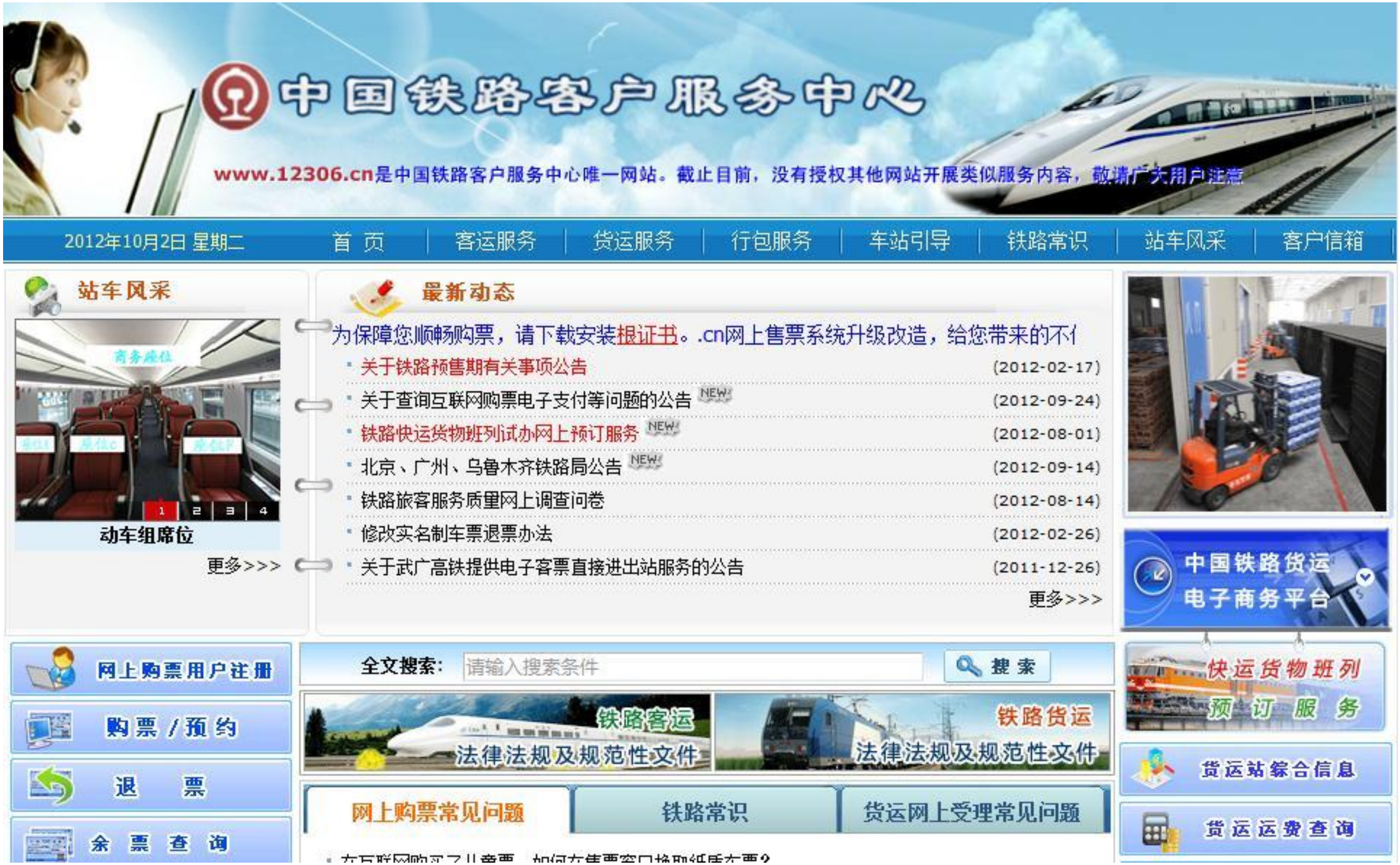
Introduction to SQL

Group members:

熊国桢、许赞、吴伊宁
曹柳星、王哲、吴宪宇

What is the hottest website during the National Holiday?



中国铁路客户服务中心

www.12306.cn是中国铁路客户服务中心唯一网站。截止目前，没有授权其他网站开展类似服务内容，敬请广大用户注意

2012年10月2日 星期二

首 页 | 客运服务 | 货运服务 | 行包服务 | 车站引导 | 铁路常识 | 站车风采 | 客户信箱

站车风采

动车组席位

更多>>>

最新动态

为保障您顺畅购票，请下载安装根证书。.cn网上售票系统升级改造，给您带来的不(

关于铁路预售期有关事项公告	(2012-02-17)
关于查询互联网购票电子支付等问题的公告 NEW!	(2012-09-24)
铁路快运货物班列试办网上预订服务 NEW!	(2012-08-01)
北京、广州、乌鲁木齐铁路局公告 NEW!	(2012-09-14)
铁路旅客服务质量网上调查问卷	(2012-08-14)
修改实名制车票退票办法	(2012-02-26)
关于武广高铁提供电子客票直接进出站服务的公告	(2011-12-26)

更多>>>

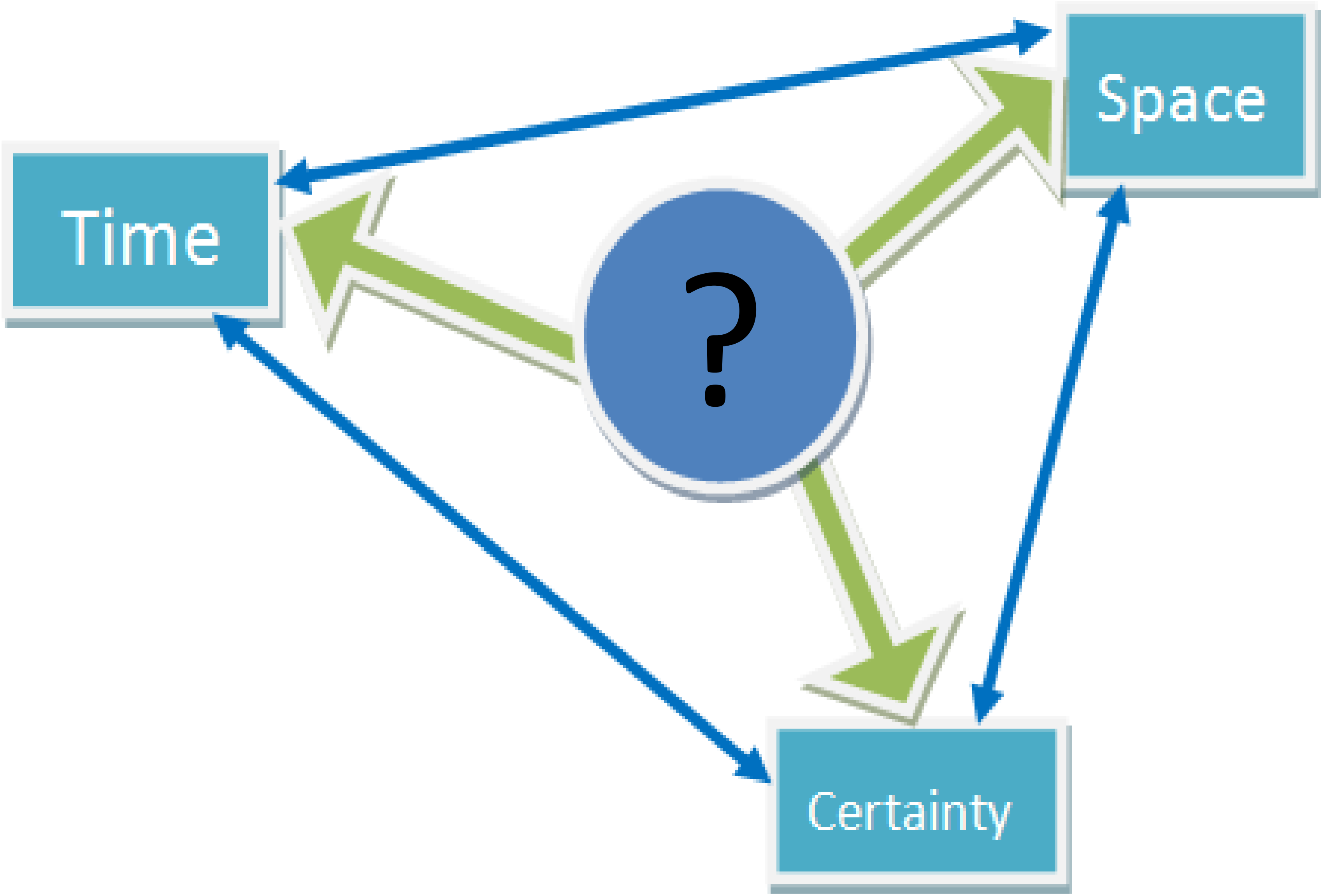
中国铁路货运
电子商务平台
网上购票用户注册
购票 / 预约
退 票
余 票 查 询

全文搜索：请输入搜索条件 搜 索

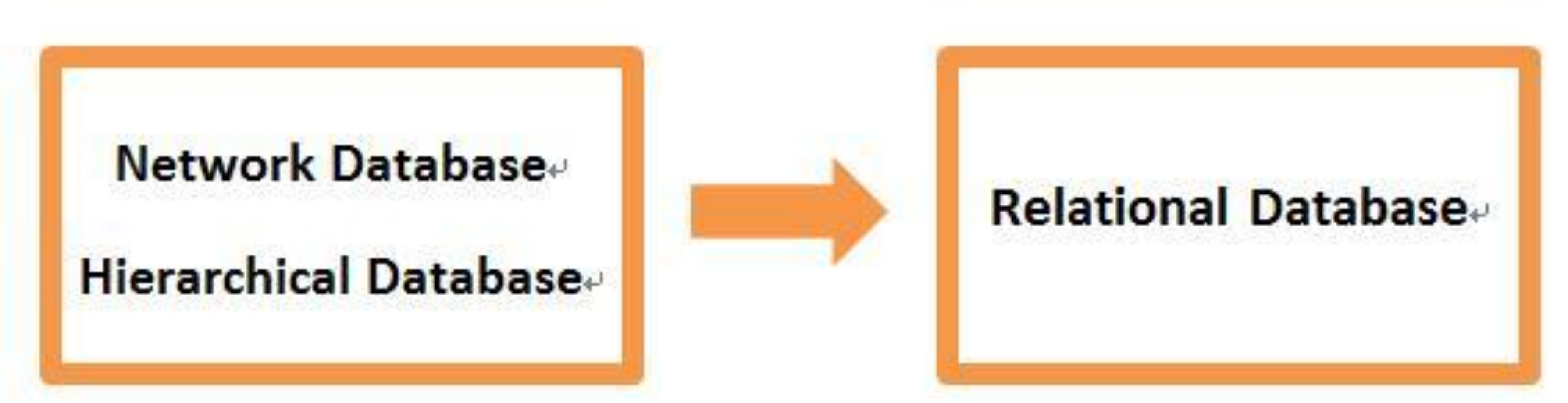
网上购票常见问题 | 铁路常识 | 货运网上受理常见问题

- 在互联网购买了儿童票，如何在售票窗口换取纸质车票？

Dance with Database


Space
Time
?
Certainty

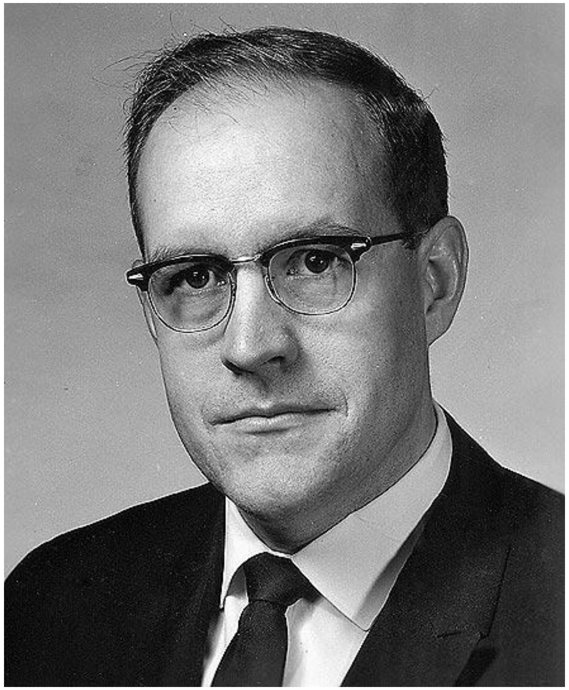
How does it work?



Network Database↵

Hierarchical Database↵

Relational Database↵

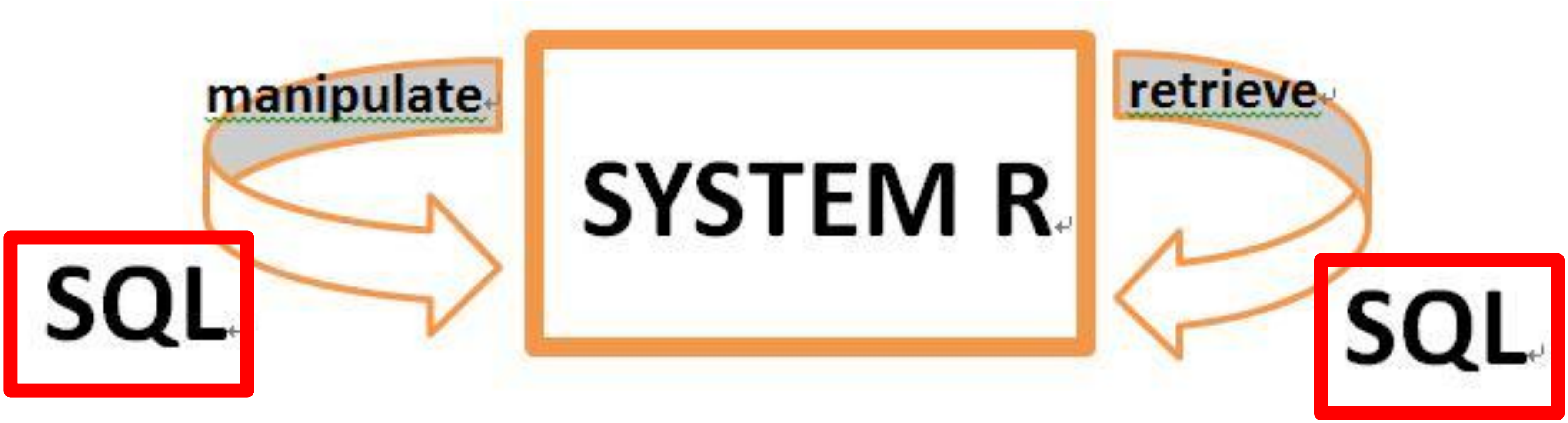


Dr. E. F. Codd

《A Relational Model of Data for Large Shared Data Banks》

IBM

Donald D. Chamberlin
Raymond F. Boyce



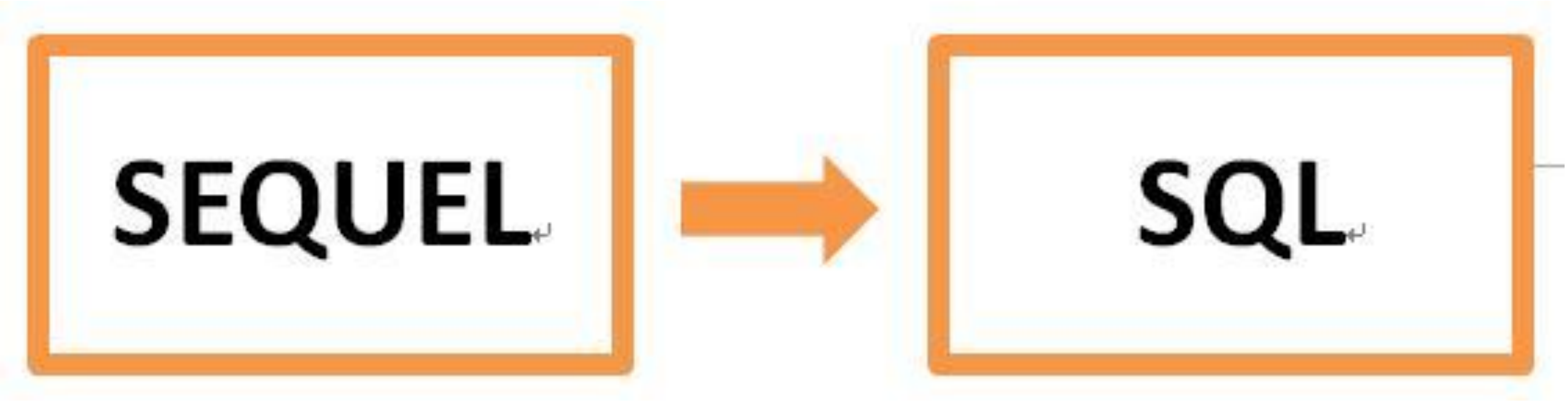
manipulate

retrieve

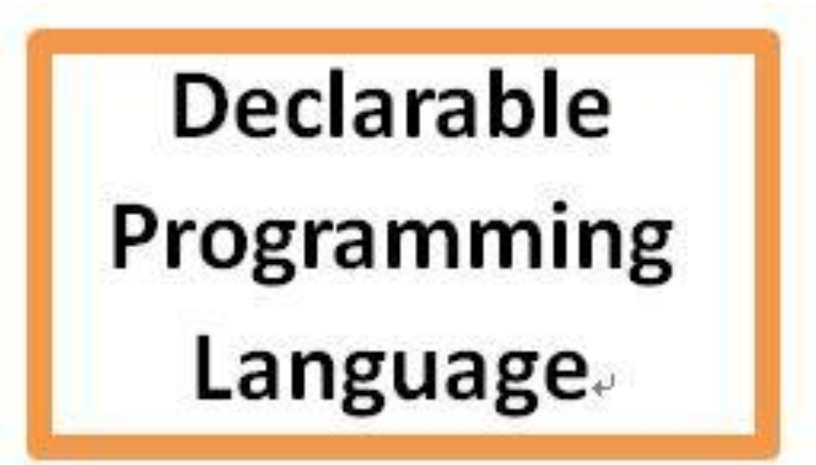
SYSTEM R

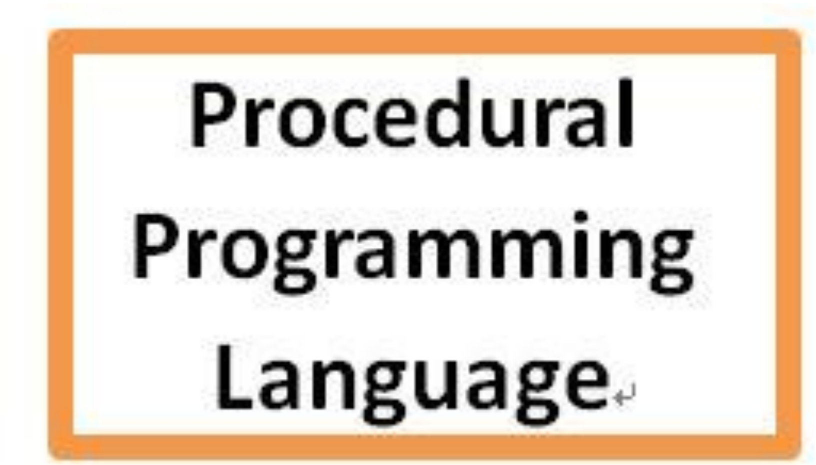
SQL

SQL



Structured English Query Language **Structured Query Language**


Declarable
Programming
Language↵


Procedural
Programming
Language↵

SQL

C, C++, Java

• Grammar	Formulation
• Optimization	More users
• NoSQL	Huge number of users

Structured Query Language

- Data Definition
- Data Manipulation
- Data Query
- Data Control

Category	Verbs	Functions
Data Definition Language	Create, Drop	Create and drop table, view, index
Data Query Language	Select	Select from tables
Data Manipulation Language	Insert, Delete, Update	Manipulate data
Data Control Language	Grant, Revoke	Give or revoke the rights to manipulate a chosen table

Grammar of SQL

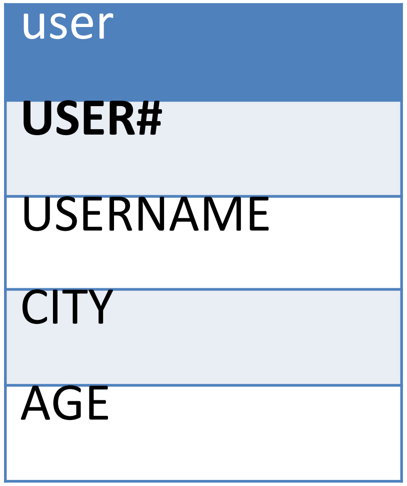
- **DDL:CREATE&DROP**
- DML:INSERT, DELETE & UPDATE
- DQL: SELECT
- DCL: GRANT&REVOKE

DDL—CREATE

CREATE TABLE <NAME>
(<Attribute1> Datatype,
<Attribute2> Datatype,…)

<u>Example:</u>

```
  CREATE  TABLE user (
  USER# INT  ,
  USERNAME VARCHAR(45) ,
  CITY VARCHAR(45)  ,
  AGE INT  ,
  PRIMARY KEY (USER#) );
```



user

USER#

USERNAME

CITY

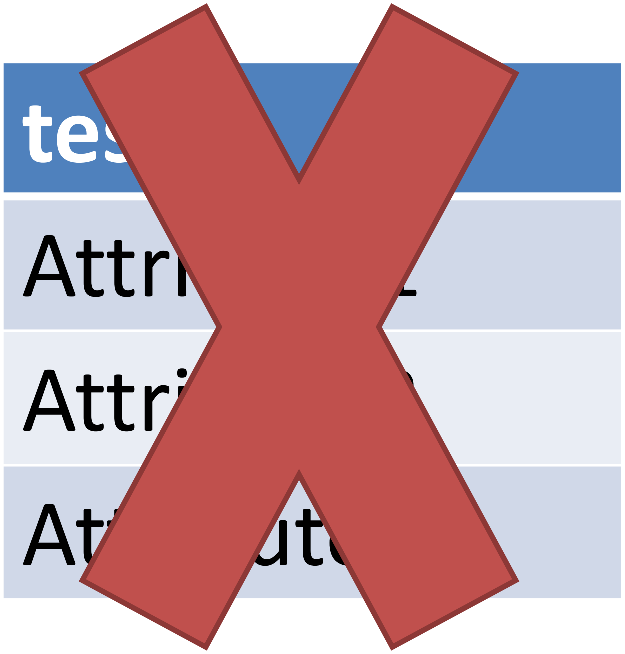
AGE

DDL—DROP

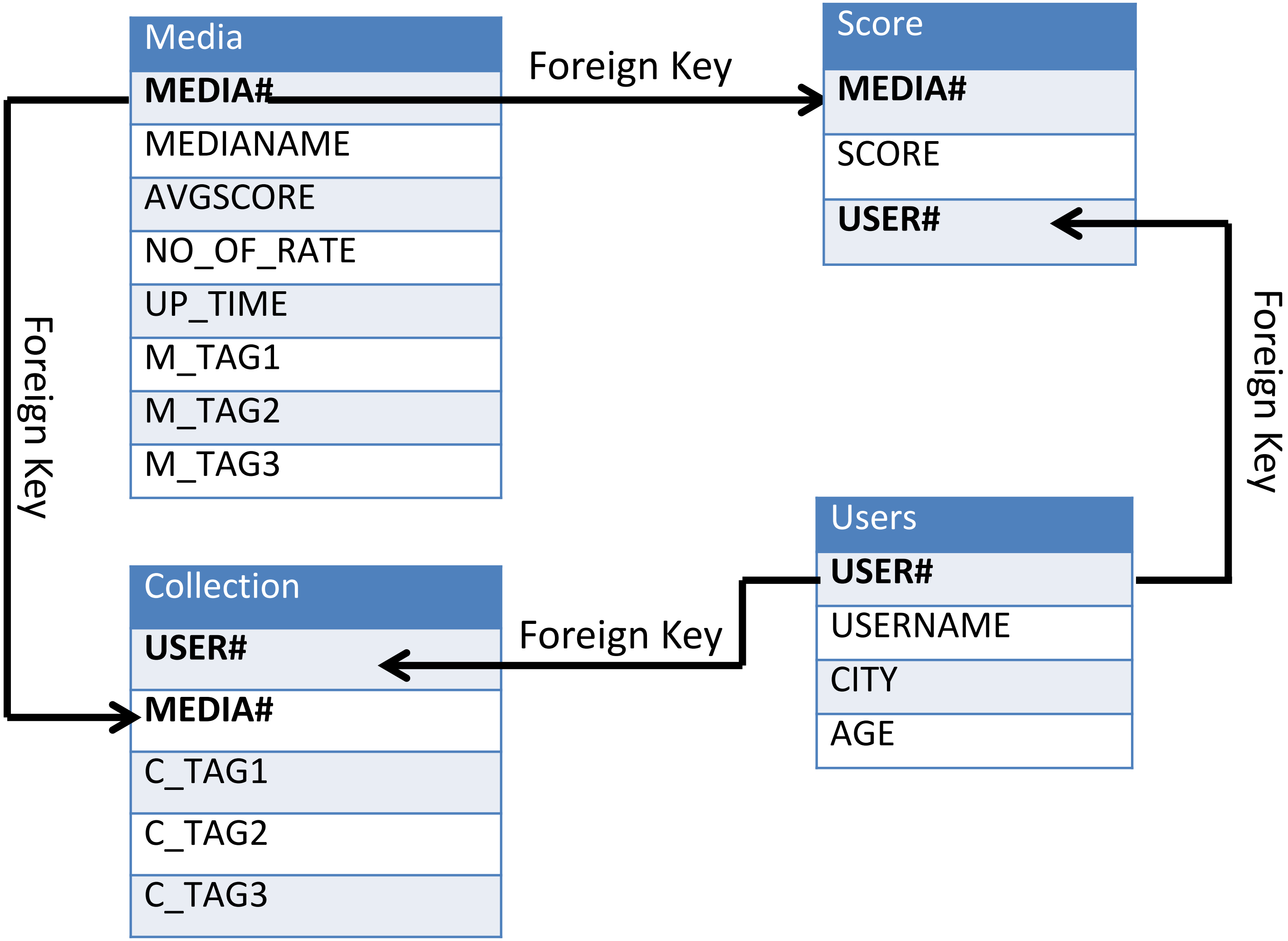
DROP TABLE <NAME>

Example:

DROP TABLE test



Glance at Our Relational DB



Media
MEDIA#
MEDIANAME
AVGSCORE
NO_OF_RATE
UP_TIME
M_TAG1
M_TAG2
M_TAG3

Score
MEDIA#
SCORE
USER#

Collection
USER#
MEDIA#
C_TAG1
C_TAG2
C_TAG3

Users
USER#
USERNAME
CITY
AGE

Grammar of SQL

- DDL:CREATE&DROP
- **DML:INSERT, DELETE & UPDATE**
- DQL: SELECT
- DCL: GRANT&REVOKE

DML—INSERT

```
INSERT INTO <NAME>(<Attribute1>,<Attribute2>,...)
VALUES
(<Value of A1>, <Value of A2>,...),
(<Value of A1>, <Value of A2>,...),
...
```

Example:

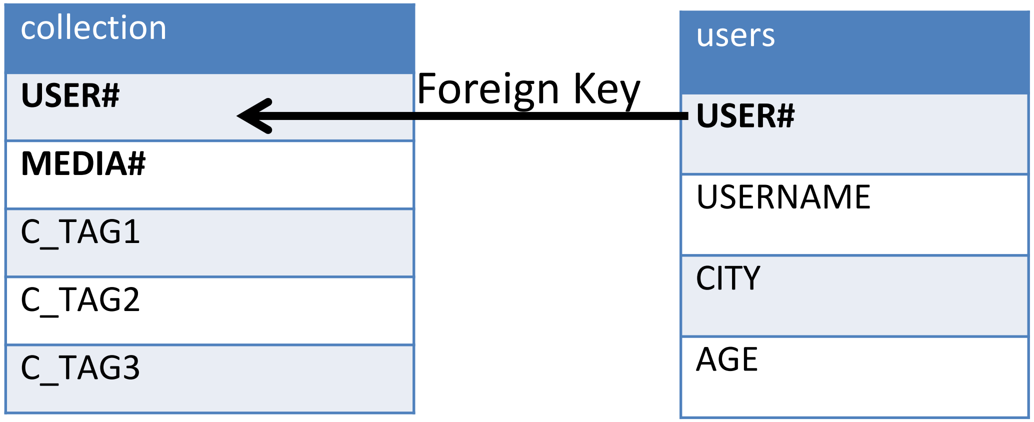
```
INSERT INTO user(`USER#`,USERNAME,CITY,AGE)
VALUES
(98,   'Penny Liu',    'Shanghai',  19),
(3,    'David Wu',     'Beijing',   21),
(105,  'John Green', 'New York',    18),
(78,   'Amy Liang',    'Xi’an',     23),
(893,  'Suki Hsu',     'Hong Kong', 17);
```

USER#	USERNAME	CITY	AGE
98	Penny Liu	Shanghai	19
3	David Wu	Beijing	21
105	John Green	New York	18
78	Amy Liang	Xi’an	23
893	Suki Hsu	Hong Kong	17

DML—INSERT(cont’d)

```
INSERT INTO ? collection
VALUES
(135, 1347, `English`, NULL,  NULL);
```

USER#	USERNAME	CITY	AGE
98	Penny Liu	Shanghai	19
3	David Wu	Beijing	21
105	John Green	New York	18
78	Amy Liang	Xi’an	23
893	Suki Hsu	Hong Kong	17



collection

USER#

MEDIA#

C_TAG1

C_TAG2

C_TAG3

Foreign Key

users

USER#

USERNAME

CITY

AGE

DML—DELETE

DELETE FROM<NAME>
WHERE<Constraints>

Example:

```
DELETE FROM media
WHERE AVGSCORE<6
```

MEDIA #	AVGSCORE	NO_OF_RATE
1045	9.0	1
1734	9.0	1
1057	7.0	1
1489	9.0	2
1347	10.0	1
~~**1984**~~	~~5.5~~	~~4~~
1578	7.0	2

MEDIA#	SCORE	USER#
1045	9	98
1057	7	98
1578	8	105
~~**1984**~~	~~3~~	~~3~~
~~**1984**~~	~~6~~	~~105~~
1347	10	78
1489	8	3
1734	9	893
1489	10	111
~~**1984**~~	~~7~~	~~111~~
1578	6	893
~~**1984**~~	~~6~~	~~893~~

DML—UPDATE

UPDATE<NAME>
SET<Attribute1>=<Value>
WHERE<Constraints>

Example:

```
UPDATE MEDIA
SET
AVGSCORE=(AVGSCORE*NO_OF_RATE+6)/(NO_OF_RATE+1),
NO_OF_RATE=NO_OF_RATE+1
WHERE MEDIA#=1984;
```

MEDIA#	AVGSCORE	NO_OF_RATE
1045	9.0	1
1734	9.0	1
1057	7.0	1
1489	9.0	2
1347	10.0	1
1984	5.3	3
1578	7.0	2

MEDIA#	AVGSCORE	NO_OF_RATE
1045	9.0	1
1734	9.0	1
1057	7.0	1
1489	9.0	2
1347	10.0	1
1984	5.5	4
1578	7.0	2

Grammar of SQL

- DDL:CREATE&DROP
- DML:INSERT, DELETE & UPDATE
- **DQL: SELECT**
- DCL: GRANT&REVOKE

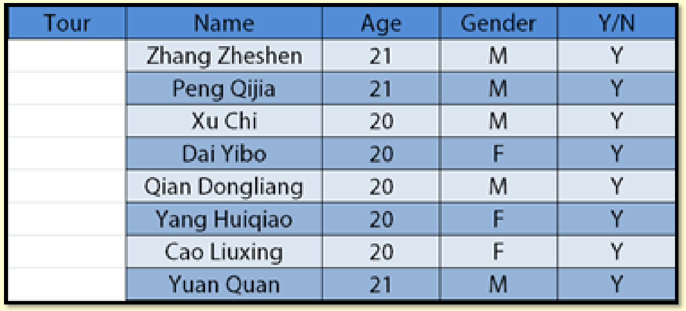
DQL—SELECT

SELECT <Attribute1>,<Attribute2>…
FROM <NAME>
WHERE <Constraints>
Group by <Attribute x>
Having <Constraints>
Order by <attribute>

Example:

```
SELECT NAME,AGE,Y/N
FROM Tour
WHERE age >= 20 AND Y/N=Y;
```

Operation Set



Tour	Name	Age	Gender	Y/N
	Zhang Zheshen	21	M	Y
	Peng Qijia	21	M	Y
	Xu Chi	20	M	Y
	Dai Yibo	20	F	Y
	Qian Dongliang	20	M	Y
	Yang Huiqiao	20	F	Y
	Cao Liuxing	20	F	Y
	Yuan Quan	21	M	Y

Selection $Tour = \sigma_{Y/N=Y \wedge Age\geq 20}(IErs)$

DQL—SELECT

USER#	USERNAME	CITY	AGE
98	Penny Liu	Shanghai	19
3	David Wu	Beijing	21
105	John Green	New York	18
78	Amy Liang	Xi’an	23
893	Suki Hsu	Hong Kong	17

```
SELECT USER#,USERNAME,CITY,AGE
FROM user
WHERE CITY = 'Beijing' OR  CITY= 'Xi’an' AND AGE > 22
```

USER#	USERNAME	CITY	AGE
3	David Wu	Beijing	21
78	Amy Liang	Xi’an	23

What’s the result?

```
SELECT USER#,USERNAME,CITY,AGE
FROM user
WHERE (CITY = 'Beijing' OR  CITY= 'Xi’an') AND AGE > 22
```

USER#	USERNAME	CITY	AGE
78	Amy Liang	Xi’an	23

DQL—SELECT

Prep. in WHERE

Prep.	Example	Meaning
NOT(<Attribute>=<Value>)	**NOT**(CITY=Beijing)	CITY< >Beijing
<Attribute> **BETWEEN** <Value1> and <Value2>	AGE **BETWEEN** 18 AND 25	AGE >=18 **AND** AGE <= 25
<Attribute> **IN**(<Value1>, <Value2>,…)	CITY **IN**(Beijing, Shanghai)	CITY=Beijing **OR** CITY=Shanghai
<Attribute> **LIKE** <Description>	CITY **LIKE** ‘Bei%’	Value of CITY is like ‘Bei%’

DQL—SELECT

Prep. in WHERE(cont’d)

LIKE for Fuzzy Query

-% represents unknown string with any length>=0

- _ represents exactly **one** character

WORD BOX

Mp3
mp3
song.mp3
Cambridge
Oxford
mails

A.LIKE m%	B.LIKE m_ _
C.LIKE %m%	D.LIKE %m_ _

DQL—SELECT

GROUP BY

USER#	MEDIA#	C_TAG1	C_TAG2	C_TAG3
98	1045	database	model	NULL
893	1734	Japan	Germany	History
893	1347	Car	Spindle	NULL
78	1489	Illumination	NULL	NULL
3	1578	Math	Markov	NULL
78	1347	Spindle	Pro/e	CAM
105	1057	English	Test	TOEFL

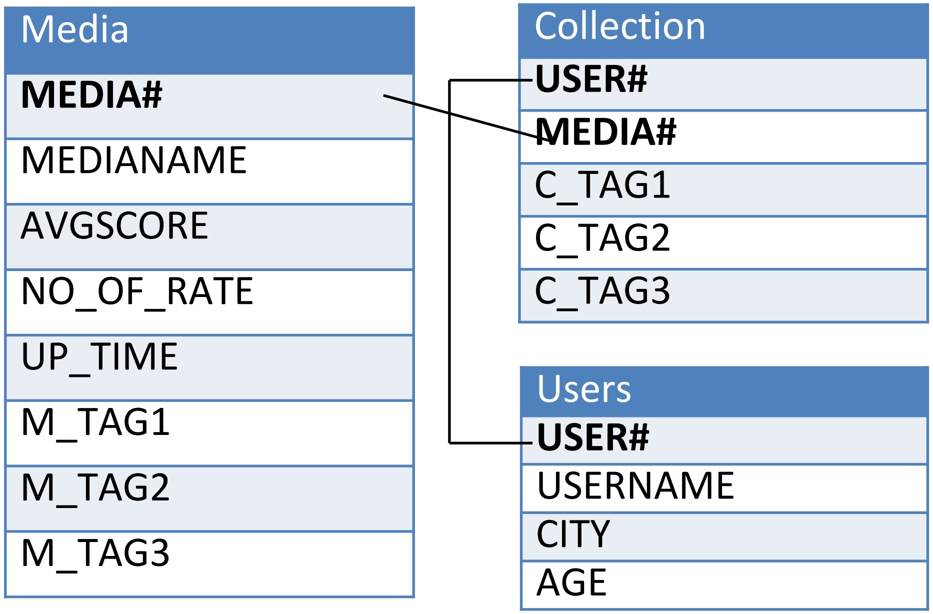
Example:

```
SELECT
USER#,COUNT(MEDIA#)
FROM collection
GROUP BY USER#
HAVING COUNT(MEDIA#)=1;
```

User#	COUNT(Media#)
98	1
3	1
105	1
3	1
105	1

DQL—SELECT

Join Tables



Media

MEDIA#

MEDIANAME

AVGSCORE

NO_OF_RATE

UP_TIME

M_TAG1

M_TAG2

M_TAG3

Collection

USER#

MEDIA#

C_TAG1

C_TAG2

C_TAG3

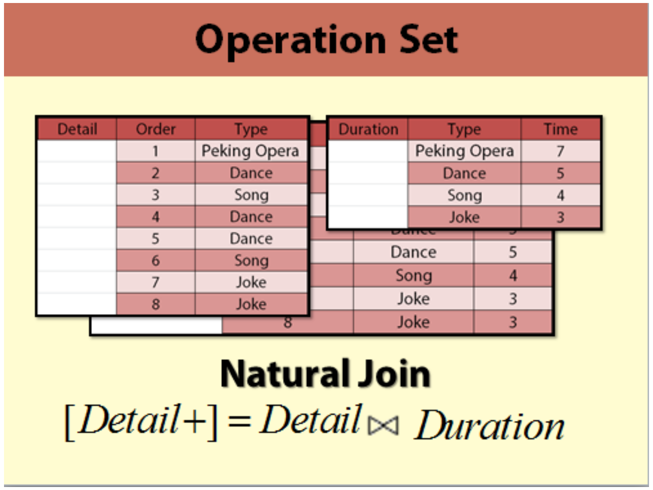
Users

USER#

USERNAME

CITY

AGE



Operation Set

Detail	Order	Type
	1	Peking Opera
	2	Dance
	3	Song
	4	Dance
	5	Dance
	6	Song
	7	Joke
	8	Joke

Duration	Type	Time
	Peking Opera	7
	Dance	5
	Song	4
	Joke	3

Dance
5
Song
4
Joke
3
8
Joke
3

Natural Join

$$[Detail+] = Detail \bowtie Duration$$

Example:

SELECT c.USER#,u.USERNAME,c.MEDIA#,MEDIANAME,C_tag1,C_tag2,C_tag3
FROM media m,user u,collection c
WHERE m.MEDIA#=c.MEDIA# AND u.USER#=c.USER#

JOIN TABLE ON m.MEDIA#=c.MEDIA#
JOIN TABLE ON u.USER#=c.USER#

DQL—SELECT

Subquery select

MEDIA#	AVGSCORE	NO_OF_RATE
1045	9.0	1
1734	9.0	1
1057	7.0	1
1489	9.0	2
1347	10.0	1
1984	5.5	4
1578	7.0	2

AVG(AVGSCORE)=8.1

```
SELECT MEDIANAME,MEDIA#,AVGSCORE
FROM media
WHERE AVGSCORE<
(SELECT AVG(AVGSCORE) FROM media)
```

Works or Not?

```
SELECT MEDIANAME,MEDIA#,AVGSCORE
FROM media
WHERE AVGSCORE< AVG(AVGSCORE)
```

DQL—SELECT

Subquery select

User#	Media#	C_tag1	C_tag2	C_tag3
98	**1045**	database	model	NULL
893	**1734**	Japan	Germany	History
893	**1347**	Car	Spindle	NULL
78	**1489**	Illumination	NULL	NULL
3	**1578**	Math	Markov	NULL
78	**1347**	Spindle	Pro/e	CAM
105	**1489**	English	Test	TOEFL

```
SELECT USER#
FROM collection
WHERE collection.media=ANY
(SELECT MEDIA# FROM collection WHERE USER#=78)
AND USER#<>78
```

What's the result?

Answer:893,105

Grammar of SQL

- DDL:CREATE&DROP
- DML:INSERT, DELETE & UPDATE
- DQL: SELECT
- **DCL: GRANT&REVOKE**

Example:

```
GRANT SELECT ON media TO ADMIN1;
REVOKE  ALL     ON media TO ADMIN1;
```

(ALL=SELECT,INSERT,DELETE,UPDATE)

• Grammar	Formulation
• Optimization	More users
• NoSQL	Huge number of users

SQL Optimization

- Change the sentence structure
- Using INDEX
- Data mining

Change the sentence structure

- SNS function: We want to know the number of users who come from Beijing or Tianjin in the database .

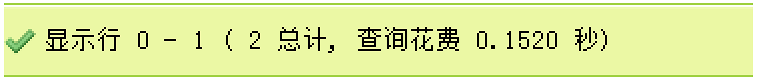
- **SELECT** city, **COUNT** (*)

  FROM user

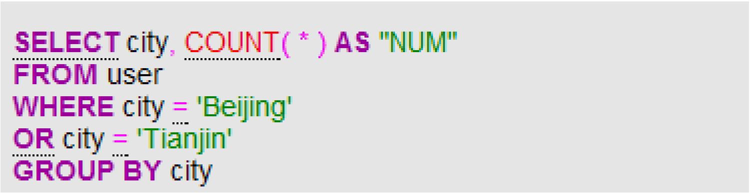
  WHERE city=‘Beijing’

  OR city=‘Tianjin’

  GROUP BY city



✓ 显示行 0 - 1 (2 总计, 查询花费 0.1520 秒)



```
SELECT city, COUNT( * ) AS "NUM"
FROM user
WHERE city = 'Beijing'
OR city = 'Tianjin'
GROUP BY city
```

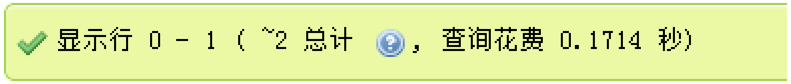
- **SELECT** city, **COUNT**(*)

  FROM user

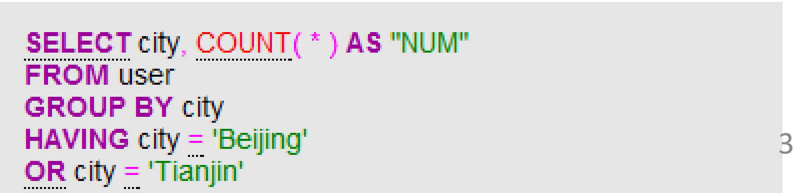
  GROUP BY city

  HAVING city = ‘Beijing’

  OR city = ‘Tianjin’



✓ 显示行 0 - 1 (~2 总计 ?, 查询花费 0.1714 秒)



```
SELECT city, COUNT( * ) AS "NUM"
FROM user
GROUP BY city
HAVING city = 'Beijing'
OR city = 'Tianjin'
```

Testing on INDEX

INDEX: Data structure that help us retrieve data efficiently and is important when data gets larger.

Step 1: Create a database by phpMyAdmin for testing

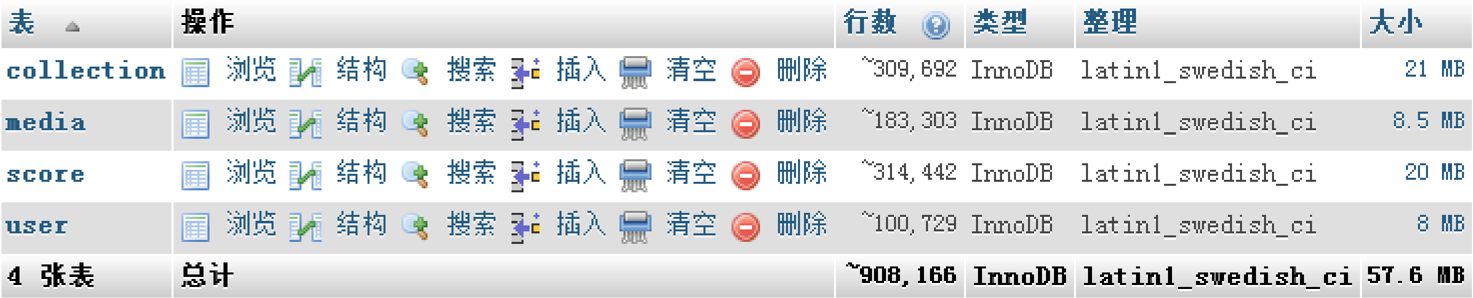
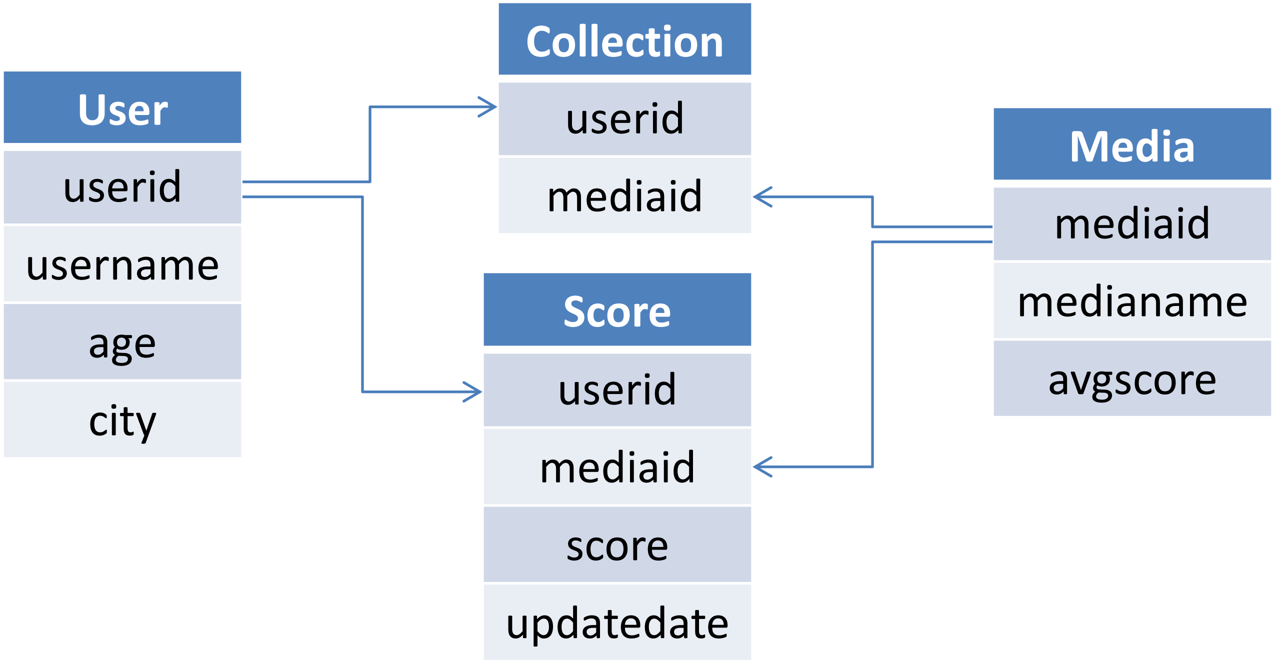


表	操作	行数	类型	整理	大小
collection	浏览 结构 搜索 插入 清空 删除	~309,692	InnoDB	latin1_swedish_ci	21 MB
media	浏览 结构 搜索 插入 清空 删除	~183,303	InnoDB	latin1_swedish_ci	8.5 MB
score	浏览 结构 搜索 插入 清空 删除	~314,442	InnoDB	latin1_swedish_ci	20 MB
user	浏览 结构 搜索 插入 清空 删除	~100,729	InnoDB	latin1_swedish_ci	8 MB
4 张表	总计	~908,166	InnoDB	latin1_swedish_ci	57.6 MB



User
userid
username
age
city

Collection
userid
mediaid

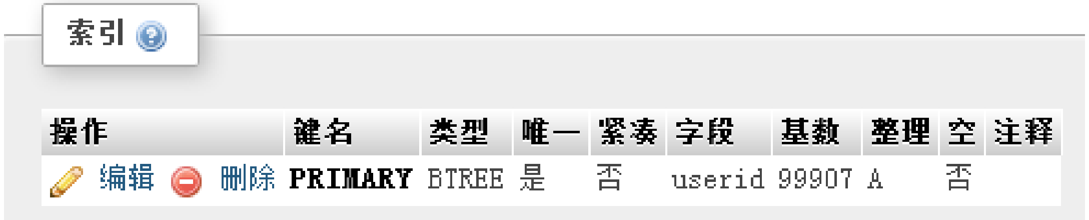
Score
userid
mediaid
score
updatedate

Media
mediaid
medianame
avgscore

Testing on INDEX

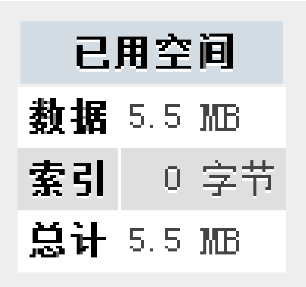
Step 2: Test a selection without index

SNS function: we want to find people who live in Beijing and have over 40 years old.

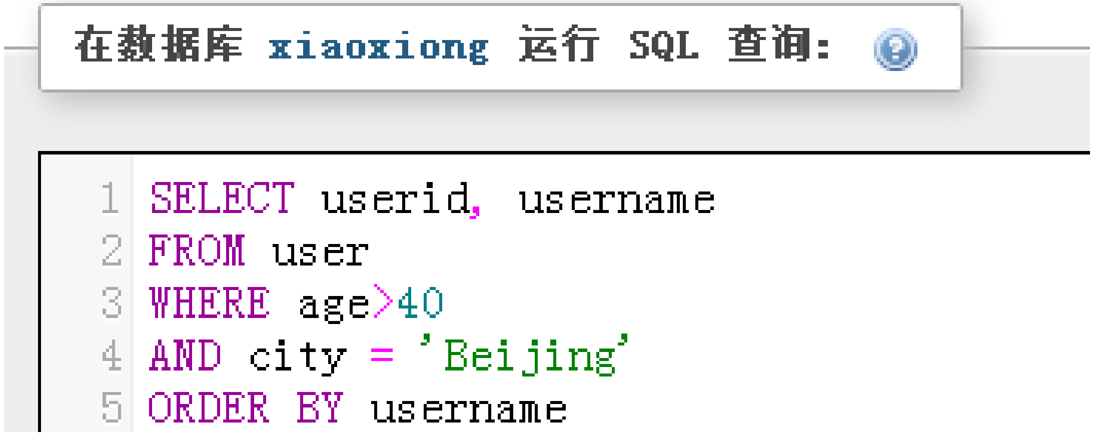


索引

操作	键名	类型	唯一	紧凑	字段	基数	整理	空	注释
编辑 删除	PRIMARY	BTREE	是	否	userid	99907	A	否	



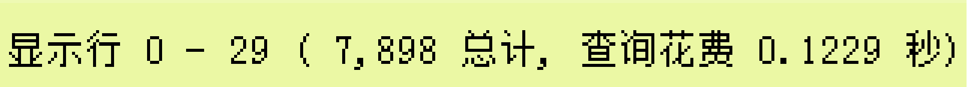
已用空间	
数据	5.5 MB
索引	0 字节
总计	5.5 MB



在数据库 xiaoxiong 运行 SQL 查询:

```
SELECT userid, username
FROM user
WHERE age>40
AND city = 'Beijing'
ORDER BY username
```

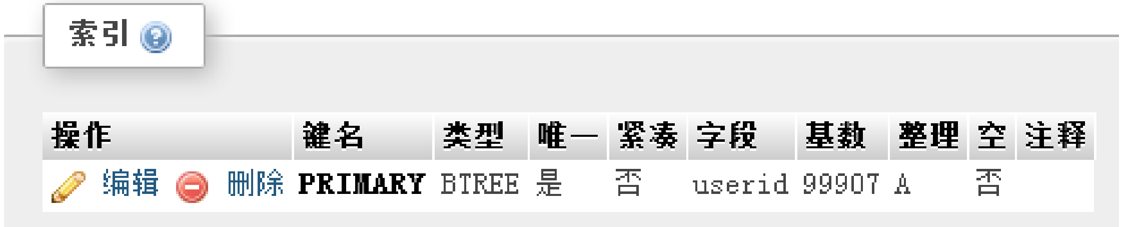
Result:



显示行 0 - 29 (7,898 总计, 查询花费 0.1229 秒)

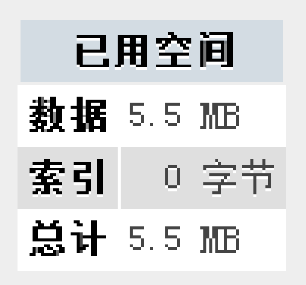
Testing on INDEX

Step 3: Create an index in ‘username’

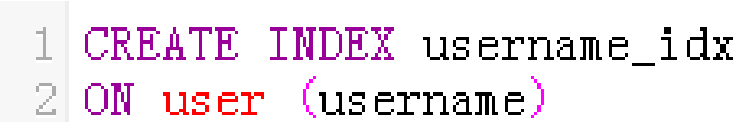


索引

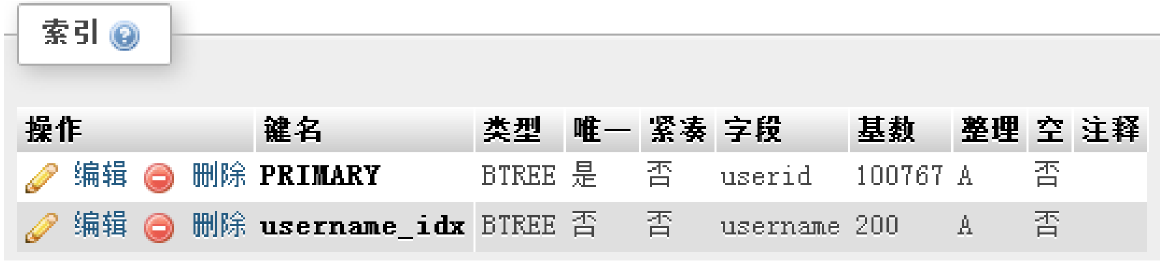
操作	键名	类型	唯一	紧凑	字段	基数	整理	空	注释
编辑 删除	**PRIMARY**	BTREE	是	否	userid	99907	A	否	



已用空间	
数据	5.5 MB
索引	0 字节
总计	5.5 MB

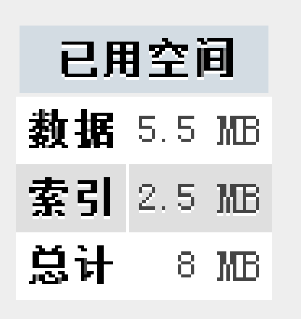


```sql
CREATE INDEX username_idx
ON user (username)
```



索引

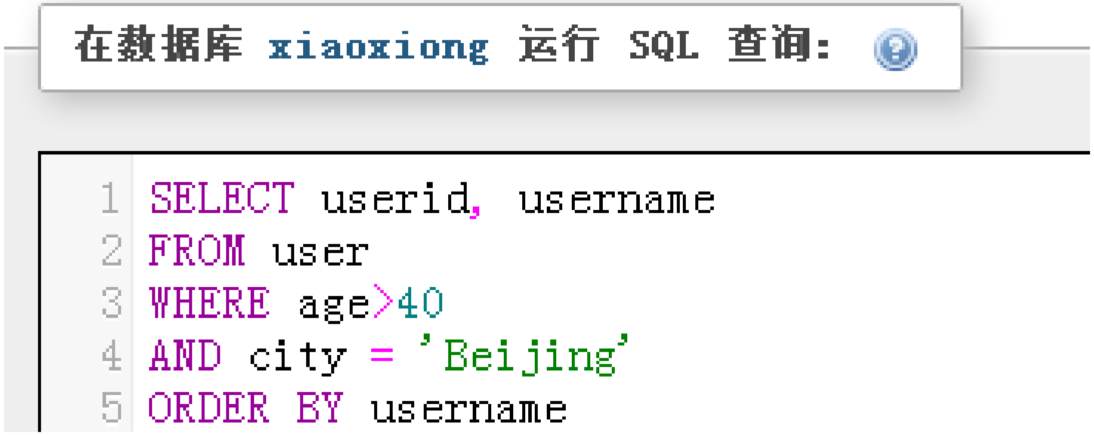
操作	键名	类型	唯一	紧凑	字段	基数	整理	空	注释
编辑 删除	**PRIMARY**	BTREE	是	否	userid	100767	A	否	
编辑 删除	**username_idx**	BTREE	否	否	username	200	A	否	



已用空间	
数据	5.5 MB
索引	2.5 MB
总计	8 MB

Testing on INDEX

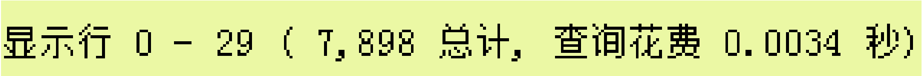
Step 4: Do the same work as Step 2



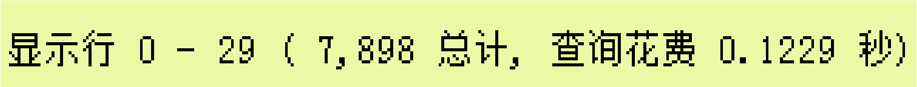
在数据库 xiaoxiong 运行 SQL 查询:

```
SELECT userid, username
FROM user
WHERE age>40
AND city = ’Beijing’
ORDER BY username
```

Result:



显示行 0 - 29 (7,898 总计, 查询花费 0.0034 秒)

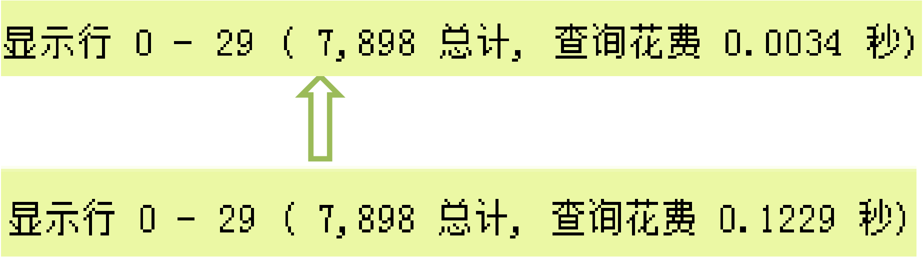


显示行 0 - 29 (7,898 总计, 查询花费 0.1229 秒)

40times!!!

Summary on INDEX

1. **Effect of INDEX**



```
显示行 0 - 29 ( 7,898 总计, 查询花费 0.0034 秒)
```

```
显示行 0 - 29 ( 7,898 总计, 查询花费 0.1229 秒)
```

40times!!!

2. **Consideration of other factors**

   Disk space: about 1-1.5 times cost of primary data

   Time spent in updating index

3. **Where needs optimization**

   Create index in frequently-searched attribute

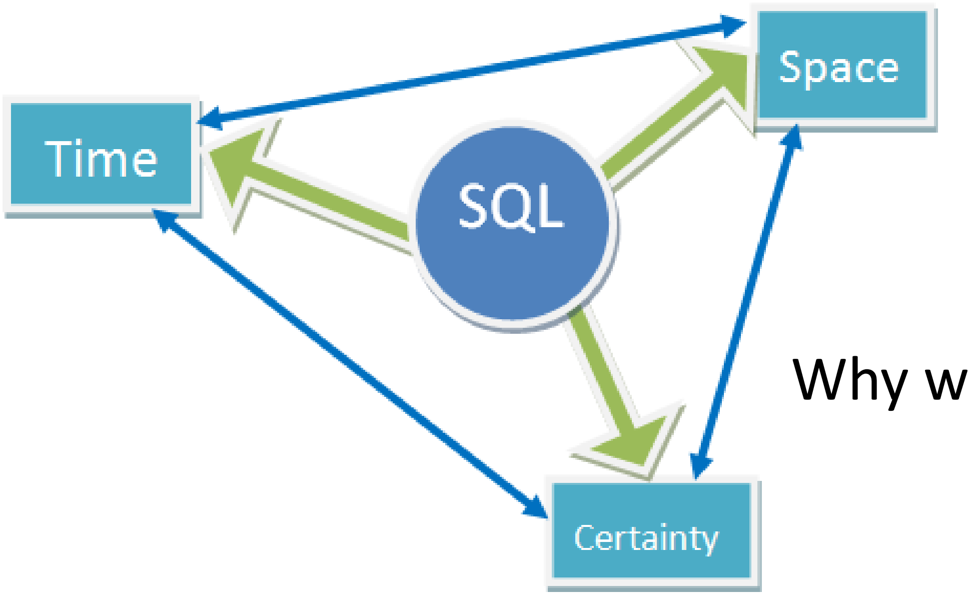
   Create index in attributes which are not often inserted

Optimization in SQL

Proper INDEX
Good structure of conditions

Optimization in time

SQL optimization



Why we need optimization?

Optimization in Data Mining

Our SNS for learning system has a big quantity of users over 100,000

How could we make progress?

Data mining

The process that attempts to discover patterns in large data sets (Wikipedia)

More information of people, more attraction to people

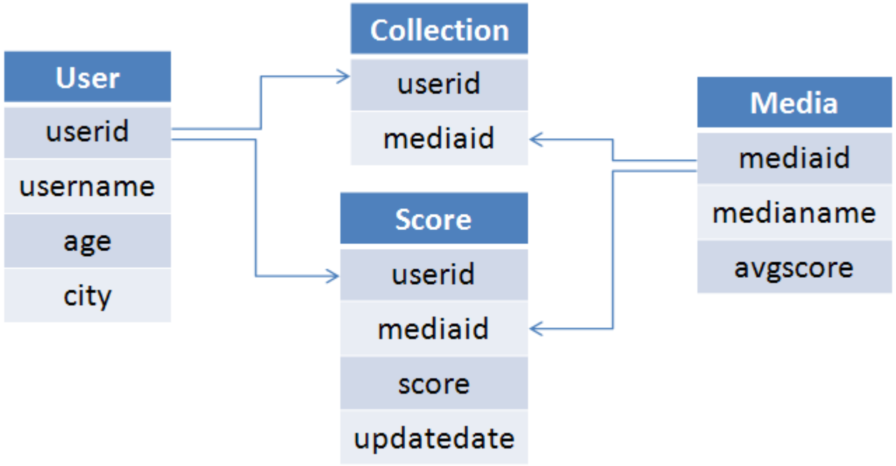
Optimization in Data Mining

SNS function: we wants to search the users who have collected a media object and also scored it in last 180days

There are there tables

user:	userid	username	city	age
collection:	userid	username		
score:	userid	username	score	updatedate

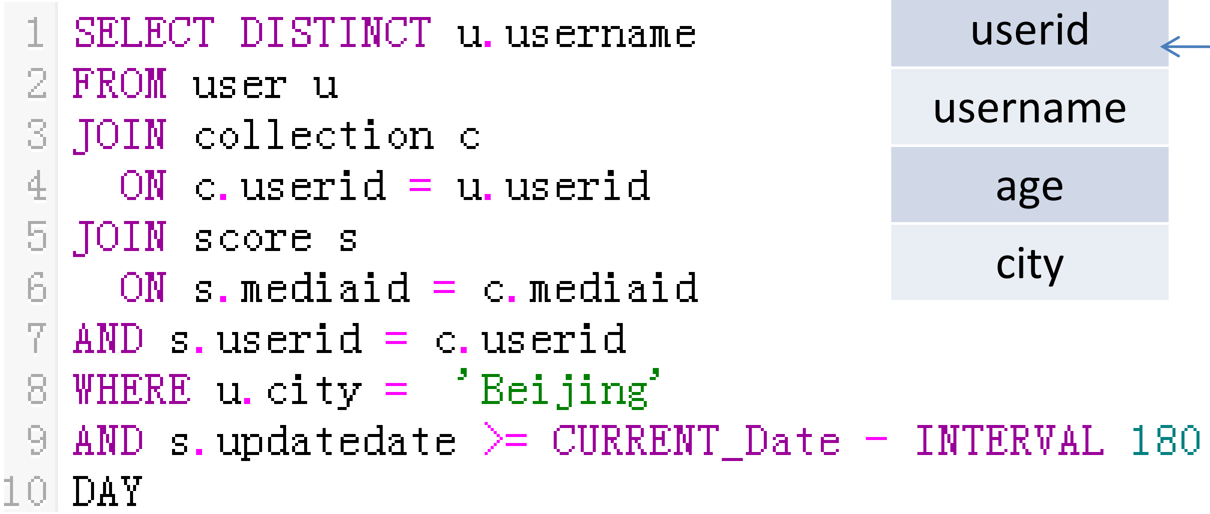
How to select their '**username**' by using SQL statement.



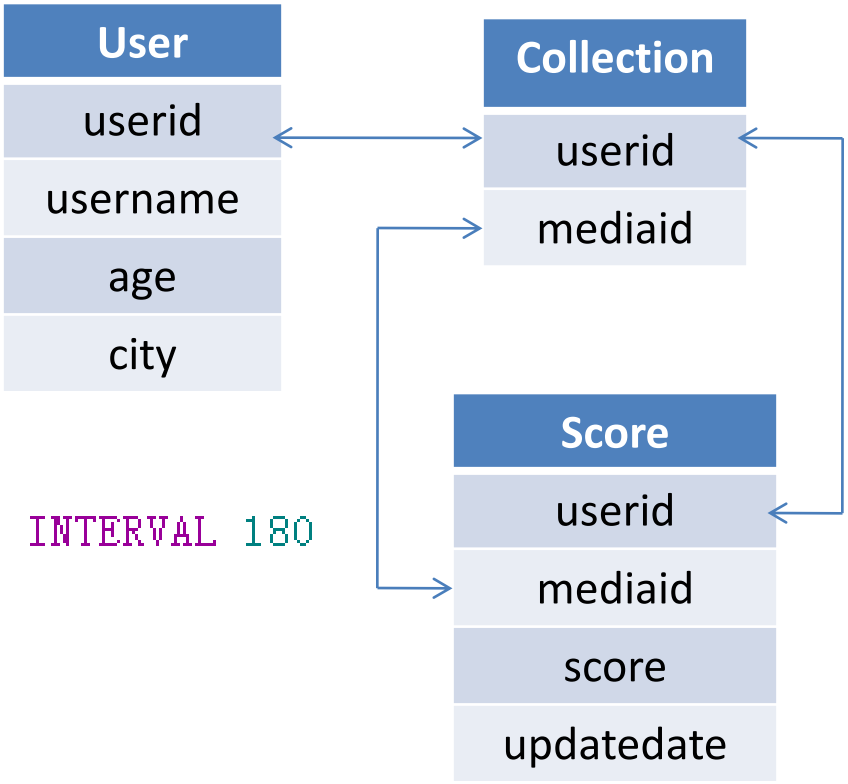
Optimization in Data Mining

SNS function: we wants to search the users who have collected a media material and also scored it in last 180days

Is it a solution?



```
SELECT DISTINCT u.username
FROM user u
JOIN collection c
  ON c.userid = u.userid
JOIN score s
  ON s.mediaid = c.mediaid
AND s.userid = c.userid
WHERE u.city = ’Beijing’
AND s.updatedate >= CURRENT_Date - INTERVAL 180
DAY
```



User
userid
username
age
city

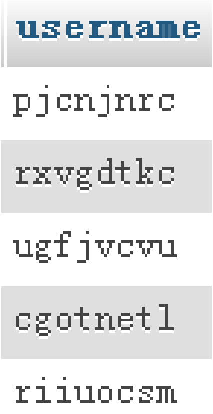
Collection
userid
mediaid

Score
userid
mediaid
score
updatedate

Or without "DISTINCT" ?

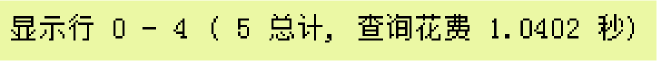
Optimization in Data Mining

Result with DISTINCT



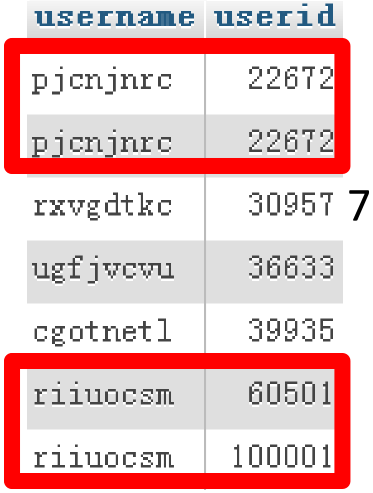
username
pjcnjnrc
rxvgdtkc
ugfjvcvu
cgotnetl
riiuocsm

5 results



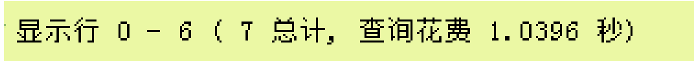
显示行 0 - 4 (5 总计, 查询花费 1.0402 秒)

Result without DISTINCT



username	userid
pjcnjnrc	22672
pjcnjnrc	22672
rxvgdtkc	30957
ugfjvcvu	36633
cgotnetl	39935
riiuocsm	60501
riiuocsm	100001

7 results



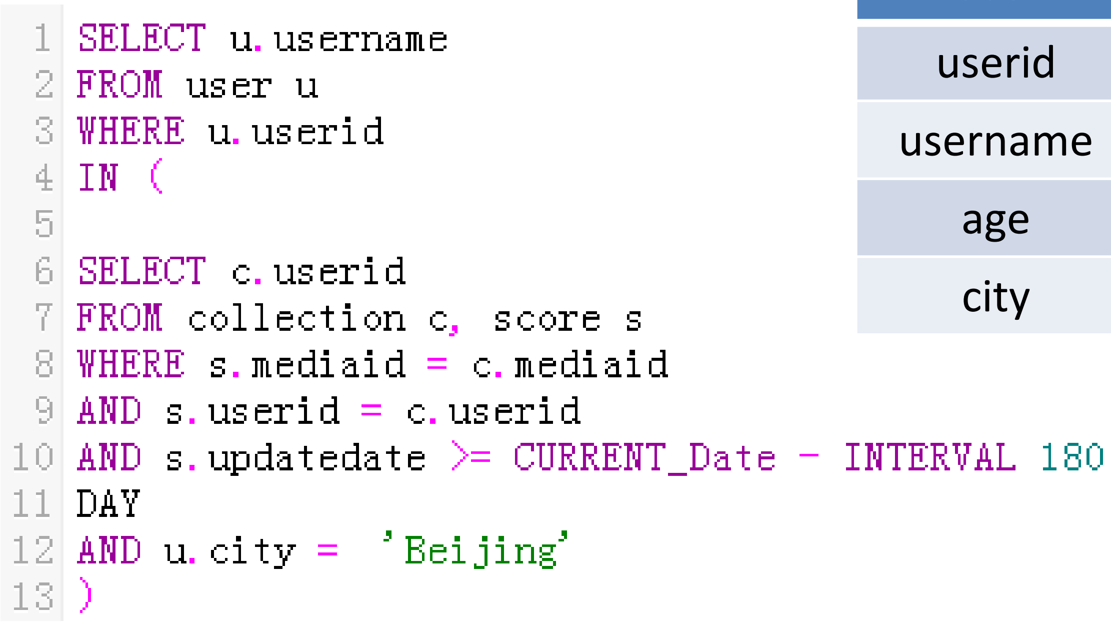
显示行 0 - 6 (7 总计, 查询花费 1.0396 秒)

Both failed!

Optimization in Data Mining

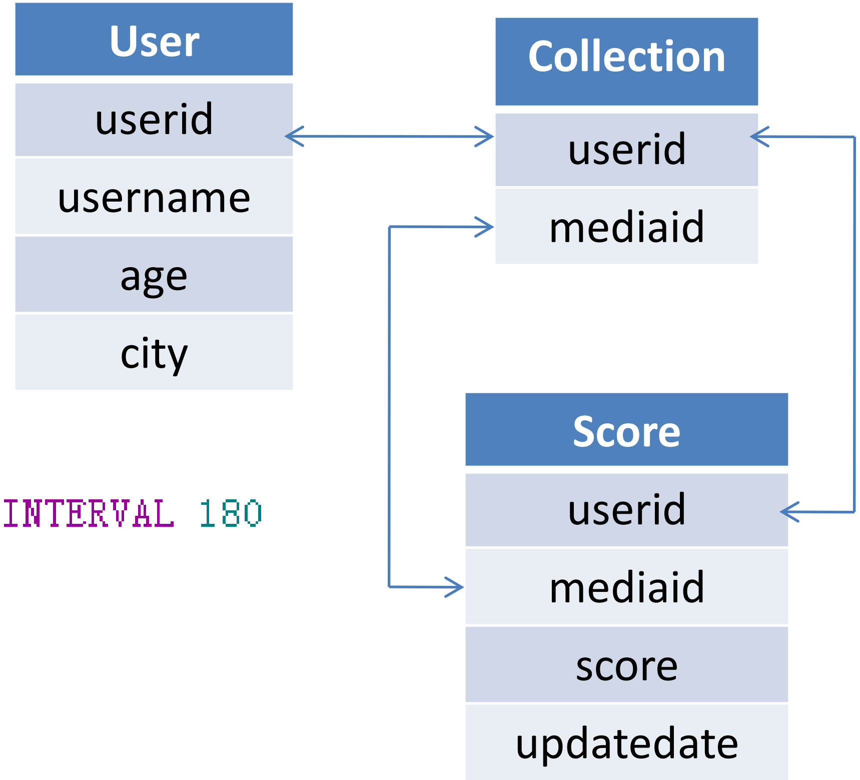
SNS function: we wants to search the users who have collected a media material and also scored it in last 180days

Using IN statement



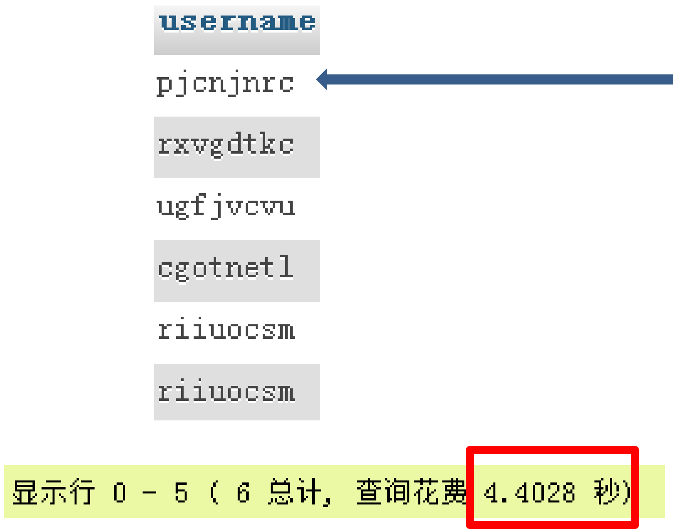
```
SELECT u.username
FROM user u
WHERE u.userid
IN (

SELECT c.userid
FROM collection c, score s
WHERE s.mediaid = c.mediaid
AND s.userid = c.userid
AND s.updatedate >= CURRENT_Date - INTERVAL 180
DAY
AND u.city = ’Beijing’
)
```



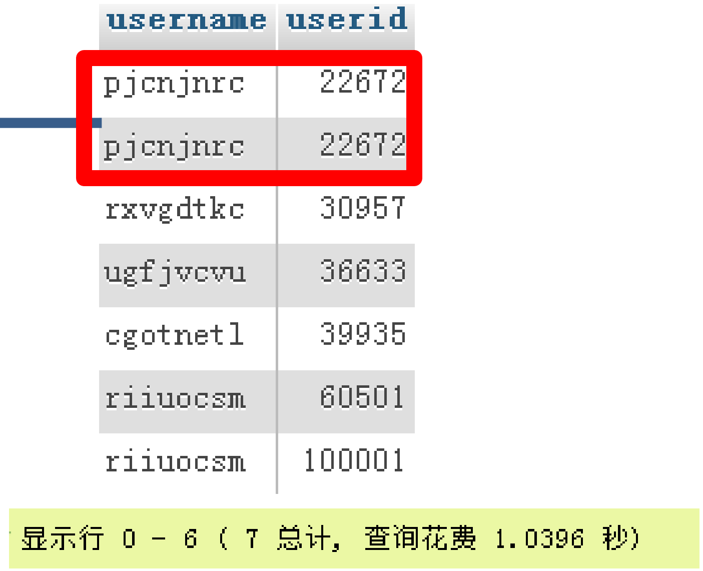
Optimization in Data Mining

Result:



username
pjcnjnrc
rxvgdtkc
ugfjvcvu
cgotnetl
riiuocsm
riiuocsm

显示行 0 - 5 (6 总计，查询花费 4.4028 秒)



username	userid
pjcnjnrc	22672
pjcnjnrc	22672
rxvgdtkc	30957
ugfjvcvu	36633
cgotnetl	39935
riiuocsm	60501
riiuocsm	100001

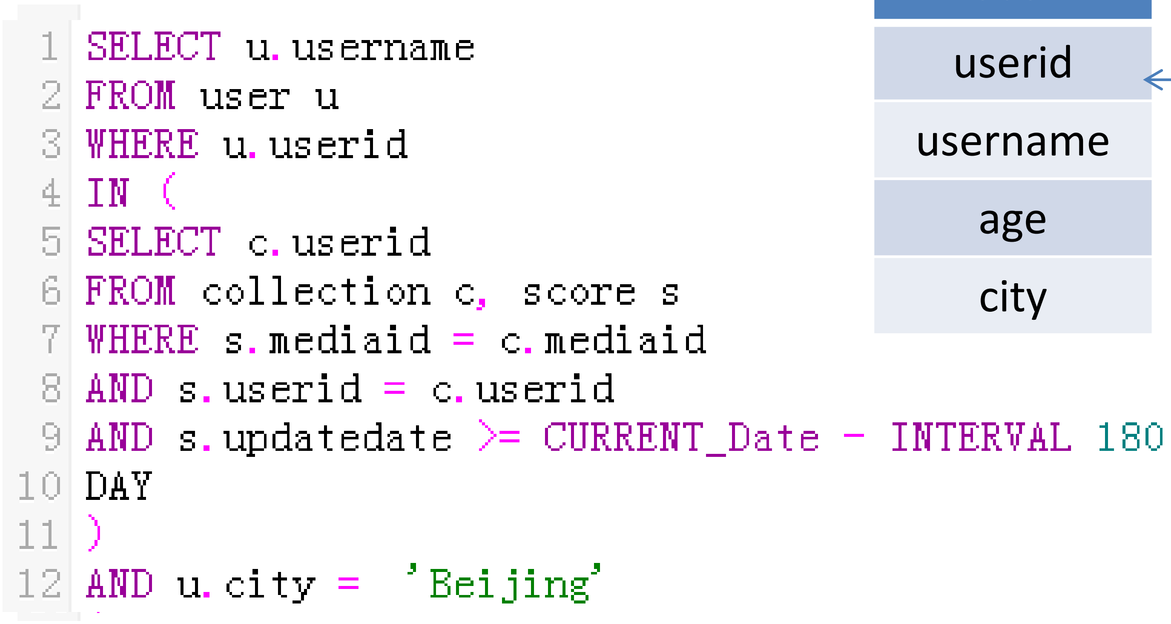
显示行 0 - 6 (7 总计，查询花费 1.0396 秒)

Certainty vs. Time

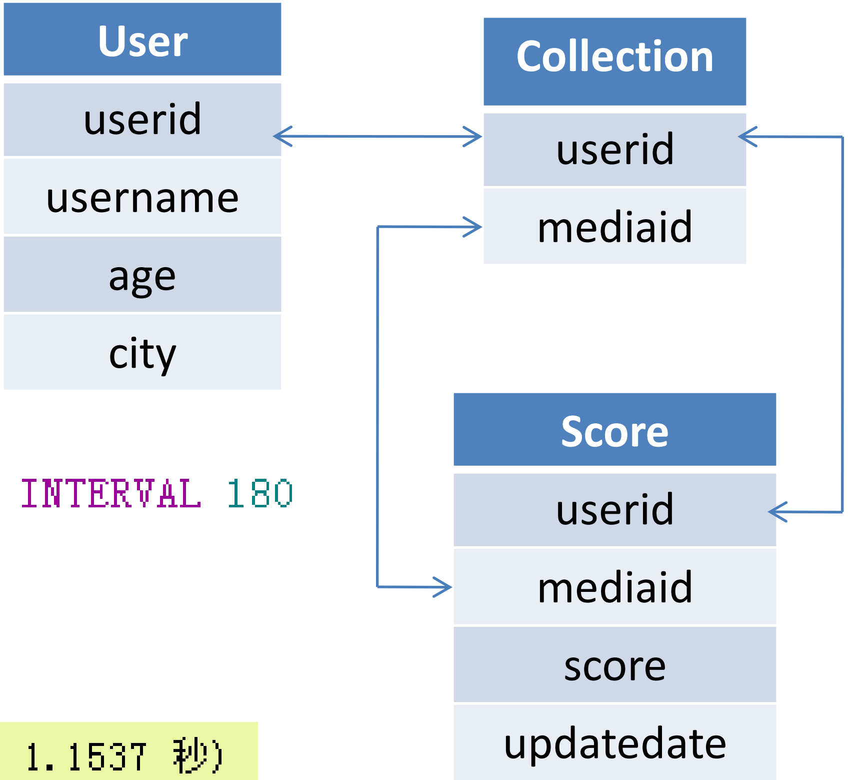
Optimization in Data Mining

SNS function: we wants to search the users who have collected a media material and also scored it in last 180days

Using IN statement



```
SELECT u.username
FROM user u
WHERE u.userid
IN (
SELECT c.userid
FROM collection c, score s
WHERE s.mediaid = c.mediaid
AND s.userid = c.userid
AND s.updatedate >= CURRENT_Date - INTERVAL 180
DAY
)
AND u.city = ’Beijing’
```



User
userid
username
age
city

Collection
userid
mediaid

Score
userid
mediaid
score
updatedate

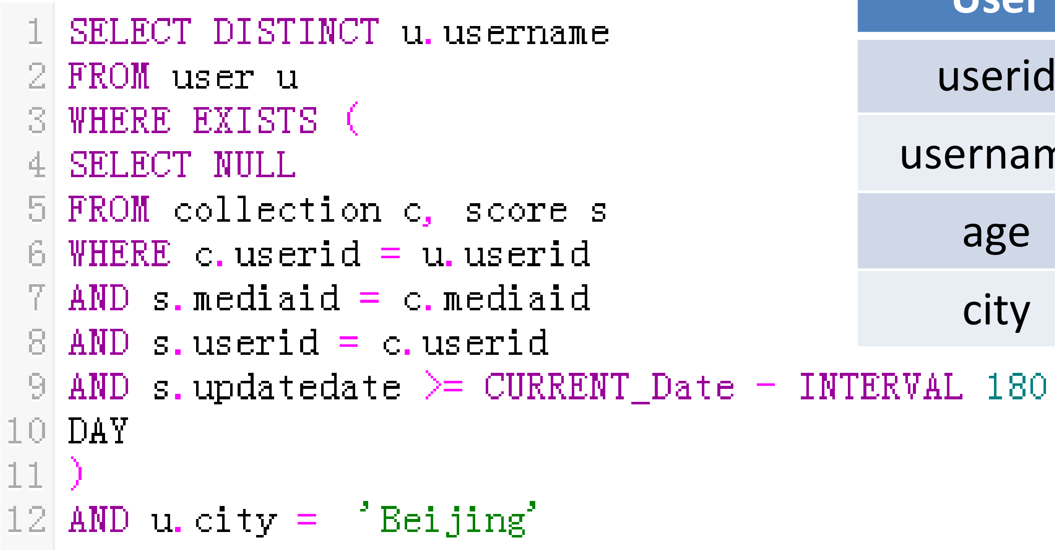
Result: 显示行 0 - 5 (6 总计, 查询花费 1.1537 秒)

显示行 0 - 5 (6 总计, 查询花费 4.4028 秒)

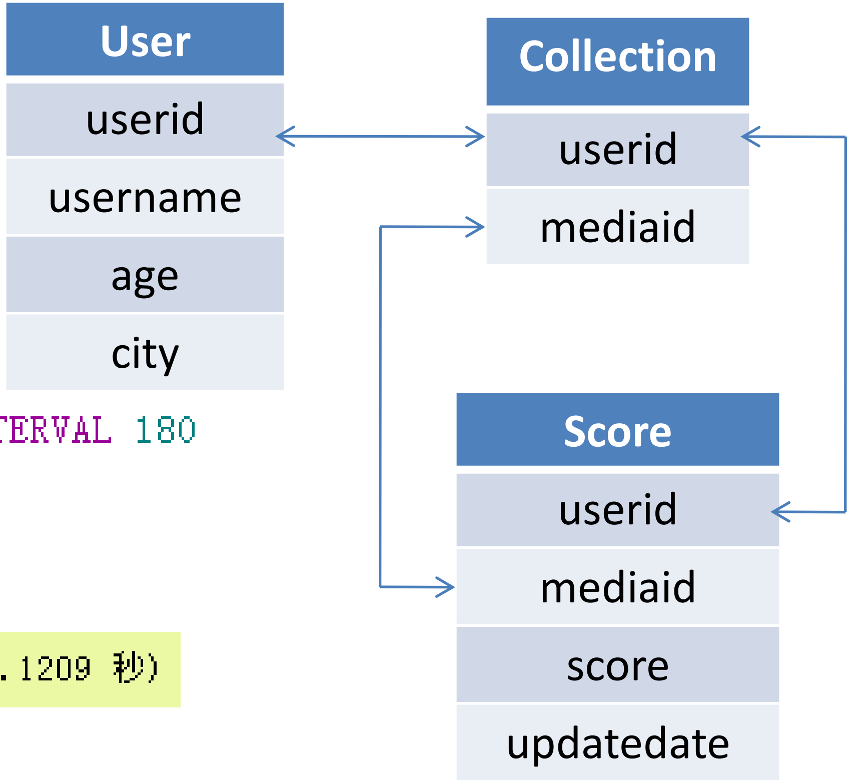
Optimization in Data Mining

SNS function: we wants to search the users who have collected a media material and also scored it in last 180days

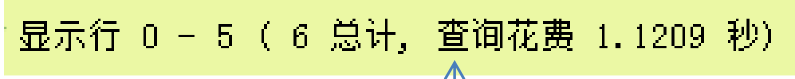
Using EXISTS statement



```
SELECT DISTINCT u.username
FROM user u
WHERE EXISTS (
SELECT NULL
FROM collection c, score s
WHERE c.userid = u.userid
AND s.mediaid = c.mediaid
AND s.userid = c.userid
AND s.updatedate >= CURRENT_Date - INTERVAL 180
DAY
)
AND u.city = ’Beijing’
```

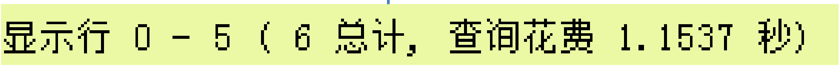


Result:



显示行 0 - 5 (6 总计, 查询花费 1.1209 秒)

Result of IN statement



显示行 0 - 5 (6 总计, 查询花费 1.1537 秒)

Optimization in Data Mining

- **DISTINCT & No DISTINCT**

  Dislodge the repeating data,
  certainty

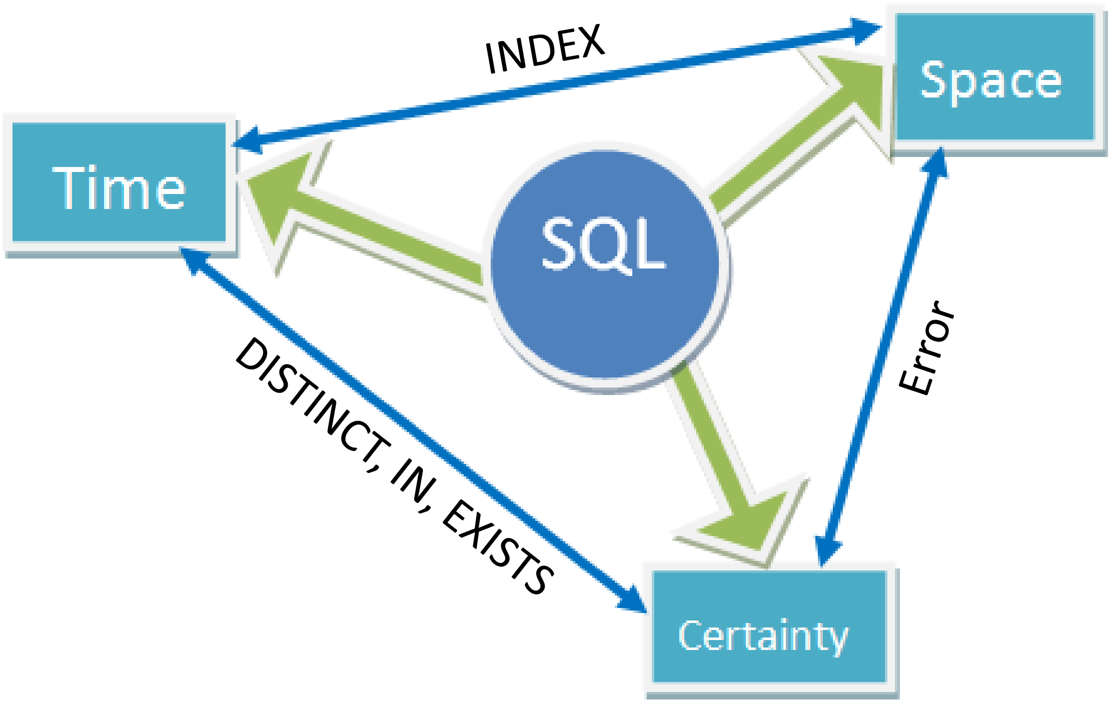
- **EXISTS & IN**

  Help selection in data mining,
  time, certainty

- **The order of conditions**

  Sharply increase speed of data mining,
  time

Relation Between
SPACE, TIME and CERTAINTY



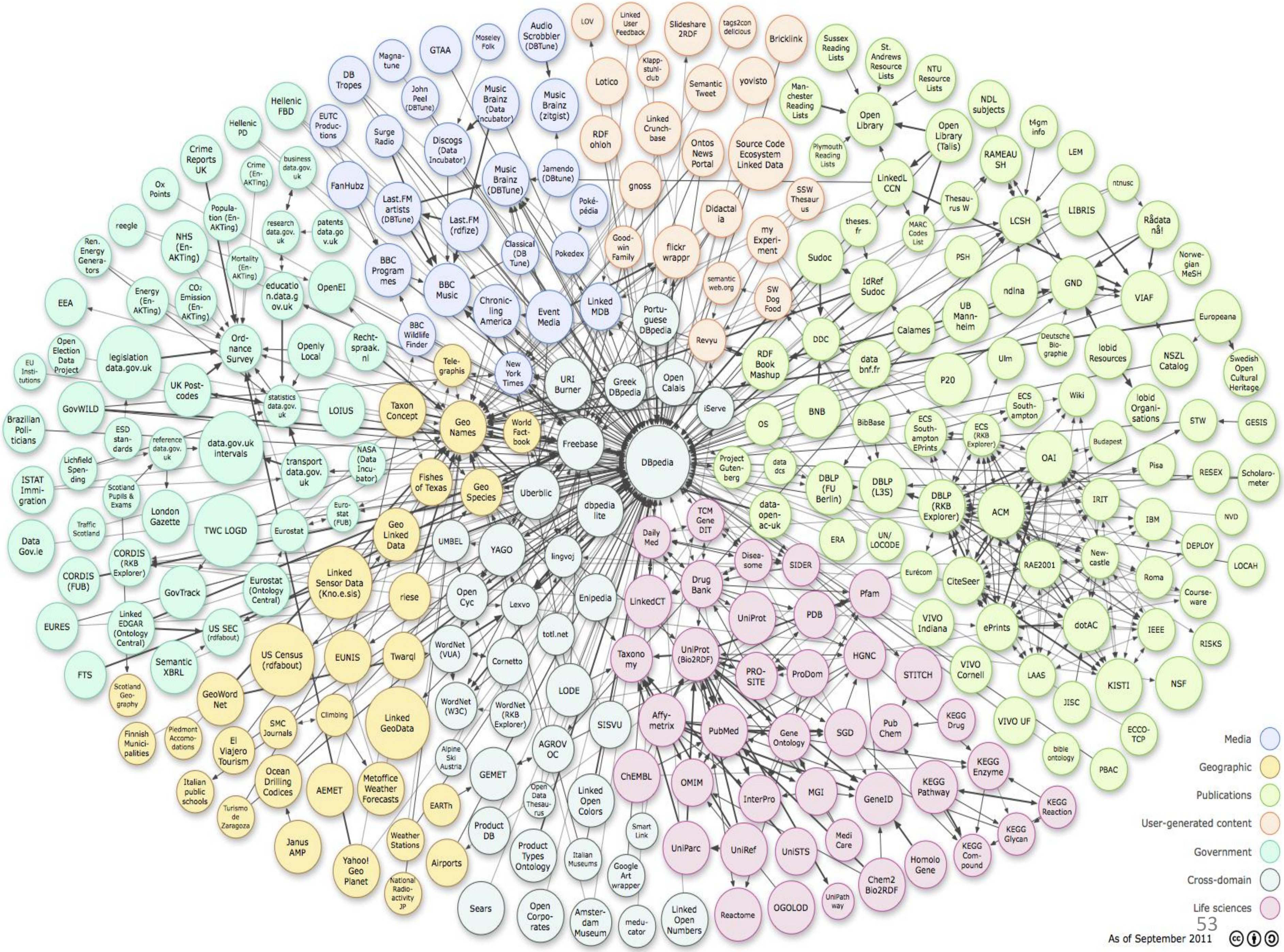
• Grammar	Formulation
• Optimization	More users
• NoSQL	Huge number of users

ORACLE
Microsoft SQL Server
IBM DB2
MySQL
Microsoft Office Access 2007

Can you sure this query language is perfect?

Or the Relational Database is perfect?

Even if the data become huge!!!!!!!!!!!!



As of September 2011

Have you heard about these DB?

DynamoDB

BigTable

NoSQL(Not Only SQL)

- **What's meaning of that???**

  NoSQL definition from *nosql-database.org:*

  Next Generation **Databases** mostly addressing some of the points: being **non-relational, distributed, open-source** and **horizontally scalable**.

- **In a word?**

  It does not use SQL as its query language and has a distributed architecture

Why NoSQL?

- High performance
- High Storage
- High scalability
- High availablity

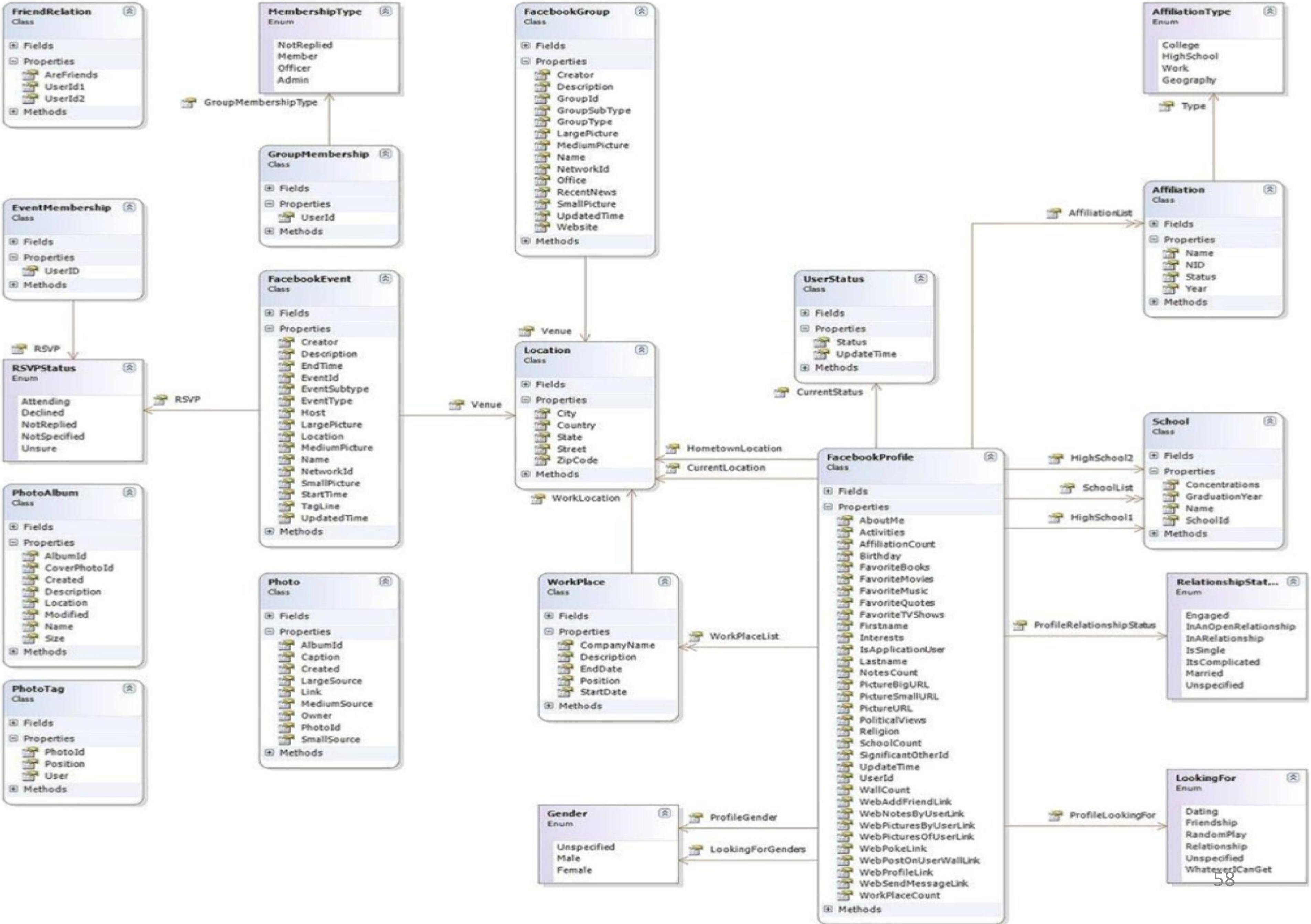
What is difference between SQL and NoSQL?

Database system	Language	Database architerture
SQL	Structured query language(SQL)	Relational
NoSQL	Java, C, C++, Python, Erlang,etc	**Distributed**

Although NoSQL are written in different languages, but they have same architecture:

Distributed Database Management System.

DDBMS(Distributed Database Management System)



FriendRelation
Class
Fields
Properties
AreFriends
UserId1
UserId2
Methods

MembershipType
Enum
NotReplied
Member
Officer
Admin

FacebookGroup
Class
Fields
Properties
Creator
Description
GroupId
GroupSubType
GroupType
LargePicture
MediumPicture
Name
NetworkId
Office
RecentNews
SmallPicture
UpdatedTime
Website
Methods

AffiliationType
Enum
College
HighSchool
Work
Geography

GroupMembership
Class
Fields
Properties
UserId
Methods

EventMembership
Class
Fields
Properties
UserID
Methods

Affiliation
Class
Fields
Properties
Name
NID
Status
Year
Methods

FacebookEvent
Class
Fields
Properties
Creator
Description
EndTime
EventId
EventSubtype
EventType
Host
LargePicture
Location
MediumPicture
Name
NetworkId
SmallPicture
StartTime
TagLine
UpdatedTime
Methods

UserStatus
Class
Fields
Properties
Status
UpdateTime
Methods

Location
Class
Fields
Properties
City
Country
State
Street
ZipCode
Methods

RSVPStatus
Enum
Attending
Declined
NotReplied
NotSpecified
Unsure

School
Class
Fields
Properties
Concentrations
GraduationYear
Name
SchoolId
Methods

FacebookProfile
Class
Fields
Properties
AboutMe
Activities
AffiliationCount
Birthday
FavoriteBooks
FavoriteMovies
FavoriteMusic
FavoriteQuotes
FavoriteTVShows
Firstname
Interests
IsApplicationUser
Lastname
NotesCount
PictureBigURL
PictureSmallURL
PictureURL
PoliticalViews
Religion
SchoolCount
SignificantOtherId
UpdateTime
UserId
WallCount
WebAddFriendLink
WebNotesByUserLink
WebPicturesByUserLink
WebPicturesOfUserLink
WebPokeLink
WebPostOnUserWallLink
WebProfileLink
WebSendMessageLink
WorkPlaceCount
Methods

PhotoAlbum
Class
Fields
Properties
AlbumId
CoverPhotoId
Created
Description
Location
Modified
Name
Size
Methods

Photo
Class
Fields
Properties
AlbumId
Caption
Created
LargeSource
Link
MediumSource
Owner
PhotoId
SmallSource
Methods

WorkPlace
Class
Fields
Properties
CompanyName
Description
EndDate
Position
StartDate
Methods

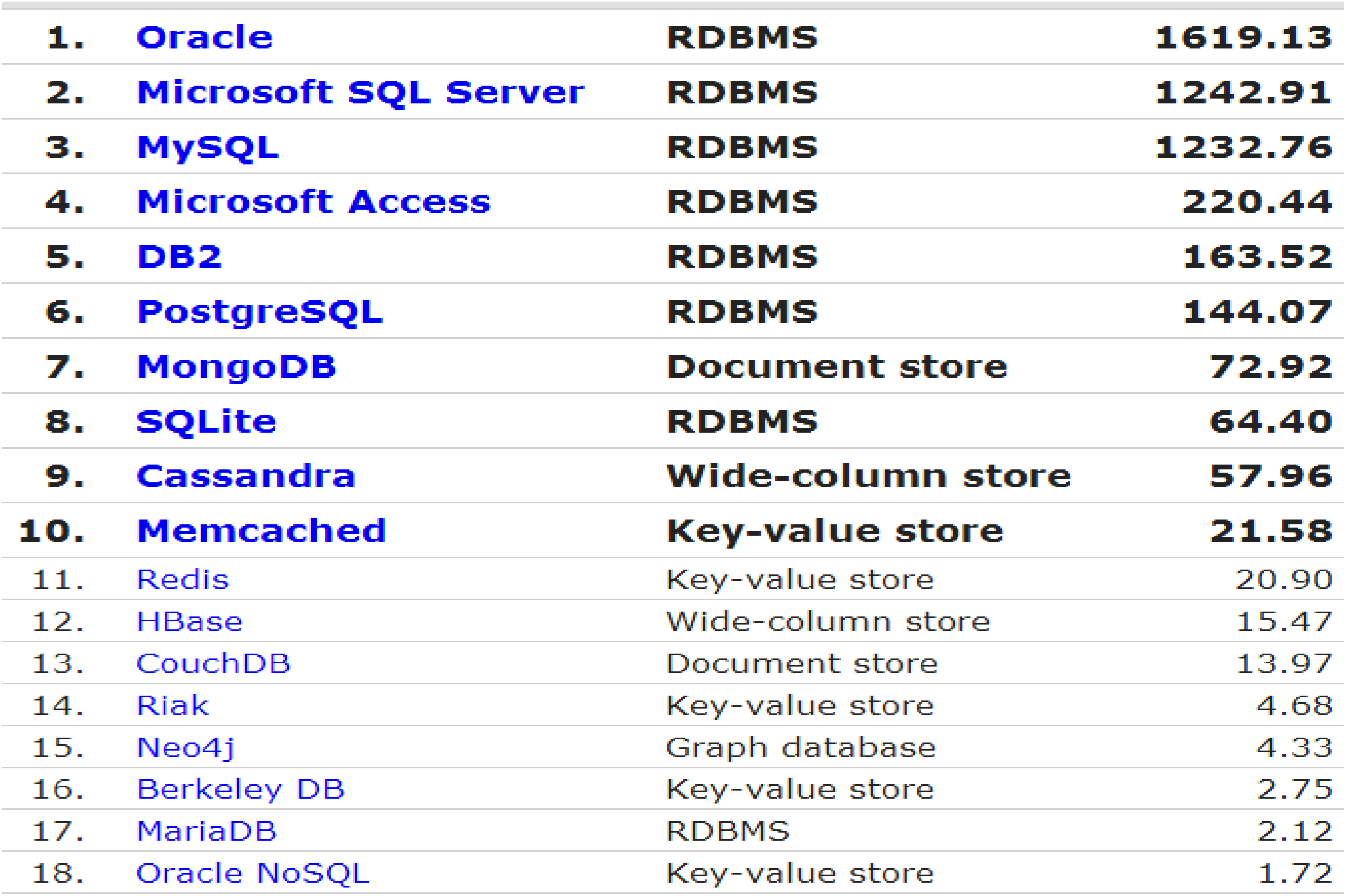
RelationshipStat...
Enum
Engaged
InAnOpenRelationship
InARelationship
IsSingle
ItsComplicated
Married
Unspecified

PhotoTag
Class
Fields
Properties
PhotoId
Position
User
Methods

Gender
Enum
Unspecified
Male
Female

LookingFor
Enum
Dating
Friendship
RandomPlay
Relationship
Unspecified
WhateverICanGet

Popularity of DB (Oct.2012)



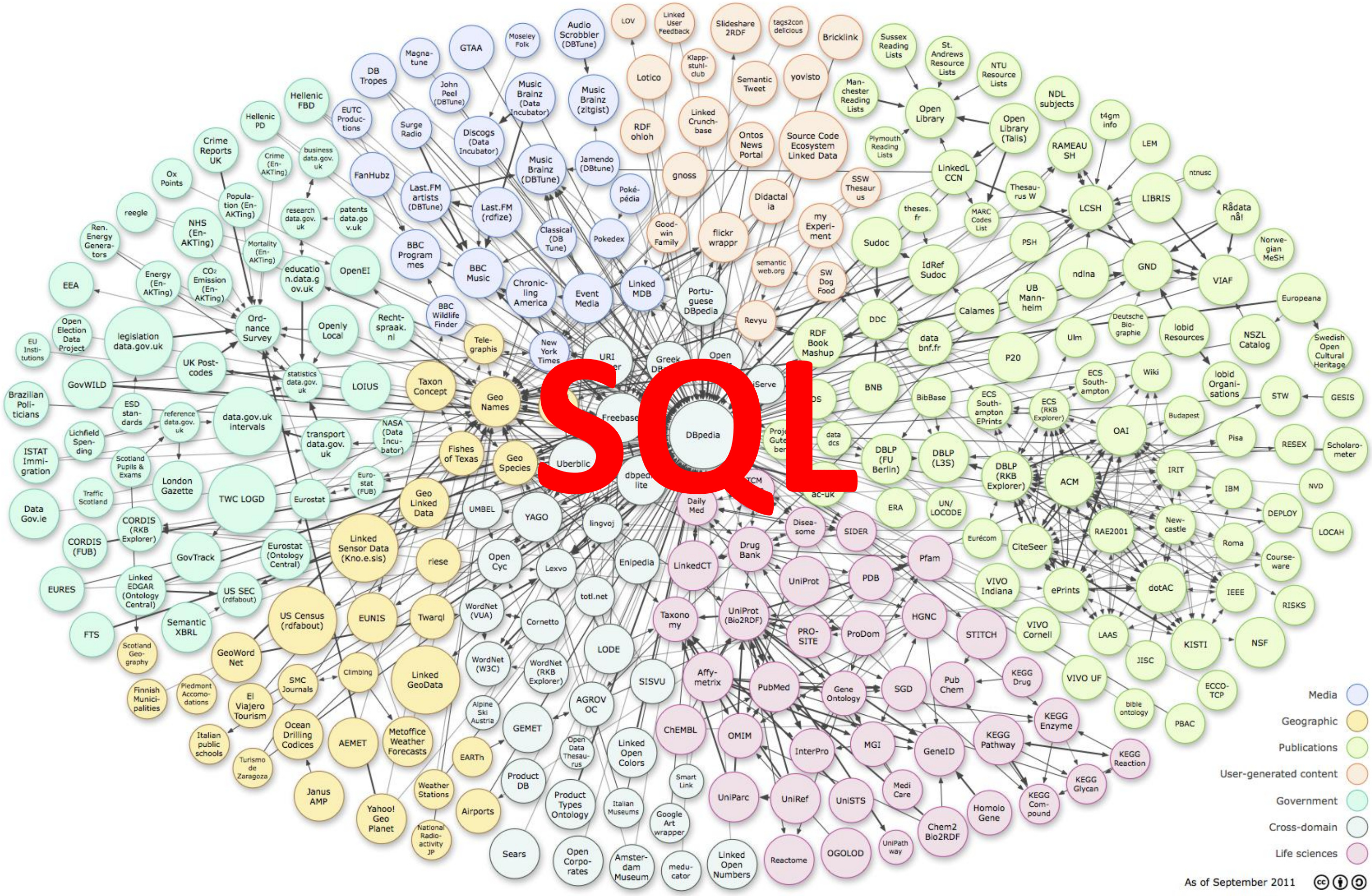
1.	Oracle	RDBMS	1619.13
2.	Microsoft SQL Server	RDBMS	1242.91
3.	MySQL	RDBMS	1232.76
4.	Microsoft Access	RDBMS	220.44
5.	DB2	RDBMS	163.52
6.	PostgreSQL	RDBMS	144.07
7.	MongoDB	Document store	72.92
8.	SQLite	RDBMS	64.40
9.	Cassandra	Wide-column store	57.96
10.	Memcached	Key-value store	21.58
11.	Redis	Key-value store	20.90
12.	HBase	Wide-column store	15.47
13.	CouchDB	Document store	13.97
14.	Riak	Key-value store	4.68
15.	Neo4j	Graph database	4.33
16.	Berkeley DB	Key-value store	2.75
17.	MariaDB	RDBMS	2.12
18.	Oracle NoSQL	Key-value store	1.72

NoSQL is a movement

If we want learn more about these,

Please learn SQL first!

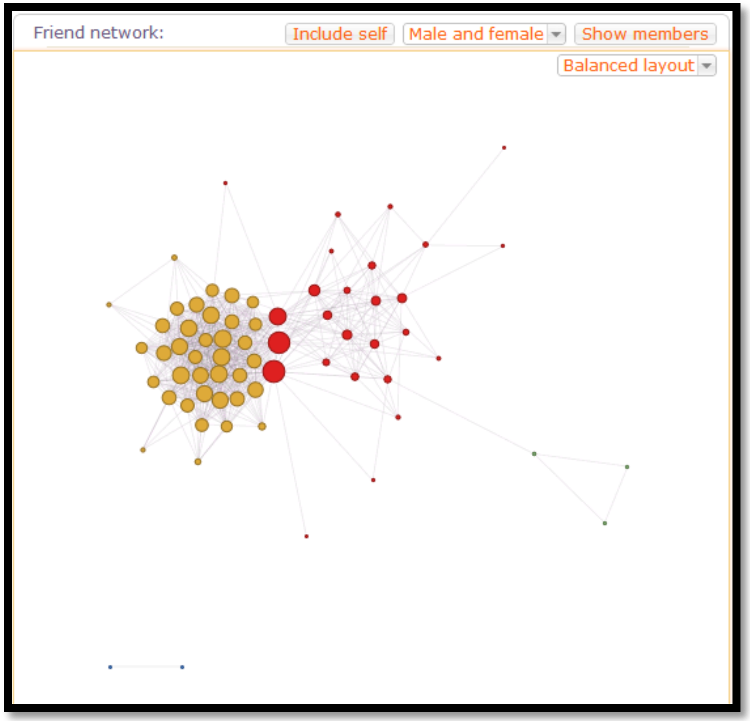
Why SQL?

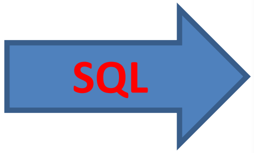

SQL
DBpedia
Freebase
Geo Names
Uberblic
dbpedia lite
YAGO
lingvoj
Open Cyc
Lexvo
totl.net
Enipedia
LinkedCT
Drug Bank
Diseasome
SIDER
Daily Med
UniProt
PDB
Pfam
HGNC
STITCH
UniProt (Bio2RDF)
Taxonomy
PRO-SITE
ProDom
PubMed
Gene Ontology
SGD
Pub Chem
KEGG Drug
KEGG Enzyme
KEGG Pathway
KEGG Reaction
KEGG Glycan
KEGG Compound
Affymetrix
ChEMBL
OMIM
InterPro
MGI
GeneID
UniParc
UniRef
UniSTS
Medi Care
Chem2 Bio2RDF
Homolo Gene
Reactome
OGOLOD
UniPath way
Linked Open Numbers
medu-cator
Amsterdam Museum
Open Corporates
Sears
Google Art wrapper
Smart Link
Italian Museums
Product Types Ontology
Product DB
Open Data Thesaurus
Linked Open Colors
AGROVOC
GEMET
SISVU
LODE
WordNet (RKB Explorer)
WordNet (W3C)
WordNet (VUA)
Cornetto
Alpine Ski Austria
UMBEL
Linked GeoData
Geo Species
Fishes of Texas
Taxon Concept
Geo Linked Data
Linked Sensor Data (Kno.e.sis)
riese
Twarql
EUNIS
US Census (rdfabout)
GeoWord Net
Climbing
Linked GeoData
SMC Journals
El Viajero Tourism
Ocean Drilling Codices
AEMET
Metoffice Weather Forecasts
EARTh
Weather Stations
Airports
Yahoo! Geo Planet
Janus AMP
National Radioactivity JP
Turismo de Zaragoza
Italian public schools
Finnish Municipalities
Piedmont Accomodations
Scotland Geography
Semantic XBRL
FTS
EURES
US SEC (rdfabout)
Linked EDGAR (Ontology Central)
CORDIS (FUB)
CORDIS (RKB Explorer)
GovTrack
Eurostat (Ontology Central)
Data Gov.ie
ISTAT Immigration
Brazilian Politicians
GovWILD
EU Institutions
Open Election Data Project
EEA
Ren. Energy Generators
reegle
Ox Points
Crime Reports UK
Hellenic PD
Hellenic FBD
Crime (EnAKTing)
business data.gov.uk
Population (EnAKTing)
research data.gov.uk
patents data.gov.uk
NHS (EnAKTing)
Mortality (EnAKTing)
Energy (EnAKTing)
CO2 Emission (EnAKTing)
educatio n.data.g ov.uk
OpenEI
Ordnance Survey
Openly Local
legislation data.gov.uk
UK Postcodes
statistics data.gov.uk
LOIUS
ESD standards
reference data.gov.uk
data.gov.uk intervals
transport data.gov.uk
NASA (Data Incubator)
Lichfield Spending
Scotland Pupils & Exams
Traffic Scotland
London Gazette
TWC LOGD
Eurostat
Eurostat (FUB)
Recht-spraak.nl
EUTC Productions
DB Tropes
Magnatune
GTAA
Moseley Folk
Audio Scrobbler (DBTune)
John Peel (DBTune)
Music Brainz (Data Incubator)
Music Brainz (zitgist)
Surge Radio
Discogs (Data Incubator)
Music Brainz (DBTune)
Jamendo (DBtune)
FanHubz
Last.FM artists (DBTune)
Last.FM (rdfize)
Classical (DB Tune)
Pokedex
Poképédia
BBC Program mes
BBC Music
BBC Wildlife Finder
Chronicling America
Event Media
Linked MDB
Telegraphis
New York Times
URI Burner
Greek DBpedia
Open Library
LOV
Linked User Feedback
Slideshare 2RDF
tags2con delicious
Bricklink
Lotico
Klappstuhlclub
Semantic Tweet
yovisto
RDF ohloh
Linked Crunchbase
Ontos News Portal
Source Code Ecosystem Linked Data
gnoss
Didactalia
Good-win Family
flickr wrappr
semantic web.org
my Experiment
SW Dog Food
Portuguese DBpedia
Revyu
SSW Thesaurus
Sussex Reading Lists
St. Andrews Resource Lists
NTU Resource Lists
Manchester Reading Lists
Plymouth Reading Lists
Open Library
Open Library (Talis)
NDL subjects
t4gm info
LEM
RAMEAU SH
LinkedL CCN
Thesaurus W
LCSH
LIBRIS
ntnusc
Rådata nå!
theses.fr
MARC Codes List
Sudoc
IdRef Sudoc
PSH
ndlna
GND
VIAF
Norwegian MeSH
UB Mannheim
Calames
DDC
RDF Book Mashup
data bnf.fr
Deutsche Biographie
lobid Resources
NSZL Catalog
Europeana
Swedish Open Cultural Heritage
Ulm
P20
BNB
BibBase
ECS Southampton EPrints
ECS Southampton
ECS (RKB Explorer)
Wiki
lobid Organisations
STW
GESIS
Budapest
OAI
Pisa
RESEX
Scholarometer
DBLP (FU Berlin)
DBLP (L3S)
DBLP (RKB Explorer)
ACM
IRIT
IBM
NVD
ERA
UN/LOCODE
Eurécom
CiteSeer
RAE2001
Newcastle
DEPLOY
LOCAH
Roma
Courseware
VIVO Indiana
ePrints
dotAC
IEEE
RISKS
VIVO Cornell
LAAS
KISTI
NSF
JISC
VIVO UF
ECCO-TCP
bible ontology
PBAC
Media
Geographic
Publications
User-generated content
Government
Cross-domain
Life sciences
As of September 2011

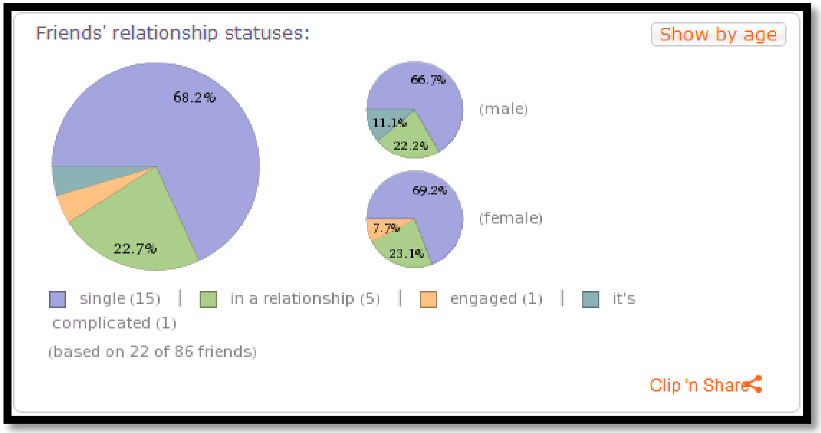
Why SQL?

- RDBMS, Declarable
- Logically simple, Standardization
- Extension, NoSQL

SQL for SNS



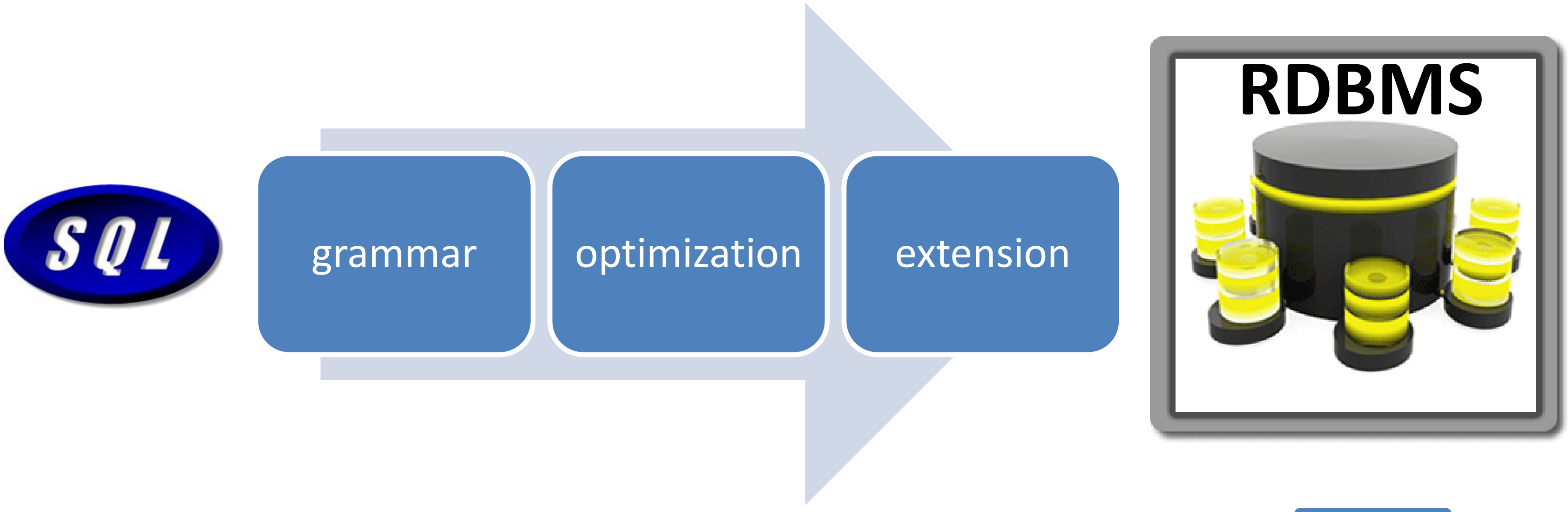

SQL


Friends' relationship statuses:

▨ single (15) | 🟩 in a relationship (5) | 🟧 engaged (1) | ▨ it's complicated (1)

(based on 22 of 86 friends)

Summary



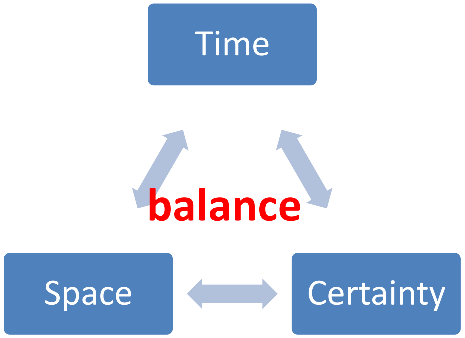
SQL

grammar

optimization

extension

RDBMS



Time

balance

Space

Certainty

Refenrence

- db-engines.com『*New DB-Engines Ranking shows the popularity of database management systems*』
- http://blogs.x2line.com/al/archive/2007/06/02/3124.aspx『*Facebook Object-Oriented Diagram*』
- Wikipedia file:『*LOD Cloud Diagram as of September 2011*』
- 《看图例解学SQL语言》,学苑出版社
- SQL history: *baidu.com*
- RDBMS, DDBMS, NoSQL: *Wikipedia.org*

Thank you