مزایای معماری توزیع شده

1-کارایی بیشتر در پردازش داده ها به ویژه در پایگاه داده های بزرگ

2- دستیابی بهتر به داده ها

3- اشتراک داده ها

4- افزایش پردازش موازی

5-کاهش هزینه ارتباطات

6- تسهیل گسترش سیستم

معایب معماری توزیع شده

1- پیچیدگی طراحی سیستم

2- پیچیدگی پیاده سازی

3-هزینه بیشتر

4-مصرف حافظه بیشتر

(4) معماری با پردازش موازی

این معماری با ساخت و گسترش ماشینهای موازی برای ایجاد پایگاه داده های خیلی بزرگ بکار میرود،گونه گسترش یافته معماری توزیع شده است و برای تامین کارائی و دستیابی پذیری بالا و گسترش پذیری سریع طراحی میشود.

این معماری بر اساس 2 طرح طراحی میشود.

1- چند پردازنده قوی (5-6)

2- تعدادی پردازنده ضعیف (100-150)

سه بخش اصلی طراحی موازی

1- مدیر تماسهای اجرائی کاربران

2-مدیر درخواستها

3- مدیر داده ها

چهار مدل مختلف در طراحی موازی

1- معماری با حافظه مشترک

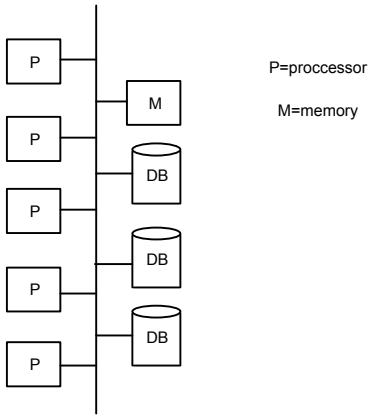
2- معماری با دیسک مشترک

3- معماری بی اجزاء مشترک

4- معماری سلسله مراتبی

(4-1) معماری با حافظه مشترک

در این طرح پردازنده ها به حافظه مشترک دسترسی دارند وضعیت این طرح این است که ارتباط بین پردازنده ها بطور کارا انجام میشود ،همچنین داده های ذخیره شده در حافظه در اختیار همه پردازنده هاقرار میگیرد،عیب این معماری در این است که نمیتوان بیش از 32 یا 64 پردازنده داشت . زیرا احتمال بروز تنگنا در Bus های حافظه یا شبکه ارتباطی افزایش میابد.



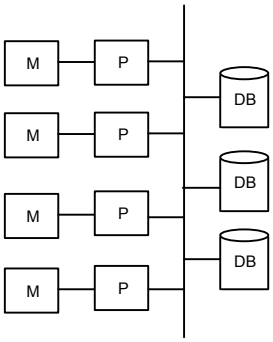
(4-2) معماری با دیسک مشترک

در این طرح تمام پردازنده ها به تمام دیسکها از طریق شبکه ارتباطی دسترسی دارند،هر پردازنده حافظه اختصاصی خود را دارد و مزیت آن

1-عدم بروز تنگنا

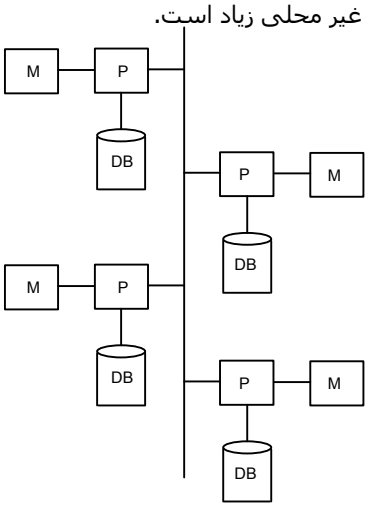
2-تسهیل تحمل خرابی

و عیب آن دشواری در گسترش سیستم است زیرا با افزایش دیسکها و پردازنده ها در ارتباط بین اجزاء تنگنا ایجاد میشودو سرعت ارتباط بین آنها کاهش می یابد.



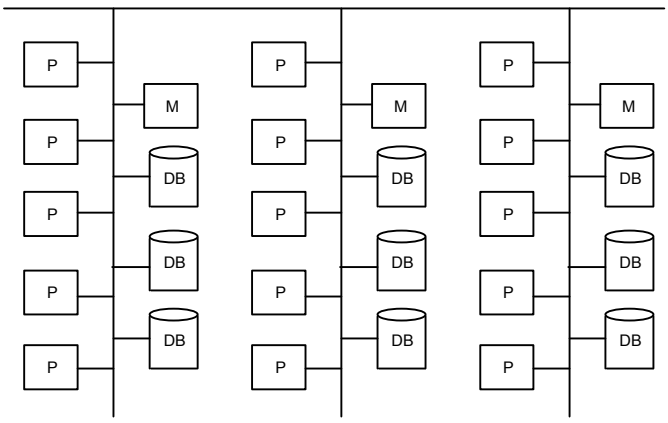
(4-3) معماری بی اجزاء مشترک

در این طرح هرماشین،پردازنده-حافظه و دیسک خود را دارد،یک شبکه ارتباطی با سرعت بالا این ماشینها را بهم مرتبط مینماید هر ماشین نوعی خدمتگذار پایگاهی است،مزیت آن تسهیل گسترش و عیب آن هزینه ارتباط و دستبابی های غیر محلی زیاد است.



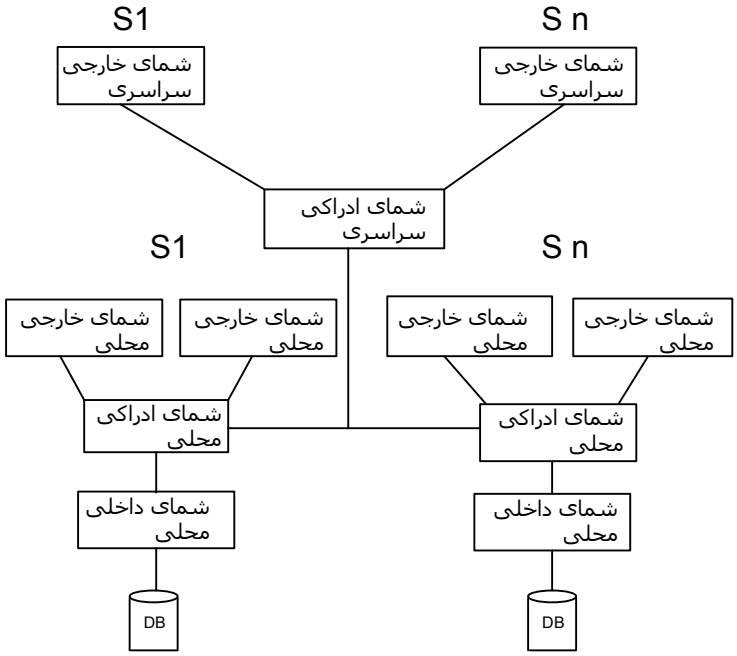
(4-4) معماری سلسله مراتبی

در این طرح ویژگیهای سه طرح پیش را ترکیب کرده و در بالاترین سطح سیستم تعدادی گره با شبکه ارتباطی بهم مرتبطند و اجزاء مشترک ندارند ،پس در این سطح معماری بی اشتراک داریم ،هر گره خود میتواند تعداد کمی پردازنده با حافظه مشترک داشته باشد و یا یک سیستم با دیسکهای مشترک داشته باشد.



معماری چند پایگاهی ⑤

این معماری نوعی معماری توزیع شده است که در آن گره ها خود مختاری کامل دارند،در این معماری معمولا چند سیستم با معماری توزیع شده از قبل وجود دارد و هر سیستم روی عملیات محلی خود کنترل کامل دارد.برای آنکه کاربران بتوانند نیازهای اطلاعاتی خود را تامین کنند بی آنکه مستقیما با اجزای معماری مرتبط باشند به یک لایه نرم افزاری خاصی نیاز است،این لایه نرم افزاری امکان میدهد تا در هر سیستم مدیر پایگاه داده هه روی داده های پایگاه خود کنترل کامل داشته باشد و نیازی به کنترل متمرکز نباشد.



دستورات SQL

ساخت جدول

```
create TABLE stt (
        stid char(10) not null PRIMARY KEY ,
        stname char (10)  ,
        stfamily char (10)  ,
        stdeg char (10)  ,
        stmaj char (10)  ,
        stdeid char (10) ,
      unique (stid) )
```

حذف جدول

```
DROP table stt
```

جدول دانشجو Stt table

	Column Name	Data Type	Length	Allow Nulls
🔑	stid	char	10	
	stname	char	10	✓
	stdeg	char	10	✓
	stmaj	char	10	✓
	stdeid	char	10	✓

جدول درس cot table

	Column Name	Data Type	Length	Allow Nulls
🔑	coid	char	10	
	cotitle	char	10	✓
	credit	int	4	✓
	cotype	char	10	✓
	codeid	char	10	✓

جدول نمره Stcot table

	Column Name	Data Type	Length	Allow Nulls
	stid	char	10	✓
	coid	char	10	✓
	tr	char	10	✓
	[year]	char	10	✓
	grade	int	4	✓

stt

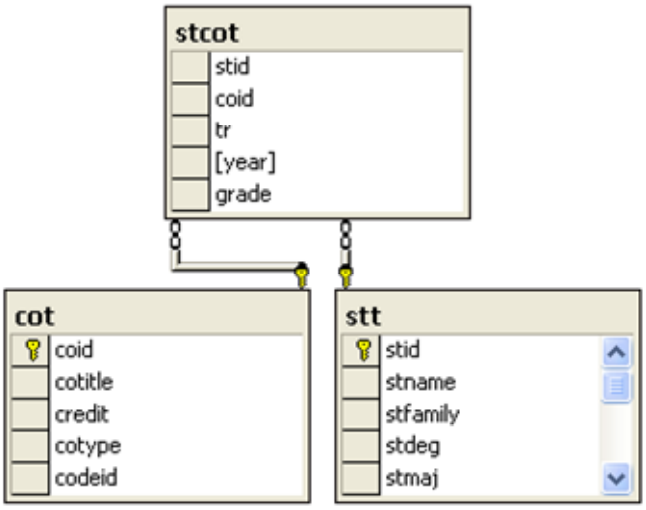
stid	stname	stdeg	stmaj	stdeid
76010222	stn2	ms	phys.	d333
76010444	stn3	ms	comp.	d777
77120333	stn1	bs	math.	d222
78110555	stn4	bs	hist.	d444
78120666	stn5	ms	comp.	d111

cot

coid	cotitle	credit	cotype	codeid
che777	chem lab	2	p	d777
com111	soft.sng.	3	t	d111
com190	data lab	1	p	d111
mat333	eng.math	4	t	d222
phy222	gen.phys.	4	t	d333

stcot

stid	coid	tr	year	grade
77120333	com111	1	77-78	11
77120333	com190	2	78-79	13
76010222	phy222	2	76-77	9
76010444	com190	1	77-78	6
78110555	com111	2	78-79	12
78110555	mat333	2	78-79	14



شماره و نام دانشجویان دوره کارشناسی را بدست آورید.

```
select stid,stname
from stt
where stdeg='ms'
```

	stid	stname
1	76010222	stn2
2	76010444	stn3
3	78120666	stn5

شماره و تعداد واحد تمام درسها را مرتب شده بصورت نزولی بر حسب مقادیر تعداد واحد بازیابی کنید.

```
select coid,credit
from cot
order by credit desc
```

	coid	credit
1	mat333	4
2	phy222	4
3	com111	3
4	che777	2
5	com190	1

شماره هر دانشجو و معدل اورا در ترم دوم 78-79 بدست آورید.

```
select stid,avg(grade)
from stcot
where tr='2' and year='78-79'
group by stid
```

	stid	(No column name)
1	77120333	13
2	78110555	13

کل تعداد درسها را بدست آورید.

```
select count(*)
from cot
```

	(No column name)
1	5

تعداد دانشجویان ثبت نام کرده در ترم دوم 78-79 را بدهید

```
select count(distinct stid)
from stcot
where tr='2' and year='78-79'
```

	(No column name)
1	2

بالاترین و پایین ترین نمره در ترم دوم سال 79-80 چیست

```
select max(grade),min(grade)
from stcot
where tr='2' and year='78-79'
```

	(No column name)	(No column name)
1	14	12

بالاترین و پایین ترین نمره در ترم دوم سال 79-80 در درس com222 چیست

```
select max(grade),min(grade)
from stcot
where tr='2' and year='78-79'and coid='com222'
```

	(No column name)	(No column name)
1	NULL	NULL

بالاترین و پایین ترین نمره با ذکر کد دانشجو در ترم دوم سال 77-78 را بدست آورید


```
select max(grade),min(grade)
from stcot
where tr='2' and year='78-79'
group by stid
```

	stid	(No column name)	(No column name)
1	77120333	13	13
2	78110555	14	12

مشخصات استادانی را بدهید که نام آنها با ma شروع میشود

```
select *
from prof
where prname like 'ma%'
```

مشخصات استادانی را بدهید که درنام آنها رشته کاراکتریzad وجود داشته باشد

```
select *
from prof
where prname like '%zad%'
```

مشخصات استادانی را بدهید که درنام آنها 8 کاراکتری بوده و کاراکتر سوم و چهارم آن ba باشد

```
select *
from prof
where prname like '--ba----'
```

شماره دانشجویان در ترم دوم 78-79 که در درس com111 را بدهید

```
select stid
from stcot
where tr='2' and year='78-79'
group by stid
```

	stid
1	78110555

شماره دانشجویان در ترم دوم 78-79 که در درسcom111 نمرهآنها هنوز اعلام نشده است را بیابید

```
select stid
from stcot
where coid='com111'and tr='2' and grade is null
```

	stid

شماره دانشجویانی را بدهید که یا دوره کارشناسی باشند یا درس com111 را ثبت نام کرده باشند.

```
select stid
from stt where stdeg='ms'
union
select stid
from stcot where
coid='com111'
```

	stid
1	76010222
2	76010444
3	77120333
4	78110555
5	78120666

دانشجویان گروه آموزشی d222 را بر حسب سطح دوره تحصیلی گروه بندی کنید.

```
select stdeid
from stt
where  stdeid='d222'
group by stdeid
```

	stdeid
1	d222

شماره درسهایی را بدهید که در ترم دوم 78-79 کمتر از 10 دانشجو در آن ثبت نام کرده اند.

```
select coid
from stcot where tr='2' and
year='78-79'
group by coid
having count(*)<10
```

	coid
1	com111
2	com190
3	mat333

درس شماره com111 را برای دانشجو با شماره 78110555 حذف کنید.

```
Delete
From stcot
Where stid='78110555' and coid='com111'
```

درسهای دانشجو با شماره 78110555 را در ترم دوم سال 78-79 حذف کنید

```
Delete
From stcot
Where stid='78110555' and tr='2' and
year='78-79'
```

گروه آموزشی با شماره d333 را حذف کنید.

```
Delete
From cot
Where codeid='d333'
```

اطلاعات زیر را درج کنید. (78110888,com888,2,78-79,12)

```
insert into stcot
('stid','coid','tr','year', grade)
values('78110888','com888','2','78-79',12)
```

تعداد دانشجویانی را بیابید که در درس com111 ترم 2 سال 78-79 نمره نیاورده اند.

```
select count(*)
from stcot
where tr='2' and year='78-79' and grade
between 0 and 10
```

	(No column name)
1	0

لیست دانشجویانی را استخراج کنید که در ترم اول 78-79 درس 3 واحدی انتخاب کرده اند.

```
select *
from stt
where stid in
(select stid from stcot where tr='1' and
year='78-79' and
coid in
(select coid from cot where credit='3'))
```

	stid	stname	stdeg	stmaj	stdeid

شماره دانشجویانی بدهید که نمره آنها در درس com190 در ترم دوم 78-79 بین 13 الی 19 باشد

```
select stid
from stcot where tr='2' and year='78-79'and
coid='com190'
and grade between 13 and 19
```

	stid
1	77120333

نام دانشجویانی را بدهید که درس com111 را انتخاب کرده اند.

```
select stt.stname
from stt,stcot
where stt.stid=stcot.stid and
stcot.coid='com111'
```

	stname
1	stn1
2	stn4

```
select stname
from stt where stid in
(select stid from stcot
where coid='com111')
```

نام دانشجویانی را بدهید که حداقل یک درس آزمایشگاهی انتخاب کرده باشند.

```
select stt.stname
from stt where stid in
(select stcot.stid
from stcot where coid in
(select cot.coid
from cot where cotype='p'))
```

	stname
1	stn3
2	stn1

عنوان کتابهایی را بدهید که قیمت آنها از بالاترین قیمت موجود در پایگاه داده ها کمتر باشد

```
Select booktitle
From book
Where bkprice<
(select max(bkprice) from book)
```

تغییر pr777 به دانشیار.

```
Update prof
Set rank='daneshyar'
Where prid='pr777'
```

تعداد واحد درسهای عملی را یک واحد افزایش دهید.

```
update cot
set credit=credit+1
where cotype='p'
```

	coid	cotitle	credit	cotype	codeid
	che777	chem lab	3	p	d777
	com111	soft.sng.	3	t	d111
	com190	data lab	2	p	d111
	mat333	eng.math	4	t	d222
	phy222	gen.phys.	4	t	d333

شماره دانشجویی 78110555 را به 78110777 تغییر دهید.

```
update stt
set stid='78110777'
where stid='78110555'
update stcot
set stid='78110777'
where stid='78110555'
```

```
select first, last, city
   from empinfo
   where first LIKE 'Er%'
```

first	last	city
Eric	Edwards	San Diego
Erica	Williams	Show Low

```
select first, last
   from empinfo
    where last LIKE '%s'

select * from empinfo
    where first = 'Eric'


select first, last, city
from empinfo

select last, city, age
from empinfo
where age > 30;

select first, last, city, state
from empinfo
where first LIKE 'J%'

select * from empinfo

select first, last
from empinfo
where last LIKE '%s'

select first, last, age
from empinfo
where last LIKE '%illia%'

select *
from empinfo
where first = 'Eric'

select first,last,city
from empinfo
where city <>'Payson'
```

first	last	city
Eric	Edwards	San Diego
Mary Ann	Edwards	Phoenix
Ginger	Howell	Cottonwood
Sebastian	Smith	Gila Bend
Gus	Gray	Bagdad
Mary Ann	May	Tucson
Erica	Williams	Show Low
Leroy	Brown	Pinetop
Elroy	Cleaver	Globe

```
select first, last
from empinfo  where last LIKE '%ay'

select *
from empinfo
where first = 'Mary'
```

Empinfo Table

id	first	last	age	city	state
99980	John	Jones	45	Payson	Arizona
99982	Mary	Jones	25	Payson	Arizona
88232	Eric	Edwards	32	San Diego	California
88233	Mary Ann	Edwards	32	Phoenix	Arizona
98002	Ginger	Howell	42	Cottonwood	Arizona
92001	Sebastian	Smith	23	Gila Bend	Arizona
22322	Gus	Gray	35	Bagdad	Arizona
32326	Mary Ann	May	52	Tucson	Arizona
32327	Erica	Williams	60	Show Low	Arizona
32380	Leroy	Brown	22	Pinetop	Arizona
32382	Elroy	Cleaver	22	Globe	Arizona

```
select "column1"
  [,"column2",etc]
  from "tablename"
  [where "condition"]
          [] = optional
```

=Equal
>Greater than
<Less than
>=Greater than or equal
<=Less than or equal
<>Not equal

Inserting into a Table

```
insert into "tablename"
  (first_column,...last_column)
    values (first_value,...last_value)
```

```
INSERT INTO empinfo
(id, [first], [last], age, city, state)
VALUES (23456, ' babak', 'ashofteh',
 '35', ' tehran', ' karaj')
```

id	first	last	age	city	state
99980	John	Jones	45	Payson	Arizona
99982	Mary	Jones	25	Payson	Arizona
88232	Eric	Edwards	32	San Diego	California
88233	Mary Ann	Edwards	32	Phoenix	Arizona
98002	Ginger	Howell	42	Cottonwood	Arizona
92001	Sebastian	Smith	23	Gila Bend	Arizona
22322	Gus	Gray	35	Bagdad	Arizona
32326	Mary Ann	May	52	Tucson	Arizona
32327	Erica	Williams	60	Show Low	Arizona
32380	Leroy	Brown	22	Pinetop	Arizona
32382	Elroy	Cleaver	22	Globe	Arizona
23456	babak	ashofteh	35	tehran	karaj

Column Name	Data Type	Length	Allow Nulls
id	int	4	✓
[first]	char	10	✓
[last]	char	10	✓
age	char	10	✓
city	char	10	✓
state	char	10	✓

Updating Records

```
update "tablename"
set "columnname" =
    "newvalue"
 [,"nextcolumn" =
   "newvalue2"...]
where "columnname"
  OPERATOR "value"
 [and|or "column"
  OPERATOR "value"]
[] = optional
```

```
update empinfo
  set first='hassan'
  where id=23456
```

id	first	last	age	city	state
99980	John	Jones	45	Payson	Arizona
99982	Mary	Jones	25	Payson	Arizona
88232	Eric	Edwards	32	San Diego	California
88233	Mary Ann	Edwards	32	Phoenix	Arizona
98002	Ginger	Howell	42	Cottonwood	Arizona
92001	Sebastian	Smith	23	Gila Bend	Arizona
22322	Gus	Gray	35	Bagdad	Arizona
32326	Mary Ann	May	52	Tucson	Arizona
32327	Erica	Williams	60	Show Low	Arizona
32380	Leroy	Brown	22	Pinetop	Arizona
32382	Elroy	Cleaver	22	Globe	Arizona
23456	hassan	ashofteh	35	tehran	karaj

Empinfo Table

id	first	last	age	city	state
99980	John	Jones	45	Payson	Arizona
99982	Mary	Jones	25	Payson	Arizona
88232	Eric	Edwards	32	San Diego	California
88233	Mary Ann	Edwards	32	Phoenix	Arizona
98002	Ginger	Howell	42	Cottonwood	Arizona
92001	Sebastian	Smith	23	Gila Bend	Arizona
22322	Gus	Gray	35	Bagdad	Arizona
32326	Mary Ann	May	52	Tucson	Arizona
32327	Erica	Williams	60	Show Low	Arizona
32380	Leroy	Brown	22	Pinetop	Arizona
32382	Elroy	Cleaver	22	Globe	Arizona

```
select *
from empinfo
where first LIKE '%Mary%'
```

id	first	last	age	city	state
99982	Mary	Jones	25	Payson	Arizona
88233	Mary Ann	Edwards	32	Phoenix	Arizona
32326	Mary Ann	May	52	Tucson	Arizona

Creating Tables

The **create table** statement is used to create a new table.

Here is the format of a simple **create table** statement:

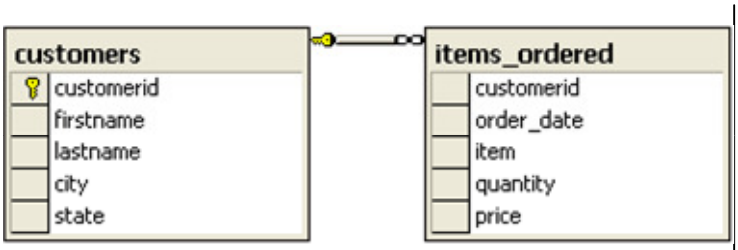
```
create table "tablename"
("column1" "data type",
 "column2" "data type",
 "column3" "data type");
```

create table if you were to use optional constraints:

```
create table "tablename"
("column1" "data type"
          [constraint],
 "column2" "data type"
          [constraint],
 "column3" "data type"
         [constraint]);
 [ ] = optional
```

```
create table employee
(first varchar(15),
 last varchar(20),
 age number(3),
 address varchar(30),
 city varchar(20),
    state varchar(20))
```

char(size) Fixed-length character string. Size is specified in parenthesis. Max 256bytes.
varchar(size) Variable-length character string. Max size is specified in parenthesis.
number(size) Number value with a max number of column digits specified in parenthesis.
date Date value
number(size,d) Number value with a maximum number of digits of "size" total, with a maximum number of "d" digits to the right of the decimal.



customers Table

customerid	firstname	lastname	city	state
10101	John	Gray	Lynden	Washington
10298	Leroy	Brown	Pinetop	Arizona
10299	Elroy	Keller	Snoqualmie	Washington
10315	Lisa	Jones	Oshkosh	Washington
10325	Ginger	Schultz	Pocatello	Idaho
10329	Kelly	Mendoza	Kailua	Hawaii
10330	Shawn	Dalton	Cannon Beach	Oregon
10338	Michael	Howell	Tillamook	Oregon
10339	Anthony	Sanchez	Winslow	Arizona
10408	Elroy	Cleaver	Globe	Arizona
10410	Mary Ann	Howell	Charleston	South Carolina
10413	Donald	Davids	Gila Bend	Arizona
10419	Linda	Sakahara	Nogales	Arizona
10429	Sarah	Graham	Greensboro	North Carolina
10438	Kevin	Smith	Durango	Colorado
10439	Conrad	Giles	Telluride	Colorado
10449	Isabela	Moore	Yuma	Arizona

Item_ordered Table

customerid	order_date	item	quantity	price
10330	30-Jun-1999	Pogo stick	1	28
10101	30-Jun-1999	Raft	1	58
10298	01-Jul-1999	Skateboard	1	33
10101	01-Jul-1999	Life Vest	4	125
10299	06-Jul-1999	Parachute	1	1250
10339	27-Jul-1999	Umbrella	1	4.5
10449	13-Aug-1999	Unicycle	1	180.79
10439	14-Aug-1999	Ski Poles	2	25.5
10101	18-Aug-1999	Rain Coat	1	18.3
10449	01-Sep-1999	Snow Shoes	1	45
10439	18-Sep-1999	Tent	1	88
10298	19-Sep-1999	Lantern	2	29
10410	28-Oct-1999	Sleeping Bag	1	89.22
10438	01-Nov-1999	Umbrella	1	6.75
10438	02-Nov-1999	Pillow	1	8.5
10298	01-Dec-1999	Helmet	1	22
10449	15-Dec-1999	Bicycle	1	380.5
10449	22-Dec-1999	Canoe	1	280
10101	30-Dec-1999	Hoola Hoop	3	14.75
10330	01-Jan-2000	Flashlight	4	28
10101	02-Jan-2000	Lantern	1	16
10299	18-Jan-2000	Inflatable Mattress	1	38
10438	18-Jan-2000	Tent	1	79.99
10413	19-Jan-2000	Lawnchair	4	32
10410	30-Jan-2000	Unicycle	1	192.5
10315	2-Feb-2000	Compass	1	8
10449	29-Feb-2000	Flashlight	1	4.5
10101	08-Mar-2000	Sleeping Bag	2	88.7
10298	18-Mar-2000	Pocket Knife	1	22.38
10449	19-Mar-2000	Canoe paddle	2	40
10298	01-Apr-2000	Ear Muffs	1	12.5
10330	19-Apr-2000	Shovel	1	16.75


```
update phone_book
  set last_name = 'Smith', prefix=555,
suffix=9292
  where last_name = 'Jones'

update employee
  set age = age+1
  where first_name='Mary' and
last_name='Williams'
```

Deleting Records

```
delete from "tablename"

where "columnname"
  OPERATOR "value"
[and|or "column"
  OPERATOR "value"]
[ ] = optional
```

```
delete from employee
  where lastname = 'May'

delete from employee
  where firstname = 'Mike' or firstname =
'Eric'

delete from myemployees_ts0211
 where lastname = 'Weber-Williams'

delete from myemployees_ts0211
  where salary > 70000
```

```
delete from empinfo
where last='yazdi' or id=23456

DELETE FROM empinfo
WHERE ([last] = 'yazdi')OR(id = 23456)
```

Drop a Table

```
drop table "tablename"

drop table myemployees_ts0211
```

customers Table

Column Name	Data Type	Length	Allow Nulls
customerid	int	4	
firstname	char	10	✓
lastname	char	10	✓
city	char	25	✓
state	char	25	✓

Item_ordered Table

Column Name	Data Type	Length	Allow Nulls
customerid	int	4	✓
order_date	char	20	✓
item	char	20	✓
quantity	bigint	8	✓
price	real	4	✓

```
SELECT customerid, order_date, item
FROM items_ordered
WHERE item LIKE 's%'
```

	customerid	order_date	item
1	10298	01-Jul-1999	Skateboard
2	10439	14-Aug-1999	Ski Poles
3	10449	01-Sep-1999	Snow Shoes
4	10410	28-Oct-1999	Sleeping Bag
5	10101	08-Mar-2000	Sleeping Bag
6	10330	19-Apr-2000	Shovel

```
SELECT DISTINCT item
FROM items_ordered
```

	item
1	Bicycle
2	Canoe
3	Canoe paddle
4	Compass
5	Ear Muffs
6	Flashlight
7	Helmet
8	Hoola Hoop
9	Inflatable Mattress
10	Lantern
11	Lawnchair
12	Life Vest
13	Parachute
14	Pillow
15	Pocket Knife
16	Pogo stick
17	Raft
18	Rain Coat
19	Shovel
20	Skateboard
21	Ski Poles
22	Sleeping Bag
23	Snow Shoes
24	Tent
25	Umbrella
26	Unicycle

Aggregate Functions

MIN returns the smallest value in a given column
MAX returns the largest value in a given column
SUM returns the sum of the numeric values in a given column
AVG returns the average value of a given column
COUNT returns the total number of values in a given column
COUNT(*) returns the number of rows in a table

SELECT Statement

```
SELECT [ALL | DISTINCT] column1[,column2]

FROM table1[,table2]

[WHERE "conditions"]

[GROUP BY "column-list"]

[HAVING "conditions]

[ORDER BY "column-list" [ASC | DESC] ]


SELECT name, age, salary
FROM employee
WHERE age > 5o

SELECT name, title, dept
FROM employee
WHERE title LIKE 'Pro%
```

ALL and **DISTINCT** are keywords used to select either ALL (default) or the "distinct" or unique records in your query results. If you would like to retrieve just the unique records in specified columns, you can use the "DISTINCT" keyword. DISTINCT will discard the duplicate records for the columns you specified after the "SELECT" statement: For example

```
SELECT DISTINCT age
FROM employee_info
```

```
SELECT customerid, item, price
FROM items_ordered
WHERE customerid=10449
```

	customerid	item	price
1	10449	Unicycle	180.78999
2	10449	Snow Shoes	45.0
3	10449	Bicycle	380.5
4	10449	Canoe	280.0
5	10449	Flashlight	4.5
6	10449	Canoe paddle	40.0

```
SELECT * FROM items_ordered
WHERE item = 'Tent'
```

	customerid	order_date	item	quantity	price
1	10439	18-Sep-1999	Tent	1	88.0
2	10438	18-Jan-2000	Tent	1	79.989998

```
SELECT state, count(state)
FROM customers
GROUP BY state
```

	state	(No column name)
1	Arizona	6
2	Colorado	2
3	Hawaii	1
4	Idaho	1
5	North Carolina	1
6	Oregon	2
7	South Carolina	1
8	Washington	3

```
SELECT item, max(price), min(price)
FROM items_ordered
GROUP BY item
```

	item	(No column name)	(No column name)
1	Bicycle	380.5	380.5
2	Canoe	280.0	280.0
3	Canoe paddle	40.0	40.0
4	Compass	8.0	8.0
5	Ear Muffs	12.5	12.5
6	Flashlight	28.0	4.5
7	Helmet	22.0	22.0
8	Hoola Hoop	14.75	14.75
9	Inflatable Mattress	38.0	38.0
10	Lantern	29.0	16.0
11	Lawnchair	32.0	32.0
12	Life Vest	125.0	125.0
13	Parachute	1250.0	1250.0
14	Pillow	8.5	8.5
15	Pocket Knife	22.379999	22.379999
16	Pogo stick	28.0	28.0
17	Raft	58.0	58.0
18	Rain Coat	18.299999	18.299999
19	Shovel	16.75	16.75
20	Skateboard	33.0	33.0
21	Ski Poles	25.5	25.5
22	Sleeping Bag	89.220001	88.699997
23	Snow Shoes	45.0	45.0
24	Tent	88.0	79.989998
25	Umbrella	6.75	4.5
26	Unicycle	192.5	180.78999

```
SELECT avg (price)
FROM items_ordered
```

	(No column name)
1	102.06656211614609

```
SELECT avg (price)
FROM items_ordered
where customerid=10101
```

	(No column name)
1	53.458332697550453

```
SELECT count (*)
FROM customers
```

	(No column name)
1	17

```
SELECT max(price)
FROM items_ordered
```

```
SELECT avg(price)
FROM items_ordered
where order_date like '%sep%'
```

	(No column name)
1	54.0

```
SELECT min(price)
FROM items_ordered
WHERE item = 'Tent'
```

	(No column name)
1	79.989998

GROUP BY clause

The GROUP BY clause will gather all of the rows together that contain data in the specified column(s) and will allow aggregate functions to be performed on the one or more columns. This can best be explained by an example:

GROUP BY clause syntax:

```
SELECT column1,
SUM(column2)
FROM "list-of-tables"
GROUP BY "column-list"
```

```
SELECT quantity, max(price)
FROM items_ordered
GROUP BY quantity
```

	quantity	(No column name)
1	1	1250.0
2	2	88.699997
3	3	14.75
4	4	125.0

```
SELECT customerid, count(customerid),
sum(price)
FROM items_ordered
GROUP BY customerid
HAVING count(customerid) > 1
```

	customerid	(No column name)	(No column name)
1	10101	6	320.74999618530273
2	10298	5	118.8799991607666
3	10299	2	1288.0
4	10330	3	72.75
5	10410	2	281.72000122070312
6	10438	3	95.239997863769531
7	10439	2	113.5
8	10449	6	930.78999328613281

ORDER BY clause

ORDER BY is an optional clause which will allow you to display the results of your query in a sorted order (either ascending order or descending order) based on the columns that you specify to order by

ORDER BY clause syntax:

```
SELECT column1, SUM(column2)
FROM "list-of-tables"
ORDER BY "column-list" [ASC | DESC]

[ ] = optional

ASC = Ascending Order - default
DESC = Descending Order
```

```
SELECT lastname, firstname,
city
FROM customers
ORDER BY lastname
```

	lastname	firstname	city
1	Brown	Leroy	Pinetop
2	Cleaver	Elroy	Globe
3	Dalton	Shawn	Cannon Beach
4	Davids	Donald	Gila Bend
5	Giles	Conrad	Telluride
6	Graham	Sarah	Greensboro
7	Gray	John	Lynden
8	Howell	Michael	Tillamook
9	Howell	Mary Ann	Charleston
10	Jones	Lisa	Oshkosh
11	Keller	Elroy	Snoqualmie
12	Mendoza	Kelly	Kailua
13	Moore	Isabela	Yuma
14	Sakahara	Linda	Nogales
15	Sanchez	Anthony	Winslow
16	Schultz	Ginger	Pocatello
17	Smith	Kevin	Durango

33

```
SELECT customerid, count(customerid),
sum(price)
FROM items_ordered
GROUP BY customerid
```

	customerid	(No column name)	(No column name)
1	10101	6	320.74999618530273
2	10298	5	118.8799991607666
3	10299	2	1288.0
4	10315	1	8.0
5	10330	3	72.75
6	10339	1	4.5
7	10410	2	281.72000122070312
8	10413	1	32.0
9	10438	3	95.239997863769531
10	10439	2	113.5
11	10449	6	930.78999328613281

HAVING clause

The HAVING clause allows you to specify conditions on the rows for each group - in other words, which rows should be selected will be based on the conditions you specify. The HAVING clause should follow the GROUP BY clause if you are going to use it

HAVING clause syntax:

```
SELECT column1,
SUM(column2)
FROM "list-of-tables"
GROUP BY "column-list"
HAVING "condition"
```

```
SELECT state,
count(state)
FROM customers
GROUP BY state
HAVING count(state) > 1
```

	state	(No column name)
1	Arizona	6
2	Colorado	2
3	Oregon	2
4	Washington	3

```
SELECT item, max(price),
min(price)
FROM items_ordered
GROUP BY item
HAVING max(price) > 190.00
```

	item	(No column name)	(No column name)
1	Bicycle	380.5	380.5
2	Canoe	280.0	280.0
3	Parachute	1250.0	1250.0
4	Unicycle	192.5	180.78999

```
SELECT item, price
FROM items_ordered
WHERE (item LIKE 'S%') OR (item LIKE 'P%')
OR (item LIKE 'F%')
order by item
```

	item	price
1	Flashlight	28.0
2	Flashlight	4.5
3	Parachute	1250.0
4	Pillow	8.5
5	Pocket Knife	22.379999
6	Pogo stick	28.0
7	Shovel	16.75
8	Skateboard	33.0
9	Ski Poles	25.5
10	Sleeping Bag	89.220001
11	Sleeping Bag	88.699997
12	Snow Shoes	45.0

IN NOT IN and BETWEEN NOT BETWEEN Conditional Operators

```
SELECT order_date, item, price
FROM items_ordered
WHERE price BETWEEN 40 AND 80
order by item
```

	order_date	item	price
1	19-Mar-2000	Canoe paddle	40.0
2	30-Jun-1999	Raft	58.0
3	01-Sep-1999	Snow Shoes	45.0
4	18-Jan-2000	Tent	79.989998

```
SELECT firstname, city, state
FROM customers
WHERE state not IN ('Arizona', 'Washington',
'Oklahoma', 'Colorado', 'Hawaii')
```

	firstname	city	state
1	Ginger	Pocatello	Idaho
2	Shawn	Cannon Beach	Oregon
3	Michael	Tillamook	Oregon
4	Mary Ann	Charleston	South Carolina
5	Sarah	Greensboro	North Carolina

Mathematical Operators

+addition
-subtraction
***multiplication**
/division
%modulo


```
SELECT lastname, firstname, city
FROM customers
ORDER BY lastname DESC

SELECT item, price
FROM items_ordered
WHERE price > 10.00
ORDER BY price ASC
```

	item	price
1	Ear Muffs	12.5
2	Hoola Hoop	14.75
3	Lantern	16.0
4	Shovel	16.75
5	Rain Coat	18.299999
6	Helmet	22.0
7	Pocket Knife	22.379999
8	Ski Poles	25.5
9	Pogo stick	28.0
10	Flashlight	28.0
11	Lantern	29.0
12	Lawnchair	32.0
13	Skateboard	33.0
14	Inflatable Mattress	38.0
15	Canoe paddle	40.0
16	Snow Shoes	45.0
17	Raft	58.0
18	Tent	79.989998
19	Tent	88.0
20	Sleeping Bag	88.699997
21	Sleeping Bag	89.220001
22	Life Vest	125.0
23	Unicycle	180.78999
24	Unicycle	192.5
25	Canoe	280.0
26	Bicycle	380.5
27	Parachute	1250.0

Combining conditions and Boolean Operators

```
SELECT customerid, order_date, item
FROM items_ordered
WHERE (item <> 'Snow shoes') AND
(order_date not like '%1999')
```

	customerid	order_date	item
1	10330	01-Jan-2000	Flashlight
2	10101	02-Jan-2000	Lantern
3	10299	18-Jan-2000	Inflatable Mattress
4	10438	18-Jan-2000	Tent
5	10413	19-Jan-2000	Lawnchair
6	10410	30-Jan-2000	Unicycle
7	10315	2-Feb-2000	Compass
8	10449	29-Feb-2000	Flashlight
9	10101	08-Mar-2000	Sleeping Bag
10	10298	18-Mar-2000	Pocket Knife
11	10449	19-Mar-2000	Canoe paddle
12	10298	01-Apr-2000	Ear Muffs
13	10330	19-Apr-2000	Shovel