پایگاه داده

SQL

مفاهیم پایگاه داده

*Database*یا دیتابیس یا پایگاه داده یا بانک اطلاعاتی چیست ؟

دیتابیس یا پایگاه داده چیست ؟ بانک اطلاعاتی یا پایگاه داده یا دیتابیس *(data base)* به مجموعه ای از اطلاعات با ساختار منظم گفته می شود. این پایگاه های اطلاعاتی معمولاً در قالبی که برای دستگاه ها و رایانه ها قابل خواندن و قابل دسترسی باشند ذخیره می شوند.

پایگاه داده اس کیو ال چیست ؟ با گسترش طراحی وب سایت های پویا در شبکه جهانی وب نیاز به یک پایگاه داده و بانک اطلاعاتی *(database)* بزرگ برای مدیریت محتوا احساس میشد. مدیریت پایگاه داده ها یک فرایند پیچیده است ، که به طور قابل توجهی با برنامه و زبان برنامه نویسی *SQL* این نیاز برطرف شده است.

پایگاه داده در اصل مجموعه ای سازمان یافته از اطلاعات است. این واژه از دانش رایانه سرچشمه می گیرد ،اما کاربر وسیع و عمومی نیز دارد، این وسعت به اندازه ای است که مرکز اروپایی پایگاه داده (که تعاریف خردمندانه ای برای پایگاه داده ایجاد می کند) شامل تعاریف غیر الکترونیکی برای پایگاه داده می باشد. در این نوشتار به کاربرد های تکنیکی برای این اصطلاح محدود می شود.

یک تعریف ممکن این است که: پایگاه داده مجموعه ای از رکورد های ذخیره شده در رایانه با یک روش سیستماتیک (اصولی) مثل یک برنامه رایانه ای است که می تواند به سوالات کاربر پاسخ دهد. برای ذخیره و بازیابی بهتر، هر رکورد معمولا به صورت مجموعه ای از اجزای داده ای یا رویداد ها سازماندهی می گردد.

بخش های بازیابی شده در هر پرسش به اطلاعاتی تبدیل می شود که برای اتخاذ یک تصمیم کاربرد دارد. برنامه رایانه ای که برای مدیریت و پرسش و پاسخ بین پایگاه های داده ای استفاده می شود را مدیر سیستم پایگاه داده ای یا به اختصار *(DBMS)* می نامیم. خصوصیات و طراحی سیستم های پایگاه داده ای در علم اطلاعات مطالعه می شود.

مفهوم اصلی پایگاه داده این است که پایگاه داده مجموعه ای از رکورد ها یا تکه هایی از یک شناخت است .نوعا در یک پایگاه داده توصیف ساخت یافته ای برای موجودیت های نگه داری شده در پایگاه داده وجود دارد: این توصیف با یک الگو یا مدل شناخته می شود. مدل توصیفی، اشیا پایگاه های داده و ارتباط بین آنها را نشان می دهد. روش های متفاوتی برای سازماندهی این مدل ها وجود دارد که به آنها مدل های پایگاه داده گوییم.

پرکاربرد ترین مدلی که امروزه بسیار استفاده می شود، مدل رابطه ای است که به طور عام به صورت زیر تعریف می شود: نمایش تمام اطلاعاتی که به فرم جداول مرتبط که هریک از سطر ها و ستونها تشکیل شده است(تعریف حقیقی آن در علم ریاضیات برسی می شود). در این مدل وابستگی ها به کمک مقادیر مشترک در بیش از یک جدول نشان داده می شود. مدل های دیگری مثل مدل سلسله مراتب و مدل شبکه ای به طور صریح تری ارتباط ها را نشان می دهند.

در مباحث تخصصی تر اصتلاح دادگان یا پایگاه داده به صورت مجموعه ای از رکورد های مرتبط با هم تعریف می شود. بسیاری از حرفه ای ها مجموعه ای از داده هایی با خصوصیات یکسان به منظور ایجاد یک پایگاه داده ای یکتا استفاده می کنند.

معمولا *DBMS* ها بر اساس مدل هایی که استفاده می کنند تقسیم بندی می شوند: ارتباطی،شی گرا، شبکه ای و امثال آن. مدل های داده ای به تعیین زبانهای دسترسی به پایگاه های داده علاقه مند هستند. بخش قابل توجهی از مهندسی *DBMS* مستقل از مدل های می باشد و به فاکتور هایی همچون اجرا، همزمانی،جامعیت و بازیافت از خطاهای سخت افزاری وابسطه است.در این سطح تفاوت های بسیاری بین محصولات وجود دارد.

هر *database* در اسکیوال از قسمتهای مختلفی تشکیل شده است،این بخشها شامل موارد زیر است:

- *Table*
- *View*
- *Stored procedre*
- *Function*
- و....

*Table*چیست ؟

جدول ***(table)*** **در پایگاه داده ها چیست** ؟ داده ها در دیتابیس یا پایگاه داده داخل جدول یا *table* ها ذخیره میشوند. هر *database* میتواند شامل چندین *table* باشد .هر جدول از تعدادی سطر و ستون تشکیل شده است.

برای تمام ستون ها در دیتابیس بسته به نوع کارکرد آن میتوانید نوع مورد نظر خود را تعریف کنید ، مقدارهایی که درون هر فیلد ذخیره میشود بایستی با نوع انتخابی فیلد مطابقت داشته باشد.

جدول یک مجموعه ای است از اطلاعات ثبت شده مرتبط و وابسته به هم که از ستون ها و ردیف ها تشکیل شده است. جداول مهمترین عناصر سیستم های پایگاه داده هستند ، که برای ذخیره و نگهداری سازمان یافته اطلاعات مورد استفاده قرار می گیرند.

جدول بخشی از پایگاه داده است. یک پایگاه داده از جداول مختلف تشکیل شده است.

مثال : برای ذخیره‌سازی انواع مختلف داده ها شما نیازمند ایجاد جداول جداگانه‌ای هستید. برای مثال، اگر شما یک نرم افزار مدیریت مدرسه دارید، ممکن است نیاز به ایجاد جداول زیر باشد:

- **دانش آموزان** – برای ذخیره لیستی از تمام اعضای دانش آموزان
- **معلمان** – ذخیره لیستی از تمام معلمان
- **حضور و غیاب** – برای پیگیری حضور همه دانش آموزان
- *MarkList* _برای ذخیره لیست علامت همه دانش آموزا

*Field*چیست ؟

به هر یک از خانه های ستون یک جدول فیلد *(Field)* می گویند . هر فیلد یکی از خصوصیات آن موجودیت را به همراه مقدار آن مشخص می کند.

هر فیلد در بر گیرنده یک صفت و ویژگی برای موجودیت می باشد ، که دارای ۲ جزء اصلی است:

۱. اسم صفت خاصه : نام صفت مورد نظر را تعیین می کند . برای مثال فیلد نام ، نام خانوادگی ، و ... در جدول اطلاعات مربوط به شخص .
۲. مقدار صفت خاصه : در برگیرنده مقدار برای صفت مورد نظر است . برای مثال مقدار " سعید " به عنوان مقدار برای فیلد نام.

یک فیلد نشانه یک ستون در جدول است. یک رکورد مجموعه ای از فیلدها است. تمام رکوردها در همان جدول همان فیلدها راخواهند داشت.

مثال درباره فیلد ها : اگر شما یک جدول با نام *"Students"* داشته باشید، ممکن است زمینه های زیر مورد نیاز باشند:

- *Name*برای ذخیره نام و نام خانوادگی دانش آموز
- *Address*برای ذخیره آدرس
- *DateofBirth*برای ذخیره تاریخ تولد دانش آموز
- *RegistrationDate*برای ذخیره تاریخ ثبت نام دانش آموز
- و...

اگر شما فیلدی را به جدول اضافه کنید، این فیلد به تمام رکوردهای موجود آن جدول اضافه خواهد شد. در مثال فوق، تمامی رکوردهای جدول در *"Students"* همان ۴ فیلد را خواهند داشت.

*Record*چیست ؟

رکورد در پایگاه داده چیست ؟ به سطرهای یک جدول رکورد *(Record)* گفته میشود. هر رکورد مجموعه ای از اطلاعات طبقه بندی شده درباره یک **موجودیت** خاص است.

موجودیت

موجودیت پدیده ، شی یا فردی در محیط پایگاه داده است که می خواهیم اطلاعات مربوط به آن را نگهداری کنیم.
مثال : به طور مثال در محیط پایگاه داده یک محیط آموزشی، انواع موجودیت ها عبارتند از : دانشجو ، کلاس ، واحدهای درسی ، استاد ، دانشکده و. ...

یک رکورد نشان دهنده یک ورودی در جدول است .یک جدول می تواند هر تعداد رکورد داشته باشد.

مثال از رکورد در پایگاه داده : اگر شما جدول *"Students"* برای ذخیره اطلاعات دانش آموزان داشته باشید، در این جدول یک **رکورد** نشان دهنده یک دانش آموز خواهد بود. برای اضافه کردن دانش آموز به برنامه، باید یک رکورد به جدول *"Students"* اضافه کنید. برای پاک کردن یا ویرایش اطلاعات دانش آموز هم باید شما یک رکورد را از این جدول حذف کنید.

مقدمات SQL

*SQL*چیست ؟ (اس کیو ال چیست)

در مدل رابطه‌ای داده‌ها، زبان ساختارمند پرسش‌ها یا اس کیو ال یا سی کوال *(Structured Query Language - SQL)*زبانی است سطح بالا مبتنی بر زبان سطح پایین و ریاضی جبر رابطه‌ای که برای ایجاد، تغییر، و بازیابی داده‌ها و نیز عملیات بر روی آنها به‌کار می‌رود.

زبان *SQL* به سمت مدل شی‌گرا - رابطه‌ای نیز پیشرفت کرده است.

سی‌کوال استاندارد *ANSI (American National Standards Institute)* را در سال ۱۹۸۶ و *ISO (International Organization for Standardization)*را در سال ۱۹۸۷ اتخاذ نمود. استانداردهای مختلفی از اس‌کیوال تاکنون عرضه شده که در جدول زیر بیان می‌کنیم:

- اس‌کیوال-۸۷
- اس‌کیوال-۸۹
- اس‌کیوال-۹۲
- اس‌کیوال:۱۹۹۹
- اس‌کیوال:۲۰۰۳
- اس‌کیوال:۲۰۰۵
- اس‌کیوال:۲۰۰۸
- اس‌کیوال:۲۰۱۰
- اس‌کیوال:۲۰۱۱

بسیاری از اصطلاحات زبان اس‌کیوال تحت استاندارد بین‌المللی بوده، و در نتیجه، از آنها شبیه بقیه زبانهای استاندارد مثل محصولات شرکت اوراکل *[PL/SQL ۲ [یا Sybase و) SQL PL* مدل رویه‌ای) از شرکت آی‌بی‌ام می‌باشد.

اس‌کیوال برای کارهای ویژه و محدودی (گزارش گیری از داده‌ها در پایگاه داده‌های رابطه‌ای) طراحی شده‌است. بر خلاف زبانهای دستوری مثل بیسیک یا سی که برای حل مسائل طراحی شده، *SQL*زبانی بر پایه اعلان است. زبانهای توسعه یافته‌ای مثل *PL/SQL* به دنبال کامل کردن زبان به هدف ایجاد زبان برنامه نویسی با حفظ مزیت‌های *SQL* می‌باشد. شیوه دیگر کار این است که به کدهای زبان برنامه نویسی اجازه دسترسی به پایگاه داده به کمک دستورات *SQL* داده شود مثلاً *PostgreSQL* به توابعش اجازه می‌دهد که درون کدهای

Perl، *Tcl*و *C* نوشته شوند. گاهی به شوخی گفته می‌شود که *SQL* نه ساخت یافته‌است، نه محدود به گزارش گیری‌ها و اصلاً یک زبان نیست!

مزایای *SQL*

- *SQL*بر پایه زبان پرس و جو ساخت یافته میباشد
- *SQL*به شما اجازه دستیابی و کنترل داده ها را می دهد
- *SQL*یک استاندارد*ANSI* (انجمن استاندارد ملی آمریکا) میباشد.
- *SQL*میتواند درخواستهای پیوسته یک پایگاه داده را اجرا کند
- *SQL*متواند دوباره اطلاعات را از پایگاه داده پس بگیرد
- *SQL*میتواند یک رکورد شامل اطلاعات را در پایگاه داده ذخیره کند
- *SQL*متواند اطلاعات پایگاه داده را به روز رسانی کند
- *SQL*میتواند هر قسمت از اطلاعات را از پایگاه داده اصلاح یا حذف کند
- *SQL*میتواند یک پایگاه داده تازه بسازد
- *SQL*میتواند جداول حاوی اطلاعات جدید را به پایگاه داده اضافه کند
- *SQL*اجازه تنظیم جداول و شیوه دستیابی به اطلاعات و نحوه نمایش اطلاعات را میدهد.

تاریخچه اس کیو ال *(SQL)*

منشا اصلی سی‌کوال به مقاله سال ۱۹۷۰ ادگار کاد تحت عنوان «مدل رابطه‌ای داده‌ها برای بانک‌های بزرگ داده‌های اشتراکی»[۱] باز می‌گردد. در دهۀ ۷۰ گروهی از شرکت آی‌بی‌ام در شهر سان خوزه بر روی سیستم پایگاه داده‌های سیستم آر بدون توجه به این مقاله کار می‌کردند و زبان *SEQUEL* را به منظور عملیات و بازیابی اطلاعات ذخیره شده در سیستم آر ایجاد کردند. اگر چه اس‌کیوال ناشی از تلاشهای کاد بود اما دونالد چامبرلین و ریموند بویس به عنوان طراحان زبان *SEQUEL* شناخته می‌شوند.

سمینارهایی در زمینه فناوری بانک اطلاعاتی و مباحثاتی در مورد مزایای مدل رابطه‌ای جدید برگزار گردید. تا ۱۹۷۶ مشخص بود که آی‌بی‌ام که طرفدار جدی فناوری بانک اطلاعاتی رابطه‌ای بوده، توجه زیادی نسبت به زبان سی‌کوال دارد. تبلیغات در زمینه سیستم آر باعث جذب گروهی از مهندسین در منلو پارک در کالیفرنیا گردید. این گروه به این نتیجه رسیدند که تحقیقات آی‌بی‌ام منجر به یک بازار تجاری برای بانک‌های اطلاعاتی رابطه‌ای خواهد گردید.

در ۱۹۷۷ این گروه شرکتی بنام اینک *(Inc)* و رلیشنال سافتویر *(Relational Software)* تأسیس نمودند تا یک سامانه مدیریت پایگاه‌های داده رابطه‌ای بر اساس سی‌کوال بسازند. محصولی بنام اوراکل در ۱۹۷۹ عرضه گردید، و اولین سامانه مدیریت پایگاه داده رابطه‌ای بوجود آمد. به این ترتیب محصول اوراکل باعث گردید اولین محصول آی‌بی‌ام برای مدت ۲ سال در بازار دچار رکود باشد. این محصول بر روی مینی کامپیوترهای وکس دیجیتال *(VAx Digital)* اجرا می‌شد که خیلی از کامپیوترهای بزرگ آی‌بی‌ام ارزان‌تر بودند.

امروزه این شرکت با نام اوراکل اولین فروشنده سیستم‌های مدیریت بانک اطلاعاتی رابطه‌ای است. استادان آزمایشگاه‌های کامپیوتر در دانشگاه برکلی کالیفرنیا نیز در نیمه دهه ۱۹۷۰ مشغول تحقیق در زمینه بانک‌های اطلاعاتی رابطه‌ای بودن (مانند تیم تحقیق آی‌بی‌ام)، گروه فوق نیز یک نمونه از سامانه مدیریت پایگاه داده رابطه‌ای ایجاد نمودند و سیستم خود را اینگرس *(Ingres)* نام نهادند.

پروژه اینگرس شامل یک زبان پرس‌وجو بنام *QUEL* بود، اگر چه از سی‌کوال خیلی ساخت یافته تر بود، اما شباهت آن به زبان انگلیسی کمتر بود.

در حالیکه اوراکل و اینگرس برای ارائه محصولات تجاری در رقابت بودند، پروژه سیستم آر شرکت آی‌بی‌ام در تلاش بوده‌است که یک محصو ل تجاری با نام *SQL/Data system)* یا *SQL/DS)* عرضه نماید. آی‌بی‌ام موجودیت *SQL/DS* را در ۱۹۸۱ اعلام، و در ۱۹۸۲ شروع به عرضه محصول خود نمود. در سال ۱۹۸۳ آی‌بی‌ام یک نسخه *SQL/DS* را برای *VM/CMS)* سیستم‌عاملی که در کامپیوتر بزرگ آی‌بی‌ام غالبا استفاده شده بود)، اعلام نمود.

همچنین در سال ۱۹۸۳ شرکت آی‌بی‌ام، محصول دی‌بی‌تو را معرفی نمود که یک سامانه مدیریت پایگاه داده رابطه‌ای برای سیستم‌های بزرگ آن شرکت بود. دی‌بی‌تو تحت سیستم‌عامل وی‌ام‌اس (سیستم‌عامل مراکز کامپیوتری بزرگ) اجرا می‌شد. اولین نسخه دی‌بی‌تو در ۱۹۸۵ عرضه گردید، و مسئولین آی‌بی‌ام اعلام نمودند که این محصول یک برنامه استراتژیک برای تکنولوژی نرم‌افزاری آی‌بی‌ام می‌باشد. از آن تاریخ تاکنون دی‌بی‌تو سامانه مدیریت پایگاه داده رابطه‌ای شاخصی بوده و آی‌بی‌ام از آن حمایت نموده و زبان «سی‌کوال دی‌بی‌تو» استاندارد عملی زبان بانک اطلاعاتی بوده‌است.

با *SQL* چه کار هایی میتوان انجام داد ؟

۱. *SQL*این توانایی را دارد که یک کوئری *(Query)* را اجرا کند.
۲. *SQL*میتواند داده ها را از دیتابیس بازیابی کند.

۳. *SQL*میتواند رکوردهایی را به دیتابیس اضافه *(Insert)* کند.
۴. *SQL*میتواند رکوردها را از دیتابیس واکشی و ویرایش *(Update)* کند.
۵. *SQL*میتواندرکوردها را از دیتابیس واکشی و حذف *(Delete)* کند.
۶. *SQL*میتواند یک پایگاه داده جدید *(New Database)* ایجاد کند.
۷. *SQL*میتواند یک جدول به دیتابیس *(New Table)* اضافه کند.
۸. *SQL*میتواند *stored procedure* در دیتابیس ایجاد کند.
۹. *SQL*میتواند *view* در دیتابیس ایجاد کند.
۱۰. *SQL*میتواند به *table* ، *procedure* و *view* ها دسترسی تعریف کند.

استفاده از *SQL* در طراحی سایت

برای ایجاد یک وب سایت داینامیک که داده ها و اطلاعات را از یک پایگاه داده بخواند شما بایستی مراحل زیر را پیگیری کنید.

- استفاده از یک برنامه سیستم مدیریت پایگاه داده رابطه ای *(RDBMS)* مانند *Access* ، *SQL Server*، *My SQL*
- یک زبان برنامه نویسی تحت سرور مانند *PHP* یا *ASP*
- استفاده از پایگاه داده*SQL*
- استفاده از *HTML* و*CSS*

داده ها در *RDBMS* در قسمت *Table* ذخیره میشوند.جدول ها مجموعه از اطلاعات مربوط به برنامه میباشند ، هر جدول شامل سطر ها و ستون ها میباشد.استفاده از *SQL* در طراحی سایت به شما کمک میکند تا یک وب سایت داینامیک و پویا ایجاد کنید.

دستورات *DML* و *DDL* در اس کیو ال

*SQL*به دو قسمت تقسیم میشود:

- زبان دستکاری داده ها*DML (Data Manipulation Language)*
- زبان تعریف داده ها*DDL (Data Definition Language)*

دستورات *DML* شامل بخشهای زیر میشوند

۱. *SELECT*واکشی اطلاعات از دیتابیس
۲. *UPDATE*ویرایش اطلاعات دیتابیس
۳. *DELETE*پاک کردن اطلاعات از دیتابیس
۴. *INSERT INTO*اضافه کردن اطلاعات جدید به دیتابیس

دستورات*DDL* شامل بخشهای زیر میشوند

۱. *CREATE DATABASE*ایجاد یک دیتابیس جدید
۲. *ALTER DATABASE*ایجاد تغییرات در دیتابیس
۳. *CREATE TABLE*ایجاد یک *table* جدید
۴. *ALTER TABLE*اعمال تغییرات در*table*
۵. *DROP TABLE*پاک کردن یک*table*
۶. *CREATE INDEX*ایجاد یک شاخصه
۷. *DROP INDEX*حذف یک شاخص

تعریف شاخصه : *Index* شاخص عبارتست از یک شماره که به هر یک از فیلدها در سطرهای یک جدول اختصاص داده می شود . شاخص ها در پشت پرده جداول ایجاد شده و از دید کاربر کاملا مخفی هستند . استفاده از شاخص ها باعث می شود تا برنامه بتواند مقادیر سطرهای مختلف را بر حسب مقدار یک فیلد و بر حسب شماره شاخص آنها از کم به زیاد یا بر عکس مرتب کند و در عملیات جستجو باعث بالا رفتن سرعت جستجو می شود .

انواع داده و متغیر در*SQL*

انواع داده رشته ای در اسکیوال *Character strings :*

این نوع فیلد برای نگهداری عبارات و یا حروف *ASCII* می‌باشد. در این نوع فیلدها، برای نگهداری هر حرف، یک بایت اشغال می‌شود و لذا نیاز به *Collation* برای تعیین زبان اطلاعات می‌باشد.

نوع داده	شرح
char(n)	اطلاعات متنی با طول ثابت از ۱ تا حداکثر ۸۰۰۰ حرف را در خود ذخیره می‌کنند.
varchar(n)	اطلاعات متنی با طول متغیر از ۱ تا حداکثر ۸۰۰۰ حرف را در خود ذخیره می‌کنند. فرق بین *Char* و *VarChar* در این است که در*Char* ، طول رشته ثابت است. یعنی اگر یک فیلد را از نوع *Char(20)* معرفی کرده و در آن کلمه *Orion* را قرار دهیم، عین ۲۰ حرف استفاده خواهد شد. یعنی ۵ کاراکتر اول را کلمه مربوطه اشغال کرده و ۱۵ کاراکتر باقی‌مانده، *Blank* خواهند بود. اما در *VarChar* اینگونه نیست..
varchar(max)	اطلاعات از ۱ تا ۲ مگا حرف ذخیره می‌شود و مکانیزم آن هم بصورت پوینتری می‌باشد.
text	اطلاعات از ۱ تا ۲ مگا حرف ذخیره می‌شود. این نوع داده همانند *Image* و *VarBinary(MAX)*در خود رکورد ذخیره نمی‌شوند. بلکه توسط یک پوینتر به جای دیگری اشاره می‌کنند. این نوع داده در *SQL 10* حذف شده و بجای آن‌ها از *VarChar(MAX)* استفاده می‌شود..

انواع داده یونیکد در اسکیوال*Unicode types* :

این نوع فیلدها برای نگهداری متون *Unicode* بوده و برای نگهداری هر حرف، از دو بایت استفاده می‌شود. پس مسلماً نسبت به نوع داده‌های کاراکتری، حافظه بیشتری را به خود اختصاص می‌دهد و در ضمن کمی هم کندتر است. این نوع فیلدها، احتیاج به *Collation* ندارند.

نوع داده	شرح
nchar(n)	در این نوع داده، اطلاعات از ۱ تا حداکثر ۴۰۰۰ حرف با طول ثابت ذخیره می‌شود.
nvarchar(n)	در این نوع داده، اطلاعات از ۱ تا حداکثر ۴۰۰۰ بایت با طول متغیر ذخیره می‌شود.

nvarchar(max)	در این نوع داده از ۱ تا ۱ مگا حرف ذخیره می‌شود. مکانیزم آن هم بصورت *Pointer*می‌باشد.
ntext	در این نوع داده، از ۱ تا ۱ مگا حرف ذخیره می‌شود. مکانیزم آن هم بصورت *Pointer* .

انواع داده باینری در اسکیوال *Binary types* :

این نوع فیلدها برای نگهداری اطلاعات بصورت باینری مانند تصاویر مناسب هستند

نوع داده	شرح
bit	یک فیلد دو بیتی است و می‌تواند ۰ و ۱ و *Null* را ذخیره کند.کاربرد آن در زمان‌هایی است که دو حالت وجود داشته باشد. مانند جنسیت زن و مرد.
binary(n)	این نوع فیلدها، از ۱ تا ۸۰۰۰ بایت را در خود جای می‌دهند.
varbinary(n)	این نوع فیلدها هم از ۱ تا ۸۰۰۰ بایت را در خود جای می‌دهند. (متغیر).
varbinary(max)	این نوع *Datatype* در *SQL 2005* معرفی شده و تقریباً همانند داده‌های *Image*هستند.
image	این نوع فیلدها از ۱ تا حداکثر ۲ گیگابایت را می‌توانند ذخیره کنند. فرق این نوع داده‌ها با دو نوع قبلی این است که در دو نوع قبلی، اطلاعات در خود رکورد ثبت می‌شوند ولی در این نوع داده‌ها، اطلاعات در یک *Page* ذخیره می‌شود و به جایش در رکورد، یک پوینتر ۱۶ بایتی ذخیره می‌شود. این نوع فیلدها در *SQL 10* حذف شده و به جایش باید از *VarBinary* استفاده کرد.

انواع داده عددی در اس کیو ال *Number types* :

این نوع فیلد برای نگهداری اعداد صحیح و بدون اعشار استفاده می‌گردد و دارای ۴ نوع به شرح زیر است. در ضمن این نوع فیلدها رتبه یک سرعت در نوع فیلدهای عددی را دارد.

نوع داده	شرح
tinyint	یک بایت را اشغال می‌کند و می‌تواند از ۰ تا ۲۵۵ را در خود ذخیره کند.
smallint	یک عدد دو بایتی است و می‌تواند از ۳۲۷۶۷ منفی تا ۳۲۷۶۷ مثبت را در خود ذخیره کند.
int	یک عدد چهار بایتی است که می‌تواند اعداد بین مثبت و منفی ۲ میلیارد را در خود ذخیره کند.
bigint	یک عدد ۸ بایتی است که می‌تواند اعداد بین مثبت و منفی ۴ میلیارد را در خود ذخیره کند.
decimal(p,s)	این نوع فیلد برای نگهداری اعداد اعشاری با تعداد اعشار مشخص استفاده می‌گردد. این نوع فیلدها بسیار کند بوده و استفاده از آنها توصیه نمی‌گردد. که در آن *Precision* به معنای تعداد کل رقم‌های عدد و *Scale* تعداد ارقام اعشار را مشخص می‌کند. مثلاً اگر فیلدری بصورت *Deciaml(6,2)* تعریف شود، حداکثر آن برابر ۹۹۹۹،۹۹ می‌باشد..
numeric(p,s)	.
smallmoney	یک عدد ۴ بایتی است که می‌تواند ۶ رقم صحیح و ۴ رقم اعشار را در خود ذخیره کند..
money	یک عدد ۸ بایتی است که می‌تواند ۱۵ رقم صحیح و ۴ رقم اعشار را در خود ذخیره کند..
float(n)	یک عدد ۸ بایتی که اعداد بصورت توانی از ۱۰ نگهداری می‌شوند. .
real	یک عدد ۴ بایتی است که اعداد بصورت توانی از ۱۰ نگهداری می‌شوند..

انواع داده تاریخ در اس کیوال *Date types*:

این نوع فیلدها برای نگهداری تاریخ میلادی و ساعت استفاده می‌شود و برای تاریخ شمسی کاربرد ندارد.

نوع داده	شرح
datetime	این نوع فیلد، ۸ بایتی است و از سال ۱۷۰۰ تا ۹۹۹۹ را با دقت هزارم ثانیه ذخیره می‌کند..
datetime2	.
smalldatetime	این نوع فیلد، ۴ بایتی است و از سال ۱۹۰۰ تا ۲۰۷۹ را با دقت هزارم ثانیه ذخیره می‌کند..
date	این نوع فیلدها برای نگهداری تاریخ میلادی استفاده می‌شود.
time	این نوع فیلدها برای نگهداری ساعت استفاده می‌شود.
datetimeoffset	
timestamp	

انواع دیگر داده ها در *sql: Other data types*

نوع داده	شرح
sql_variant	این نوع فیلد برای نگهداری انواع داده استفاده می‌شود و نوع آن با توجه به اولین مقداری که در آن قرار می‌گیرد تعیین خواهد شد. چون نوع و حجم فیلد مشخص نیست، لذا تنها یک اشاره‌گر ۱۶ بایتی در آن قرار گرفته و داده اصلی در فایل جداگانه نگهداری می‌شود. استفاده از این نوع فیلد، توصیه نمی‌گردد..
uniqueidentifier	.این فیلد ۱۶ بایتی، به ما کدی *Unique* یا تک می‌دهد که به اصطلاح *GUID*می‌گویند. یکی از کاربردهای آن در *Replication* است.
xml	این فیلد بیشتر برای انتقال اطلاعات و دستورات تحت *web* استفاده می‌شود و شامل انواع *MetaData* های مختلف است.

cursor	این فیلد مربوط به کنترل *Cursor* است.
table	.

دستورات SQL

دستور *SELECT* در *sql*

دستور *SELECT* در اسکیوال برای انتخاب و استخراج اطلاعات مورد نظر از یک یا چند جدول و سپس مشاهده نتایج در یک جدول موقت استفاده می شود:

```
SELECT column_name(s)
FROM table_name
```

***SELECT*در اینجا به مفهوم فعل است و مشخص میکند که میخواهیم اطلاعاتی را بخوانیم.**

- *SELECT*فهرست ستونها مشخص میکند که قصد خواندن کدام ستونها را از بانک داریم در صورتی که در این فهرست * قرار دهیم کلیه ستونهای جدول نمایش داده میشود.
- با استفاده از کلمه *FROM* مشخص میکنیم که از کدام جدول یا جداول این ستونها باید انتخاب شوند .

دستور * *SELECT*

دستور * *SELECT* همه اطلاعات موجود در جدول را انتخاب میکند.برای انتخاب کلیه ستون های جدول در قسمت نام ستون باید علامت * را گذاشت استفاده کرد .

```
SELECT * FROM table_name
```

نکته مهم : برخی از برنامه نویسان بجای نوشتن تمام ستونهایی که در تهیه گزارش به آنها احتیاج دارند از * استفاده میکنند که این کار باعث بالا رفتن بار شبکه شده و کارایی سیستم را پایین می آورد لذا بهتر است بجای استفاده از * کمی بخود زحمت دهید و لیست تمامی ستونها را بطور کامل قید کنید.

دستور *SELECT DISTINCT* در *sql*

چنانچه در ستون های مورد جستجو ، موارد تکراری وجود داشته باشد در نتیجه خروجی نمایش داده خواهند شد . برای جلوگیری از چنین موردی و عدم نمایش موارد تکراری پس از دستور *Select*عبارت *DISTINCT* نوشته می شود.

```
SELECT DISTINCT column_name(s)
FROM table_name
```

مثال : تصور کنید یک جدول مشتری با ۱۰۰۰ رکورد با ۹۰ درصد مشتری از کالیفرنیا، *Query* زیر کد *CA* را ۹۰۰۰ بار خواهد باز گرداند که اصلاً یک نتیجه مفیدی نمی باشد.

```
SELECT State From Customer
```

کلید واژه *Distinct* در این موقعیت شما را کمک می کند .*Distinct* که درست بعد از *SELECT* قرار می گیرد، به *SQL Server* دستور داده که سطرهای تکراری در نتایج را حذف نماید. بنابراین *Query* زیر هر کد ایالتی را فقط یک بار باز می گرداند به طور وضوح لیستی که شما جستجو می کنید.

```
SELECT DISTINCT State From Customer
```

نکته : همتای کلید واژه *Distinct* ، *All*می باشد که *SQL Server* را برای بازگرداندن همه سطرها آگاه می سازد خواه آن واحد باشد یا خیر .*All* پیش فرض دستور *select* است ، پس نیازی به نوشتن آن نیست.

دستور شرطی *WHERE* در *sql*

دستور *Where* برای اضافه کردن شرط یا شرط هایی جهت محدود کردن نتایج جستجو و یا استخراج نتایج دقیقتر برای داشتن خروجی که در ذهن ما وجود دارد استفاده می شود . این دستور باید پس از دستور *Select* و تعیین ستون ها از جدول مورد نظر به کار رود.

با استفاده از عملگرهای *AND* ، *OR* و پرانتز می توان چندین شرط را با هم ترکیب کرد .خروجی برنامه با شرط هایی که روی دستور داده شده است مطابقت داده خواهد شد .

```
Select Name , Family
From Person
Where IdNumber= " 1111"
```

در مثال بالا نام و فامیلی اشخاصی که فیلد *IdNumber* آنها برابر ۱۱۱۱ باشد توسط دستور *where* در اسکیوال برگردانده می شوند.

عملگر های And و Or در SQL

عملگرهای And و Or برای ترکیب شرط ها در دستور Where در sql استفاده می شود .

گاهی اوقات خروجی که ما میخواهیم در اس کیوال بایستی چند شرط مختلف داشته باشد . به طور مثال افرادی را میخواهیم که سن بالای ۲۳ سال و مدرک تحصیلی بالای لیسانس داشته باشند . در این حالت بایستی هر کدام از شرط ها را جداگانه تعریف کرده و سپس آنها را با هم ترکیب کنیم . برنامه هر کدام از شرط ها را بررسی میکند و خروجی را نمایش میدهد.

عملگر And برای اجرای دستور نیاز دارد تا تمام شرط های تعیین شده برای آن درست باشد.

```
SELECT * FROM Persons
WHERE FirstName=' saeed '
AND LastName=' rajabi '
```

عمگر Or فقط نیاز دارد که حداقل یکی از شرط ها درست باشد.

```
SELECT * FROM Persons
WHERE FirstName='saeed'
OR FirstName='rajabi'
```

ترکیب عملگرهای And و Or

```
Select * From Persons
Where ( ( Grade = 16 AND Major = 'Hard Ware' ) OR ( Garde = 12 AND Major
= 'SoftWare' ) ) " ;
```

دستور Order By در sql

اطلاعاتی که در دستور select در اسکیوال به عنوان خروجی نمایش داده میشود ی بی نظم و یا بهتر بگوییم بدون نظم مد نظر ما است. مقادیر خروجی در ستون های جدول بر اساس مقدار هیچ ستونی مرتب نمیشوند . با دستور دستور Order By میتوان اطلاعات جدول را بر اساس مقادیر یک یا چند ستون برحسب شاخص هایی مثل ترتیب حروف الفبا ، بزرگتر یا کوچکتر بودن اعداد و ...مرتب کرد.

ساختار دستور *Order By*

```
SELECT column_name(s)
FROM table_name
ORDER BY column_name(s) [ASC|DESC]
```

علامات [] در اطراف *where* بدین معناست که میتوانید *where* را بکار نبرید. اما اگر بکار بردید حتما باید قبل از *order* باشد *ASC* . به معنای صعودی بودن*(a to z)* و *DESC* به معنای نزولی بودن است.*(z to a)* پیش فرضش *ACS* است.

نکته : همچنین این نیز امکان پذیرست که مرتب سازی را بر مبنای بیش از یک ستون انجام دهید .برای این منظور بخش مرتب سازی کد بالا بصورت زیر در میاید:

```
ORDER BY "column_name1" [ASC, DESC], "column_name2" [ASC, DESC]
```

مثال:

```
Select Name , Family From Person
Where idnumber="11"
Order By Family
```

خروجی دستور بالا ، اشخاص با شرط ذکر شده در دستور *where* را بر اساس نام فامیلی مرتب صعودی میکند.

دستور *Group By* در *sql*

از دستور *Group By* در اسکیوال برای دسته بندی یک ستون بر حسب مقادیر مشابه فیلدهای یک ستون دیگر استفاده می شود.

در هنگام استفاده از برخی از توابع درون ساخته *SQL* که عمل محاسبه (مثل مجموع و میانگین) را بر روی داده ها انجام می دهند ، این مشکل وجود دارد که این توابع قادر به جدا کردن و متمایز کردن اطلاعات موجود در دو ستون نسبت به هم نیستند و نتایج محاسبات را به صورت کلی برای همه آنها در نظر می گیرند . در این مواقع از دستور *Group By* استفاده میکنیم.

ساختار این دستور به صورت زیر است:

```
SELECT column_name, aggregate_function(column_name)
FROM table_name
WHERE column_name operator value
GROUP BY column_name
```

دستور *Group By* در *sql* وقتی استفاده میشود که ما در حال انتخاب چند ستون هستیم و حداقل یک عملگر محاسباتی در دستور *select* داریم . در این زمان ما باید تمام ستونهای دیگر را گروه کنیم.

مثال : در مثال زیر مجموع نمرات هر دانش آموز را بصورت گروه بندی شده بر اساس فیلد نام دانش آموز به خروجی داده می شود :

```
Select Name ,Sum ( Grade ) As مجموع نمرات
From Class
Group By Name ;
```

نکته : دستور *group by* در اسکیوال از جمله ی پرکاربردترین دستورات است که یادگیری کامل این دستور ، میتواند تفاوت اسکریپت نویسی حرفه ای در *sql* باشد.

دستور شرطی Having در sql

دستور Having در اسکیوال برای افزودن شرط به توابع درون ساخته SQL استفاده می شود ، زیرا از دستور Where نمی توان برای کار با مقادیر خروجی توابع درون ساخته SQL استفاده کرد .

به عبارت دیگر دستور Having در sql برای اعمال شرط به ستون ها اعمال می شود و همان کاری را می کند که Where در رکوردها انجام می دهد . دستور Having معمولا با دستور Group By می آید.

ساختار دستور شرطی Having در sql به صورت زیر است:

```
SELECT column_name, aggregate_function(column_name)
FROM table_name
WHERE column_name operator value
GROUP BY column_name
HAVING aggregate_function(column_name) operator value
```

مثال : در مثال زیر همچون **مثال بخش دستور** group by عمل شده ، با این تفاوت که بعد از گروه بندی بر اساس نام دانش آموزان و دادن خروجی جمع نمرات آنها ، خروجی شامل دانش آموزانی است که مجموع نمرات آنها بالاتر از ۲۵ باشد:

```
Select Name  , Sum ( Grade ) As جمع نمره مجموع نمرات
From Students
Group By Name
Having Sum ( Grade ) > 25
Order By Family
```

نکته مثال : در مثال بالا به جای having نمیتوان از where استفاده کرد و برای اعمال شرط بر روی فیلدهای محاسباتی آمده در دستور group by باید از جمله شرطی having استفاده کرد.

دستور Insert در اسکیوال

دستور Insert در sql برای وارد کردن و ایجاد یک رکورد جدید در جدول استفاده می شود.

دو روش استفاده از این دستور وجود دارد:

```
INSERT INTO table_name
VALUES (value1, value2, value3,...)

INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

نکته : می توان در دستور *Insert* در اسکیوال تعیین کرد که مقادیر مورد نظر به ترتیب به کدام ستون های جدول وارد شوند . برای این منظور ابتدا نام ستون های مورد نظر را به ترتیب جلوی نام جدول در یک پرانتز وارد کرده و با کاما از هم جدا می کنیم . سپس مقادیر متناظر را به همان ترتیب پس از واژه *Vaues* در پرانتز وارد می کنیم.

نکته : چنانچه برای فیلد یا فیلدهایی مقداری در دستور *insert* در نظر گرفته نشود ، مقادیر پیش فرض تعیین شده و در صورت تعیین نکردن مقدار پیش فرض جای آنها در جدول خالی می ماند . فقط باید به ترتیب نام ستون ها و مقادیر دقت کرد.

```
Insert into Student ( Name , Family ) Values ( "Saeed" , "Rajabi" )
```

در مثال بالا نام *Saeed* در فیلد *Name* و نام *Rajabi* در فیلد *Family* میرود. بقیه ی فیلد ها در صورت تعیین کردن مقدار پیش فرض آن مقدار را میگیرند و در صورت تعیین نکردن آن ، مقدار خالی یا *Null* میگیرند.

دستور *Update* در *sql*

دستور *Update* در اسکیوال برای تغییر اطلاعات موجود در یک فیلد و جایگزینی آن با یک مقدار جدید به کار می رود . ویرایش اطلاعات در بانک اس کیوال از جمله مهمترین اعمال هست که با استفاده از دستور *Update* امکان پذیر می باشد.

```
UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value
```

برای ویرایش کردن بیش از یک فیلد به روش زیر عمل میکنیم:

```
Update  Person
Set Name = " bahar" , Family = " shokri" , id-number=" 11 "
Where  ID =3222
```

برای ویرایش یک فیلد دستور زیر را مینویسیم:

```
Update Person
Set Name =" Ali "
Where Name = " Ahmad "
```

دستور *Delete* در اسکیوال

دستور *Delete* در *sql* برای حذف اطلاعات یک رکورد در جدول بانک اسکیوال استفاده می شود. از دستور *delete* باید با دقت استفاده شود زیرا امکان بازگرداندن رکورد های حذف شده به این راحتی ها نیست!

```
DELETE FROM table_name
WHERE some_column=some_value
```

برای پاک کردن یک رکورد با شناسه مشخص از دستور زیر استفاده میکنیم.

```
Delete From Person
Where ID = "16 "
```

دستور *Delete **

این امکان وجود دارد که با دستور *Delete ** بدون حذف یک جدول ، کلیه رکوردهای درون آن را پاک کرد.

```
Delete From Person
```

دستورات پیشرفته

SQL

دستور *TOP* در *sql*

دستور *TOP* در دستور *select* اسکیوال ، تعداد رکورهای خروجی را مشخص میکند.

دستور *top* در *sql* برای جداولی که بالای هزاران رکورد دارند بسیار مفید میباشد .خروجی با تعداد رکوردهای بالا بر روی کارایی برنامه ی مرتبط با *sql* ممکن است تاثیر بگذارد.

مثال : با مثال زیر ما ۲ رکورد از جدول را انتخاب می کنیم:

```
SELECT TOP 2 * FROM Persons
```

مثال : با مثال زیر ما میتوانیم ۵۰ درصد از رکوردها در جدول بالا را انتخاب کنیم:

```
SELECT TOP 50 PERCENT * FROM Persons
```

عملگر *Like* در *SQL*

عملگر *Like* در شرطهای *sql* ، برای الگوی خاص جستجو مورد استفاده قرار میگیرد. ساختار دستور *like* در اسکیوال به صورت زیر است:

```
SELECT column_name(s)
FROM table_name
WHERE column_name LIKE pattern
```

مثال : دستور زیر افرادی که شهر آنها با حرف *s* شروع میشوند را انتخاب مبکند.

```
SELECT * FROM Persons
WHERE City LIKE 's%'
```

مثال : دستور زیر افرادی که شهر آنها به حرف *s* ختم میشوند را انتخاب مبکند.

```
SELECT * FROM Persons
WHERE City LIKE '%s'
```

مثال : دستور زیر افرادی که شهر آنهاشامل حرف *st* میباشد را انتخاب مبکند.

```
SELECT * FROM Persons
WHERE City LIKE '%st%'
```

مثال : دستور زیر افرادی که شهر آنهاشامل حرف *st* نمیباشد را انتخاب مبکند.

```
SELECT * FROM Persons
WHERE City NOT LIKE '%st%'
```

نکته : در **مقاله ی بعدی** علامت های شرطی در *Like* را توضیح خواهیم داد.

علامت های شرطی در *Like*

یادآوری از دستور *Like* : همانطور که در **مقاله ی قبلی** در مورد دستور *like* در *sql* گفته شد ، هنگام کارکردن با رشته ها همیشه خواهان مطابقت کامل رشته ها نیستیم ، بلکه بخشی از رشته یا الگوی خاصی از آن بیشتر موردنظر است. در این هنگام می توان از دستور *LIKE* در شرط های اسکیوال ، به جای علامت = استفاده کرد.

در دستور *like* ، دو کاراکتر *%* **و** _ به ترتیب به معنی چند کاراکتر و یک کاراکتر را برای تطابق می توان بکار برد.

در آموزش *sql* این علامت ها کاربرد بسیاری دارند.

مثال : اسامی کلیه مشتریانی که آدرس آنها شامل کلمه *Main* است بطوریکه قبل *Main* هر چند کاراکتری که خواست بیاید ولی بعد از *Main* فقط یک کاراکتر بیاید را پیدا کن:

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

عملگر *In* در *sql*

عملگر *In* در اسکیوال برای مشاهده اطلاعات رکوردهایی از جدول به کار می رود که شما مقدار دقیق حداقل یکی از فیلدهای آنرا می دانید . دستور *in* در اسکیول برای کوئری های پیچیده در اس کیو ال بسیار کاربردی است.

ساختار این دستور به صورت زیر است :

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1,value2,...)
```

مثال : از جدول *Persons* مشخصات افرادی را بدهید که نام خانوادگی آنها رجبی یا یوسفی است:

```
Select * From  Persons
Where Family IN ( "رجبی" , "یوسفی" );"
```

عملگر *BETWEEN* در اسکیوال

عملگر *BETWEEN* در *sql* برای انتخاب اطلاعات در یک محدوده خاص ، در بین دو مقدار تعیین شده استفاده می شود . این مقادیر می تواند از نوع عددی ، متن یا تاریخ باشد.

نکته مهم در مورد دستور *Between* در *sql* اینست که باید نوع مدنظر با نوع داده ای فیلد های استفاده شده در دستور یکسان باشد.
نکته ی دیگر در مورد *Between* در مورد داده های متنی ، ترتیب قرار گیری حروف الفبا مد نظر برنامه است.

ساختار این دستور به صورت زیر است :

```
SELECT column_name(s)
FROM table_name
WHERE column_name
BETWEEN value1 AND value2
```

مثال : برای نمایش اطلاعات درون محدوده تعیین شده به صورت زیر عمل میکنیم :

```
SELECT * FROM Persons
WHERE LastName
BETWEEN 'Hansen' AND 'Pettersen'
```

مثال : برای نمایش اطلاعات خارج از محدوده تعیین شده از یک عملگر *NOT* قبل از دستور *Between ...* *And*استفاده می شود.

```
SELECT * FROM Persons
WHERE LastName
NOT BETWEEN 'Hansen' AND 'Pettersen'
```

*Alias*در اس کیو ال

زمانی که با دستور *select* در *sql* یک کوئری اجرا کرده ایم نام یا عنوانی که در بالای هر ستون در خروجی نمایش داده می شود ، همان نام فیلد مربوط به داده در جدول اصلی پایگاه داده است.

گاهی اوقات میخواهیم نام ستون ما در خروجی برابر با نام دلخواه ما باشد یا در مواردی که در بخشتوابع *SQL* خواهیم دید ، یک ستون جدید ایجاد می کنیم که باید نامی برای آن تعیین شود در این موارد ازویژگی *Alias* استفاده میکنیم .

از ویژگی *Alias* برای در نظر گرفتن یک نام مستعار و مجازی برای قرار گرفتن در بخش عنوان هر ستون در خروجیدستور *Select* در اسکیوال استفاده می شود.

تعریف *Alias* برای جداول : *sql* کلمه کلیدی *as* در اسکیوال این کار را انجام می دهد.

```
SELECT column_name(s)
FROM table_name
AS alias_name
```

تعریف *Alias* برای ستونها در : *sql*

```
SELECT column_name AS alias_name
FROM table_name
```

مثال:

```
Select Name As نام , Family As خانوادگی From Persons
Where id >15
Order By Family ;
```

می توان بر روی مقادیر ستون هایی که داده عددی دارند ، عملیات ریاضی (مثل ضرب ، تقسیم و ...) انجام داده و سپس نتایج محاسبات را در یک ستون جدید با عنوان دلخواه تعیین شده توسط ویژگی *Alias* در خروجی دستور *Select* نمایش داد .در این حالت ستون یا ستون های به جدول خروجی اضافه می شود.

مثال : مثال زیر کاربردی از این مورد را در دستور *select* نشان میدهد:

```
Select Name+Family As " نام و نام خانوادگی From Persons "
Where id >15
Order By Family ;
```

دستورهای پیوند (*Join*) در *SQL*

از مجموعه دستورات *Join* در اسکیوال ، برای پیوند جدول ها در پایگاه داده ها استفاده می شود .از دستور های *Join* ، برای ارتباط بین چندین جدول که با یکدیگر ارتباط دارند ، درواقع کلید خارجی برای آنها تعریف شده است، میتوان استفاده کرد.

تفاوتهای بین دستورات پیوند (*JOIN*) در *SQL*

دستور : *INNER JOIN* خروجی دستور *JOIN* یا دستور *INNER JOIN* از بین سطرهایی انتخاب میشود که حداقل یک رابطه در هر دو جدول وجود داشته باشد.

دستور : *LEFT JOIN* خروجی دستور *LEFT JOIN* از از جدول سمت چپ انتخاب میشود، حتی اگر هیچ رابطه ای با جدول سمت راست نداشته باشد.

دستور : *RIGHT JOIN* خروجی دستور *RIGHT JOIN* از از جدول سمت راست انتخاب میشود، حتی اگر هیچ رابطه ای با جدول چپ نداشته باشد.

دستور : *FULL JOIN* خروجی دستور *FULL JOIN* از بین سطرهایی انتخاب میشود که یک رابطه در یکی از جداولبایستی وجود داشته باشد.

تفاوت *Join* و *InnerJoin*

تفاوت *Join* و *InnerJoin* در *performance* آنها می باشد. نتیجه ی اجرای هر دوی آنها یکسان است اما دستور *InnerJoin* دارای *Prformance* بالاتری می باشد و به همین دلیل توصیه می شود که از دستور *InnerJoin*استفاده شود.

در مقالات زیر یک به یک این پیوند ها در اسکیوال با مثال توضیح داده شده اند:

- **مقاله ی دستور** *inner join* **در** *sql*
- **مقاله ی دستور** *left join* **در** *sql*
- **مقاله ی دستور** *right join* **در** *sql*
- **مقاله ی دستور** *full join* **در** *sql*

دستور *INNER JOIN* در *sql*

خروجی دستور *INNER JOIN* در اس کیو ال از بین سطرهایی انتخاب میشود که حداقل یک رابطه در هر دو جدول وجود داشته باشد.

تفاوت دستور *Join* و دستور *InnerJoin* در *performance* آنها می باشد. نتیجه ی اجرای هر دوی آنها یکسان است امادستور *InnerJoin* دارای *Prformance* و کارایی بالاتری می باشد و به همین دلیل توصیه می شود که از دستور *InnerJoin* استفاده شود.

شکل کلی این دستور به صورت زیر است که در آن کلمه ی کلیدی *ON* فیلد رابطه رو در دو جدول مشخص میکند:

```
SELECT column_name(s)
FROM table_name1
INNER JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

در دستور *INNER JOIN* در اسکیوال بایستی حداقل یک رابطه در هر دو جدول وجود داشته باشد که بعد از کلمه کلیدی *On* می آید.

نکته مهم : مهمترین دستور ها در آموزش *sql* ، دستور *inner join* می باشد که یادگیری کامل دستور *join*میتواند تفاوت اصلی آموزش حرفه ای اسکیوال باشد.

مثال : اگر رکوردی (خروجی) که در جدول *"Persons"* وجود دارد هیچ تطابقی با خروجی های جدول *"Orders"* نداشته باشد ، این رکوردها در خروجی این دستور *sql* نمایش داده نمیشوند.

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
INNER JOIN Orders
ON Persons.P_Id=Orders.P_Id
ORDER BY Persons.LastName
```

دستور *LEFT JOIN* در *sql*

خروجی دستور *LEFT JOIN* در اسکیوال از از جدول سمت چپ انتخاب میشود، حتی اگر هیچ رابطه ای با جدول سمت راست نداشته باشد.

شکل کلی دستور *left join* در *sql* بصورت زیر است:

```
SELECT column_name(s)
FROM table_name1
LEFT JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

نکته : در بعضی دیتابیس ها دستور *LEFT JOIN* با نام دستور *LEFT OUTER JOIN* استفاده میشود که تفاوتی در این زمینه وجود ندارد.

مثال : در این مثال جدول اصلی ، جدول *Persons* میباشد که در سمت چپ *JOIN* قرار دارد و خروجی ها بر اساس این جدول تنظیم میشوند.

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
LEFT JOIN Orders
ON Persons.P_Id=Orders.P_Id
ORDER BY Persons.LastName
```

در صورت نداشتن رابطه ی بعضی ای رکورد ها با جدول سمت راست ، فیلد های آن *null* انتخاب میشوند که با ترکیب آن با دستور *isNull* در اسکیول میتوانید خروجی های دلخواه خود را با استفاده از دستور *left join*بدست آورید

دستور *RIGHT JOIN* در *sql*

خروجی دستور *RIGHT JOIN* در اسکیوال از از جدول سمت راست انتخاب میشود، حتی اگر هیچ رابطه ای با جدول چپ نداشته باشد .ساختار دستور *right join* به صورت زیر است:

```
SELECT column_name(s)
FROM table_name1
RIGHT JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

نکته : در بعضی دیتابیس ها دستور *RIGHT JOIN* با نام دستور *RIGHT OUTER JOIN* استفاده میشود.

مثال : در این مثال جدول اصلی ، جدول *Orders* میباشد که در سمت راست *JOIN* قرار دارد و خروجی ها بر اساس این جدول تنظیم میشود.

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
RIGHT JOIN Orders
ON Persons.P_Id=Orders.P_Id
ORDER BY Persons.LastName
```

در صورت نداشته رابطه ی بعضی ای رکورد ها با جدول سمت چپ ، فیلد های آن *null* انتخاب میشوند که با ترکیب آن با دستور *isNull* در اسکیول میتوانید خروجی های دلخواه خود را با استفاده از دستور *right join*بدست آورید.

دستور *FULL JOIN* در *sql*

خروجی دستور *FULL JOIN* در *sql* از بین سطرهایی انتخاب میشود که یک رابطه در یکی از جداول بایستی وجود داشته باشد.

به طور خلاصه دستور *full join* **(** *full outer join* **) دستور بصورت** *or* **کار میکند و دستور** *inner join* **بصورت**. *and*

```
SELECT column_name(s)
FROM table_name1
FULL JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

مثال : در مثال زیر ما میخواهیم با استفاده از دستور *full join* در اسکیوال ، همه ی افراد با سفارش هایشان و همه ی سفارشات با افراد مربوط به آنها را در خروجی نمایش دهیم.

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
FULL JOIN Orders
ON Persons.P_Id=Orders.P_Id
ORDER BY Persons.LastName
```

دستور *UNION* در *sql*

از دستور *Union* برای ترکیب و ادغام دو یا چند ستون مختلف از ۲ یا چند جدول و نشان دادن آنها در یک ستون مشترک استفاده می شود .

در دستور *union* **، نوع داده ای ستون های انتخاب شده برای ترکیب باید یکسان باشند.**

دستور *Union* در هنگام ترکیب فیلد ها ، در صورت برخورد با مقادیر تکراری آنها را حذف کرده و از هر مقدار یک نمونه را نمایش می دهد . ساختار دستور *union* به صورت زیر است:

```
SELECT column_name(s) FROM table_name1
UNION
SELECT column_name(s) FROM table_name2
```

مثال:

```
SELECT E_Name FROM Employees_Norway
UNION
SELECT E_Name FROM Employees_USA
```

برای مشاهده تمام مقادیر ، حتی مقادیر تکراری باید از دستور *Union ALL* استفاده کرد

ساختار این دستور به صورت زیر است:

```
SELECT column_name(s) FROM table_name1
UNION ALL
SELECT column_name(s) FROM table_name2
```

مثال:

```
SELECT E_Name FROM Employees_Norway
UNION ALL
SELECT E_Name FROM Employees_USA
Result
```

در آموزش *sql* ، دستور *union* و دستور *union all* کاربرد های ویژه و خاصی دارند.

دستور *SELECT INTO* در *sql*

از دستور *Select Into* در موارد زیر استفاده می شود:

۱. ایجاد یک (*Back Up* نسخه پشتیبان) از یک جدول.
۲. ایجاد یک آرشیو از رکوردهای یک جدول.
۳. قرار دادن برخی از رکوردها یا فیلدهای مورد نظر از یک جدول در یک جدول جدید.
۴. ایجاد یک نسخه پشتیبان از کل یک پایگاه داده در یک پایگاه داده جدید.

تمامی مثالهای زیر نحوه استفاده از دستور *select into* را نشان میدهد:

- در این مثال (ساختار) ما می میخواهیم برخی از رکوردها یا فیلدهای مورد نظر از یک جدول در یک جدول جدید قرار دهیم:

```
We can select all columns into the new table:
SELECT *
INTO new_table_name [IN externaldatabase]
FROM old_tablename
```

- در این مثال (ساختار) ما تنها می توانیم ستون ها را برای ایجاد یک جدول جدید انتخاب کنیم:

```
SELECT column_name(s)
INTO new_table_name [IN externaldatabase]
FROM old_tablename
```

- در این مثال ما میخواهیم یک (*Back Up* نسخه پشتیبان) از یک جدول *Persons* ایجاد کنیم:

```
SELECT *
INTO Persons_Backup
FROM Persons
```

- در این مثال ما میخواهیم یک کپی از جدول *Persons* را به یک پایگاه داده دیگر *(Backup.mdb)* اضافه کنیم:

```
SELECT *
INTO Persons_Backup IN 'Backup.mdb'
FROM Persons
```

- در این ساختار ما میتوانیم یک تعدادی از فیلدها را به جدول جدید کپی کنیم:

```
SELECT LastName,FirstName
INTO Persons_Backup
FROM Persons
```

- در این مثال ما میخواهیم با داشتن یک دستور شرطی فیلدهایی را از جدول *Persons* به جدول *Persons_Backup* اضافه کنیم:

```
SELECT LastName,Firstname
INTO Persons_Backup
FROM Persons
WHERE City='Sandnes'
```

- انتخاب داده از بیش از یک جدول هم در دستور *select into* امکان پذیر است . در مثال زیر جدول *"Persons_Order_Backup"* شامل داده ها از دو جدول *"Persons"* و *"Orders"* میباشد.

```
SELECT Persons.LastName,Orders.OrderNo
INTO Persons_Order_Backup
FROM Persons
INNER JOIN Orders
ON Persons.P_Id=Orders.P_Id
```

ذکر نام یک پایگاه داده جدید در دستور *Select Into* اختیاری است . چنانچه نامی در این قسمت **ذکر نشود** ، برنامه نسخه پشتیبان را در همان پایگاه داده موجود ایجاد می کند و چنانچه نامی **ذکر شود** ، برنامه نسخه پشتیبان را در پایگاه داده ذکر شده ایجاد می کند.

مقدار *Null* در اس کیو ال

در حالت کلی مقادیر ستون ها میتوانند مقدار *null* داشته باشند. زمانی که یک ستون اختیاری باشد ، شما میتوانیدمقدار *Null* در آن ذخیره کنید. اختیاری به این معنی که میتوانید خالی رهایش کنید .

مقدار پیش فرض در صورت مقدار ندادن به فیلدی در یک رکورد , مقدار *Null* است.

با استفاده از دستور *is null* در *sql* میتوانیم فیلدهایی که مقادیر تهی دارند را با در دستور *Select* انتخاب کنیم.

مقدار *null* از جمله مقادیری است که زیاد باهاش در آموزش *sql* سر و کار خواهیم داشت.

نکته ی بسیاری مهم در مورد *null* یا تهی اینست که این مقدار با مقدار رشته ای *space* یا ' ' تفاوت کامل دارد. همچنین *null* را نباید با مقدار رشته ای آن یعنی *'null'* اشتباه بگیریم.

دستور *ISNULL* در *sql*

از دستور *ISNULL* در اسکیوال زمانی استفاده میشود که شما میخواهید تمام فیلدهای *null* را در خروجی داشته باشید.

مقدار *null* در اس کیو ال با *space* ' ' تفاوت دارد.

ساختار دستور *is null* به همراه مثال آن به صورت زیر است که در این مثال تمام رکوردهایی که حاوی فیلد آدرس *null* هستند انتخاب می شوند.

```
SELECT LastName,FirstName,Address FROM Persons
WHERE Address IS NULL
```

بر عکس دستور *is null* در *sql* ، دستور *is not Null* هست که مقادیری را برمیگرداند که *Null* نباشد که بیشتر از این شکل از این دستور استفاده مشود.

مثال : در مثال زیر رکوردهایی که آدرس آنها تهی (*null*) نیستند ، انتخاب و در خروجی این دستور در اسکیوال می روند.

```
SELECT LastName,FirstName,Address FROM Persons
WHERE Address IS NOT NULL
```

دستورات تعریف داده ها

(DDL)

دستور *CREATE DATABASE* در *sql*

برای ایجاد یک پایگاه داده جدید از دستور *CREATE DATABASE* استفاده می شود.

ساختار دستور *create database* در اسکیوال به صورت زیر است:

```
CREATE DATABASE database_name
```

برای ایجاد یک دیتابیس با نام *mySite_db* به صورت زیر عمل میکنیم:

```
CREATE DATABASE mySite_db
```

دستور ***Create Table*** در ***sql***

برای ایجاد یک جدول جدید در پایگاه داده *sql* ، از دستور *Create Table* استفاده می کنیم .
برای ایجاد جدول در اسکیوال بایستی نکات زیر را در نظر گرفت:

۱. تعیین یک نام منحصر به فرد برای جدول.
۲. تعیین تعداد ستون های (فیلد ها) جدول و نام آنها که نباید تکراری باشد.
۳. تعیین نوع داده ای ستون های جدول و اندازه آنها در صورت نیاز.

ساختار دستور *create table* در *sql* در ساده ترین حالت ، به صورت زیر است:

```
CREATE TABLE table_name
(
column_name1 data_type,
column_name2 data_type,
column_name3 data_type,
....).
```

برای مثال برای ایجاد جدول *Persons* با فیلدهای زیر به صورت زیر عمل میکنیم:

```
CREATE TABLE Persons
(
P_Id int,
```

```
LastName varchar(255),
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)
```

دستور ***Drop*** در اسکیوال

تمامی جدول ها ، اندیس ها ، و حتی دیتابیس ها میتوانند با دستور *Drop* در *sql* حذف شوند. شیوه های استفاده ازدستور *drop* در *sql* به شرح زیر است:

دستور*Drop Index*

از دستور *DROP INDEX* برای حذف یک اندیس موجود در یک فیلد جدول استفاده می شود.

```
DROP INDEX table_name.index_name
```

دستور*Drop Table*

از دستور *DROP TABLE* برای حذف یک جدول در پایگاه داده استفاده می شود.

```
DROP TABLE table_name
```

دستور*Drop Database*

از دستور *DROP DATABASE* برای حذف یک پایگاه داده به صورت کامل استفاده می شود

```
DROP DATABASE database_name
```

دستور*TRUNCATE*

برای حذف کلیه اطلاعات موجود در یک جدول بدون حذف ستون های آن از دستور *TRUNCATE* استفاده می کنیم:

```
TRUNCATE TABLE table_name
```

این دستور باعث خالی شدن جدول از اطلاعات می شود ، بدون اینکه خود آن و ساختارش تغییری کند .

مثال:

```
Truncate Table Mydb;
```

دستور ***Create Index*** در ***sql***

از دستور *Create Index* در اسکیوال برای ایجاد اندیس در ستون های جدول استفاده می شود.

در یک جدول می توان برای یک یا چند ستون جدول اندیس ایجاد کرد ، که این اندیس ها باعث بالا رفتن سرعت جستجو در رکوردهای جدول می شود .

اندیس یک شماره است که به هر سطر جدول اختصاص داده می شود و معمولا از صفر شروع می شود . اندیس ها از دید کاربر مخفی هستند و هر اندیس یک نام منحصر به فرد دارد.

برای ایجاد یک اندیس با داده های تکراری به صورت زیر عمل میکنیم:

```
CREATE INDEX index_name
ON table_name (column_name)
```

مثال:

```
CREATE INDEX PIndex
ON Persons (LastName)
```

برای ایجاد یک اندیس یا *index* ، با داده های منحصر به فرد *(unique)* به صورت زیر عمل میکنیم:

```
CREATE UNIQUE INDEX index_name
ON table_name (column_name)
```

برای ایجاد یک اندیس *(index)* برای بیش از یک فیلد ، باید نام فیلد های مورد نظر را به ترتیب در پرانتز بعد از نام جدول وارد کرد.

```
CREATE INDEX PIndex
ON Persons (LastName, FirstName)
```

دستور ***Alter Table*** در ***sql***

از دستور *Alter Table* برای اضافه کردن یا حذف کامل یک ستون از یک جدول و یا تغییر نوع داده ی فیلد موجود در پایگاه داده *sql* استفاده می شود.

اضافه کردن فیلد به جدول در اسکیوال

برای اضافه کردن یک ستون (فیلد) جدید به یک جدول موجود در اسکیوال از دستور *ALTER TABLE* استفاده می شود:

```
ALTER TABLE table_name
ADD column_name datatype
```

حذف یک فیلد جدول در ***sql***

برای حذف یک ستون از یک جدول موجود به روش زیر عمل می شود:

```
ALTER TABLE table_name
DROP COLUMN column_name
```

تغییر نوع داده ی یک فیلد جدول در ***sql***

برای تغییر نوع داده ی یک فیلد از جدول در اسکیوال از دستور زیر استفاده میکنیم:

```
ALTER TABLE table_name
ALTER COLUMN column_name datatype
```

همانطور که دیدید ، دستور *alter table* یکی از مهمترین دستورات در *sql* است که با ۳ پارامتر و کلمه ی کلیدی زیر ، تغییرات جدول را با استفاده از کد در *sql* راحت میکنند:

- *ADD*
- *DROP COLUMN*
- *DROP COLUMN*

نکته مهم : برای تغییر نام های فیلدهای جدول در اسکیوال باید از *sp* های اسکیوال استفاده کرد . در مقاله ی زیر ، تغییر نام فیلدها همراه با مثال توضیح داده شده است:

تغییر نام فیلد در یک جدول با استفاده از دستور *sql*

برای تغییر نام فیلد در جدول از تغییر کد باید به روش زیر و با کمک *stored procedure* های خود اسکیوال ، عمل کرد:

```
sp_RENAME 'table_name.old_name' , 'new_name', 'COLUMN'
```

مثال از تغییر نام ستون و فیلد *TerritoryID* به *TerrID* از جدول *Sales* از پایگاه داده ی: *Db_name*

```
USE Db_name;
GO
EXEC sp_rename 'Sales.TerritoryID', 'TerrID', 'COLUMN';
GO
```

ایجاد محدودیت برای فیلد ها در اسکیوال *(Constraints)*

*Constraint*ها در *sql* یا همان محدودیت ها در اس کیو ال ، برای محدود کردن نوع داده هایی که میتواند در جدول تعریف شود مورد استفاده قرار میگیرد.

این محدودیت ها را زمانی که یک جدول جدید ایجاد میکنیم یا زمانی که نوع داده ای جدول را تغییر میدهیم بایستی در نظر داشته باشیم.

Constraint **های** *sql* شامل موارد زیر میشوند:

- *NOT NULL*
- *UNIQUE*
- *PRIMARY KEY*
- *FOREIGN KEY*
- *CHECK*
- *DEFAULT*

محدودیت ***Primary Key , SQL PRIMARY KEY Constraint*** یا کلید اصلی در اس کیو ال

از محدودیت *Primary Key* زمانی استفاده میکنیم که میخواهیم فیلد کلید اصلی در *sql* ، تعریف کنیم.

محدودیت *Primary Key* یکی از مهمترین محدودیت‌ها می‌باشد. فیلد کلید، مقدار تکراری قبول نمی‌کند و بیشتر برای تفکیک و جستجوی رکوردها مورد استفاده قرار می‌گیرد.

- **فیلد کلید اصلی در اسکیوال هیچ وقت *Null* نمی‌پذیرد. پس ستون *Allow Nulls* برای این فیلد نباید چک خورده باشد.**
- **هر جدول فقط می تواند یک کلید اولیه *(Primary Key)* داشته باشد.**

اعمال محدودیت ***PRIMARY KEY*** در هنگام ساخت جدول

برای تعریف محدودیت *Primary Key* در دستور *create table* از دستور *SQL* زیر استفاده می کنیم:

```
CREATE TABLE Persons
(
P_Id int NOT NULL PRIMARY KEY,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)
```

برای تعریف محدودیت *Primary Key* روی چند ستون از دستور *SQL* زیر استفاده می کنیم:

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
```

```
City varchar(255),
CONSTRAINT pk_PersonID PRIMARY KEY (P_Id,LastName)
)
```

محدودیت ***PRIMARY KEY*** پس از ساخته شدن جدول

برای تغییر دادن محدودیت *Primary Key* روی یک ستون از دستور *SQL* زیر استفاده می کنیم:

```
ALTER TABLE Persons
ADD PRIMARY KEY (P_Id)
```

برای تغییر دادن محدودیت *Primary Key* روی چند ستون از دستور *SQL* زیر استفاده می کنیم

```
ALTER TABLE Persons
ADD CONSTRAINT pk_PersonID PRIMARY KEY (P_Id,LastName)
```

برای از بین بردن یک***PRIMARY KEY***

برای از بین بردن محدودیت *Primary Key* یا کلید اصلی ، از دستور *SQL* زیر پیروی می کنیم:

```
ALTER TABLE Persons
DROP CONSTRAINT pk_PersonID
```

محدودیت ***Foreign Key , SQL FOREIGN KEY Constraint*** یا کلید خارجی در اس کیوال

یک *FOREIGN KEY* یا کلید خارجی در اسکیوال ، در یک جدول به یک *PRIMARY KEY* در جدولی دیگر اشاره می کند.

- **برای ارتباط بین جداول از محدودیت** *Foreign Key* **استفاده می‌شود و در اصل** *Relational Integrity***را فراهم می‌کند.**
- **به عبارت ساده تر ، کلید خارجی همان مقدار کلید اصلی از جدول دیگر است.**

- **مقدار یک کلید خارجی میتواند تکراری باشد.**

فیلدهای کلید خارجی ، سه نوع محدودیت را ایجاد می‌کنند:

۱. اجازه حذف رکوردی از جدول پدر نداریم که فرزندی داشته باشد.
۲. اجازه *insert* رکورد در جدول فرزند با کد پدری که در جدول پدر موجود نیست را نداریم.
۳. مقدار کلید پدری را که دارای فرزند هست را نمی‌توانیم تغییر دهیم.

همچنین محدودیت *FOREIGN KEY* از وارد کردن داده های نامعتبر در جدول جلوگیری میکند. (در فیلد *foreign key)* ، دلیل آن هم، این است که با مقادیر جدول دیگر که به آن اشاره می کند تطابق ندارد.

برای تعریف یک محدودیت *FOREIGN KEY* در دستور *create table* ، روی یک ستون از دستور *SQL*زیر استفاده می کنیم:

```
CREATE TABLE Orders
(
O_Id int NOT NULL PRIMARY KEY,
OrderNo int NOT NULL,
P_Id int FOREIGN KEY REFERENCES Persons(P_Id)
)
```

برای تعریف یک محدودیت *FOREIGN KEY* روی چند ستون از دستور *SQL* زیر استفاده می کنیم:

```
CREATE TABLE Orders
(
O_Id int NOT NULL,
OrderNo int NOT NULL,
P_Id int,
PRIMARY KEY (O_Id),
CONSTRAINT fk_PerOrders FOREIGN KEY (P_Id)
REFERENCES Persons(P_Id)
)
```

محدودیت ***FOREIGN KEY*** پس از ساخته شدن جدول

برای تغییر دادن یک محدودیت *FOREIGN KEY* روی یک ستون از دستور *SQL* زیر استفاده می کنیم:

```
ALTER TABLE Orders
ADD FOREIGN KEY (P_Id)
REFERENCES Persons(P_Id)
```

برای تغییر دادن محدودیت *FOREIGN KEY* روی چند ستون، از دستور *SQL* زیر استفاده می کنیم:

```
ALTER TABLE Orders
ADD CONSTRAINT fk_PerOrders
FOREIGN KEY (P_Id)
REFERENCES Persons(P_Id)
```

از بین بردن ***FOREIGN KEY***

برای از بین بردن محدودیت *FOREIGN KEY* ، از دستور *SQL* زیر استفاده می کنیم:

```
ALTER TABLE Orders
DROP CONSTRAINT fk_PerOrders
```

محدودیت *SQL CHECK Constraint* , *Check* یا محدودیت های خاص در *sql*

از محدودیت *Check* یا محدودیت خاص در اس کیو ال زمانی استفاده میکنیم که بخواهیم برای یک فیلد، یک سری محدودیت خاص روی فیلد تعریف کنیم.

برای مثال : مقادیر داخل فیلد بایستی حتما بزرگتر از ۱۰ باشد در غیر اینصورت در هنگام ورود اطلاعات، کاربر با خطا مواجه گردد.

نکته : محدودیت *Check* را میتوان برای ستون و جدول تعریف کرد.

محدودیت *CHECK* هنگامی که جدول ساخته می شود

برای تعریف یک محدودیت *Check* در دستور *create table* ، روی یک ستون از دستور *SQL* زیر استفاده می کنیم:

```
CREATE TABLE Persons
(
P_Id int NOT NULL CHECK (P_Id>0),
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)
```

برای تعریف یک محدودیت *Check* روی چند ستون از دستور *SQL* زیر استفاده می کنیم :

```
CREATE TABLE Persons

(P_Id int NOT NULL,

LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
CONSTRAINT chk_Person CHECK (P_Id>0 AND City='Sandnes')
)
```

محدودیت *CHECK* پس از ساخته شدن جدول

برای تغییر دادن یک محدودیت *Check* روی یک ستون از دستور *SQL* زیر استفاده می کنیم:

```
ALTER TABLE Persons
ADD CHECK (P_Id>0)
```

برای تغییر دادن یک محدودیت *Check* روی چند ستون از دستور *SQL* زیر استفاده می کنیم:

```
ALTER TABLE Persons
ADD CONSTRAINT chk_Person CHECK (P_Id>0 AND City='Sandnes')
```

از بین بردن محدودیت *CHECK*

برای از بین بردن محدودیت *Check* یا محدودیت خاص ، از دستور زیر استفاده می کنیم:

```
ALTER TABLE Persons
DROP CHECK chk_Person
```

محدودیت ***Default , SQL DEFAULT Constraint*** یا مقدار پیش فرض در ***sql***

از محدودیت *Default* یا مقدار پیش فرض در اس کیو ال ، میتوانیم زمانی استفاده کنیم که ، در هنگام ساخت جدول و تعیین نوع فیلدها، مقدار *Default Value* و یا پیش فرض به یک فیلد بدهیم.

در واقع محدودیت در اینجا به این معنی می باشد که اگر کاربر در هنگام درج یا به روزرسانی مقدار آن فیلد را وارد نکرد، مقدار *Default Value* به جای آن قرار بگیرد.

محدودیت *DEFAULT* در هنگام ساخت جدول

برای تعریف یک محدودیت *Default* در دستور *create table* از دستور *SQL* زیر استفاده می کنیم:

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255) DEFAULT 'Sandnes'
)
```

از محدودیت *Default* می توان در مواقعی که میخواهیم از *function* ها در *SQL* استفاده کنیم،ز دستور *SQL*به صورت زیر استفاده می کنیم:برای مثال تابع *()GETDATE*

```
CREATE TABLE Orders
(
O_Id int NOT NULL,
OrderNo int NOT NULL,
P_Id int,
OrderDate date DEFAULT GETDATE()
)
```

محدودیت *DEFAULT* پس از ساخت جدول

برای تغییر دادن محدودیت *Default* از دستور *SQL* زیر استفاده می کنیم:

```
ALTER TABLE Persons
ALTER COLUMN City SET DEFAULT 'SANDNES'
```

از بین بردن محدودیت*DEFAULT*

برای از بین بردن محدودیت *Default* یا مقدار پیش فرض در *sql* ، از دستور *SQL* زیر استفاده می کنیم:

```
ALTER TABLE Persons
ALTER COLUMN City DROP DEFAULT
```

محدودیت ***NOT NULL*** در ***sql***

از محدودیت *NOT NULL* در دستور *create table* زمانی استفاده مکنیم که میخواهیم مقدار *Allow Null*فیلدها را غیر فعال کنیم.

در حقیقت با تعریف این محدودیت اجازه نمیدهیم که فیلد ما مقدار *Null* بگیرد. پس در هنگام درج یا به روزرسانی، اگر کاربر مقداری را در این فیلدها وارد نکرد، جلوی انجام عملیات گرفته می‌شود.

برای تعریف محدودیت *NOT NULL* از دستور *SQL* زیر استفاده می کنیم:

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
```

```
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)
```

که در مثال بالا *p_Id* و *LastName* اجازه ی *null* بودن ندارند ولی دیگر فیلدها *allow null* هستند.

نکته : پیش فرض برای فیلدها در دستور *create table* ، مقدار دهی آنها *allow null* است.

محدودیت ***Unique , SQL UNIQUE Constraint*** در ***sql***

از محدودیت *Unique* زمانی استفاده میکنیم که بخواهیم مقادیر بعضی از فیلدها تکراری نباشند.

این فیلدها ممکن است کلید نباشند ، اما بنابر لزوم برنامه بایستی غیر تکراری باشند. به عنوان مثال فیلد آدرس ایمیل، کلید نیست اما باید *Unique* یا واحد باشد. برای این کار از *Unique Index* ها استفاده می‌کنیم.

- **محدودیت *PRIMARY KEY* به صورت خودکار، محدودیت *Unique* را نیز دارد.**
- **در هر جدول بیش از یک فیلد نیز میتواند از محدودیت *Unique* استفاده کند.**

محدودیت ***UNIQUE*** در هنگام ساخت ***TABLE***

برای تعریف محدودیت *Unique* در دستور *create table* در *sql* ، روی یک ستون از کلمه کلیدی *UNIQUE*بصورت زیر استفاده می کنیم:

```
CREATE TABLE Persons
(
P_Id int NOT NULL UNIQUE,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)
```

برای تعریف محدودیت *Unique* روی چند ستون از دستور *SQL* زیر استفاده می کنیم:

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
CONSTRAINT uc_PersonID UNIQUE (P_Id,LastName)
)
```

محدودیت ***UNIQUE*** پس از ساخت جدول

برای تغییر دادن محدودیت *Unique* روی یک ستون از دستور *SQL* زیر استفاده می کنیم:

```
ALTER TABLE Persons
ADD UNIQUE (P_Id)
```

برای تغییر دادن محدودیت *Unique* روی چند ستون از دستور *SQL* زیر استفاده می کنیم:

```
ALTER TABLE Persons
ADD CONSTRAINT uc_PersonID UNIQUE (P_Id,LastName)
```

از بین بردن یک محدودیت***UNIQUE***

برای از بین بردن محدودیت *Unique* از دستور *SQL* زیر استفاده می کنیم:

```
ALTER TABLE Persons
DROP CONSTRAINT uc_PersonID
```

فیلد *identity* در *sql*

فیلد *identity* در *sql* به این معناست که به ازای هر رکوردی که در جدول وارد می کنیم ، فیلد *identity*شده ، به صورت اتوماتیک مقدار دهی خواهد شد و نمی توانیم این فیلد را مقدار دهی کنیم .

فیلد *identity* دو خصوصیت دارد:

Identity Increment : که مشخص می کند مقدار فیلد چند تایی افزایش پیدا کند، به طور پیش فرض مقدار آن یک است و می تواند اعداد مثبت و همین طور منفی نیز بگیرد.

Identity Seed : مشخص می کند مقدار این فیلد از چه عددی شروع شود، به طور پیش فرض از یک شروع می شود ولی می تواند از اعداد منفی و هر عدد مثبتی برای شروع فیلد *Identity* استفاده کرد.

برای استفاده از فیلد *identity* در *sql* از ساختار زیر استفاده مکنیم:

```
CREATE TABLE Persons
(
P_Id int PRIMARY KEY IDENTITY,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255)
)
```

نکته مهم : اگر رکوردهای جدول را با *Delete* حذف کنیم مقدار آخرین *Identity* حفظ خواهد شد و درج با عدد بعدی آن صورت می گیرد، اگر بخواهیم بعد از حذف داده ها درج با مقدار *Seed* تعریف شده صورت گیرد به جای *Delete* از **دستور** *TRUNCATE table_name* استفاده می کنیم، به این ترتیب لاگ هم ذخیره نخواهد شد ، ضمن اینکه سرعت حذف رکوردها بسیار بیشتر است.

آموزش *view* در *sql*

view در *sql* در واقع همان جداول مجازی هستند که توسط آن می توانید عملیات خاصی رو که شامل دستورات *sql* میشه رو انجام بدهید و به صورت خروجی داشته باشید.

کاربرد *view* در اسکیوال

از مزیت های *view* این است که بعد از یک بار اجرا به صورت موقت در سرور ذخیره می شود و برای مراجعات بعدی از همان استفاده می شود و سرعت خیلی بالایی دارد.

ساختار دستور *View* در *sql* به صورت زیر است:

```
CREATE VIEW view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition
```

توابع *SQL*

تابع Avg در sql

تابع Avg در اسکیوال میانگین یک ستون عددی را برمیگرداند.

ساختار دستور avg به صورت زیر است:

```
SELECT AVG(column_name) FROM table_name
```

تابع COUNT در اسکیوال

تابع COUNT در sql تعداد سطرهای موجود در یک فیلد را شمرده و مقدار آنرا بر می گرداند . به تابع count در اسکیوال ، تابع سطر شمار هم می گویند.

نکته مهم : تابع count در اس کیو ال ، فیلدهایی که مقدار آنها خالی یا تکراری باشد را نمی شمارد . برای شمارش کلیه فیلدها (حتی تکراری) باید قبل از نام ستون در دستور Count عبارت Distinct ذکر شود.

ساختار این دستور به صورت زیر است:

```
SELECT COUNT(*) FROM table_name
```

تابع First در sql

تابع First در اسکیوال مقدار اولین رکورد را در یک فیلد بر می گرداند . ترتیب در تابع first ، همان ترتیب قرار گیری رکوردها در جدول است.

ساختار دستور first در sql به صورت زیر است:

```
SELECT FIRST(column_name) FROM table_name
```

تابع Last در sql

تابع Last در اسکیوال مقدار آخرین رکورد را در یک فیلد بر می گرداند . ترتیب در دستور و تابع last ، همان ترتیب قرار گیری رکوردها در جدول است.

ساختار تابع last در sql به صورت زیر است:

```
SELECT LAST(column_name) FROM table_name
```

تابع Max در اسکیوال

تابع Max در sql بیشترین مقدار موجود در بین مقادیر فیلدهای یک ستون را بر می گرداند.

تابع max در sql با ستون هایی که داده آنها از نوع عددی یا حروفی باشد ، می تواند به کار رود . در فیلدها با مقادیر عددی تابع بزگترین عدد و در فیلدها با مقادیر متنی ، تابع max کلمه ای که به ترتیب حروف الفبا از آ تا ی در فارسی و A تا Z در انگلِسی در آخرین رده باشد ، را بر می گرداند.

ساختار این دستور به صورت زیر است:

```
SELECT MAX(column_name) FROM table_name
```

تابع Min در اس کیو ال

تابع Min در sql کمترین مقدار موجود در بین مقادیر فیلدهای یک ستون را بر می گرداند .

تابع min در اس کیوال با ستون هایی که داده آنها از نوع عددی یا حروفی باشد ، می تواند به کار رود . در فیلدها با مقادیر عددی تابع کوچکترین عدد و در فیلدها با مقادیر متنی ، تابع کلمه ای که به ترتیب حروف الفبا از آ تا ی در فارسی و A تا Z در انگلِسی در بالاترین رده باشد ، را بر می گرداند.

ساختار دستور min در sql به صورت زیر است:

```
SELECT MIN(column_name) FROM table_name
```

تابع Sum در اسکیوال

تابع Sum در sql مجموع مقادیر اعداد در یک فیلد را محاسبه کرده و به عنوان خروجی بر می گرداند .
تابع Sum() باید با فیلدهایی که داده آنها از نوع عددی است ، به کار رود.

ساختار دستور sum در sql به صورت زیر است:

```
SELECT SUM(column_name) FROM table_name
```

تابع UCase در sql

تابع UCase در اسکیوال مقادیر فیلد را به حروف بزرگ تبدیل میکند.

ساختار دستور Ucase در sql به صورت زیر است:

```
SELECT UCASE(column_name) FROM table_name
```

تابع lcase در sql

تابع lcase در اسکیوال مقادیر فیلد را به حروف کوچک تبدیل میکند.

ساختار دستور LCase در sql به صورت زیر است:

```
SELECT LCASE(column_name) FROM table_name
```

تابع Mid در اسکیوال

تابع mid برای استخراج کاراکتر از فیلد متنی ، از کاراکتر تعیین شده تا چند کاراکتر بعد که تعیین میکنیم ، بکار میرود.

نکته : مقدار تعداد کاراکتر (lenght) در دستور mid اسکیوال ، اختیاری است و در صورت نزاشتن این مقدار ، برش و استخراج کاراکتر از نقطه شروع تا آخر رشته ، انجام میگیرد.

ساختار دستور mid در sql به صورت زیر است:

```
SELECT MID(column_name,start[,length]) FROM table_nam
```

تابع Len در اسکیوال

تابع len در sql ، طول (تعداد کاراکترهای) یک فیلد متنی رشته ای را برمیگرداند.

ساختار دستور len در sql به صورت زیر است:

```
SELECT LEN(column_name) FROM table_name
```

تابع Round در اسکیوال

تابع round در sql برای round (روند کردن) کردن یک مقدار عددی به تعداد اعشار مشخص استفاده میشود. نام ستون و تعداد رقم اعشار را داخل پرانتز این دستور تعیین میکنیم.

ساختار دستور round در sql به صورت زیر است:

```
SELECT ROUND(column_name,decimals) FROM table_name
```

تابع Now در sql

تابع now در اسکیوال تاریخ و ساعت جاری سیستم را برمیگرداند.

ساختار دستور now در sql به صورت زیر است:

```
SELECT NOW() FROM table_name
```

تابع Format در اسکیوال

تابع format در sql چگونگی نمایش یک فیلد را مشخص میکند.

ساختار دستور format در sql به صورت زیر است:

```
SELECT FORMAT(column_name,format) FROM table_name
```

مثال چگونگی کار تابع format در اسکیوال را توضیح میدهد:

```
SELECT ProductName, Price, FORMAT(Now(),'YYYY-MM-DD') AS PerDate
FROM Products
```

تابع isNull در اسکیوال

با تابع *isNull* در اسکیوال میتوان تعیین کرد که در صورتی که مقدار یک فیلد از رکورد برابر مقدار *NULL*بود ، خروجی به جای مقدار *NUll* ، چه مقدار دیگری شود.

به این صورت که این تابع میگوید اگر *Null* بود خروجی چه شود.

ساختار تابع *isNull* در *sql* به صورت زیر است:

```
SELECT ISNULL(column_name,value)

FROM table_name
```

مثال : استفاده از این تابع مهم در فیلدهای ترکیبی یا محاسباتی بسیار زیاد است:

```
SELECT ProductName,UnitPrice*(UnitsInStock+ISNULL(UnitsOnOrder,0))
FROM Products
```

نکته مهم : در مثال بالا در صورتی که *ISNULL UnitsOnOrder,0* را قرار ندهیم و مستقیما *UnitsOnOrder* را قرار دهیم ، ممکن است در صورت *Null* بودن یکی از مقادیر این فیلد در رکوردهای جدول محصولات ، با ارور مواجه شویم.

نکته : یکی از توابع بسیار کاربردی که برنامه نویسان با نادیده گرفتن آن و ارجاع کار این تابع به بخش برنامه نویسی برنامه هاشون از کارایی سیستم میکاهند، تابع و دستور *isNull* در اسکیوال می باشد.