SQL

SQLیک زبان ساختیافته برای بازیابی و بهنگام سازی داده از یک پایگاه داده رابطه ای است. دستوراتی برای ایجاد، حذف و تغییر اشیای مختلف در پایگاه داده دارد. پرکاربردترین دستور آن برای اجرای پرس و جوهای مختلف روی پایگاه داده استفاده می شود .

SQL (Structured Query Language)یک زبان برنامه نویسی تعاملی استاندارد برای بازیابی و بهنگام سازی پایگاه داده رابطه ای است .

SQLاجازه می دهد ایجاد جدول، اضافه و حذف داده، اصلاح داده و اجرای پرس و جوی روی داده به شکل یک زبان فرمانی در آیند .

اولین نسخه SQL در دهه ۱۹۷۰ در IBM توسط Donald D. Chamberlin و Raymond F. Boyce پیاده شد. این نسخه که ابتدا SEQUEL (Structured English Query Language)نامیده شد برای کارکردن و بازیابی داده ذخیره شده درپایگاه داده رابطه ای System R بود .

SQLبه عنوان یک استاندارد توسط ANSI در سال ۱۹۸۶ و توسط ISO در سال ۱۹۸۷ پذیرفته شد .

ANSIبیان کرد که تلفظ رسمی آن es queue el است. در حالیکه افراد حرفه ای انگلیسی زبان پایگاه داده هنوز نام آنرا sequel تلفظ می کنند .

گونه هائی ازSQL ، به عنوان یک زبان پرس وجو و کار با داده، توسط عرضه کنندگان DBMS همراه با ضمایمی ایجاد شد. با بیرون آمدن استاندارد SQL:1999 بسیاری از ضمیمه ها به عنوان بخشی از زبان SQL پذیرفته شدند .

در این بخش فرامینی از SQL که در اکثر گونه ها وجود دارد شرح داده خواهد شد.

فضاهای خالی در عبارات SQL ندیده گرفته می شوند و برای خوانائی کدهای SQL استفاده می شوند. سمیکولن (;) به عنوان پایان دهنده عبارت است .

CREATE TABLE

عبارت Create یک فرمان DDL در SQL است که برای ایجاد یک شیء در پایگاه داده استفاده می شود. احتمالا معمول ترین فرمان Create فرمان CREATE TABLEاست .

CREATE TABLEاجازه ایجاد شمای یک جدول را می دهد. فرم کلی آن به صورت زیر است :

```
CREATE TABLE tablename (
colname datatype coloptions
,colname datatype coloptions
,colname datatype coloptions
,additionalinfo
);
```

هر سطر یک فیلد جدول را مشخص می کند. تعریف هر فیلد شامل نام، نوع داده و اطلاعات اضافی مربوطه می تواند باشد. سطرها با کاما (,) از هم جدا می شوند .

نوع های داده مختلفی در یک DBMS وجود دارد که مهمترین آنها عبارتند از:

- CHAR(n) •رشته کاراکتری ASCII با طول ثابت n کاراکتر .
- VARCHAR(n) •رشته کاراکتری ASCII با طول متغیر با حداکثر n کاراکتر .
- NVARCHAR(n) •رشته کاراکتری Unicode با طول متغیر با حداکثر n کاراکتر .
- INT •عدد صحیح (زیرمجموعه متناهی از اعداد صحیح که وابسته به ماشین است(.
- SMALLINT •عدد صحیح کوچک (زیرمجموعه وابسته به ماشین از نوع صحیح(.
- DECIMAL(p,d) •عدد ممیز ثابت، با دقت تعریف شده p رقم و با d رقم در سمت راست ممیز .
- REAL,DOUBLE precision •اعداد ممیزشناور با دقت مضاعف (وابسته به ماشین(.
- FLOAT(n) •عدد ممیز شناور با دقت تعریف شده حداقل n رقم .
- DATE •تاریخ با فرمتday/month/year .

در انتهای هر ستون می توان اطلاعات اضافی داشت. معمول ترین آنها عبارتند از:

PRIMARY KEYیعنی این فیلد کلید اصلی است

NOT NULLیعنی این فیلد باید مقداری داشته باشد و نمی تواند تهی باشد

REFERENCES *othertable* (*primarykeyname*)یعنی این فیلد یک کلید خارجی است که در جدول دیگری کلید اصلی است

در انتهای تعریف می توان اطلاعات اضافی دیگری را داشت. برای نمونه:

PRIMARY KEY (*column1,column2,...*)اگر جدول کلید اصلی ترکیبی دارد باید آنرا در انتهای تعریف مشخص کنید.
FOREIGN KEY (*column1,column2,...*) REFERENCES *othertable*اگر جدول ارتباطی با جدول دیگر دارد که یک کلید ترکیبی دارد بنابراین ستون های این جدول که کلید های خارجی هستند باید به این صورت لیست شوند.

مثال. فرمان ایجاد یک جدول به نام employeesبا چند فیلد نمونه به صورت زیر می تواند باشد :

```
CREATE TABLE employees (
id INTEGER PRIMARY KEY
,first_name CHAR(50)
,last_name CHAR(75) NOT NULL
,date_of_birth DATE
);
```

مثال. تعریف جداول Car و Driver می تواند به صورت زیر باشد:

```
CREATE TABLE driver (
name varchar(30)
,dob DATE NOT NULL
,PRIMARY KEY (name)
);

CREATE TABLE car (
regno VARCHAR(8)
,make VARCHAR(20)
,colour VARCHAR(30)
,price DECIMAL(8,2)
,owner VARCHAR(30)
,PRIMARY KEY(regno)
,FOREIGN KEY(owner) REFERENCES driver
);
```

DROP TABLE

عبارت DROP برای از بین بردن یک شیء در پایگاه داده است .فرمان DROP TABLE زمانی بکار می رود که بخواهید جدول را حذف کنید. فرم کلی آن به صورت زیر است :

DROP TABLE *tablename;*

مثال. فرمان زیر جدول employees را حذف می کند .

DROP TABLE employees;

تنها نکته در حذف یک جدول این است که اگر جدولی توسط کلید خارجی با این جدول در ارتباط باشد نمی توانید آنرا حذف کنید .

مثال. چون جدول car توسط کلید خارجی با جدول driver در ارتباط است می توانید ابتدا جدول Car و سپس Driverرا حذف کنید ولی عکس آن نمی شود .

DROP TABLE car;
DROP TABLE driver;

SELECT

معمولا بیشترین عملی که روی پایگاه های داده توسط SQL انجام می گیرد جستجو است، که توسط عبارت SELECTانجام می پذیرد .

دستور SELECT داده ها را از یک یا چند جدول مرتبط بازیابی می کند و اغلب تاثیری روی داده ذخیره شده در پایگاه داده ندارد .

SELECTپیچیده ترین عبارت SQL است. فرم کلی عبارت SELECT به صورت زیر است:

SELECT DISTINCT columns AS columns
FROM table
WHERE rule
GROUP BY columns

HAVING rule
ORDER BY columns;

دستور SELECT دارای چند عبارت اختیاری به شرح زیر است:

FROM •جدول یا جداولی را که از آنها داده بازیابی می شود را مشخص می کند. برای الحاق جداول بر اساس ضابطه خاصی می تواند همراه با عبارت JOIN بیاید.
WHERE •همراه با یک گزاره شرطی برای محدود کردن سطرهای برگردانده شده استفاده می شود .
GROUP BY •اغلب همراه با توابع تجمعی(AVE)، SUM، MAX، MINو (COUNT برای ترکیب یا گروه بندی سطرها یا حذف سطرهای تکراری در مجموعه نتیجه استفاده می شود .
HAVING •همراه با یک گزاره شرطی روی نتیجه GROUP BY کار می کند. توابع تجمعی می توانند در گزاره شرطی HAVING هم استفاده شوند .
ORDER BY •برای تعیین ستون های که بر اساس آنها داده نتیجه مرتب می شود (صعودی و نزولی . (

FROM

در ساده ترین دستور SELECT کلیه سطرهای یک جدول که بعد از عبارت FROM ذکر شده است را بازیابی می کند. لیستی از فیلدهای موردنظر در مقابل عبارت SELECT قرار می گیرد. نام فیلدها با کاما (,) از هم جدا می شوند .

علامت ستاره (*) برای بیان کلیه فیلدهای یک جدول (یا چند جدول) می تواند استفاده شود .

مثال. اسامی کلیه شعبه ها در loan را پیدا کن

```
SELECT branch_name
FROM loan;
```

مثال. مشخصات کلیه مشتریان را پیدا کن.

```
SELECT *
FROM customer;
```

عبارات محاسباتی +، -، * و / روی یک فیلد در لیست فیلدها می توانند بکاربرده شوند .

مثال. دستور زیر جدولی مشابه **loan** را بر می گرداند که مقدار صفت خاصه **amount** آن ۱۰ برابر شده است .

```
SELECT loan_number, branch_name, amount * 100
FROM loan;
```

ممکن است لازم باشد داده های موردنیاز را از دو یا چند یک جدول استخراج کنیم .

مثال. اسامی و مقدار وام کلیه مشتریانی را که وامی از شعبه **Perryridge** گرفته اند را پیدا کن .

```
SELECT customer_name, borrower.loan_number, amount
FROM borrower, loan
WHERE borrower.loan_number = loan.loan_number AND
branch_name = 'Perryridge';
```

مثال. دقت کنید که اگر شرطی ذکر نشود ضرب دکارتی دو جدول حاصل می شود.

```
SELECT *
FROM borrower, loan;
```

DISTINCT

SQLاجازه تکرار در نتیجه **SELECT** را می دهد. بنابراین جدول حاصل ممکن است دارای سطرهای مشابه باشد. اگر این موضوع موردنظر نباشد عبارت **DISTINCT** را استفاده می کنیم. در این صورت کلیه سطرهای جدول حاصل منحصر بفرد خواهند بود. و سطرهای تکراری حذف می شوند .

مثال. اسامی کلیه شعب بانک که از آنها وامی گرفته شده است را با حذف تکراری ها لیست کن .

```
SELECT DISTINCT branch_name
FROM loan;
```

عبارت **all** مشخص می کند که تکراری ها حذف نشوند.

```
SELECT ALL branch_name
FROM loan;
```

WHERE

عبارت WHERE برای انتخاب سطرهای برگردانده شده از دستور SELECT بر طبق شرط خاصی بکار می رود .

برای بیان شرط می توان عملگرهای مقایسه ای=) ، =!، <>، <، >، =<و (=>را استفاده کرد .

نتایج مقایسه را می توان توسط عملگرهای منطقیNOT) ، ANDو (ORو پرانتز با هم ترکیب کرد. اجرای عملگرهای منطقی به ترتیب الویت آنها است NOT .الویت بالاتر و OR الویت کمتر را دارد. پرانتز می تواند برای تعیین ترتیب انجام عملیات استفاده شود. عمل داخل پرانتز همیشه اول انجام می گیرد .

مثال. کلیه شماره وام هائی که مقدار وام آنها از ۱۲۰۰ بیشتر است را پیدا کن .

```
SELECT loan_number
FROM loan
WHERE amount > 1200;
```

مثال. کلیه شماره وام های شعبه Perryridge که مقدار وام آنها از ۱۲۰۰ بیشتر است را پیدا کن .

```
SELECT loan_number
FROM loan
WHERE amount > 1200 AND branch_name = 'Perryridge';
```

توجه کنید که هنگام مقایسه با رشته باید آنرا در کوتیشن (' ') قرار داد.

مثال.شماره وام هائی که مقدار وام آنها بین ۹۰،۰۰۰ و ۱۰۰،۰۰۰ می باشد را پیدا کن .

```
SELECT loan_number
FROM loan
WHERE amount BETWEEN 90000 AND 100000;
```

عملگر IN برای تعیین اینکه آیا مقدار مشخصی درون مجموعه ای از مقادیر وجود دارد یا خیر بکار می رود .

مثال. مقدار وامهایی که از شعبPerryridge ، Downtownیا Redwood گرفته شده اند را پیدا کن .

```
SELECT amount
FROM loan
WHERE Branch_name IN (' Perryridge' , 'Downtown' , 'Redwood');
```

هنگام کارکردن با رشته ها وقتی خواهان مطابقت کامل رشته ها نیستیم بلکه بخشی از رشته یا الگوی خاصی از آن بیشتر موردنظر است، می توان از عبارت LIKE به جای علامت (=) استفاده کرد. دو کاراکتر '٪' و '_' به ترتیب به معنی یک کاراکتر و بیشتر از یک کاراکتر را برای تطابق می توان بکار برد .

مثال. اسامی کلیه مشتریانی که آدرس آنها شامل کلمه Main است را پیدا کن.

```
SELECT customer_name
FROM customer
WHERE customer_street LIKE '%Main%';
```

AS

SQLاجازه تغییر نام جدول را توسط عبارت AS می دهد.

مثال. تعداد وام و مقدار وام کلیه وام ها را پیدا کرده، نام ستون loan_number به loan_id تغییر بده .

```
SELECT loan_number AS loan_id, amount
FROM loan;
```

مثال. نام و تعداد وام کلیه مشتریانی که وامی در یک شعبه دارند را پیدا کن.

```
SELECT customer_name, T.loan_number, S.amount
FROM borrower AS T, loan AS S
WHERE T.loan_number = S.loan_numbe;
```

توابع تجمعی

توابع تجمعی (aggregation function) عملگرهایی هستند که محاسبه آماری روی گروهی از مقادیر داده ای را انجام می دهند. این توابع روی مقادیر یک ستون از یک جدول عمل می کند و یک مقدار را به عنوان نتیجه بر می گردانند. این توابع شاملAVG ، SUM، MAX، MINو COUNT هستند .

نتیجه تجمع نامی ندارد می تواند از AS برای نامگذاری آن استفاده کرد .

مثال. میانگین موجودی حساب ها در شعبه Perryridge را پیدا کن.

```
SELECT AVG (balance)
FROM account
WHERE branch_name = 'Perryridge';
```

COUNTتعداد سطرهای موجود در جواب که حاوی NULL نیستند را می دهد. برای اینکه تعداد مستقل از NULLباشد COUNT(*)را استفاده کنید .

مثال. تعداد مشتریان بانک را محاسبه کن.

```
SELECT COUNT (*)
FROM customer;
```

گاهی در جواب تعدادی سطرها مشابه می شوند، اگر می خواهید تعداد سطرهای متمایز را بدست آورید از COUNT DISTINCTاستفاده کنید .

مثال. تعداد افرادی که در بانک پول دارند را پیدا کن.

```
SELECT COUNT (DISTINCT customer_name)
FROM depositor;
```

مثال. تعداد افرادی که در هر شعبه بانک حساب دارند را پیدا کن.

```
SELECT branch_name, COUNT (DISTINCT customer_name)
FROM depositor, account
WHERE depositor.account_number = account.account_number
GROUP BY branch_name;
```

GROUP BY

در بسیاری موارد تحلیل آماری روی گروهی از داده ها موردنیاز است. برای گروه بندی از عبارت GROUP BY استفاده کنید .

مثال. اسامی کلیه شعب و میانگین موجودی حساب آنها را پیدا کن

```
SELECT branch_name,AVG (balance)
FROM account
GROUP BY branch_name;
```

HAVING

توابع تجمعی در عبارت WHERE کار نمی کنند. اگر می خواهید با توجه به نتیجه توابع تجمعی شرطی داشته باشید از عبارت HAVING استفاده کنید .مانند عبارت WHERE کار می کند با این تفاوت که روی آخرین داده حاصل کار می کند و اجازه استعمال توابع تجمعی را هم می دهد. البته هزینه اجرای آن بالاست بنابراین فقط در زمانی که واقعا نیاز است استفاده کنید .

مثال. اسامی کلیه شعب را که میانگین حساب آنها بیشتر از ۱۲۰۰ است را پیدا کن

```
SELECT branch_name,AVG (balance)
FROM account
GROUP BY branch_name
HAVING AVG (balance) > 1200;
```

ORDER BY

ترتیب رکوردها در نتیجه پرس و جو معمولا بدون نظم است .اگر می خواهید جدول حاصل دارای نظم خاصی بر طبق یک یا چند فیلد باشد عبارت ORDER BYرا به همراه فیلدهای موردنظر اضافه کنید .

برای ترتیب نزولی از DESC و برای ترتیب صعودی از ASC روی هر صفت خاصه استفاده می شود. پیش فرض ترتیب صعودی است.

مثال. اسامی کلیه مشتریانی که وامی در شعبه Perryridge دارند را به ترتیب حروف الفبا لیست کن.

```
SELECT DISTINCT customer_name
FROM borrower, loan
WHERE borrower loan_number = loan.loan_number AND
branch_name = 'Perryridge'
ORDER BY customer_nam;
```

NULL

ممکن است مقدار بعضی از صفات خاصه در رکوردها تهی باشد که توسط NULL مشخص می شود. وقتی فیلدی حاوی NULL است بیان کننده این است که مقدار آن فیلد نامعلوم است یا مقداری در دنیای واقعی ندارد .

عملگرهای مقایسه ای اگر روی NULL عمل کنند مقدار Unknown را برمی گردانند. گزاره IS NULL می تواند برای بررسی مقادیر NULL استفاده شود. عملگر متضاد آن IS NOT است که مقادیری که NULL نیستند را پیدا می کند .

نتیجه هر عبارت ریاضی روی NULL برابر با NULL است. کلیه توابع تجمعی به استثنای COUNT از مقدار NULLصرفنظر می کنند .

مثال. تعداد وام هائی که میزان وام آنها معین نیست را پیدا کن.

```
SELECT loan_number
FROM loan
WHERE amount IS NULL;
```

پرس و جوهای تودرتو

در SQL مکانیسمی برای پرس و جوهای تودرتو فراهم شده است .به عبارت دیگر یک عبارت SELECT می تواند درون دیگری قرار بگیرد تا نتیجه اجرای آن در شرط WHERE عبارت SELECT دیگر استفاده شود. عبارت SELECTدوم را یک پرس و جوی فرعی می نامند و حتما باید یک فیلد را برگرداند یعنی فقط یک صفت خاصه در دستور SELECT آن باید باشد .

وقتی حاصل پرس و جوی فرعی بیشتر از یک سطر باشد از عملگرهایALL ، ANY، IN، NOT IN، EXISTSو NOT EXISTSبرای گرفتن نتیجه مطلوب باید استفاده کرد .

مثال. اسامی کلیه مشتریانی که هم حساب وهم وام در بانک دارند را پیدا کن.

```
SELECT DISTINCT customer_name
FROM borrower
WHERE customer_name IN (SELECT customer_name FROM depositor );
```

مثال. اسامی کلیه مشتریانی که از بانک وام گرفته اند ولی حساب ندارند را پیدا کن.

```
SELECT DISTINCT customer_name
FROM borrower
WHERE customer_name NOT IN (SELECT customer_name FROM depositor );
```

مثال. اسامی کلیه مشتریانی که هم حساب وهم وام در شعبه Perryridge دارند را پیدا کن.

```
SELECT DISTINCT customer_name
FROM borrower, loan
WHERE borrower.loan_number = loan.loan_number AND branch_name = 'Perryridge'
AND
branch_name, customer_name IN (SELECT branch_name, customer_name
FROM depositor, account WHERE depositor.account_number = account.account_number
);
```

توجه. پرس و جوهای بالا ساده تر هم می تواند نوشته شود.

ترکیب پرس و جوها

گاهی می خواهیم نتیجه دو پرس و جو را با هم به نحوی ترکیب کنیم و یک جدول را بدست بیاوریم. عملگرهای UNION، INTERSECTو EXCEPT برای ترکیب نتیجه دو پرس و جو می توانند استفاده شوند که به ترتیب مشابه عملگرهای اجتماع، اشتراک و تفاضل در جبر رابطه ای عمل می کنند .

مجموعه فیلدهای دو پرس و جوئی که با هم ترکب می شوند باید از نظر تعداد و نوع مطابق هم باشند .

عملگر UNION جدولی شامل کلیه سطرهای هردو پرس و جو را می دهد. سطرهای تکراری حذف می شوند مگر اینکه از عبارت UNION ALL استفاده شود .

عملگر INTERSECT سطرهای مشترک در نتیجه دو پرس و جو را بر می گرداند. سطرهای تکراری حذف می شوند مگر اینکه از عبارت INTERSECT ALL استفاده شود .

عملگر EXCEPT سطرهائی از نتیجه پرس و جوی اول که در نتیجه پرس و جوی دوم ظاهر نشده است را بر می گرداند EXCEPT ALL .سطرهای تکراری را حذف نمی کند .

مثال. اسامی کلیه مشتریانی که هم حساب وهم وام در بانک دارند را پیدا کن.

```
SELECT customer_nameFROM borrower
UNION
SELECT customer_name FROM depositor;
```

JOIN

وقتی بخواهیم اطلاعاتی را از دو جدول بدست بیاوریم می توانیم عمل الحاق را روی دو جدول انجام دهیم. عملگر JOINرکوردهای گرفته شده از دو جدول را با هم ترکیب می کند و جدول دیگری را به عنوان نتیجه می دهد. شرط الحاق نحوه جفت کردن رکوردهای دو جدول را تعیین می کند .

الحاق دارای انواع مختلفی نظیر الحاق طبیعی و الحاق خارجی است. نوع الحاق تعیین می کند چه رکوردهائی از هر جدول که جفتی در جدول دیگر ندارند در جدول نتیجه باید اضافه شوند .

Natural Join

در الحاق طبیعی کلیه سطرهائی که فیلدهای همنام آنها که در هردو جدول دارای یک مقدار هستند، درنظر گرفته می شود. جدول حاصل تنها شامل یک ستون از ستونهای هم نام خواهد بود .

مثال. اسامی وام گیرنده ها به همراه وام های گرفته شده از بانک را پیدا کنید.

```
SELECT *
FROM borrower NATURAL JOIN loan;
```

Outer Join

در الحاق خارجی نیازی نیست رکوردهای دو جدول حتما رکورد مطابقی در جدول دیگر داشته باشند. الحاق خارجی، بسته به جدولی که همه سطرهایش نگهداشته می شود، به سه دسته الحاق چپ، راست و کامل تقسیم می شود .

LEFT OUTER JOINکلیه مقادیر جدول سمت چپ خود را بعلاوه مقادیری از جدول سمت راست که مطابقت دارند می دهد RIGHT OUTER JOIN .کلیه مقادیر جدول سمت راست خود را می دهد بعلاوه مقادیری از جدول سمت چپ که رکوردهایش جور هستند FULL OUTER JOIN .نتیجه الحاق خارجی چپ و راست را با هم ترکیب می کند .

الحاق خارجی داده های مفقود را، برای سطرهائی که شرط الحاق در آنها برقرار نبوده، با NULL پر می کند .

مثال. اسامی کلیه وام گیرنده ها به همراه میزان وامی که گرفته اند را پیدا کنید.

```
SELECT *
FROM borrower LEFT OUTER JOIN loan
ON borrower.loan_no = loan.loan_no;
```

مثال. مقدار کلیه وام های گرفته شده از شعبه Perryridge را به همراه نام وام گیرنده ها پیدا کنید.

```
SELECT *
FROM borrower RIGHT OUTER JOIN loan
ON borrower.loan_no = loan.loan_no
WHERE loan.branch_name = 'Perryridge';
```

DELETE

فرمان DELETE اجازه حذف سطرهائی از یک جدول را می دهد. فرم کلی دستور به شکل زیر است :

DELETE FROM *table_name* WHERE *condition*;

کلیه رکوردهائی که شرط WHERE در آنها برقرار است از جدول حذف می شوند. اگر شرطی بیان نشود کلیه رکوردهای جدول حذف خواهند شد .

دستور DELETE هیچ رکوردی را به عنوان خروجی بر نمی گرداند.

مثال. کلیه رکوردهای وام گیرندگان را حذف کن.

```
DELETE FROM borrower;
```

مثال. کلیه رکوردهایی که حسابی در شعبه Perryridge دارند را حذف کن.

```
DELETE FROM account
WHERE branch_name = 'Perryridge';
```

INSERT

دستور INSERT اجازه اضافه کردن رکوردی به یک جدول را می دهد. فرم کلی آن به صورت زیر است :

INSERT INTO *table_name*
(*column_list*)
VALUES (*value_list*);

Column_listلیست فیلدهائی است که مقادیر به آنها نسبت داده خواهد شد و اگر برای همه فیلدها مقداری درنظرگرفته شود می تواند حذف شود value_list .مجموعه ای از مقادیر است که برای هر فیلد در لیست column_listیا فیلدهای جدول که در دستور CREATE TABLE تعریف شده اند مقداری دارد .

تعداد ستون ها و مقادیر آنها باید یکسان باشد. اگر فیلدی ذکر نشود مقدار پیش فرض آن درنظر گرفته می شود .

به فیلدهائی که در دستور CREATE TABLE به عنوان PRIMARY KEY یا با محدودیت NOT NULL تعریف شده اند باید مقداری نسبت داده شود .

مثال. حساب جدیدی را با شماره A-9732 و با موجودی ۱۲۰۰ در شعبه Perryridge اضافه کن .

```
INSERT INTO account
VALUES ('A-9732', 'Perryridge',1200);
یا
INSERT INTO account
(branch_name, balance, account_number)
VALUES ('A-9732', 'Perryridge',1200);
```

مثال: اضافه کردن رکورد جدیدی در جدول account با مقدار موجودی null به صورت زیر انجام می شود .

```
INSERT INTO account
VALUES ('A-777','Perryridge', null );
```

UPDATE

دستور UPDATE اجازه تغییر داده های درون یک جدول را می دهد. این دستور هیچ رکوردی را اضافه یا حذف نمی کند. شکل کلی آن به صورت زیر است :

UPDATE *table_name*
SET *column_name = value, column_name=value, ...*
WHERE *condition*;

در کلیه رکوردهائی که شرط در آنها برقرار بوده است مقدار فیلدی که نامش در عبارت SET تعیین برابر با مقدار جدید می شود. دستور UPADTEممکن است روی یک یا چند رکورد در یک جدول تاثیر بگذارد .

مثال. حساب هائی که موجودی آنها بیشتر از ۱۰۰۰۰ است را به میزان ۶٪ افزایش بده .

```
UPDATE account
SET balance = balance * 1.06
WHERE balance > 10000;
```

ایرادهایSQL

به چند نمونه از انتقادهائی که درباره استفاده کاربردی از SQL وجود دارد اشاره می شود :

•پیاده سازی های مختلفی از SQL توسط سازندگان DBMS وجود دارد که گاهی با هم متناقض و نا موافق است. خصوصا مواردی نظیر فرمت زمان و تاریخ، الحاق رشته ها، مقادیر تهی و حساسیت به متن در مقایسه از یک سازنده به دیگری ممکن است متفاوت باشد .
•نوشتن شرط اشتباه در الحاق جداول به راحتی ضرب دکارتی دو جدول را نتیجه می دهد که به ندرت ممکن است در عمل سودمند باشد .
•ممکن است عبارت شرط در بهنگام سازی و حذف اشتباهی ساخته شود و روی کلیه سطرهای جدول ناخواسته تاثیر بگذارد .
SQL •و مدل رابطه ای راهی برای حمایت ساختارهای درختی و ارجاع بازگشتی به سطرهای دیگر یک جدول ندارند .
