

《交通大数据技术》



2024/6/14

交通大数据技术

马晓磊
交通科学与工程学院
2024年

结构化查询语言 (SQL) I



为什么SQL发音为SEQUEL?

- Edgar F.Codd博士（1970年）关于关系型数据库理论的论文。
- IBM系统/R研究小组。
- 原型 DB2 和System/R的多表和多用户访问的支持语言称为结构化英语查询语言（Structured English Query language，简称SEQUEL）。
- 最终，这种语言被称为SQL，是关系型数据库的行业标准。

SQL是一种非常高级的语言

- SQL采用一种类似表的结构，这有助于程序员避免指定传统编程语言中需要的大量数据操作细节
- SQL查询得到了很好的“优化”，从而产生了高效的查询执行

SQL方言

- SQL
- SQL-92 (SQL2)
- SQL-99 (SQL3)
- T-SQL (Transact-SQL) –Microsoft SQL Server

不区分大小写

- SQL将大小写字母视为同一个字母

简单SQL查询

查询的主要形式是：

```
SELECT attributes  
FROM tables  
WHERE conditions
```

SELECT做什么？

- **投影**数据。这可以是现有字段的列表或某些表达式。

FROM做什么？

- 关系/表格。这可以是现有关系、临时表和子查询。

WHERE做什么？

- **选择**数据。限制查询返回的行的条件表达式。

简单SQL查询

Movies

title	year	length	budget	rating	votes	mpaa	genres
'G' Men	1935	85	450000	7.2	281	NULL	Drama
'Manos' the Hands of Fate	1966	74	19000	1.6	7996	NULL	
'Til There Was You	1997	113	23000000	4.8	799	PG-13	Comedy, Romance
.com for Murder	2002	96	5000000	3.7	271	NULL	
10 Things I Hate About You	1999	97	16000000	6.7	19095	PG-13	Comedy, Romance
100 Mile Rule	2002	98	1100000	5.6	181	R	Comedy
...

```
SELECT title, year, length, rating  
FROM movies  
WHERE title = 'titanic'
```



title	year	length	rating
Titanic	1997	194	6.9

简单SQL查询

以下所有查询都会得到相同的结果：

```
SELECT title, year, length, rating  
FROM movies  
WHERE title = 'titanic'
```

```
select TITLE, YEAR, LENGTH, RATING  
from MOVIES  
where TITLE = 'TITANIC'
```

```
SELECT title, year, length, rating FROM movies WHERE title = 'titanic'
```

简单SQL查询

title	year	length	budget	rating	votes	mpaa	genres
'G' Men	1935	85	450000	7.2	281	NULL	Drama
'Manos' the Hands of Fate	1966	74	19000	1.6	7996	NULL	
'Til There Was You	1997	113	23000000	4.8	799	PG-13	Comedy, Romance
.com for Murder	2002	96	5000000	3.7	271	NULL	
10 Things I Hate About You	1999	97	16000000	6.7	19095	PG-13	Comedy, Romance
100 Mile Rule	2002	98	1100000	5.6	181	R	Comedy
...

```
SELECT *  
FROM movies  
WHERE title = 'titanic'
```



title	year	length	budget	rating	votes	mpaa	genres
Titanic	1997	194	200000000	6.9	90195	PG-13	Drama, Romance

简单SQL查询

输入模式：

Movies(title, year, length, budget, rating, votes, mpaa, genre)

```
SELECT title, year, length, rating  
FROM movies  
WHERE title = 'titanic'
```

输出模式：

Result(title, year, length, rating)

SQL中的投影

在结果中重命名列

```
SELECT title AS Name, year, length AS Duration, rating  
FROM movies  
WHERE title = 'titanic'
```



Name	year	Duration	rating
Titanic	1997	194	6.9

WHERE子句中有什么？

常用布尔运算符

Operator	Description	Operator	Description
=	等于	<=	小于或等于
<> or !=	不等于	BETWEEN a AND b	位于包含范围内
>	大于	LIKE	匹配一种字符串模式
<	小于	IN (a, b, c)	等于多个可能值中的一个
>=	大于或等于	IS NULL or IS NOT NULL	与null做比较 (缺失数据)

!= 不是标准的SQL运算符，但在大多数DBMS中受支持

- 对于数字，它们具有通常的含义
- 对于字符和文本：字母顺序
- 对于日期和时间：time1 < time2意味着time1早于time2。

LIKE运算符

- 根据模式匹配比较两个字符串。
- 表达式：s LIKE p, 其中s是字符串，p是一种模式。
- p可以包含两个特殊符号：
 - `_` = 任意单个字符
 - `%` = 任意字符序列

SQL中的选择

```
SELECT title, year, length, rating
FROM movies
WHERE title LIKE 'star ____'
```



title	year	length	rating
Star Dust	1940	86	6.6
Star Maps	1997	86	6.1
Star Trak	1998	9	4.3
Star Wars	1977	125	8.8

```
SELECT title, year, length, rating
FROM movies
WHERE title LIKE '%star%'
```



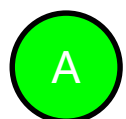
title	year	length	rating
Last Starfighter, The	1984	101	6.2
Star Kid	1997	101	5
Star Trek III: The Search for Spock	1984	105	6.2
...

从下表中选出电影名称包含The的行的所有列的语句应当是

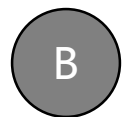
SELECT *

FROM movies

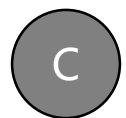
WHERE title LIKE ' [填空1] '



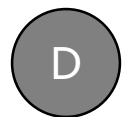
A %The_



B %The%



C The%



D _The_

title	year	length	budget	rating	votes
'G' Men	1935	85	450000	7.2	281
'Manos' the Hands of Fate	1966	74	19000	1.6	7996
'Til There Was You	1997	113	23000000	4.8	799
.com for Murder	2002	96	5000000	3.7	271
10 Things I Hate About You	1999	97	16000000	6.7	19095
100 Mile Rule	2002	98	1100000	5.6	181
...

提交

SQL中的选择

逻辑运算符

- **AND**: 如果两边都为TRUE, 则返回TRUE
- **OR**: 如果任何一方为TRUE, 则返回TRUE
- **NOT**: 如果以下谓词为FALSE, 则返回TRUE, 反之亦然

括号可以更改求值顺序

- TRUE **OR** TRUE **AND** FALSE
- (TRUE **OR** TRUE) **AND** FALSE
- FALSE **AND** FALSE **OR** TRUE
- FALSE **AND** (FALSE **OR** TRUE)

运算符优先级

Operators	↓ 低优先级
*(Multiply), / (Division), % (Modulo)	
=, >, <, >=, <=, <>, != (Comparison operators)	
NOT	
AND	
BETWEEN, IN, LIKE, OR	
= (Assignment)	

对结果进行排序

选择2000年以后发行并且评分高于8.5的电影，或者其他评分高于9的电影。

```
SELECT title, year, length, rating
FROM movies
WHERE rating > 9 OR year > 2000 AND rating > 8.5
```



title	year	length	rating
Looking Out	2002	15	9.4
Looking Up	1977	94	9.1
Lord of the Rings: The Fellowship of the Ring	2001	208	8.8
Lord of the Rings: The Return of the King	2003	251	9
Lord of the Rings: The Two Towers	2002	223	8.8
...

对结果进行排序

使用ORDER BY对结果进行排序

- 示例：查找预算高于10,000,000美元的所有电影，并按其评分对电影进行排序。

```
SELECT title, year, length, budget, rating
FROM movies
WHERE budget > 10000000
ORDER BY rating
```



title	year	length	budget	rating
From Justin to Kelly	2003	90	12000000	1.7
Son of the Mask	2005	94	74000000	1.9
Alone in the Dark	2005	96	20000000	2.1
Glitter	2001	104	22000000	2.1
...

这是我们想要的吗？

对结果进行排序

- 排序是升序，除非你指定了DESC关键字
- 你可以按不在 SELECT 列表中的字段排序

```
SELECT title, year, length
FROM movies
WHERE budget > 10000000
ORDER BY rating DESC
```



title	year	length
Shawshank Redemption	1994	142
Lord of the Rings: The Return of the King	2003	251
Godfather: Part II	1974	200
Schindler's List	1993	195
Lord of the Rings: The Two Towers	2002	223
Lord of the Rings: The Fellowship of the Ring	2001	208
Star Wars	1977	125
...

对结果进行排序

- 按多个字段排序

```
SELECT title, year, length, rating
FROM movies
WHERE budget > 10000000
ORDER BY year DESC, rating DESC
```



title	year	length	rating
Sin City	2005	124	8.3
Eternal Sunshine of the Spotless Mind	2004	108	8.6
Taegukgi hwinalrimyeo	2004	140	8.5
Incredibles, The	2004	121	8.3
Kill Bill: Vol. 2	2004	136	8.3
Million Dollar Baby	2004	132	8.3
Lord of the Rings: The Return of the King	2003	251	9
Kill Bill: Vol. 1	2003	111	8.3
Finding Nemo	2003	100	8.3
...

消除重复

使用 **DISTINCT** 关键字删除查询结果中的重复项。

- 示例：数据库中有多少种不同的电影类型？

```
SELECT DISTINCT genres  
FROM movies
```



比较

```
SELECT genres  
FROM movies
```

genres
Animation, Documentary
Action, Animation
Action, Comedy, Documentary
Drama, Documentary, Romance
Action, Animation, Comedy, Romance
Action, Comedy, Drama, Romance
Drama, Short
Action, Comedy, Drama, Short
Action, Animation, Comedy
...

genres
Drama
Comedy
Animation, Comedy, Short
Comedy, Drama
Animation, Comedy, Short
Drama
Drama
Drama
...

SQL中的联接

Product

PName	Price	Category	Manufacturer
Gizmo	19.99	Gadgets	GizmoWorks
Powergizmo	29.99	Gadgets	GizmoWorks
SingleTouch	149.99	Photography	Canon
MultiTouch	203.99	Household	Hitachi

Company

他们之间有什么联系？

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

SQL中的联接

Product (PName, Price, Category, Manufacturer)

Company (CName, StockPrice, Country)

例如，要查找在日本制造的所有产品的名称和价格，我们需要运行查询：

```
SELECT pname, price  
FROM product, company  
WHERE manufacturer = cname AND country = 'Japan'
```

Join between Product
and Company

SQL中的联接

Product

PName	Price	Category	Manufacturer
Gizmo	19.99	Gadgets	GizmoWorks
Powergizmo	29.99	Gadgets	GizmoWorks
SingleTouch	149.99	Photography	Canon
MultiTouch	203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

```
SELECT pname, price
FROM product, company
WHERE manufacturer = cname
AND country = 'Japan'
```

pname	price
SingleTouch	149.99
MultiTouch	203.99

SQL中的联接

Product (PName, Price, Category, Manufacturer)

Company (CName, StockPrice, Country)

查找所有生产 “Gadgets” 类别产品的国家

```
SELECT country  
FROM product, company  
WHERE manufacturer = cname AND category = 'Gadgets'
```


SQL中的联接

Product


PName	Price	Category	Manufacturer
Gizmo	19.99	Gadgets	GizmoWorks
Powergizmo	29.99	Gadgets	GizmoWorks
SingleTouch	149.99	Photography	Canon
MultiTouch	203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

```
SELECT country
FROM product, company
WHERE manufacturer = cname
AND category = 'Gadgets'
```

有什么问题吗？解决办法是什么？



country
USA
USA

SQL中的联接


Product

PName	Price	Category	Manufacturer
Gizmo	19.99	Gadgets	GizmoWorks
Powergizmo	29.99	Gadgets	GizmoWorks
SingleTouch	149.99	Photography	Canon
MultiTouch	203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

```
SELECT DISTINCT country
FROM product, company
WHERE manufacturer = cname
AND category = 'Gadgets'
```



country
USA

SQL中的联接

Product (PName, Price, Category, Manufacturer)

Purchase (Invoice, Buyer, Seller, Store, Product)

Person(PerName, PhoneNumber, City)

查找居住在西雅图购买 “Gadgets” 类产品的人的姓名，以及购买这些产品的商店的名称。

```
SELECT DISTINCT pername, store
FROM person, purchase, product
WHERE pername=buyer AND product = pname
AND city='Seattle' AND category='Gadgets'
```

SQL中的联接

假设我们有两个大表：

Accident(ReportNum, Route, Milepost, **Date**, Severity)

Loopdata(LoopID, **Date**, Time, Speed, Volume)

如果我们只使用公共的 “Date” 属性联接这些表，会发生什么？

- 每天发生多起事故，每天都有许多个探测器和时间。
- 结果：每次事故都将与相应日期收集的所有线圈数据匹配。
- 巨大的无用响应，可能内存过载。
- 需要事故时间和线圈位置来完全定义连接。

SQL中的联接

内部联接

- 交叉联接Cross join
- Θ -联接Theta join
- 自然联接 Natural join

```
SELECT *  
FROM product JOIN company  
ON manufacturer = cname
```

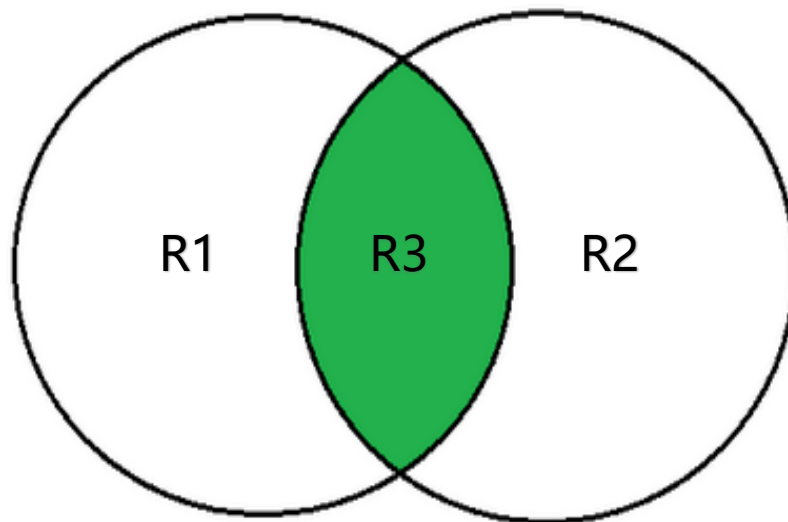
- 内部联接可以在FROM或 WHERE 子句中指定。
- 外部联接只能在 FROM子句中指定

外部联接

- 全外联接Full outer join
- 右外联接Right outer join
- 左外联接Left outer join

内部联接

“内联接Inner join” 意味着我们只返回两个表的“ON”子句都为真的行。也就是说，只返回两个表中都匹配的行。



交叉联接

交叉联接也被称为笛卡尔积(Cartesian product):

在关系代数中 $R3 := R1 \text{ CROSS JOIN } R2$

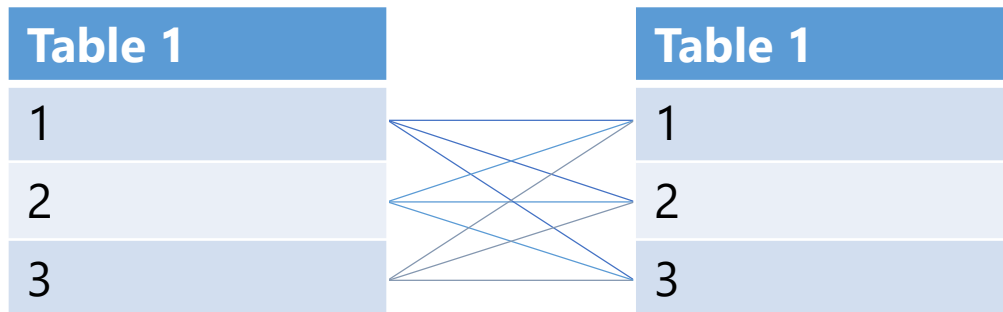
- 无连接字段。
- 将R1的每个元组 $t_{1,i}$ 与R2的每个元组 $t_{2,j}$ 配对。
- 串联 $t_{1,i}t_{2,j}$ 是R3的元组。
- R3的模式是R1和R2的属性，按顺序。
- 但是要注意R1和R2中同名的属性 A：使用 $R1.A$ 和 $R2.A$ 。

交叉联接

交叉连接在哪里使用？

首先，考虑内部联接的工作方式 = 匹配所有可能的行，然后只返回与 **WHERE** 子句中条件匹配的行。

如果在 **FROM** 子句中包含一个表，而没有在 **WHERE** 子句中指定联接条件，则会产生交叉联接。



示例：假设你正在管理计算机实验室，部门刚刚购买了一个软件列表。

您希望确保所有软件都安装在所有计算机上。

在这种情况下，您可以将计算机表与软件许可证表交叉联接，以创建任务清单。

Computers

ComputerID	Make	Model	Year	OperatingSystem
001	Dell	OptiPlex 9010	2012	Windows 10 64bit
002	Dell	OptiPlex 9030	2014	Windows 10 64bit

Software

SoftwareID	Name	Version	Developer	License Server
022	VISSIM	9	PTV	TLAB-2
023	AutoCAD	2016	Autodesk	TLAB-6

结果



- 对于从现有数据中创建表非常有用
- 在实践中使用频率相对较低（仍需了解）

ComputerID	SoftwareID	Completed
001	022	Yes
001	023	No
002	022	Yes
002	023	No

θ -联接

θ -联接也被称为条件联接 (conditional Join)

在关系代数 $R3: = R1 \text{ JOIN}_C R2$

- 取乘积 $R1 \times R2$ (交叉连接)
- 然后将 SELECT_C 应用于结果

在 SQL Server 中, 我们使用 $R1 \text{ JOIN } R2 \text{ ON } C$

条件 C 可以是任何布尔值条件。

- 表示为: $A\theta B$, 其中 θ 为 $=$, $<$, 等等。
- 因此得名为 “ θ -连接”

以下形式的查询在功能上没有区别。

- 使用WHERE

```
SELECT *  
FROM product, company  
WHERE manufacturer = cname
```

- 使用 JOIN or INNER JOIN

```
SELECT *  
FROM product JOIN company  
ON manufacturer = cname
```

消除属性歧义

有时两个关系可以具有相同的属性：

Student(Name, Address, StudyAt)

University(Name, Address)

```
SELECT DISTINCT name, address
FROM student, university
WHERE studyat = name
```

哪个name?
哪个Address?



```
SELECT DISTINCT student.name, university.address
FROM student, university
WHERE student.studyat = university.name
```

假设我们进行R **OUTER JOIN** S :

- 一个R的元组如果没有与其连接的S的匹配元组，则被称为悬浮
- 与 S 的元组相似

OUTER JOIN通过在结果中填充一个特殊的NULL符号来保留悬浮元组

我们可以使用FULL、LEFT或RIGHT来指定要执行的外部联接的类型

根据product_id进行以下两表的内连接，保存商店商品表的商店编号、商品编号和商品表的商品名称列

```
SELECT SP.shop_id,SP.product_id,P.product_name
FROM ShopProduct AS SP INNER JOIN Product AS P
ON [填空1]
```

- ☐ A product_id
- ☐ B SP.product_id=P.shop_id
- ☒ C SP.product_id=P.product_id
- ☐ D product_type

Product（商品）表

product_id (商品编号)	product_name (商品名称)	product_type (商品种类)
0001	T恤衫	衣服
0002	打孔器	办公用品
0003	运动T恤	衣服
0004	菜刀	厨房用具
0005	高压锅	厨房用具

ShopProduct（商店商品）表

shop_id (商店编号)	shop_name (商店名称)	product_id (商品编号)
000A	东京	0001
000A	东京	0002
000A	东京	0003
000B	名古屋	0001
000B	名古屋	0004

提交

外部联接

R

A	B
1	2
4	5

S

A	C
1	3
6	7

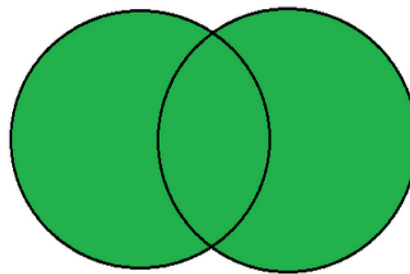
```
SELECT *  
FROM R FULL OUTER JOIN S  
ON R.A = S.A
```

R.A	B	S.A	C
1	2	1	3
4	5	NULL	NULL
NULL	NULL	6	7

**如果我们进行左外连接
(LEFT OUTER JOIN)
会发生什么?**

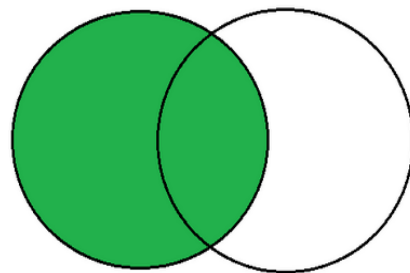
全外连接Full outer join

- 包括两个表中的所有行，不管是否找到匹配项



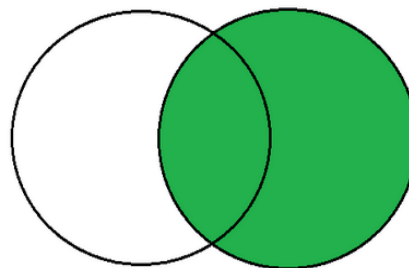
左外连接Left outer join

- 包括 JOIN 子句左边指定的表中的所有行



右外连接Right outer join

- 包含 JOIN 子句右侧指定的表中的所有行



联接示例

candy

Name	Price	type	Calories
M&M	\$1.29	Chocolate	190
Sourpatch	\$1.09	Sour	180
Cognac	\$3.29	Novelty	210

person

Pername	Age	Predilection
James	4	M&M
Tina	29	Sourpatch
Seth	16	Gummybear

Select * FROM candy RIGHT OUTER JOIN person
ON Name=predilection



Name	Price	Type	Calories	Pername	Age	Predilection
M&M	\$1.29	Chocolate	190	James	4	M&M
Sourpatch	\$1.09	Sour	180	Tina	29	Sourpatch
NULL	NULL	NULL	NULL	Seth	16	Gummybear

联接示例

candy

Name	Price	type	Calories
M&M	\$1.29	Chocolate	190
Sourpatch	\$1.09	Sour	180
Cognac	\$3.29	Novelty	210

person

Pername	Age	Predilection
James	4	M&M
Tina	29	Sourpatch
Seth	16	Gummybear

```
SELECT *  
FROM candy INNER JOIN person  
ON Name= predilection
```



内 (θ) 连接结果

Name	Price	Type	Calories	Pername	Age	Predilection
M&M	\$1.29	Chocolate	190	James	4	M&M
Sourpatch	\$1.09	Sour	180	Tina	29	Sourpatch

有时，一个查询需要使用同一关系的两个副本

与重名列类似，可以通过在关系名称后面加上元组变量的名称，使用关键字 AS 来区分副本。

示例：预测未来15分钟的流量状况

- 在表格中，需要一组列来包含某个时间点的交通状况，另一组列具有下一个15分钟的交通状况。
- 从一个表中选择数据，这两组列的选择标准不同。

元组变量

Purchase(buyer, seller, store, product)

查找至少销售一种 “BestBuy” 商店销售产品的所有商店

```
SELECT DISTINCT x.store  
FROM purchase AS x, purchase AS y  
WHERE x.product = y.product AND y.store = 'bestbuy'
```

SQL中的数据定义

SQL由两个组件组成

- 数据定义语言 (Data definition language, DDL)
- 数据操作语言 (Data manipulation language, DML)

对于数据定义，SQL还可以用于定义

- 数据类型 (例如：数字、字符、日期和时间)
- 数据库模式 (创建、更改和删除表)

SQL中的数据类型

精确数字

- TINYINT
- SMALLINT
- INT
- BIGINT
- DECIMAL(p,s)

近似数字

- FLOAT
- REAL

字符串

- CHAR(n) – 固定长度
- VARCHAR(n) – 可变长度
- TEXT

Unicode字符串

1. NCHAR(n)
2. NVARCHAR(n)
3. NTEXT

如果123.451被定义为DECIMAL (6,2) , 它的表达式是什么?

0123.45

日期和时间

- TIME: hh:mm:ss
- DATE: YYYY-MM-DD
- DATETIME: YYYY-MM-DD hh:mm:ss

转换数据类型

一些有用的数据转换函数：

- Convert

```
SELECT CONVERT(DECIMAL(4,2), height)
FROM players
```

- Cast

```
SELECT CAST(height AS DECIMAL(4,2))
FROM players
```

- Datepart

```
Select DATEPART(MONTH, timestamp)
FROM loopdata
```

创建表

示例：基于以下关系模式创建一个表

Person(name, ssn, age, city, gender, birthdate)

```
CREATE TABLE Person(  
    name          VARCHAR(100),  
    ssn           INT,  
    age           SMALLINT,  
    city          VARCHAR(30),  
    gender        CHAR(1),  
    birthdate     DATE  
)
```

我遗漏了什么吗？

定义主键

以下两个查询相等：

```
CREATE TABLE Person(  
  name          VARCHAR(100),  
  ssn           INT PRIMARY KEY,  
  age           SMALLINT,  
  city          VARCHAR(30),  
  gender        CHAR(1),  
  birthdate     DATE  
)
```

```
CREATE TABLE Person(  
  name          VARCHAR(100),  
  ssn           INT,  
  age           SMALLINT,  
  city          VARCHAR(30),  
  gender        CHAR(1),  
  birthdate     DATE,  
  PRIMARY KEY (ssn)  
)
```

修改模式

使用CREATE table命令创建表后，得到一个空的表。

可以使用关键字DROP table从数据库中删除表：

```
DROP TABLE person
```

表结构和内容都将被删除。

小心使用！

- 这是不可返回和不可恢复的。
- 在SQL中删除表时没有警告。

修改模式

修改表结构

- 添加新列

```
ALTER TABLE person  
ADD phone CHAR(20)
```

- 删除列

```
ALTER TABLE person  
DROP birthdate
```

修改模式

修改表结构

- 添加新列

```
ALTER TABLE person  
ADD phone CHAR(20)
```

- 删除列

```
ALTER TABLE person  
DROP birthdate
```

默认值

```
CREATE TABLE Person(  
    name          VARCHAR(100),  
    ssn           INT PRIMARY KEY,  
    age           SMALLINT,  
    city          VARCHAR(30) DEFAULT 'Seattle',  
    gender        CHAR(1),  
    birthdate     DATE  
)
```

默认值: NULL

插入值

常规形式：

```
INSERT INTO R(c1,..., cn) VALUES (v1,..., vn)
```

列名

值

示例：在表中插入一个新的人

```
INSERT INTO person(name, ssn, age, city, gender)  
VALUES ('Kevin', 123456789, 28, 'Bellevue', 'M')
```

- 缺少属性 → NULL。
- 如果按顺序给出值，则可以省略属性名（每个属性必须有一个值）