Fuzzy Search（模糊查询）

```
create table movie_title_ft_index
    (title_word    varchar(30) not null,
     movieid       int not null,
           primary key(title_word, movieid),
           foreign key (movieid)
              references movies(movieid)  )
```

建一个表，使多种不同的情况（如不是全文本查询）都能在该表中找到

```
select movieid
from (select movieid,
          rank()
          over (order by hits desc) as rnk
      from (  select movieid,
                   count(*) as hits
              from movie_title_ft_index
              where title_word in
                ('SPACE', 'ODYSSEY', '2001')
              group by movieid ) q1  ) q2
where rnk = 1
```

Such a query might serve the purpose. We count how many of the words we find by film for which at least one is found.

Then we rank. Notice that here we are interested by ties.

Transaction（事务处理）

This idea that one business operation may translate into several database operations that must all succeed or fail is of prime importance to a DBMS. Some products require a special command to start a transaction (BEGIN is sometimes START)

begin transaction

insert
update
delete

Other products such as Oracle or DB2 automatically start a transaction if you aren't already in one when you start modifying data.

```
runoobdb=# BEGIN;
DELETE FROM MOVIES WHERE MOVIEID = 25;
ROLLBACK;
```

```
runoobdb=# BEGIN;
DELETE FROM MOVIES WHERE MOVIEID = 25;
COMMIT;
```

事务会在commit或rollback后结束

commit表示提交操作，rollback表示删除操作。

（自begin开始删，但也可以删transaction的一部分，但少用）

A transaction ends when you issue either COMMIT (which is like an OK button) or ROLLBACK, which cancels everything you have done since the beginning of a transaction (you can sometimes cancel a subpart of a transaction, but it's not much used)

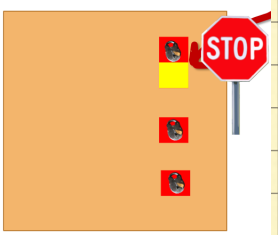
commit OK

ROLLBACK automatically undoes everything. You can no longer do it after COMMIT.

rollback Cancel

begin transaction

One important thing is concurrency: databases are usually meant to share data between many users. While you change data, and for all the duration of the transaction, other users are prevented by the DBMS from changing the same data.

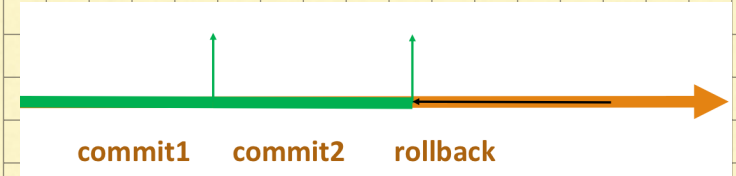


commit

在并发操作中，数据库常需要在不同用户间分享数据。

当在transaction途中改变数据时，DBMS会阻止他人改变同一个地方的数据

每一个commit会定义数据的连续状态，rollback会返回上一个已知的连续状态，这些过程都是由跳跃进行的，每个transaction都是跳跃。



有许多DBMS有Autocommit模式，意味着改变会自动commit且不能取消，只能通过逆操作返回原先位置。JDBC默认是自动提交模式。

在一些数据库中(如ORACLE和MySQL),使用DDL改变数据库的结构时,会自动commit对数据的延时操作.

DDL操作通常不会立刻更新.

Insert

```
insert into table_name
  (column1, column2, ..., columnn)
values (value1, value2, ..., valuen)
       ...
       (valuep, valueq, ..., valuez)
```

Most products (exceptions are Oracle and SQLite) allow inserting several rows in one statement, with a comma separated list of row data between parentheses.

SQL Server · IBM DB2 · MySQL · PostgreSQL

许多数据库都支持一次性插入许多行。(除了ORACLE和SQLite)

```
insert into table_name
values (value1, value2, ..., valuen)
```

若不指定对应列，则会默认插入全部列，按数据库中的顺序。
(非常危险！因为列可能会改顺序或删除、添加)

```
insert into table_name
  (col1, col2, col4)
values (value1, value2, value4)
```

若忽略了某些行，则会自动补上默认值，或是NULL。

```
create table <table_name>
        (...
         <column_name>    <data type>
              default <default_value> not null,
        ...)
```

想要有默认值，在建表时指定。
但在插入时也可以显性地指定该处为NULL。
在指定日期时间的默认值时，常用 CURRENT_TIMESTAMP 默认。

在数据库中，同时有两个用户读取/插入相同的值的可能性很高，对第一个有效，但第二个会报约束违背错误。
解决方法是让系统管理新标识符的生成。

Sequence

```
create sequence movie_seq
```

数字生成器，从1开始每次加1。

IBM DB2 · ORACLE · SQL Server 2012 · PostgreSQL

```
insert into movies(movieid, ...)
values(movie_seq.nextval, ...)
insert into credits(movieid, ...)
values(movie_seq.currval, ...)
```

```
insert into movies(movieid, ...)
values(next value for movie_seq, ...)
insert into credits(movieid, ...)
values(previous value for movie_seq,
       ...)
```

```
insert into movies(movieid, ...)
values(nextval('movie_seq'), ...)
insert into credits(movieid, ...)
values(currval('movie_seq'), ...)
```

可以在插入后得到新数字(保证唯一)
插入的时候不能用currval，会报错

AUTO-NUMBERED COLUMN

```
create table movies
   (movieid      int not null identity primary key,
   ...
```

SQL Server · IBM DB2 · PostgreSQL · MySQL · SQLite

```
create table movies
   (movieid      int generated as identity primary key,
   ...
```

SQL Server · IBM DB2 · PostgreSQL · MySQL · SQLite

```
create table movies
   (movieid      serial primary key,
   ...
```

As usual, syntax differs. PostgreSQL actually creates a sequence behind the scene, which it "attaches" to the table so that dropping the table drops the sequence.

SQL Server · IBM DB2 · PostgreSQL · MySQL · SQLite

```
create table movies
   (movieid      int not null auto_increment primary key,
   ...
```

SQL Server · IBM DB2 · PostgreSQL · MySQL · SQLite

```
create table movies
   (movieid      integer primary key,
   ...
```

SQL Server · IBM DB2 · PostgreSQL · MySQL · SQLite

```
define
movieseq.nextval
as default value for movieid
```

Oracle (since version 12, it wasn't possible before) can do it PostgreSQL style, but more explicitly.

ORACLE >= 12c

在添加了自增列后，在插入时就可以忽略了。
若想回溯上一个值，可使用 special variable such as @@identity with SQL Server,
或 lastval() with PostgreSQL or last_insert_id() with MySQL

Microsoft SQL Server

```
insert into movies(title, ...)
values('Some Movie Title', ...)
insert into credits(movieid, ...)
values(@@identity, ...)
```

IBM DB2

```
insert into movies(title, ...)
values('Some Movie Title', ...)
insert into credits(movieid, ...)
values(identity_val_local(), ...)
```

PostgreSQL

```
insert into movies(title, ...)
values('Some Movie Title', ...)
insert into credits(movieid, ...)
values(lastval(), ...)
```

MySQL

```
insert into movies(title, ...)
values('Some Movie Title', ...)
insert into credits(movieid, ...)
values(last_insert_id(), ...)
```

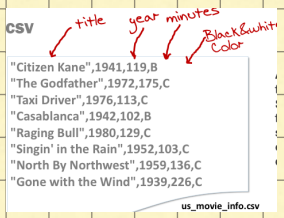
SQLite

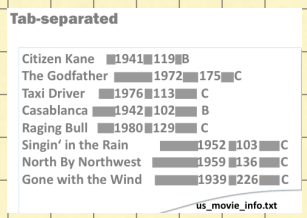
```
insert into movies(title, ...)
values('Some Movie Title', ...)
insert into credits(movieid, ...)
values(last_insert_rowid(), ...)
```

Loading data from file

CSV (Comma Separated Values)

Tab-separated





MySQL

txt或csv

```
load data infile '/tmp/us_movie_info.fmt'
into table us_movie_info
fields terminated by ','
optionally enclosed by '"'
```

→ 文件在服务器上.

```
load data local infile 'us_movie_info.csv'
into table us_movie_info
fields terminated by ','
optionally enclosed by '"'
```

→ 文件在本机中.

Microsoft SQL Server

文件在服务器上.

```
bulk insert us_movie_info
from 'C:\temp\us_movie_info.txt'
with (rowterminator='0x0a')
```

注意 Linux文件, 默认是tab分隔的.

```
bulk insert us_movie_info
from '/tmp/us_movie_info.csv'
with (formatfile='C:\temp\us_movie_info.fmt')
```

对csv文件, 需要指定格式

formatfile 样式如下:

```
11.0 ← SQL Serve 版本
4 ← 原csv几行数据
1 SQLCHAR 0 150 "\"," 1 title ""
2 SQLCHAR 0 4 "," 2 year_released ""
3 SQLCHAR 0 3 "," 3 duration ""
4 SQLCHAR 0 1 "\n" 4 color ""
```

每行都是对数据每列的描述.

number, type, 0, maxlength, 终止符, 对应哪列

实际对应的CSV文件

```
"Citizen Kane",1941,119,B
"The Godfather",1972,175,C
"Taxi Driver",1976,113,C
"Casablanca",1942,102,B
"Raging Bull",1980,129,C
"Singin' in the Rain",1952,103,C
"North By Northwest",1959,136,C
"Gone with the Wind",1939,226,C
```

Virtual Table: SQL Serve中有一特殊函数 openrowset() 可以以表的形式看文件.

(虚表, 实际上不存在)

可以在loading过程中执行 INSERT、SELECT、转换数据等操作.

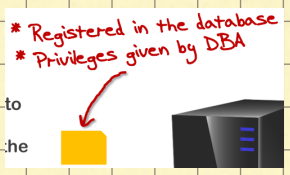
(省略了再次构建表)

```
select replace(title, '"', '') title,
    year_released,
    duration,
    case color
      when 'B' then 'N'
      when 'C' then 'Y'
    end color
from openrowset(bulk 'C:\temp\us_movie_info.csv',
        formatfile='C:\temp\us_movie_info.fmt') as virtual_table
```

从本机中加载文件需要特殊的组件 BCP(Bulk Copy)

ORACLE

ORACLE与SQL Server类似，但需要在数据库中为文件目录注册，且帐号需有特别的权限




```
create table virtual_us_movie_info
      (title       varchar2(150),
       year_released  number(4),
       duration      number(3),
       color        char)
organization external (default directory input_dir
    access parameters
      (records delimited by '\n'
       fields terminated by '   ')
    location ('us_movie_info.txt'))
```

name given by DBA

Tab

使用ORACLE后，可以创建一个永久的虚表用于投影到文件中，默认格式是CSV.
当查询该表时，实际上从文件中读取了数据
可做 INSERT...SELECT 操作.

要从本机中加载文件，要用特殊的组件 sqlldr
有一种特殊的叫 control file 的的文件描述了该文件及其数据如何投影到表中.

PostgreSQL

从服务器上

```
copy us_movie_info
from '/tmp/us_movie_info.txt'
```

从本机中

```
copy us_movie_info
from '/tmp/us_movie_info.csv'
with (format csv)
```

默认格式是 tab-separated.

```
\copy us_movie_info
from '/tmp/us_movie_info.csv'
with (format csv)
```

psql command

在psql（命令行工具）中有类似的命令.

SQLite

```
.separator ','
.import 'us_movie_info.csv' us_movie_info

.separator '   '
.import 'us_movie_info.txt' us_movie_info
```

（只能本机，服务器和客机都是本机）

XML格式：

```
<row><title>Citizen Kane</title><year>1941</year>...</row>
<row><title>The Godfather</title><year>1972</year>...</row>
<row><title>Taxi Driver</title><year>1976</year>...</row>
<row><title>Casablanca</title><year>1942</year>...</row>
...
```

JSON

对于一些比较奇特的文件格式，只能用脚本语言生成 SQL 语句再运行