9	Bicycle Thieves	it	1948	1
10	The Battleship Potemkin	ru	1925	1
11	Sholay	in	1975	1
12	A Better Tomorrow	hk	1986	1

- row_number

row_number() assigns distinct, sequential numbers to everyone

row_number()



- rank()

rank() assigns the same number to ties, but there is a gap in ranks.

rank()



- dense_rank

dense_rank() also assigns the same number to ties, with no gap

dense_rank()



Lecture 7

1. Fuzzy Search

What people use is known as "full-text search"

- isolating and standardizing words
- 按照单个的词 hit 的次数排名

2. Transaction

Many products (MySQL is one of them) start in "autocommit" mode, which means that every change is automatically committed and cannot be undone otherwise than by running the reverse operation (not always easy)

Some interfaces such as JDBC (Java DataBase Connectivity) always start in autocommit mode, even when accessing products such as Oracle that never natively work in such a mode

DDL

- For products such as Oracle and MySQL, any change to the structure of the database (DDL operations) automatically commits all pending changes to data
- This isn't the case with SQL Server or PostgreSQL, for which a transaction can contain DDL statements (they can be rolled back). Switching to another DBMS is a mine field

Deletes often are virtual deletes

- In fact they are updates of a flag
- One good reason is foreign keys, which prevent physical deletion
- A web store cannot physically delete an item no longer on sale but which appears in many past orders, you would have to delete much of your sales history

3. Insert

What happens when you omit a column?

- The value inserted is the default one if defined
 - Otherwise it will be NULL
-

Sequence

如果你想要一个自增的 id

- You can use special database objects called sequences, which are simply number generators. By default they start with 1 and increase by 1 (they can reach values that are very, very big)
- Or AUTO-NUMBERED COLUMN

4. Loading Data from a File

PostgreSQL uses the (non-standard) SQL COPY command to load a file located on the server. The default format is tabseparated

Sometimes you encounter really weird text file formats. Using a scripting language to generate INSERT statements is usually the simplest solution

Lecture 8

1. Update

Update is the command that changes column values. You can even set a non-mandatory column to NULL

UPDATE is a SET operation

PostgreSQL support the same older join type of syntax allowing to join the updated table to the one from which we are getting data.

```
update movies_2
set duration = i.duration,
    color = case i.color
              when 'C' then 'Y'
              when 'B' then 'N'
            end
from us_movie_info i
where i.title = movies_2.title
      and i.year_released = movies_2.year_released
      and movies_2.country = 'us'
```

} Join

MySQL allows a join with the newer syntax


```
update movies_2 m
  inner join us_movie_info i
    on i.title = m.title
   and i.year_released = m.year_released
set m.duration = i.duration,
    m.color = case i.color
                when 'C' then 'Y'
                when 'B' then 'N'
                end
where m.country = 'us'
```

2. Delete

However, to keep volumes under control, it's frequent to copy old rows to a history table, then delete them from the "active" table

- 把不活跃的数据（删除 flag = true）备份到 history table 中

除了 DELETE，你还可以使用 TRUNCATE，它不会保留记录

- DELETE saves data for rollback before removing it, 所以会慢一点
 - TRUNCATE cannot be rolled back and is far more efficient. It's better not to use it
 - 所以最好别用...
-

Constraints

- 它保证了数据的一致性 guarantee that data remains consistent
- 你在 INSERT，UPDATE 和 DELETE 都可能触发它
- they ensure that whatever happens, you'll always be able to make sense of ALL pieces of data in your database

3. Functions

Procedural extensions provide all the bells and whistles of true programming languages (they were often inspired by programming languages such as PL/I or ADA)

Procedural

variables

conditions

loops

arrays

error management

They also support all DML statements

To retrieve data from the database into your variables, you can use

```
SELECT ... INTO ...
```

```
select col1, col2, ...  
into local_var1, local_var2, ...  
from ...
```

能用 SQL 写出来的命令一定不要用自定义函数！

```
select country_name(country_code), title, ...  
from movies  
where ...
```

```
create function country_name(p_code varchar2)  
return countries.country_name%type  
as  
    v_name countries.country_name%type;  
begin  
    select country_name  
    into v_name  
    from countries  
    where country_code = p_code;  
    return v_name;  
end;
```

NO
禁
Het
Không

自定义函数必须一行一行的执行，相比 SQL 可以调用优化器来说，它的效率太低了！函数不可以用来执行一个查询！！

Lecture 9

1. Procedure

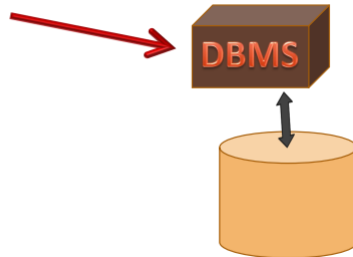
你对数据库执行的一系列操作，可以将它们转化成一个 **procedure**

我们可以发出一个命令来执行服务器上的所有内容，而不是发出几个 SQL 语句，检查它们的结果

create procedure myproc

```
delete from ...  
where ...  
+ Conditional logic  
update ...  
set ...  
where ...
```

execute myproc



Transaction" and "commit"/"rollback") can be performed inside the procedure or outside it

Procedure 的好处

- 首先，调用过程是与数据库的单一交互，使用 Procedure 不会浪费时间通过网络与远程服务器聊天
 - 安全性，你不知道 Procedure 背后实际做了什么
-

你可以使用 perform 在一个 procedure 中调用另一个 procedure

2. Trigger

SELECT will never trigger anything

Only write operations do

Function

- Modify input on the fly
before insert / update
for each row

- Check complex rules
before insert / update / delete
for each row
- Manage data redundancy
after insert / update / delete
for each row

3. Auditing

当有人修改表的时候，触发器会把这些更改记录下来

比如你在执行下面的操作

```
insert into people(first_name, surname, born)
values('Ryan', 'Gosling', 1980);
```

将会有审计表记录在后面

people_audit

auditid	peopleid	type_of_change	column_name	old_value	new_value	changed_by	time_changed
1	95	I	first_name	NULL	Ryan	root@localhost	... 23:05:01
2	95	I	surname	NULL	Gosling	root@localhost	... 23:05:01
3	95	I	born	NULL	1980	root@localhost	... 23:05:01
4	96	I	first_name	NULL	George	root@localhost	... 23:05:02
5	96	I	surname	NULL	Clooney	root@localhost	... 23:05:02
6	96	I	born	NULL	1961	root@localhost	... 23:05:02

你在用 trigger 更改某一特定行的时候，或者用 EACH ROW TRIGGER 的时候

- 一定不要跟其它的行扯上关系，很容易出问题

DON'T
look at
other rows
of the
modified table
in
for each row
triggers

- trigger 增加了很多复杂性，一个简单的操作可能会因为编写糟糕的触发器而表现怪异
- 此外，它们经常被用来“修复”本来就不应该存在的问题，而且往往是由于糟糕的数据库设计造成的

- 尽可能不要用 `trigger`
 - 如果用户可以不是通过您的程序访问数据库 `multiple access point`，你最好写一个 `trigger` 去监控
-

注意点

- Use functions to reuse complex expressions
- Don't query the database in functions
- Use procedures for business operations
- Procedures aren't where to be heavily procedural
- Triggers: the last line of defense. Only when you can't do otherwise

Lecture 10

1. Index

indexed values are binary values

You have actually already created indexes without knowing it whenever you declare a

- PRIMARY KEY or
- UNIQUE constraint

an index is created behind your back.

The only way to find quickly in a big table that a supposedly unique value is already recorded is to index the column

-
- indexes use a lot of storage, sometimes more than data!
 - 对表的增删改操作 requires moving things around in the tree
-

Unique index 和 Unique constraint 一般使用哪个

- Unique constraint 是轻量级的，只是一种逻辑约束
- Unique index 是要用数据结构去实现的东西

一般，如果只是要求数据 Unique，使用 Unique constraint

2. Performance

Indexes massively improve performance

- Scanning a half-million row table for a unique value may take a fraction of a second, but it will be instant(0.001s) with an index.

根据不同的特点选择是否使用索引

- "OLTP" (OnLine Transaction Processing) usually makes heavy use of indexes
- massive batch processes scan a lot, should better not use index

Indexing every column means, once again, that every INSERT will have to

- write into the table
- and into every index

应该索引哪些列？

- often used search criteria
- 有区分性 **Selective**：该列中的值的不会有很多重复的（所以 **PRIMARY KEY** 和 **UNIQUE** 比较好）

3. Execution Plan

If there is an index on a column such as COUNTRY, will the DBMS use it?

Not necessarily

The optimizer may decide not to use an index.

Depending on the DBMS, the "explain" command will display the plan directly or populate a table that must be queried



```

movies=# explain select distinct m.title, m.year_released
movies=# from movies m
movies=#         inner join credits c
movies=#             on c.movieid = m.movieid
movies=#         inner join people p
movies=#             on p.peopleid = c.peopleid
movies=# where p.surname = 'Bogart';

```

QUERY PLAN

```

HashAggregate (cost=5.29..5.30 rows=1 width=222)
-> Nested Loop (cost=2.16..5.28 rows=1 width=222)
    -> Hash Join (cost=2.16..4.51 rows=1 width=4)
        Hash Cond: (c.peopleid = p.peopleid)
        -> Seq Scan on credits c (cost=0.00..1.97 rows=97 width=8)
        -> Hash (cost=2.15..2.15 rows=1 width=4)
            -> Seq Scan on people p (cost=0.00..2.15 rows=1 width=4)
                Filter: ((surname)::text = 'Bogart'::text)
    -> Index Scan using movies_pkey on movies m (cost=0.00..0.76 rows=1 width=226)
        Index Cond: (movieid = c.movieid)

(10 rows)

movies=#

```

- 即使是完全相同的表结构、数据、索引，在不同的 DBMS 下 **execution plan** 也不一定一样，因为它们用的算法不同
- 事实上很难通过比较两个不同的 **execution plan** 直观判断哪个更好
- 查看 **execution plan** 只是想看看你做的一些比如索引的东西是不是真的会被 optimizer 用到

4. Use Index or Not

- 联合索引要注意 WHERE 的顺序，如果顺序不对不会用到
- Like 也不会用到
- 函数也不会用到
- lower()、upper() 也不会用到
- 日期函数也不会用到
- 数据类型的强制转换也不会用到

如果实在想在用函数的情况下还能用到索引，做一点 **cheating**

先把使用函数的列放在后面，不对外显示，然后对这个列做索引

```

select *
from people
where soundex(surname) = soundex('Stuart')

```

people

peopleid	first_name	surname	born	died	surname_soundex
1	James	Stewart	1908	1997	S363
2	Humphrey	Bogart	1899	1957	B263



有的 DBMS 支持用函数建虚拟列，来背后做索引加速，函数需要

- 是 确定的 Deterministic
 - 只要输入不变、输出一定不变
 - 如果你改变数据库配置，函数输出结果变化，索引无效
- 当然有很多函数是 确定的 函数

Lecture 11

1. View

它只是一个有命名的查询

视图可以是您想要的复杂查询，并且通常会返回一些不像表那样规范化的东西，但更容易理解

when you change something in a table you change its value and it's reflected in the view

Once the view is created, I can query the view exactly as if it were a table

views are permanent objects in the database

View 创建后就固定了查找表的各列，即使有其它表中之后创建了新的列，View 也不会显示它们！

2. Deeper in Views

View 的问题在于，只要您还没有看到它们是如何定义的，您就不知道它们可能有多复杂，那么索引和优化的东西都有可能无效

View 检索出来的很多列和内容在真实场景下可能都用不到，而你调用 View 意味着它需要先查询好所有的信息，很浪费时间

3. Security

不同的 View 给不同的用户，你可以对他们隐藏某些数据或某些列

在这种 View 下，不同用户运行的完全相同的查询将返回不同的行

Beware when you modify the definition of a view

- If you simply drop and recreate it
- you lose the privileges
- Use CREATE OR REPLACE

Lecture 12

1. Change data through views

不要随便通过 View 去更新一些数据，View 大部分不是符合表的设计原则，对它进行修改可能会对数据库造成破坏

如果是一个来自关联表的 View

```
[55000] ERROR: cannot update view "vmovies"  
详细: Views that do not select from a single table or view are not automatically updatable.  
建议: To enable updating the view, provide an INSTEAD OF UPDATE trigger or an unconditional ON UPDATE DO INSTEAD rule.
```

Most products will express concern and prevent you from doing it

WITH CHECK OPTION

可以检查你对 View 的一些行为

If updating the view directly is impossible, in many cases (remember when we were displaying the country name) what should be applied to base tables is fairly obvious and can be performed by dedicated stored procedures

There is a special type of trigger called an

- BEFORE TRIGGER: update 之前触发，触发完后继续 update
- AFTER TRIGGER: 先update，update 完后触发
- INSTEAD OF TRIGGER: 直接接管，执行触发器内部的 procedure，无视你的 update

2. Data Dictionary

the set of tables that contain information about the objects in the database, collectively known as the Data Dictionary Or Catalog

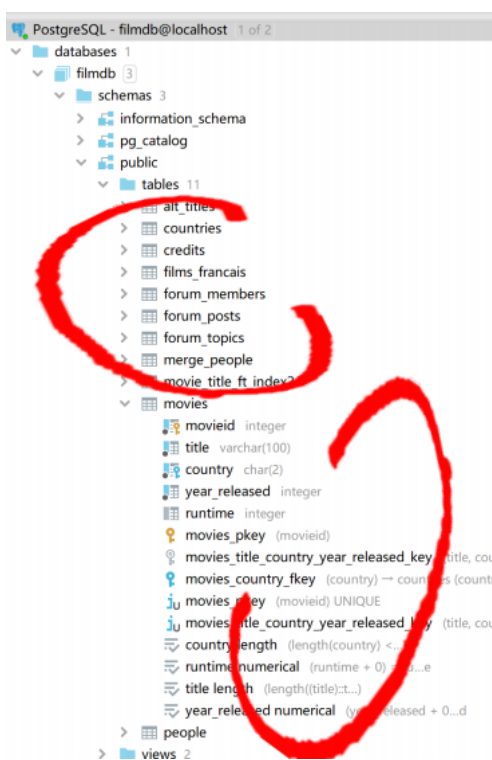
可以理解为目录

They are using all the features we have seen (you only have privileges to read views and only see what is relevant to your account)

You always have ONE catalog per database

you can reference tables in another schema

Any database stores "metadata" that describes the tables in your database (and not only them)



你不可以用 DDL 来修改 system table

Read access to these tables is provided through system views

DBAs often use queries on the catalog to generate other SQL queries

Lecture 13

1. Query Optimizer

One way to improve efficiency is to keep data dictionary information (meta-data) in a shared cache to avoid additional queries

Query Optimizer

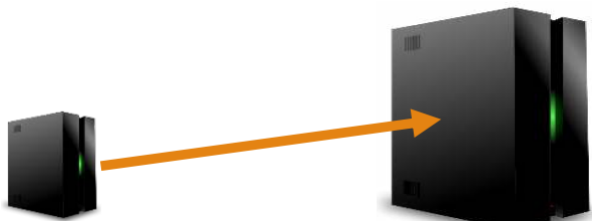
- Logical transforms
- Indexes Volumes Storage
- Hardware performance
- System load
- Settings

As most applications run exactly the same SQL statements again and again, a DBMS will cache a parsed query for reuse

- For MySQL, it will be cached for a session
- Least Recently Used
- We primarily recognize identical queries by computing a text checksum

2. Scaling

- Scaling UP



- Scaling OUT



- 集群的问题在于它遵循收益递减的规律：添加第二台服务器将不会使您的容量增加一倍，而且在实践中，人们拥有最多 2、3 或 4 台机器的集群

一个大问题是涉及多个服务器的事务。请记住，事务是原子操作

- TWO-PHASE COMMIT
- 在提交之前，要询问所有相关的服务器是否准备好提交。如果你没有得到安全信号，你可以 roll back



延迟 Latency

- 此外，还有延迟问题。集群中的所有机器可能并不相邻，出于安全原因(火灾、洪水.....)，它们可能在不同的数据中心相距几英里
- 即使信息传播得很快，多交换(TWO-PHASE COMMIT)也可能成为一个敏感问题

Lecture 14

1. NoSQL

NOT ONLY SQL

Non-Relational database (No Tables)

used for big data & real-time web apps

Advantages of NoSQL over RDBMS

- Handles Big Data
- Data Models – No predefined schema
- Data Structure – NoSQL handles unstructured data
- Cheaper to manage
- Scaling – Scale out / horizontal scaling

Advantages of RDBMS over NoSQL

- Better for relational data
 - Normalization
 - Well known language (SQL)
 - Data Integrity
 - ACID Compliance (atomicity, consistency, isolation, durability)
-

Type

- Document Databases [MongoDB, CouchDB]
- Column Databases [Apache Cassandra]
- Key-Value Stores [Redis, Couchbase Server]
- Cache Systems [Redis, Memcache]
- Graph Databases [Neo4]