منبع : مرکز اطلاع رساني و پژوهش هاي برنامه نويسي ايران (irAsp.Net)
آموزش: مونا غفاري نژاد
ايميل: mona3000m@yahoo.com

عنوان: **SQLچيست ؟**

SQL :

مخفف Structerd Query Language ميباشد زباني ساختيافته براي پرس و جو از بانك اطلاعاتي ميباشد . با استفاده از SQL ميتوانيد داده هايي را كه در بانك اطلاعاتي خود داريد به نحوي كه ميخواهيد بازيابي كنيد و مورد تحقيق واگر خواستيد تغيير دهيد .
براي تمرين كردن با SQL به يك بانك اطلاعاتي همچون SQL – server و Access يا اوراكل و يا هر بانك اطلاعاتي ديگري نيازمنديد .
گرچه SQL زباني تحت استاندارد ANSI (American National Standards Institute) ميباشد اما هر بانك اطلاعاتي از نسخه دلخواه خودش استفاده ميكند بنابراين ممكن است بعضي از دستورات در يك بانك اطلاعاتي بنا به نسخه اي از اس كيو ال كه استفاده ميكند با دستورات در بانك اطلاعاتي ديگر كمي متفاوت باشد .
دستورات زبان SQL به پنج دسته تقسيم بندي ميشوند كه بترتيب :

Quote:

DDL (Data Definition Language)زبان تعريف داده ها .
DML (Data Manipulation Language)زبان پردازش داده ها .
DCL (Data Control Language)دستورات مديريت و كنترل داده ها .
DQL (Data Query LAnguage)دستورات پرس و جوي داده ها .
دستورات كنترلي تراكنشي .
دستورات مديريت داده ها .

در دسته اول DDL زباني است كه امكان ايجاد وساخت يا پاك كردن يك جدول از بانك اطلاعاتي را بر عهده دارد دستورات اين رده بترتيب زير است :

Code:
CREATE , ALTER , DROP , CREATE INDEX , ALTER INDEX , DROP INDEX

دسته دوم دستورات DML هستند كه براي اعمال تغييرات بر جداول درست شده اند كه از سه دستور اصلي زيرتشكيل شده اند :

Code:
INSERT , UPDATE , DELETE

دسته سوم دستورات DQL هستند كه تنها از يك دستور بنام SELECT تشكيل شده اند كه بيشترين حجم كاري را براي يك كاربر كه با SQL كار ميكند را در بر ميگيرد . اين دستور بهمراه چندين دستوري كه درون خود وجود داردسبب گستردگي استفاده از اين دستور ميشود . كه بعدا با اين دستور بيشتر اشنا خواهيم شد .

دسته چهارم دستورات DCL هستند . همانطور كه از نامشان پيداست جهت كنترل داده هاي بانك

اطلاعاتي كه بيشتر در رابطه با دسترسي اين داده ها به ديگر كاربران است كاربرد دارد . كه متشكل از دستورات زير است :

Code:
ALTER PASSWORD , REVOKE , GRANT , CREATE SYNONYM

دسته بعد دستورات كنترلي TRANSACTION ها ميباشد كه امكان مديريت تراكنشي بانك اطلاعاتي را براي كاربر فراهم ميسازد . كه از دستورات زير تشكيل شده اند :
Code:
COMMIT , ROLLBACK , SAVEPOINT , SET TRANSACTION

و دسته اخر كه مكمل كننده ان دستورات دسته قبل بودند دستورات مديريت داده ها هستند كه امكان بررسيو تحليل عمليات داخل بانك اطلاعاتي را فراهم ميكنند . در ضمن يادتان باشد كه اين مديريت را با مديريت بانك اطلاعاتي اشتباه نگيريد .

Code:
START AUDIT , STOP AUDIT

و اما انچه كه در اين اموزش فرا ميگيريم سعي ميشود كاملا واضح نحوه استفاده و كاربرد تك تك دستورات فوق بيان شود.

عنوان: **مقدمه اي بر SQL و اشنايي با select وwhere**

Quote:
What Is SQL ?

SQLمخفف Structured Query Language ميباشد كه خوانده ميشود ess-que-el
SQLبراي ارتباط با يك بانك اطلاعاتي استفاده ميشود . اين زبان كه بر طبق سازمان استاندارد سازي زبانها ANSI American National Standard Institute تنظيم شده است براي مديريت سيستمهاي بانكهاي اطلاعاتي رابطه اي به كار ميرود . دستورات SQL بيشتر براي انجام عملياتي همچون بروز رساني و يا پاك كردن و اضافه كردن مطالب به بانك اطلاعاتي است . بعضي نرم افزارهاي مديريت بانك اطلاعاتي همچون Ingres , Sybase , SQL Server , Oracle , Informix , DB2 , Access و غيره از اين زبان استفاده ميكنند . البته هر كدام از اين سيستمها داراي SQL با مشخصات خاص خودشان هستند اما بعضي دستورات همچون Create , Delete , Update , Insert , Select و Drop تقريبا در تمامي اينها با كمي تفاوت مشابه است . سعي ميكنم تا حد توان ا انجايي كه بتوانم وارد اين دستورات شده و تك تك انها را بررسي كنم .
SQLميتواند ركوردها را از جدولي پاك كند يا اضافه كند . ميتواند همانطور كه گفتم جداول را بروز كند . و ميتواند ازجداول كپي برداري كند . به هر حال فراگيري SQL ساده و اسان است .

جدول چيست ؟ TABLE

يك سيستم مديريت بانك اطلاعاتي رابطه اي از چندين شئ تشكيل ميشود كه يكي از اين اشياء مهم جداول هستند . داده ها و اطلاعات بانك اطلاعاتي در اين جداول ضبط ميشوند . جداول همانطور كه از نامشان پيداست مشخصند كه چه هستند . هر جدول داراي سطرها و ستونهايي است . ستونهاي جدول داراي نام , نوع و ديگر خصيصه ها attribute هستند . در اين جداول كليه عناصر ستون از يك نوع هستند . مثلا همگي با هم عدد هستند يا متن يا هر چيز ديگري . رديفها يا ركوردها كه سطرهاي جدول را تشكيل ميدهند متشكل از كليه ستونهاي ان جدول در يك سطر هستند در واقع مجموعه اطلاعاتي كه مرتبط با هم هستند در يك سطر اورده ميشوند . جدول زير را ببينيد .
Lastnameرا يك ستون گوييم كه بر هر يك از سلولهاي اين ستون يك فيلد ميگوييم Firstname . و Addressو city نيز هر كدام يك ستون هستند .

این جدول سه سطر دارد که مشخصات سه فرد را به ما میدهد .که هر کدام از این سطرها از مجموعه ستونها تشکیل شده است .

Code:
```
LastName | firstName |Address | City
vahid | imani | Tehran 10 |Tehran
arash | vishoniee | roodehen 23 |Tehran
amir | matrixi | karaj20 |isfahan
```

دستور SELECT

این دستور براي پرس و جو از بانك اطلاعاتي به كار ميرود . در اين دستور داده اي را از جدول انتخاب كرده و نمايش ميدهيم . اين دستور با شرايطي كه نويسنده دستور براي ان ذكر ميكند به كار ميرود :
نوع قالب اصلي اين دستور به اين شرح است :

Code:
```
select "column1"
[,"column2",etc]
from "tablename"
[where "condition"];
```

عباراتيكه در قالب فوق درون كروشه ميبينيد بيانگر اختياري بودن دستورات هستند . در دستور فوق به جاي column از نام ستون استفاده ميكنيد . اگر خواستيد كليه ستونهاي يك جدول را بازيابي كنيد كافيست بجاي نوشتن تك تك ستون ها از * استفاده كنيد .
بعد از دستور from نام جدولي كه ميخواهيم فيلد مربوطه را بازيابي كنيم مينويسيم . براي where ميتوان شرطي گذاشت كه از عملگرهاي Between , LIKE , = , <> , <= , >= , < , > استفاده كند .

در بعضي بانكهاي اطلاعاتي بايد از سميكالن ; در انتهاي دستور استفاده كرد . اما استفاده از ان در بانكهاي اطلاعاتي چون Access و SQL server اختياري است .
SQL Data Manipulation Language (DML)
SQLداراي دستوراتي براي ساخت گزارشهاست . اما زبان SQL شامل دستوراتي است كه براي بروزرساني و قراردادن پاك كردن دكوردها به كار ميروند . اين دستورات در دسته DML ها هستند . كه براي دستكاري داده ها به كار ميروند .
SELECT , UPDATE , DELETE , INSERT INTOدر اين دسته هستند .

SQL Data Definition Language (DDL)
بخشي از دستورات SQL كه براي ساخت و يا پاك كردن جداول به كار ميروند در اين دسته قرار ميگيرند . دستورات مهم اين بخش شامل Create Table , ALTER TABLE , DROP TABLE , VREATE :
INDEX , DROP INDEXميباشد .
استفاده از دستور DISTINCT در دستور select باعث ميشود انتخابات ما همگي بصورت منحصر بفرد در خروجي نمايش داده شود . فرضا دستور زير :

Code:
```
SELECT DISTINCT Company FROM jadvalname
```

اين دستوراز كليه ركوردهاي جدول تنها يك نمونه نشان ميدهد در خروجي ركورد تكراري نخواهيم داشت ..
بنابراين چند مثال از دستورات ساده select مي اورم :

Code:
```
select column1 from jadval1 ;
```

اين دستور column1 را از جدول jadval1 استخراج ميكند و به شما نمايش ميدهد .

Code:

```
select DISTINCT column1 from jadval1 ;
```

اين دستور كليه ركوردهاي فيلد column1 را از جدول jadval1 استخراج ميكند منتهي بصورت منحصر فرد و ركوردهاي تكراري را نشان نميدهد . .

Code:
```
select DISTINCT column1,column2 from jadval1 ;
```

اين دستور كليه محتويات دو فيلد column1 و column2 را از جدول jadval1 را استخراج ميكند بصورت منحصر بفرد

Code:
```
select cloumn1,column2 from jadval2 ;
```

ستونهاي column1 , column2 را از جدول jadval2 استخراج ميكند .

Code:
```
select * from jadval2
```

كليه ستونهاي جدول jadval2 را نمايش ميدهد . در واقع همان جدول را بدون تغيير نمايش ميدهد .

Code:
```
select firstname, lastname from student ;
```

ستونهاي firstname و lastname را از جدول student استخراج ميكند . در واقع نام و نام خانوادگي دانش اموزاني كه اطلاعات انها در student است را نمايش ميدهد .

دستور: where

هر وقت از دستور select استفاده كرديد ميتوانيد از دستور Where براي شرط گذاري بر دستور select نيز استفاده كنيد .

Code:
```
SELECT column FROM table
WHERE column operator value
```

عملگرهاي where بترتيب تساوي = عدم تساوي <> بزرگتر < كوچكتر > و كوچكتر مساوي =< و بزرگتر مساوي => و بالاخره between و like ميباشند . توجه داشته باشيد كه در بعضي زبانهاي SQL بجاي <> از != استفاده ميكنند .
مثلا براي انتخاب كردن فردي از جدول persons كه در شهر تهران زندگي ميكند . كه البته در ان جدول ستوني بنام city وجود دارد از

Code:
```
SELECT * From Persons where City='Tehran'
```

از تك كوتيشن به ان سبب كه نوع فيلد city از نوع text است استفاده كرديم .
در بعضي نسخه هاي SQL از جفت كوتيشن استفاده ميشود اما يادتان باشد كه اگر نوع فيلد ما عددي بود نبايد از كوتيشن استفاده كنيم

Code:
```
SELECT * from persons where year >1984
```

اين دستور غلط است :
Code:

```
SELECT * from persons where year>'1984'
```

براي LIKE ميتوان از % يا * استفاده كرد . اين دو علامت يعني انكه هر چه بعد از ان يا هر چه قبل از ان (كاراكتر) بود مثال زير را ببينيد

```
Code:
SELECT * from persons where firstname LIKE '%a'
select * from persons where firstname like '*a*'
```

عنوان: **دستورات UPDATE وDELETE**

پس از وارد كردن اطلاعات در بانك اطلاعاتي حتما برايتان پيش خواهد آمد كه بخواهيد محتويات فيلدي را تغيير دهيد . براي اينكار بايد از دستور UPDATE به صورت قالب زير استفاده كنيد :

```
Code:
UPDATE table_name
SET column_name = new_value
WHERE column_name = some_value
```

فرض كنيد در جدولي به نام students ميخواهيم نام دانشجويي كه فاميلي آن aghamohammadi هست را به vahid تغيير دهيم . اگر نام فاميلي ها در فيلد lname و نام كوچك در فيلدي به نام fnameباشد انگاه دستور به صورت زير خواهد بود :

```
Code:
UPDATE STUDENTS
SET FNAME="VAHID"
WHERE LNAME="AGHAMOHAMMADI"
```

اما با استفاده از قالب كه در زير نشان خواهم داد ميتوانيد در ان واحد چند فيلد را همزمان UPDATE كنيد :

```
Code:
UPDATE TABLE "table_name"
SET ("column_1", "column_2") = ([new value 1], [new value 2])
WHERE {condition}
```

در دستور UPDATE مجاز خواهيد بود كه از زيرپرس وجو استفاده كنيد :

```
Code:
UPDATE supplier
SET supplier_name = ( SELECT customer.name
FROM customers
WHERE customers.customer_id = supplier.supplier_id)
WHERE EXISTS
( SELECT customer.name
FROM customers
WHERE customers.customer_id = supplier.supplier_id);
```

براي پاك كردن ركوردها از جدولي خاص نيز از دستور DELETE استفاده ميكنيم كه قالب آن به صورت

زير است :

Code:
```
DELETE FROM "table_name"
WHERE {condition}
```

عنوان: **دستور INSERT TABLE ساخت جدول**

Code:
```
INSERT TABLE
```

اين دستور جزو دستورات DML است
اگر فرض كنيم كه جدول خود را با دستور CREATE TABLE درست كرده ايم . با استفاده از دستور INSERTميتوانيم ان جدول را با داده ها پر كنيم .
نكته حائز اهميت و سوال برانگيز اين بخش انستكه دستورات SQL خاصيت CASE SENSITIVE را ندارند . اما هنگامي كه با استفاده از دستوراتي اينگونه داده اي را واردجدول ميكنيد . ان داده ها بنا به انچه شما وارد كرده ايد به جدول ميروند و خاصيت CASE SENSITIVE را دارند .
قالب اصلي دستور iNSERT چيزي شبيه به قالب زير است :

Code:
```
INSERT INTO TABLE_NAME
VALUES ('vALUE1"VAlue2' , [NULL]);
```

طبق انچه در ظاهر قالب فوق ميبينيد . فيلدهاي جدول در يك ركورد با استفاده از دستور فوق پر ميشوند . طبق قانوني كه ميدانيم اگر از داده هاي كاراكتري و تاريخ استفاده ميكنيم بايد اين مقادير بين علامت نقل قول (كوتيشن) قرار بگيرند . اما اگر مقدار پوچ يا عددي وارد ميكنيد نبايد انرا بين علامت كوتيشن قرار دهيد .
براي وارد كردن مقدار پوچ كافيست از كلمه كليدي NULL بدون كوتيشن يا از دو كوتيشن بدون هيچ فاصله و يا كاراكتري استفاده كنيد ^ بصورتي كه ميبينيد . ''
ترتيب قرارگيري فيلدهايي كه قرار است پر شوند نيز مهم است . يعني مقاديري كه با دستور iNSERT وارد جدول ميشوند بترتيب ورود در ركورد جدول بر اساس آرايشي كه در ساختار جدول فيلدها قرار گرفته اند قرار ميگيرند .
اما قالب كلي تر ديگري داريم كه در آن ذكر نام فيلدها نيز مي آيد كه قالب ان بشكل زير است :

Code:
```
INSERT INTO table
(column-1, column-2, ... column-n)
VALUES
(value-1, value-2, ... value-n);
```

نكته اي كه در قالب فوق بايد تذكر داده شود انستكه : نيازي نيست نام تمامي فيلدها وارد دستور شود بلكه تنها اندسته از فيلدهايي كه قرار است پر شود كافيست) البته يادتان باشد كه ستونهايي كه نبايد مقدار پوچ داشته باشند را در دستور فوق از قلم نياندازيد) در ضمن ترتيب ستونها در اين دستور مهم نيست . يعني مانند قالب اولي كه آورده شد و ترتيب ستونها مهم بود . در اين دستور ترتيب ستونها مهم نيست .
مثال زير را ببينيد :

Code:
```
INSERT INTO supplier
(supplier_id, supplier_name)
```

```
VALUES
(24553, 'IBM');
```

در مثال فوق در جدولي به نام supplier در فيلدهاي supplier_id و supplier_name بترتيب مقدار عددي ۲۴۵۵۳ و مقدار كاراكتري IBM را قرار داديم .
در استفاده از دستور INSERT INTO شايد بخواهيد داده هايي را ازجدول ديگري وارد جدول فعلي خود بكنيد . كه در اين صورت در قالب دستور فوق از يك SUBQUERY استفاده ميكنيم . در اين زير پرس وجو با استفاده از دستور SELECT ميتوان از جدول ديگري اطلاعات را وارد كرد .

Code:
```
INSERT INTO TABLENAME(‘COLUMN1’ , ‘COLUMN2’ , ‘COLUMNN’)
SELECT [*|FIELDNAMES] FROM ANOTHER_tbl [WHERE CONDITION][AND ANY OTHER
FRAMES OF SELECT PROMPT]
```

اگر نميدانيد دستور SELECT چيست ميتوانيد به بخش مربوطه مراجعه كنيد .
مثال زير را ببينيد :

Code:
```
INSERT INTO supplier
(supplier_id, supplier_name)
SELECT account_no, name
FROM customers
WHERE city = 'Newark';
```

در جدول supplier در فيلدهاي supplier_id و supplier_name بترتيب فيلدهاي account_no و name را از جدول customers ا وارد ميكنيم كه نام شهر انها Newark است . در استفاده از زيرپرس و جو ها يادتان باشد كه نوع فيلدها نيز بايد با هم يكسان باشند .
در موارد بسياري پيش مي ايد كه فرد فراموش ميكند كه آيا با استفاده از دستور insert مقادير مورد نظرش را وارد كرده است يا خير ؟ !
كه در اين موارد بايد از دستور شرطي exist كه بعدا مفصلا راجع به ان نيز بحث ميكنيم استفاده كنيد .
مثال زير را ببينيد :

Code:
```
INSERT INTO clients
(client_id, client_name, client_type)
SELECT supplier_id, supplier_name, 'advertising'
FROM suppliers
WHERE not exists (select * from clients
where clients.client_id = suppliers.supplier_id);
```

يا اگر خواستيد تنها يك مقدار را ببينيد كه وارد شده است يا نه به صورت زير دستور فوق را تغيير دهيد :

Code:
```
INSERT INTO clients
(client_id, client_name, client_type)
SELECT 10345, 'IBM', 'advertising'
FROM dual
WHERE not exists (select * from clients
where clients.client_id = 10345);
```

عنوان: **ترکیب جداول در دستورSELECT**

Code:
```
SELECT stno,Avg(grade) FROM grade GROUP BY stno HAVING count(*)>2
```

دستور فوق باعث ميشود كه
.1فيلد يا ستون يا همان خصيصه كه هر سه يكي هستند كه براي راحتي كار از نام فيلد استفاده ميكنيم . بنابراين فيلد stno را از جدول grade انتخاب ميكند .
2. avg(grade)يكي از توابع جمعي است وظيفه اين تابع خروجي دادن ميانگين مقداريست كه درون اين تابع قرار ميگيرد در صورت بالا داخل اين تابع از grade استفاده كرده ايم يعني ميانگين نمرات يا gradeها را حساب كن .
.3با استفاده از group by كه در اين دستور استفاده از ان بدليل حضور يك تابع جمعي و نام فيلد با يكديگر گروه بندي بايد بر اساس Stno قرار بگيرد . بنابراين ميانگين بر اساس شماره دانشجويي كه همان stno است انجام ميپذيرد .
.4استفاده از دستور HAVING منجر به ورود شرط در دستور فوق ميشود اين شرط مرتبط با دستور group byاست . اين شرط باعث ميشود كه عمليات تابع avg محدود شود . اگر فرض كنيم كه چيزي به نام تابع جمعي نداشتيم ميتوانيم انرا با where مقايسه كنيم چراكه دقيقا مانند Where به اعمال شرط در دستورات ميپردازد . شرط ورودي بر اين دستور coun(*)>2 است بنابراين تعداد كليه ركوردها شمرده ميشود ركوردي كه تكرار ان بر اساس stno بيشتر از 2 مورد بود در خروجي چاپ ميشود . يعني ميانگين نمراتي از دانشجويان را ميبينيم كه در جدول حداقل ۳ نمره دارند .
بنابراين ميتوانيد بعد از شرط having دوباره از يكي از توابع شرطي استفاده كنيد . فرضا براي مثال فوق اگر خواستيد معدل دانشجوياني را حساب كنيد كه نمرات انها بيشتر از ۱۵ بوده است ميتوانيد از max(grade)>15استفاده كنيد .

سوال: تقاضايي بنويسيد كه شماره دانشجويي stno و نام fname و نام خانوادگي lname و نمرات gradeهر دانشجو را در خروجي نمايش دهد !

در مرحله اول ظاهرا دستور به شرح زير است :

Code:
```
select stno,fname,lname,grade from student,grade
```

دستور فوق يك تفاوت اساسي با ديگر دستورات sql كه تا كنون استفاده كرده بوديم دارد و ان اينكه بعد از form به جاي نام يك جدول نام دو جدول امده است ! و اين به ان دليل است كه در جدول student فيلدي براي grade نداشته ايم بنابراين ان را از جدول grade انتخاب كرده ايم . اول از همه روشن كنيم كه مشكل اساسي دستور زير چيست !
دستور فوق همانطور كه ميبينيد از دو جدول student و grade استفاده كرده است اما مشكل اينجاست كه فيلدهايي كه بعد از دستور select انتخاب كرده ايم همگي در جداول مربوط به خودشان قابل تشخيص هستند جز Stno ميدانيم كه فيلد Stno هم در جدول student قرار دارد هم در جدول grade بنابراين بايد مشخص كنيم كه اين stno مربوط به كدام يكي از اين جداول است پس دستور فوق را به صورت زير تصحيح ميكنيم :

Code:
```
select student.stno,fname,lname,grade from student,grade
```

براي مشخص كردن انكه stno مربوط به كدام جدول است با اوردن نام جدول قبل از نام stno و يك نقطه بلافاصله بعد از ان مشخص شد كه stno از جدول student است .
اما مشكل دومي كه دستور فوق دارد انستكه دستور فوق تنها يك خروجي ساده از فيلدهاي انتخابي بيرون ميدهد . ميدانيد كه دستور Select تنها وظيفه انتخاب فيلدها از جداول مربوطه را دارد اينكه اين فيلدها چگونه به يكديگر مرتبط هستند و ايا اينكه ترتيب اين فيلدها بر اساس ركورد هاي جدول است يا خير بر عهده اين دستور نيست . بنابراين بايد به سراغ دستورات شرطي رفت !
براي انكه مشخط كنيم كه كدام دانش اموزان كدام نمرات را گرفته اند بايد محدوديتي اعمال كنيم بر اينكه چون جدول grade يك جدول جداگانه است بنابراين بايد به طوري ترتيبي بين اين دو ايجاد كرد .
ترتيب را ميتوان از همان مشكل اول كه گفته شد به دست اورد اينكه : همانطور كه ديديد فيلد مشترك اين دو جدول stno بود بنابراين ميتوانيم شرطي بگذاريم كه تنها فيلدهايي را نشان بدهد كه شماره دانشجويي stno انها يكسان است ! اين يكساني منجر به ان ميشود كه :

.1ميدانيم Stno در جدول student يك كليد اصلي است بنابراين هر شماره دانشجويي تنها يك نام و نام خانوادگي دارد .
.2در مورد جدول grade ميدانيم نمرات هر كدام به شماره هاي دانشجويي مرتبط با خودشان مرتبط هستند بنابراين هر شماره دانشجويي تنها نمرات خاص خودش را دارد .
حال براي انكه در خروجي ما نمرات و شماره دانشجويي و نام خانوادگي همگي با همديگر مرتبط باشند بايد مقايسه اي بين اين دو جدول انجام دهيم مقايسه به اينصورت است شماره دانشجويي هر جدول با جدول ديگر چك ميشود در صورتي كه شماره دانشجويي ها برابر بود محتويات با يكديگر تركيب ميشوند و بصورت ركوردهايي از جدول در خروجي چاپ ميشوند .
دستور صحيح سوال فوق به صورت زير است

Code:
SELECT student.stno,fname,lname,grade from student,grade where student.stno=grade.stno

اگر كمي در توضيحات فوق بانديشيد ميتوانيد به اين نكته پي ببريد كه عملا چه اتفاقي مي افتد . در واقع ما با شرط فوق نوعي رابطه بين دو جدول ايجاد كرده ايم . رابطه ما به اين صورت است كه هر گاه stnoاز جدول رئيس كه همان stnudent باشد با stno از جدول grade مطابقت داشت خروجي ها را بر اساسي كه خواسته شده است نمايش بده
اگر به تشريح دستور فوق بپردازيم متوجه ميشويد كه دستور تنها زماني اجرا ميشود كه شماره دانشجويي هر دو جدول يكي باشد .
فرض كنيم شماره دانشجويي ۱۰ نام و نام خانوادگي خود را در جدول student دارد حالا اين شماره دانشجويي با تك تك شماره دانشجويي هاي موجود در جدول grade چك ميشود در صورتيكه به اولين تساوي مقداري رسيد) يعني به محض رسيدن به اولين شماره دانشجويي كه مطابق با شماره ۱۰ است) شروع به Select كردن يا انتخاب كردن فيلدهايي كه در جلوي دستور select امده است ميكند يعني student.stno را به همراه fname و lname و grade از جدول student به خروجي چاپ ميكند .
سپس به ساغ مقادير بعدي Stno ميرود كه بايد مقايصه شوند .
تذكر : براي چاپ خروجي همانطور كه ميبينيد stno جلوي دستور Select مربوط به جدول student است بنابراين عملا دو شماره دانشجويي يكسان در كنار هم در ركورد نخواهيم داشت .
سوال ۲ : شماره دانشجويي نام و فاميلي و دروسي كه دانشجويان گرفته اند را نمايش دهد !
سوال از ما چند فيلد ميخواهد كه در خروجي نمايش دهيم فيلدها به ترتيب stno و fname و lname و nameهستند name نام درسي است كه دانشجو گرفته است .

Code:
SELECT stno,fname,lname,name from ????

مرحله اول طي شده است يعني ما فيلدها را انتخاب كرده ايم اما بايد ديد اين فيلدها از كدام جدول انتخاب ميشوند . جدول اولي كه حاوي شماره دانشجويي و نام و نام خانوادگي است جدول student است و جدول دومي كه از ان name يا همان نام درس را گرفته ايم جدول course است . بنابراين داريم

Code:
SELECT stno,fname,lname,name FROM student,course

تا اينجا اگر دستور را اجرا كنيم چون از دو جدول متمايز استفاده كرده ايم اطلاعات ما بالطبع طبق همانچه قبلا براي مثال بالا توضيح داده شد بدون هيچ ترتيبي و بدون هيچ ارتباطي است !
بنابراين بايد ارتباطي بين اين دو جدول براي ترتيب دهي پيدا كنيم .
خوب فيلد مشتركي بين اين دو جدول نداريم . اما ميدانيم از طريق جدول ديگري ميتوانيم فيل مشتركي بين اين دو جدول پيدا كنيم . جدول مشترك ما جدول grade است چون ميدانيم دانشجويي كه درسي را برداشته حتما كد درسي كه به صورت كليد اصلي در ان جدول است براي وي مقداري فرض كرده است . بنابراين جدول ديگري را بايد در دستور فوق وارد كنيم .
دستور به صورت زير عوض ميشود

Code:
SELECT stno,fname,lname,name FROM student,course,grade

همانطور كه در دستور فوق ميبينيد جدول grade را نيز به بازي وارد كرديم اما همانطور كه ميدانيم چون در دستور select فيلدي از ان انتخاب نشده است تاثيري بر انچه در خروجي خواهيم داشت بصورت عيني نخواهد داشت . اما به محض وارد كردن جدول grade متوجه اشتراك اين جدول با جدول student

از طريق فيلد تكراري Stno خواهيم شد .
خدايا چه كنم !؟
كافيست همانطور كه در مثال قبل توضيح داده شد قبل از فيل انتخابي stno در دستور Select نام جدول مربوط به اين فيلد را بياوريم :

Code:
SELECT student.stno,fname,lname,name FROM student,course,grade

اين هم از اين ! اما كار به پايان نرسيده است ما تازه جدول grade را براي ايجاد نوعي اشتراك يا ارتباط بين دو جدول student و course به بازي اورده ايم . بنابراين اشتراك دو جدول قديمي ما مهيا شده است ! به اين صورت كه جدول student از طريق فيلد Stno با جدول grade در ارتباط است و جدول gradeاز طريق courseno با جدول course در ارتباط است .
بنابراين نوبت به اعمال محدوديت ها ميرسد !
براي اعمال محدوديت بايد اينگونه عمل كنيم كه هرگاه Stno از جدول student با stno از جدول grade هماهنگ شد يا برابر شد سراغ جدول course رفته و فيل هاي courseno با يكديگر چك شوند در صورت ايجاد تساوي بين اين دو نام روبروي نام و نام خانوادگي چاپ شود . اعمال اين ايجاد تساوي را قبل از انكه به جواب نگاه كنيد . ميتوانيد حدث بزنيد ...

Code:
SELECT student.stno,fname,lname,name FROM student,course,grade WHERE student.stno=grade.stno and course.courseno=grade.courseno

Code:
SELECT student.stno, fname, lname, name
FROM student, course, grade
WHERE student.stno=grade.stno and course.courseno=grade.courseno;

عنوان: **دستور ALTER TABLE براي اعمال تغييرات در جداول**

Quote:
ALTER TABLE

اين دستور همانطور كه ميدانيد جزو دستورات DDL ميباشد .
پس از ساخت جدول و ورود داده ها به آن شايد لازم باشد پس از طي شدن زماني و استفاده از جدول متوجه شويد كه جدول شما ستوني كم دارد ! يا انكه نوع داده هاي ستوني از جدول شما مناسب ان داده ها نميباشد . مثلا ابتدا در هنگام ساخت جدول ستوني بنام GRADE را با نوع داده صحيح انتخاب كرده ايد . اما بعدا متوجه شديد كه اين مقدار براي اين ستون كافي نيست. بنابراين نياز داريد به انكه نوع فيلدهاي جدول را تغيير دهيد .
البته اين دستور تنها براي تغيير دادن در نوع فيلد نيست كه براي تغيير اندازه سايز فيلد . تغيير نوع . يا انكه فيلد شما ميتواند مقدار تهي داشته باشد يا خير و تعدادي ديگر از تغييرات اين دستور مورد استفاده قرار ميگيرد . حتي شايد بخواهيد ستوني را حذف يا اضافه كنيد .
شماي اصلي دستور به صورت زير است :

Code:
ALTER TABLE table {ADD {COLUMN field type[(size)] [NOT NULL] [CONSTRAINT index] |
ALTER COLUMN field type[(size)] |
CONSTRAINT multifieldindex} |

DROP {COLUMN field I CONSTRAINT indexname} }

در قالب فوق :
tableنام جدولي است كه قرار است تغييرات بر ان اعمال شود .
fieldنام فيلدي است كه قرار است در جدول اضافه پاك يا تغييرات بر ان اعمال شود .
typeنوع فيلد مورد نظر است كه ميتواند عددي يا رشته اي و ياهر يك از انواع استاندارد داده اموزش داده شده دربخش قسمت دوم : انواع داده ها در SQL باشد .
Sizeسايز فيلدهاي كاراكتري و باينري كه تنها در اين دو نوع داده قابل استفاده است .
indexشاخصي كه قرار است براي اين فيلد در نظر گرفته شود .
multifieldindexشاخصهايي كه قرار است براي چندين فيلد در نظر گرفته شود .
indexnameنام ايندكسي كه قرار است از چندين فيلد پاك شود
يك مثال ساده به همراه قالب دستور :

Code:
```
ALTER TABLE table_name
ADD column_name column-definition;
```

فرضا :

Code:
```
alter table student add studet_test1 varchar(250)
```

براي افزودن چندين ستون :

Code:
```
ALTER TABLE table_name
ADD ( column_1 column-definition,
column_2 column-definition,
...
column_n column_definition );
```

براي تغييرات نوع دستور :

Code:
```
ALTER TABLE table_name
MODIFY column_name column_type;
```

مثال :

Code:
```
ALTER TABLE student
MODIFY stno varchar2(100) not null;
```

براي چندين ستون :

Code:
```
ALTER TABLE table_name
MODIFY ( column_1 column_type,
column_2 column_type,
...
column_n column_type );
```

براي تغيير نام يك ستون :

Code:

```
ALTER TABLE table_name
RENAME COLUMN old_name to new_name;
```

در مثال زير يك كليد خارجي به جدول student با نام test_fk ايجاد ميكنيم كه با فيلد test_pk از جدول gradeارتباط برقرار ميكند : .

Code:
```
ALTER TABLE student ADD CONSTRAINT test_fk FOREIGN KEY (test_pk) REFERENCES grade
(test_pk);
```

حال همين فيلد را پاك ميكنيم :

Code:
```
ALTER TABLE Students DROP CONSTRAINT test_fk;
```

اگر بخواهيد با برنامه اي اينكار را بكنيد فرضا با visual basic از دستور زير استفاده ميكنيد :كه فقط يك مثال است :

Code:
```
Sub Create_Table_Script()

Dim db As Database

Set db = OpenDatabase("yourcustomers.mdb")

'alter employees table to add foreign key reference as above

db.Execute "ALTER TABLE student ADD CONSTRAINT test_fk FOREIGN KEY (test_pk)
REFERENCES grade (test_fk);"

db.Close
End Sub
```

در دستور فوق سه گزينه اختياري داريم . اولي MODIFY براي اعمال تغييرات در نوع جدول . و گزينه دومي ADD براي اضافه كردن ستوني به جدول و همچنين DROP براي پاك كردن ستون .
مثالي براي تغيير نوع فيلد STNO را ميبينيم :

Code:
```
ALTER TABLE STUDENT MODIFY STNO VARCHAR(12);
```

در استفاده از دستور Add يادتان باشد كه اگر داده اي از قبل در ستون وارد كرده ايد نوع اين ستون نميتواند not null باشد بنابراين بايد ابتدا انرا از نوع null تعريف كنيد . سپس انرا با داده هايي پر كنيد و سپس انرا به نوع not null تغيير دهيد .
باز هم مثال :

Code:
```
ALTER TABLE Employees ALTER COLUMN Emp_Email TEXT(50);
```

عنوان: **دستور ساخت جدولCREATE TABLE**

هر بانك اطلاعاتي متشكل از اشيايي ميباشد كه از انها براي ذخيره سازي يا ارجاع به داده ها استفاده

ميكند . جداول از مهمترين اشياي بانكهاي اطلاعاتي به شمار ميروند .
Schemaنيز يكي از اشياي بانك اطلاعاتي است . به مجموعه اي از اشيايي كه مرتبط با يك نام كاربري در يك بانك اطلاعاتي موجود است گفته ميشود . بنابراين هر كاربري كه شيئي را در بانك اطلاعاتي توليد ميكند در واقع شماي خود را ايجاد و تكميل ميكند . فرض كنيم كه كاربري با نام كاربري user01 در بانك اطلاعاتي login ميكند . پس از ان يك جدول با نام tbl_parsx ميسازد . اين جدول در كل بانك اطلاعاتي با نام user01.tbl_parsx شناسايي ميشود . اما براي خود كاربر نيازي نيست كه از كلمه user01در ابتداي نام جدول استفاده كند .
اما براي ساخت يك جدول در بانك اطلاعاتي با استفاده از دستورات SQL بايد از خودمان سوالاتي بپرسيم از جمله انكه :
نام جدول را چه خواهيد گذاشت ؟
چه تعداد ستون در جدول خود خواهيد داشت ؟
ستون هاي مربوطه چه نامي خواهند داشت ؟
در اين ستون ها چه نوع داده هايي بايد تعريف شود ؟
طول ستونهاچه مقدار بايد باشد ؟
كدام يك از ستون ها بايد كليد اصلي جدول ما باشد ؟
آيا ستوني ميتواند خالي بماند ؟
پس از انكه پاسخي براي كليه سوالات فوق پيدا كرديد ميتوانيد با قالب دستور زير جدول خود را بسازيد :

Code:
```
CREATE TABLE table_name(
column1 datatype [NOT NULL] [primary key],
column2 datatype [NOT NULL],
.
.
columnnameN datatype [NOT NULL]);
```

در دستور فوق مقاديري كه داخل كروشه قرار دارند اختياري هستند و ميتوانند نباشند . اگر NOT NULL را در هر يك از ستونهاي فوق نگذاريم بصورت پيشفرض انرا NULL در نظر ميگيرد . در ضمن در دستور فوق تنها ميتوانيم يك كليد اصلي براي جدول داشته باشيم . لزومي ندارد كه كليد اصلي مختص ستون اول باشد بلكه ميتوانيد ان مقدار را برا ستونهاي بعدي در نظر بگيريد .
در مثال زير جدول STUDENT را با دستور CREATE TABLE بازسازي ميكنيم :

Code:
```
CREATE TABLE STUDENT11(
stno char(8) primary key,
fname char(20),
lname char(50),
id char(5));
```

همانطور كه ميبينيد در انتهاي دستور CREATE TABLE از سمي كالن استفاده كرديم . اين علامت بيانگر به پايان رسيدن دستور SQL است كه در بعضي از بانكهاي اطلاعاتي متفاوت است . فرضا در TRANSACT-SQLاز GO استفاده ميشود . در بانك اطلاعاتي ACCESS هم ميتوانيد اين مقدار را اصلا نگذاريد . كه در اينصورت خود ACCESS اين مقدار را قرار ميدهد .
در بعضي از نسخه هاي SQL ميتوان در ادامه دستور CREATE TABLE دستوري به نام STORAGE را اضافه كنيد كه اين دستور دو مقدار را بدنبال خود دارد مقدار اوليه INITIAL مقدار فضايي كه در ابتداي تشكيل جدول لازم است توليد ميكند . و مقدار بعدي در صورتي كه فضاي اوليه اي كه با دستور INITIALدر نظر گرفته شده بود پر شود اين مقدار فضا را اضافه ميكند :

Code:
```
CREATE TABLE PARSX( ...
...
...)
STORAGE
( INITIAL 4K
```

```
NEXT 2K );
```

اما همانطور كه ديديد براي ساخت كليد اصلي با استفاده از دستور PRIMARY KEY تنها ميتوانستيم يك كليد اصلي داشته باشيم . استفاده از كليد اصلي باعث ميشود كه مقادير ما در يك جدول مقدار منحصر بفردي در ان ستون داشته باشند . يعني دو ركورد در ستون كليد اصلي نميتواند برابر باشد .
اين مثال را ميتوان براي فرضا كد ملي يا شماره دانش اموزي در نظر گرفت كه هر فردي تنها ميتواند يك كد ملي ان هم متفاوت با بقيه داشته باشد .
گاهي لازم است در جدول بيش از يك كليد اصلي داشته باشيم . بنا براين براي اين كار به جاي استفاده از PRIMARY KEY از دستور UNIQUE استفاده ميكنيم. با استفاده از UNIQUE ميتوانيم فيلدهاي ديگر مورد نظر را نيز منحصر بفرد كنيم .
به هر حال هر گاه خواستيد خواصي از كليد اصلي را به جدول خود اضافه كنيد ميتوانيد از دستورات indexبراي انجام اين كار استفاده كنيد .
علاوه بر كليد اصلي در اين دستور ميتوانيد كليد خارجي را نيز تعيين كنيد . با تعيين كردن اين كليد ميتوانيد ارتباط بين جداول را ايجاد كنيد . كافيست از دستور زير براي ايجاد كليد فرعي استفاده كنيد :

Code:
```
CREATE TABLE table_name(
column1 datatype [NOT NULL] [primary key],
column2 datatype [NOT NULL],
.
.
columnnameN datatype [NOT NULL])
FOREIGN KEY STD_ID_FK (EMP_ID) REFRENCES GRADE (STNO)
;
```

با استفاده از زير پرس و جو يا SUB QUERY ميتواند جدول مشابهي را كه ساختار يكساني با جدول درست شده دارد درست كرد :

Code:
```
CREATE TABLE NAME AS SELECT * FROM PARSX
```

دستور فوق در بعضي از نسخه هاي SQL متفاوت است...

عنوان: **انواع داده ها درSQL**

داده ها و انواع آن در : SQL

داده ها به اطلاعاتي گفته ميشوند كه در يكي از انواع استاندارد تعريف شده در بانك اطلاعاتي ذخيره ميشوند . همانطور كه از نام داده بر مي ايد . داده به هر ان چيزي گفته ميشود كه كاربر در بانك اطلاعاتي وارد ميكند كه از جمله : نام . عدد و هر متن كه ميتواند تركيبي از اعداد و حروف باشد و حتي گرافيك و هر چيزي كه به ذهنتان ميرسد ميتواند داده باشد . اين داده ها در بانك اطلاعاتي ميتوانند پردازش شوند و يا حتي تغيير و ويرايش شوند .
انواع داده ها در بانك هاي اطلاعاتي براي مشخص كردن نوع فيلد به كار ميروند كه الزاما پس از تعيين نوع فيلد تمامي داده ها در ان فيلد يا ستون از بانك اطلاعاتي از همان نوع داده بايد باشند . داده ها در بانك اطلاعاتي به سه نوع اصلي : متن و اعداد و تاريخ يا زمان دسته بندي ميشوند .
و اما هر يك از سه نوع فوق انواعي دارند كه به انها اشاره ميكنيم :

نوع كاراكتر هاي ثابت :
اين نوع داده ها براي ذخيره رشته كاراكترها و حتي اعداد و يا تركيبي از انها استفاده ميشود طرز تعريف انها نيز به اين گونه است كه ابتدا كلمه char و سپس عددي كه حداكثر طول اين نوع رشته ها را

مشخص میکند وارد میکنیم :

Code:
CHAR (N)

البته در بانکهاي اطلاعاتي به صورت استاندارد از

Code:
CHARACTER (N)
استفاده میشود .

این نوع داده ها به این ترتیب هستند که فرضا اگر حداکثر طول انها یعني N=10 باشد هر مقداري که شما بعنوان ورودي به انها بدهید به شرطي که کمتر از ۱۰ باشد را قبول میکنند . فرض کنیم داده اي پنج کاراکتري میدهیم که در این صورت این پنج کاراکتر در این نوع داده ذخیره میشود و پنج کاراکتر باقیمانده با SPACE پر میشود . معایبي که این نوع داده ها دارند انستکه اگر بصورت نادرستي از انها استفاده شود فضاي زیادي را بیهوده از بانك اطلاعاتي شما میگیرد که این خود يك نقطه ضعف است . فرضا میتوان از این نوع داده ها براي طولهاي با مقدار ثابت استفاده کرد مانند شماره ملي . توصیه میکنم براي مقادیري همچون نام افراد که میتواند طول متغیري داشته باشد از این نوع استفاده نکنید

نوع کاراکترهاي با طول متغیر :

این نوع کاراکترها بر خلاف نوع فوق میتوانند داده ها با طول متغیري را بدون اتلاف حافظه ذخیره کنند . که طرز اعلان ان بصورت
Code:
VARCHAR (N)

یا بصورت استاندارد بصورت :

Code:
CHARACTER VARYING (N)

میباشد که N عددي است که نشانگر بیشینه طول رشته کاراکتري شما در ان فیلد است .
البته نوع دیگري بنام VARCHAR2 نیز هست که همان VARCHAR است منتهي از اولي در بانك اطلاعاتي اوراکل استفاده میشود و از دومي در بانك اطلاعاتي Sql سرور .
بنابراین بهتر است براي مقادیري با طول متفاوت از این نوع استفاده کنید . در ضمن این نوع داده ها مانند نوع قبل میتوانند ترکیبي از اعداد و کاراکتر ها و یا یکي از انها باشند .

اعداد از نوع صحیح :

اعداد صحیح تنها اعدادي هستند که میتوانند مقادري مثبت یا منفي داشته باشند که فاقد دقت اعشاري میباشند . در واقع این مقادیر تنها اعداد کامل را میپذیرند . از این نوع بیشتر بنام integer یاد میکنند .

اعداد دسیمال :

این نوع مقادیر برعکس مقادیر از نوع صحیح میتوانند مقادیر اعشاري را نیز مشتمل شوند و طرز تعریف انها به صورت زیر است :

Code:
DECIMAL (n,m)

که n در مقدار فوق طول عدد است که علاوه بر مقدار صحیح مقدار اعشاري را نیز شامل میشود . و منظور از m میزان دقت اعشار است . بعبارتي دیگر تعداد ارقام پس از نقطه اعشار است .
مانند مثال زیر :

Code:
decimal (5,2)

123.45
543.21

عدد زيرين به دليل افزايش مقدار دقت اعشار گرد ميشود يعني به ۱۲,۱۲ تبديل ميشود

Code:
12.123

دسيمال با ممیز شناور :

اين نوع اعداد به گونه اي هستند كه نقطه ممیز انها قابل جابجايي هست و محدوديت انواع دسيمال معمولي را ندارند كه انواع REAL براي مقاديري هستند كه دقت نقطه ممیز از اين نوع است كه براي ان عدد N بايد مقداري بين [۱-۲۱] باشد و براي دقت اعشاري مضاعف از نوع DOUBLE PRECISION استفاده ميشود كه مقدار N ان نيز بايد بين [۲۲-۵۳ [باشد .

Code:
FLOAT (10)
FLOAT(23)

نوع تاريخ و زمان :

اين نوع همانطور كه از نامش پيداست براي ذخيره زمان و تاريخ به كار ميرود كه بعدا مفصل توضيح داده ميشوند . اين انواع با DATE و TIME مشخص ميشوند .
كه براي DATE مقادير year , month , day و براي تاريخ مقادير hour , minute , second ميتوانيد داشته باشيد

داده هاي از نوع تهي :

اين نوع داده ها داده هايي هستند كه هيچ مقداري ندارند و گاها پيش مي اید كه در يك فيلد از ركوردي خاص داده اي براي ورود نداشته باشيم كه مقدار NULL براي ان در نظر گرفته ميشود . براي NULLكردن كافيست از خود اين كلمه استفاده كنيد يا از دو تك كوتيشن به هم چسبيده كه فاصله اي با هم ندارند ^ استفاده كنيد .
در ضمن اين مقدار يك مقدار NULL نيست 'NULL' :
بلكه رشته اي است كه بصورت ليترال تعيرف ميشود

انواع ليترال :

نوعي است كه عموما بصورت يك رشته است كه توسط خود كاربر وارد ميشود . اين نوع مخصوص يك فيلد يا ستون نيست . بلكه از اين نوع تنها براي مشخص كردن سريع ان توسط كاربر استفاده ميشود .
در زير چند مقدار ليترال را ميبينيد :

Code:
'234'
234
'DAL'
'DAL23L'

هر يك از رشته هاي فوق ليترال هستند . كه بين تك كوتيشن امده اند هر گاه تك كوتيشن حذف شود يعني مقدار شما عددي است . حتي مقداري كه در مثال اول امده است بصورت عدد بين كوتيشنها امده كه با اين عدد نيز به دليل انكه بين تك كوتيشن است بصورت رشته برخورد ميشود.

عنوان: **پیوند جداول در پرس و جو ها**

پیوند جداول در پرس و جو ها:

در استفاده از دستورات SQL گذرکردن از رابطه جداول دقیقا مانند انستکه کلا سیستم بانك اطلاعاتي رابطه اي را حذف كرده باشيم و كلا سيستم بانك اطلاعاتي رابطه اي را نادیده بگیریم .
میدانیم که هر پیوند دو یا بیش از دو جدول را به یکدیگر مرتبط میکند .

انواع پیوندها در SQL عبارتند از , EQUIJOIN , NATURAL JOIN , NON-QUIJOINS , OUTER JOINS
SELF JOINS
EQUIJOIN
مهمترین پویند این پیوند است که پیوند داخلي INNER JOIN نیزبه ان گفته میشود EQUIJOIN . دو جدول را با يك ستون مشترك كه معمولا در هر دوي انها كليد اصلي است به یکدیگر پیوند میزند .
نمونه قالب ان بصورت زیر است :

Code:
```
SELECT TABLEN.COLUMNN , TABLEM.COLUMNM ...
FROM TABLEN,TABLEM
WHERE TABLEN.COLUMNN = TABLEM.COLUMNM
[AND ... ]
```

همانطور كه قبلا هم گفتم براي مشخص كردن فيلدهايي كه در دو جدول در يك دستور SELECT مي ايند با يكديگر مي ايند بايد نام جدول را قبل از نام فيل نوشت و با يك نقطه ازنام فيلد جدا كرد . اينكار براي تنها فيلدهايي لازم است كه در دو جدول استفاده شده ما در دستور SELECT مشترك هستند و اگر براي نام ديگر فيلدها هم نام جدول را قبل از انها بنويسيد كاملا اختياري است .
چنانچه بخواهید از دستوري مانند دستور زیر استفاده کنید :

Code:
```
SELECT S.STNO,S.FNAME,G.GRADE FROM STUDENT S,GRADE G ;
```

دستور فوق فيلدهاي STNO و FNAME را از جدول student و فيلد grade را نيز از جدول grade انتخاب ميكند g . و s كه قبل از نام فيلدها ميبينيد نام مستعاريست (alias) براي دو جدول student و grade كه با استفاده از student s و grade g اين نام هاي مستعار را براي راحتي كار انتخاب كرده ايم . ميتوان بجاي دستور فوق از student as s و يا grade as g نيز استفاده كرد .
در دو دستور فوق بدليل عدم مشخص كردن پيوندي براي جداول طبق قانون ضرب تمامي فيلدهاي جدول اولي در دومي ضرب ميشود . اين بان دليل است كه طبق قانون ضرب هرگاه n مسير را بتوان به mطريق طي كرد بنابراين ميتوان m*n مسير را انتخاب كرد است .
به اين ضرب كه ضرب كارتزين معروف است كاملا در هنگام استفاده از دستورات SQL توجه داشته باشيد چرا كه هرگاه اشتباهي اينگونه انجام دهيد اگر هر جدول ۱۰۰۰ سطر داشته باشد و بين ان رابطه اي برقرار نكنيد طبق اين قانون ۱۰۰۰۰۰۰ سطر در خروجي خواهيد داشت !
بنابراين يادتان باشد كه هرگاه داده ها را از بيشتر از دو جدول انتخاب كرديد . رابطه را برقرار كنيد . وگرنه سرويس دهنده بانك اطلاعاتي شما چون پيوندي نميبيند براي هر سطر جدول اولي كليه سطرهاي جدول دومي را در خروجي ميدهد .
بنابراين بايد شرطي در دستور فوق اعمال كنيم كه ديگر چنين حادثه اي پيش نيايد بنابراين با استفاده از دستور Where يك رابطه equijoin درست ميكنيم . در واقع براي دسترسي به خروجي صحيح بصورت زير دستور فوق را عوض ميكنيم :

Code:
```
SELECT S.STNO,S.FNAME,G.GRADE FROM STUDENT S,GRADE G WHERE S.STNO=G.STNO
```

با شرط فوق فرمان داديم كه تنها اندسته از فيلدهايي را كه مقادير فيلد Stno در هر دوي جداول يكسان هستند نمايش دهد .

درمورد استفاده از نام مستعار براي جداول كه در بالا نيز از اين روش استفاده كرديم بايد بگويم كه استفاده از اين روش كاملا اختياري است اما استفاده از ان بهتر از استفاده نكردن از انست جراكه استفاده از نام مستعار سبب كوتاهتر شدن دستور sql ميشود كه منجر به كاهش احتمالي خطاي تايپ توسط كاربر است . در ضمن نام مستعار قرار دادن براي يك جدول تنها يك نام دادن موقتي است و در همان QUERY اين نام معتبر است وگرنه نام اصلي جدول دست نخورده باقي ميماند ...
توجه داشته باشيد كه نوع پيوند EQUIJOIN تنها يك نام است و انتظار دستور خاصي براي استفاده از اين پيوند نداشته باشيد . شما بدون انكه بدانيد استفاده از دستور WHERE در دستورات SQL سبب ايجاد اين نوع پيوند است ميتوانيد به كار خود ادامه دهيد .
در نقطه مقابل اين نوع پيوند نام ديگري بنام NON-EQUALITY قرار گرفته است كه اين پيوند تنها تفاوتي كه با پيوند EQUIJOIN دارد انستكه تنها در عبارت مقايسه اي كه در دستور WHERE استفاده ميكنيم از عملگر عدم تساوي استفاده ميكنيم ... دستور زير يك دستور از نوع اخري است :

Code:
SELECT S.STNO,S.FNAME,G.GRADE FROM STUDENT S,GRADE G WHERE S.STNO!=G.STNO

در ضمن بجاي عملگر != بسته به نوع نسخه اي كه نرم افزار مديريت بانك اطلاعاتي شما استفاده ميكند شايد بايد از اين <> استفاده كنيد . وظيفه تست كردن اينكه كدامين يك از دو عملوند ذكر شده در دستور sql شما قرار ميگيرد بر عهده خود شما .
مراقب باشيد هنگام استفاده از رابطه NON-EQUALITY با مشكل قانون ضرب كه در بالا گفتم مواجه نشويد .
در ضمن اينكه چرا قانون ضرب رخ ميدهد دليل دارد و ان به اين دليل است كه براي هر فيلد از ركورد مورد نظر در جدول اولي به تعداد تمامي ركوردهاي جدول دومي مقدار وجود دارد بنابراين فيلد اول از جدول اول N بار براي هر كدام از N ركورد جدول دومي متناظر ميشود و در خروجي نمايش داده ميشود .

OUTER JOIN

اين اتصال براي بدست اوردن تمامي سطرهاي جدولي مورد استفاده قرار ميگيرد كه در سطرهاي متناظر جدول ديگر وجود ندارد . يعني هر گاه بازاي مقدار مشتركي از يك فيلد كه در مقدار WHERE امده است مقداري نداشته باشيم اما بخواهيم باقي مقادير را نيز نمايش دهيم بايد از اين اتصال استفاده كرد .
اگر دستوري كه در SQUIJOIN استفاده كرديم را مرور كنيم :

Code:
SELECT S.STNO,S.FNAME,G.GRADE FROM STUDENT S,GRADE G WHERE S.STNO=G.STNO

ميبينيد كه جلوي شرط WHERE ذكر كرده ايم كه هر گاه STNO هر دو جدول برابر بود مقادير فيلد مورد نظر در مقابل SELECT را نمايش بده .
حال شايد مقداري از فيلد STNO در جدول داشته باشيم كه بخواهيم ان نيز نمايش داده شود . بنابراين بايد با استفاده از OUTER JOIN مشكل را حل كرد .
استفاده از نماد (+) در يك دستور SQL نمايانگر اين رابطه است كه انرا با عملوند جمع نبايد اشتباه بگيريد . چرا كه اين نماد هميشه بين دو پرانتز قرار ميگيرد . اين نماد در انتهاي نام جدول كه در مقابل عبارت WHERE امده است اورده ميشود . جدولي كه درمقابل ان از اين نماد استفاده ميكنيم بايستي جدولي باشد كه سطرهاي متناظر ما را نداشته باشد . در واقع جدولي كه نقصان فيلد مورد نظر دارد بايد اين نماد را با خودش حمل كند . البته استاندارد مورد نظر در بيشتر سرويس دهنده ها استفاده از اين نماد نيست بلكه رابطه ها به بخش هاي left , right , full تقسيم ميشوند اما اين نماد را نيز ميپذيرند .
نكته اي كه بايد فهميده باشيد انستكه استفاده از اين نماد تنها در يك طرف عمليات جايز است . يعني تنها يك جدول است كه اين نماد را ميتواند داشته باشد .
اما ياداوري ميكنم كه نكته فوق دليل بر ان نيست كه فيلدهاي يك جدول نتوانند از اين نماد استفاده كنند . اگر شرط فوق را رعايت كنيد يعني اين نماد را تنها براي يك جدول استفاده كنيد محدوديتي در استفاده ان براي چند فيلد از همان جدول نداريد .

Code:
SELECT S.STNO,S.FNAME,G.GRADE FROM STUDENT S,GRADE G WHERE

S.STNO=G.STNO(+)

در دستور فوق اگر STUDENT را جدول ليست دانش اموزان بدانيم و GRADE را ليست نمرات . و از طريق فيلد STNO كه بيانگر شماره دانش اموزي است با يكديگر رابطه داشته باشند .
حال با استفاده از (+) ميتوانيم بفهميم كه كدام دانش اموز نمره نداشته و كدام دانش اموز نمره دارد .
اگر نماد OUTER JOIN را پاك كنيم تنها ليستي از دانش اموزاني نمايش داده ميشود كه نمره دارند .
پيش مي ايد كه فيلد مشتركي بين دو جدول نداريد ! اما چون بانك اطلاعاتي شما رابطه اي است با نگاهي به رابطه هاي جداول ميتوانيد واسطه دو جدولي كه فيلد مشترك ندارند را پيدا كنيد . اينجاست كه از جدول واسط نيز براي ايجاد ارتباط استفاده ميكنيد .
فرض كنيد جدول A از طريق جدول B به جدول C مرتبط است . حال چون دو جدول A , C از طريق جدول Bبهم مرتبط شده اند . بنابراين جدول A با فيلدي از جدول B ارتباط دارد و جدول C نيز با فيلدي از جدول واسطه B ارتباط دارد . بنابراين دو جدول A , C را ميتوان بهم مرتبط كرد به اين صورت كه
اگر جدول A,B با فيلدي بنام KABK مرتبط به هم هستند و جدول B,C با فيلدي بنام KBCK به هم مرتبط هستن بنابراين با دستور زير ميتوان دو جدول A ,C را بهم مرتبط كرد :

Code:
WHERE (A.KABK=B.KABK) AND (B.KBCK=C.KBCK)

عنوان: **group Functionsتوابع گروهي وGROUP BY**

Sum - Avg - Min - Max - Count - SetDev - Var - First - Last - Expression - Where

Code:
Select Min(grade) from grade

براي وارد كردن دستور فوق مراحل زير را در بانك اطلاعاتي Access طي كنيد . ابتدا بانك اطلاعاتي جديد باز كنيد و در ان جدولي به نام grade كه فيلد هاي ان حاوي نمرات ميباشد را وارد كنيد سپس از انجا دكمه Queries را انتخاب كنيد . دكمه create Query in Design view را انتخاب كنيد سپس از انجا به View/SQL View برويد و دستور فوق را وارد كنيد براي اجرا شدن ان از دكمه علامت تعجبي كه در بالاي صفحه ميبينيد استفاده كنيد .
استفاده از تابع Min باعث ميشود كه كمترين نمره از جدول grade در خروجي تقاضا (Qurey) نمايش داده شود . فرضا اگر نمرات به ترتيب ۱۹،۲۰،۵،۷،۶،۳،۱۴،۱۸،۲۰ بود پايين ترين نمره كه در اين ليست ۳ است اورده ميشود . اما چيزي كه در خروجي به نظر ميرسد چندان جالب توجه نيست بنابراين براي انكه نام فيلد خروجي ما چيزي باشد كه كاربري كه ميخواهد نتيجه گزارش را ببيند مناسب باشد دستور فوق را به ترتيب زير عوض ميكنيم :

Code:
Select Min(grade) as Minimum from grade

بنابراين خروجي مينيموم نمره با مشخصه Minimum در نام فيلد را ميبينيد . با استفاده از AS ميتوان نام جديدي براي ستون خروجي درست كرد بنابراين كلمه اي كه بعد از as مي ايد كاملا سليقه ايست و پيرو قانون خاصي نيست ! به عبارت بعد از AS اصطلاحا alias ميگويند .
مثال ديگري ميزنيم : اين مثال در خروجي ميانگين نمرات جدول student را كه كدگروه انها ۸ ميباشد را در خروجي نمايش ميدهد .

Code:
Select Avg(Grade) from student where groupcode=”11”

همانطور كه ميبينيد براي انكه كدگروه ۱۱ را در خروجي ببينيم از دستور Where استفاده كرديم دستور فوق ميانگين نمراتي كه كدگروه ان نمرات ۱۱ ميباشد را نشان ميدهد .
اگر توجه كنيد متوجه ميشويد كه كد گروه در دو دابل كتويشن قرار گرفته است اين به ان دليل است كه نوع groupcode از نوع Text ميباشد ميتوانيد ان نوع را به number تغيير دهيد تا ديگر به اين جفت

كوتيشن احتياجي نداشته باشيد .

Code:
Select Sum(grade*unit) from grade

مجموع نمرات ضربدر تعداد واحد درسي را از جدول نمرات grade محاسبه ميكند و در خروجي نشان ميدهد . ميتوانيد از as newname در جلوي تابع sum استفاده كنيد .

Code:
Select Stno,avg(grade) from grade group by stno

و اما دستور فوق كه ميبينيد دستوريست كاملا شبيه به دستورات بالا منتهي از دستوري ديگري به نام group byنيز استفاده كرده ايم . استفاده از اين دستور كاملا اجباريست ! چرا كه از دستورات توابع گروهي استفاده كرده اند . همانطور كه ميبينيد از تابع avg براي اين دستور استفاده كرده اند ; منتهي قبل از ان از يكي از فيلد ها نيز براي نمايش داده شدن استفاده كرده اند . اين فيلد چون از نوع توابع گروهي نيست بنابراين بايد با استفاده از دستور group by گروه بندي را بر اساس ان انجام داد . بهتر است كمي درباره دستور فوق بحث و بررسي انجام دهيم . همانطور كه در دستور فوق ميبينيد از تابع avg براي ميانگين گرفتن استفاده كرده ايم اما قبل از ان stno را نيز اورده ايم . با كمي تفكر بر روي اينكه چرا stno امده است ميتوانيد به اين نتيجه برسيد كه stno دليل نداشته است كه در اين تقاضا بيايد جز انكه بر اساس ان تابع گروهي ما عمل كند . بنابراين چون تابع گروهي ما كه در اينجا avg است به همراه stno امده است بنابراين بايد دستور ميانگين را بر اساس Stno يا همان شماره دانشجويي انجام دهد . بنابراين با استفاده از دستور group by ميانگين نمرات را بر اساس Stno گروه بندي ميكنيد . لذا اگر علاوه بر stno از فيلد ديگري نيز استفاده كنيد . يعني تابع avg خود را ميخواهيد باز هم كوچك تر كنيد . و بر اساس ان بايد گروه بندي خود را جامع تر كنيد . يعني اگر فرض كنيم علاوه بر Stno از term نيز استفاده ميكنيم . لذا گروه بندي بر اساس شماره دانشجويي و ترم انتخابي انجام ميشود . اين به ان معنيست كه ديگر استفاده از تابع ميانگين براي كليه نمرات يك دانش نيست بلكه براي انستكه ميانگين نمرات هر دانشجو در هر ترم اندازه گيري شود و در خروجي نمايش داده شود بنابراين براي انكه از تابع ميانگين براي جدا كردن درس ها از ترم هاي تحصيلي استفاده كنيم دستور فوق به شكل زير تغيير پيدا ميكند

Code:
select stno,term,avg(grade) from grade group by stno, term

از توضيحات فوق ميتوان اين نتيجه را نيز به دست اورد كه هر گاه از تابعهاي گروهي در دستوري استفاده كرديم براي انكه از group by استفاده نكنيم نبايد بر اساس فيلدي انرا محدود كنيم چراكه استفاده از همان فيلد مذكور منجر به استفاده از group by ميشود .

در هر صورت هر گاه از توابع جمعي استفاده كرديم براي گسترش قابليت بيشتر آنها بايد از Group by استفاده كنيم Group by . براي مشخص كردن گروههايي استفاده ميشود كه سطرهاي خروجي بايد در انها قرار بگيرند و وقتي از توابع جمعي استفاده ميكنيد خلاصه داده ها برا ان گروه محاسبه ميشود . . گروه بندي عموما مستقل از ستونهاي دستور SELECT انجام ميشود . هر يك از ستونهايي كه فاقد توابع جمعي بوده و در دستور SELECT مشخص شده اند بايد در عبارت group by فهرست شوند . دليل اين امر قانوني است كه بر طبق ان هر فيلدي كه در فهرست انتخاب قرار ميگيرد بايد تنها يك مقدار عبارت براي Group by برگرداند.