مقدمه

تمام كارها كارها داده هائي دارند , اين امرمستلزم داشتن يك روش يا مكنيزم سازمان يافته براي نگهداشت داده است.اين مكانيزم تحت عنوان سيستم مديريت بانك اطلاعاتي (DBMS)شناخته شده است. سيستم هاي مديريت بانك اطلاعاتي سا لهاست كه مطرح هستند, و بسياري از آنها كار خود را بصورت سيستم هاي مديريت فايل در كا مپيوترهاي بزرگ آغاز نموده اند.با وجود تكنولوژيهاي امروزي, كاربرد سيستمهاي مديريت بانك اطلاعاتي در جهات ديگر آغاز شده, و دلايل اين امر, كارهاي اداري وتجاري رو به رشد, حجم بالاي داده هاي مجتمع, و ا لبته تكنولوژيهاي اينترنت است.

موج نوين مديريت اطلاعات از طريق كاربرد سيستم مديريت بانك اطلاعاتي رابطه اي(RDBMS), كه از طريق DBMS هاي قديمي مشتق شده, انجام مي شود. تكنولوژيهاي بانكهاي اطلاعاتي رابطه اي و سرويس گيرنده/ سرويس دهنده تركيبهائي هستند كه در كارهاي اداري و تجاري جاري براي مديريت موفق داده ها وباقي ماندن در بازارهاي مربوطه مورد استفاده قرار مي گيرند. چند قسمت آتي تكنولوژي بانك اطلاعاتي رابطه اي و سرويس گيرنده/سرويس دهنده را براي فراهم كردن فونداسيون مستحكم تري از دانش لازم براي زبان استاندارد بانك اطلاعاتي رابطه اي(SQL) بررسي مي كنند.

SQL زبان برس و جوي ساختيافته, زبان استاندارد مورد استفاده براي برقراري ارتباط با يك بانك اطلاعاتي رابطه اي است. نخستين نمونة اين زبان در ابتدا با استفاده از مقا لة دكتر E..F.Codd (مقا له اي تحت عنوان يك مدل رابطه اي از دادها براي بانكهاي اطلاعاتي مشترك بزرگ) توسط شركت IBM ارائه شد. در سال 1979 , مدت كوتاهي پس از نمونه IBM , نخستين محصول SQL يعني اراكل, توسط شركت Relational Software (اين شركت سپس شركت اراكل ناميده مي شود) عرضه شد. امروزه اين شركت يكي از پيشروان متمايز در تكنولوژي بانكهاي اطلاعاتي رابطه اي است. SQL به دو شكل تلفظ مي شود: به صورت حروف S-Q-L , يا ((سيكوال)), هر دو تلفظ قابل قبول هستند. انسيتوي استانداردهاي ملي آمريكا(ANSI) سازماني است كه برخي از استانداردهاي بسياري از صنايع مختلف را به تصويب مي رساند. SQL به عنوان زبان استاندارد برقراري ارتباط با بانكهاي اطلاعاتي رابطه اي در نظر كرفته شده است. اين زبان نخستين بار در سال 1986 بر اساس نسخة پياده سازي شده توسط IBM تصويب شد. استاندارد SQL ANSI توسط سازمان بين ا لمللي تعيين استانداردها(ISO)به عنوان استاندارد بين ا لمللي پذيرفته شده است. استاندارد جاري تحت عنوان SQL/92 شناخته شده و نسبت به سال 1986 بسيار تغيير يافته است.علاوه بر مزاياي آشكار زياد,معايبي نيز به همراه هر استاندارد وجود دارند. مهمترين مزيت آن است كه

هر استاندارد فروشندگان را براي توسعه در جهت صنعتي مناسب هدايت مي كند, اين چهارچوب به عنوان نتيجة نهائي, سازگار بين نسخعه هاي پياده سازي شدة گوناگون را فراهم نموده و امكان استفاده از آنها در محيطهاي مختلف را بهتر مي كند.

برخي از اشخاص ممكن است ادعا بكنند استانداردي كه انعطاف پذيري و قابليتهاي پياده سازي نسخه خاصي را محدود مي كند, چندان خوب نيست. اما بيشتر فروشندگاني كه از اين استاندارد پيروي ميكنند ويژگيهاي را به استاندارد SQLافزوده اند تا اين گونه محدوديتها را جبران كند.

هر استاندارد با در نظر گرفتن مزايا و معايب خوب است. استاندارد مورد نظر ويژگيهاي را طلب مي كند مي بايست در هر نسخة پياده سازي شده كامل از SQL موجود باشند, و مفاهيم پايداري را مشخص مي كند كه نه تنها سازگاري و پيوستگي را در بين تمام نسخه هاي پياده سازي شده به وجود مي آورند, بلكه ارزش يك برنامه ساز SQL يا كاربر بانك اطلاعاتي رابطه اي را در بازار بانك اطلاعاتي امروز افزايش مي دهند

انواع دستورات SQL

قسمتهاي ذيل دسته هاي اصلي دستورات مورد استفاده براي انجام كارهاي مختلف در SQL را بررسي مي كنند. اين كارها شامل ساختن شيءهاي بانك اطلاعاتي, پردازش شيءها,پر كردن جداول بانك اطلاعاتي با داده ها, به روزرساني داده هاي موجود در جداول, حذف داده ها, اجراي پرس وجوها, كنترل دستيابي به بانك اطلاعاتي, و مديريت كلي بانك اطلاعاتي مي شوند.

ذسته هاي اصلي عبارتند از:

1. DDL : زبان تعريف داده ها
2. DML : زبان پردازش داده ها
3. DQL : زبان پرس وجوي داده ها
4. DCL : دستورات مديريت داده ها
5. دستورات مديريت داده ها
6. دستورات كنترل تراكنشي

تعريف ساختار بانك هاي اطلاعاتي (DDL)

زبان تعريف داده ها (DDL) بخشي از SQL است كه امكان ايجاد و تعريف مجدد ساختار شيء هاي بانك اطلاعاتي, همچون ايجاد يا حذف يك جدول, را براي كاربر بانك اطلاعاتي فراهم مي كند.

دستورات اصلي كه در قسمتهاي آتي مورد بررسي قرار خواهند گرفت عبارتند از:

CREATE TABLE
ALTER TABLE
DROP TABLE

CREATE INDEX
ALTER INDEX
DROP INDEX

پردازش داده ها(DML)

زبان پردازش داده ها, DML , بخشي از SQL است كه براي پردازش داده ها در شيء هاي يك بانك اطلاعاتي رابطه اي مورد استفاده قرار مي گيرد.

سه نوع دستور اصلي DML وجود دارد:

INSERT
UPDATE
DELETE

اين دستورات به تفصيل در قسمتهاي بعدي آمده اند.

انتخاب داده ها(DQL)

گرچه زبان پرس و جوي داده ها؛ DQL از تنها يك دستور تشكيل شده است ،اما بيشترين توجه كاربران يك بانك اطلاعاتي رابطه اي در SQL به آن است. تنها دستور اين زبان به شكل زير است:

SELECT

اين دستور كه با گزينه ها وعبارات زيادي همراه است، براي تشكيل پرس و جوهاي(Query) يك بانك اطلاعاتي رابطه اي مورد استفاده قرار مي گيرد. پرس و جوها را، از ساده گرفته تا پيچيده و از عمومي گرفته تا خاص، مي توان به آساني ايجاد نمود. توضيحات بيشتر راجع به اين دستور در قسمتهاي بعدي آمده است.

زبان كنترل داده ها(DCL)

دستورات كنترل داده ها در SQL به شما امكان مي دهند تا دستيابي به داده هاي بانك اطلاعاتي را كنترل كنيد. دستورات DCL عموماً براي ايجاد شيء هاي مرتبط با دستيابي كابر و نيز كنترل توزيع حقوق دستيابي در بين كاربران مورد استفاده قرار مي گيرند. برخي از دستورات كنترل عبارتند از :

ALTER PASSWORD
GRANT
REVOKE
CREATE SYNONYM

دستورات مديريت داده ها

دستورات مديريت داده ها امكان بررسي و آزمايش داده ها و تحليل عمليات درون بانك اطلاعاتي را فراهم مي كنند؛ با استفاده از آنها مي توان كارايي سيستم را نيز تحليل نمود. دو دستور عمومي مديريت داده ها عبارتند از:

START AUDIT
STOP AUDIT

بايد توجه داشت كه منظور از مديريت بانك اطلاعاتي، مديريت كلي يك بانك اطلاعاتي است، كه در آن تمامي سطوح دستورات مورد استفاده قرار مي گيرند.

دستورات كنترلي تراكنشي

علاوه بر سه دسته دستورات پيشين، دستوراتي وجود دارند كه امكان مديريت تراكنشهاي بانك اطلاعاتي را براي كاربر فراهم مي سازد.

1. COMMIT براي ذخيرة تراكنشهاي بانك اطلاعاتي مورد استفاده قرار مي گيرد
2. ROLLBACK براي بازپس گرفتن تراكنشهاي بانك اطلاعاتي مورد استفاده قرار مي گيرد
3. SAVEPOINT نقاطي در گروه هاي تراكنشها ايجاد مي كند كه بتوان ROLLBACK را به كار برد
4. SET TRANSACTION نامي را براي يك تراكنش تعيين مي كند

استانداردهاي نامگذاري جداول

استانداردهاي نامگذاري جداول، و همچنين ساير استانداردهاي يك كار، براي حفظ كنترل حياطي هستند. پس از مطالعة جداول و داده هاي قسمتهاي پيشين، احتمالآ متوجه شده ايد كه پسوند هر جدول TBL– است. اين استاندارد نامگذاري است كه براي استفاده در تمام جداول انتخاب شده است. همچون چيزي كه در سايتهاي سرويس گيرنده گوناگون مورد استفاده قرار گرفته است. TBL– صرفآ مشخص مي كند كه شيء يك جدول است؛ انواع مختلفي از شيء ها در يك بانك اطلاعاتي رابطه اي وجود دارد. به عنوان مثال، در قسمتهاي بعدي خواهيد ديد كه از پسوند INX– براي شناسايي شاخص هاي جداول استفاده شده است. استانداردهاي نامگذاري تقريبآ براي سازماندهي كلي هستند و در مديريت هر بانك اطلاعاتي رابطه اي كمك بسيار زيادي مي كنند.

موارد تشكيل دهندة يك جدول

ذخيره سازي و نگهداشت داده هاي ارزشمند، دليل وجود هر بانك اطلاعاتي است. در قسمتهاي زير عناصر موجود در يك جدول را بررسي مي كنند. توجه بايد داشت كه يك جدول متداولترين فرم ذخيره سازي داده ها در يك بانك اطلاعاتي رابطه اي است.

فيلد: هر جدول به موجوديتهاي كوچكتري به نام فيلد تجزيه مي شود. فيلدهاي جدول PRODUCTS_TBL عبارتند از PROD-ID، PROD-DESC و COST. اين فيلدها براي دسته بندي اطلاعات خاصي هستند كه در جدول مورد نظر نگهداري مي شوند.

هر فيلد، ستوني در يك جدول است كه براي نگهداري اطلاعات خاص تمام ركوردهاي جدول طراحي مي شود.

ركورد،يا سطري از داده ها

هر ركورد، سطري از داده ها نيزناميده مي شود، يك ورودي(entry) مجزا است كه در يك جدول وجود دارد. به عبارت ديگر هر سطر از داده ها ، يك ركورد در جدولي از يك بانك اطلاعاتي رابطه اي است.

ستون: هر ستون، يك موجوديت عمومي در يك جدول است كه از تمام داده هاي يك فيلد خاص از يك جول تشكيل مي شود.

ساختار داده ها در SQL

داده: داده ها، مجموعه اي از اطلاعاتي هستند كه به صورت يكي ازانواع مختلف در يك بانك اطلاعاتي ذخيره مي شوند. داده ها شامل نام ها، اعداد، مقاديرارزي، متن،گرافيك،دسيمال ها، ارقام، محاسبات، خلاصه مطالب، وتقريباً هر چيز ي ة بتوانيد تصور كنيد مي باشند. داده ها را مي توان پردازش يا تغيير داد؛ بيشتر داده ها در طي دوره حيات خود ايستا نيستند.

داده ها هدف هر بانك اطلاعاتي هستند و بايد محافظت شوند. محافظ داده ها عموماً DBA است، گرچه هر كاربر بانك اطلاعاتي وظيفه دارد تا اقداماتي براي محافظت از داده ها انجام دهد. امنيت داده ها به تفصيل در قسمتهاي بعدي آمده است.

انواع داده هاي اصلي

قسمتهاي ذيل داده هاي اصلي مطرح در SQL را بررسي مي كنند. انواع داده ها ويژگيهاي خاص خود را دارند كه صفات مشخصه آنها يك جدول نسبت داده مي شوند. به عنوان مثال مي توانيد مشخص كنيد كه يك فيلد بايد داراي مقادير عددي باشد، و از وارد شدن رشته هاي حرفي – عددي پيشگيري كنيد. گذشته از اين ها، احتمالاً نمي خواهيد كاراكترهاي الفبايي در فيلد كه براي مقادير ارزي است وارد شوند.

انواع داده هاي اصلي در اكثر زبانها عبارتند از:

1. رشته هاي كاراكتري
2. رشته هاي عددي
3. مقادير تاريخ و زمان

كاراكترهاي با طول ثابت: كاراكترهاي ثابت، رشته هايي كه هميشه طول ثابت دارند، با طول ثابت ذخيره مي شوند. استاندارد زشته هاي با طول ثابت در SQL به شكل زير است:

CHARACTERS (n)

در اين استاندارد n عددي است كه نشانگر طول اختصاص يافته، يا بيشينه طول فيلد مورد نظر در اين تعريف است.

برخي از نسخه هاي پياده سازي شدة SQL از CHAR براي ذخيره سازي داده هاي با طول ثابت استفاده مي كنند. داده هاي حرفي- عددي را مي توان در اين موارد ذخيره كرد. از آنجا كه تمامي ايالات (در اايالات متحده آمريكا) با دو كاركتر نشان داده مي شوند، اين داده ها مثالي از داده هاي با طول ثابت هستند.

زماني كه از داده هاي با طول ثابت استفاده مي شود، عموماً از فاصله براي پر كردن نقاط اضافي استفاده مي شود؛ اگر طول فيلدي 10 باشد و دادة وارد شده تنها 5 نقطه را پر كند، بقيه باقيمانده با فاصله پر مي شوند. افزودن فاصله تضمين مي كند كه طول هر مقدار در فيلد ثابت است.

رشته هاي با طول متغيير: SQL از رشته هاي با طول متغيير پشتيباني مي كند، رشته هايي كه طول آنها براي تمام داده ها ثابت نيست. استاندارد رشته هاي با طول متغيير در SQL به شكل زير است:

CHARACTER VARYING (n)

در اين تعريف عدد n نشانگر طول اختصاص يافته، يا بيشينه طول فيلد مورد نظر است.

داده هاي نوع VARCHAR وVARCHAR2 از متداولترين نوع داده ها براي مقادير كاراكتري با طول متغيير هستند. VARCHAR استاندارد ANSI است كه ميكروسافت Server SQL از آن استفاده مي كند؛ VARCHAR2 توسط اراكل مورد استفاده قرار مي گيرد و بايد در خود آن مورد استفاده قرار گيرد، چرا كه در آينده ممكن است كاربرد VARCHAR تغيير يابد. داده هاي ذخيره شده مي تواند حرفي- عددي باشند.

به خاطر داشته باشيد كه داده هاي با طول ثابت عموماً نقاط پر نشده در فيلد را با فاصله پر مي كنند. داده هاي با طول متغيير به اين صورت عمل نمي كنند. به عنوان مثال، اگر فيلدي با طول متغيير 10 تعريف شده باشد، و رشته اي با طول 5 كاراكتر در آن وارد شود، در آن صورت طول كل آن مقدار خاص تنها 5 است. از فاصله براي پر كردن نقاط استفاده نشده در يك ستون استفاده نمي شود.

نكته:به منظور صرفه جويي در فضاي بانك اطلاعاتي، هميشه از داده هاي با طول متغيير براي رشته هاي كاراكتري با طول غير يكسان استفاده كنيد.

مقادير عددي: مقادير عددي در فيلده اي ذخيره مي شوند كه به عنوان نوعي عدد تعريف مي شوند. و عموماً تحت عنوان NUMBER,INTEGER,REAL,DECIMAL و غيره شناخته شده هستند.

BIT (n)
BIT VARYING (n)
DECIMAL (n,n)
INTEGER
SMALLINT
FLOAT (p)
DOUBLE PRECISION

در هر تعريف n عددي است كه نشانگر طول اختصاص يافته، يا بيشينه طول مورد نظر در هر تعريف است.

NUMBER يكي از متداولترين انواع عددي در نسخه هاي پياده سازي شدة SQL است كه براي مقادير عددي ANSI است. مقادير عددي را مي توان به صورت صفر، اعداد مثبت، منفي، با طول ثابت و با نقطه مميز شناور ذخيره نمود. مثا لي از كاربرد NUMBER :

NUMBER (s)

اين مثال بيشينه مقدار اين فيلد خاص را به 99999 محدود مي كند.

مقادير دسيمال: مقادير دسيمال، مقادير عددي هستند كه در آنها از نقطه اعشار استفاده مي شود. استاندارد مطرح براي يك مقدار دسيمال در SQL به شكل زير است كه در آن n اول نشانگر دقت وn دوم نشانگر مقياس است:

DECIMAL (n,n)

منظور از دقت، طول كلي مقدار عددي است. در مقدار عددي كه به صورت DECIMAL (4,2)تعريف شده است، دقت 4 است، يعني طول اختصاص يافته به يك مقدار عددي.

منظور از مقياس، تعداد ارقام سمت راست نقطه اعشار است. در مثال پيشين، مقياس 2 است.

اگر يك مقدار عددي به صورت زير تعريف شده باشد، در آن صورت بيشينه مقدار مجاز، 99.99 خواهد بود:

DECIMAL (4,2)

«دقت» در اين مثال 4 است، يعني طول كل اختصاص يافته به يك مقدار. «مقياس» در اين مثال 2 است، يعني محل ها، يا بايت هايي كه در سمت راست نقطه اعشار كنار گذاشته مي شوند. نقطه اعشار، خود به عنوان يك كاركتر شمارش نمي شود.

مقادير مجاز ستوني كه به صورت DECIMAL (4,2) تعريف شده عبارتند از:

12
12.4
12.44
12.449

آخرين عدد به 12/449 ، به 12/45 گرد مي شود.

اعداد صحيح: هر عدد صحيح، مقدار عدديي است كه فاقد نقطه اعشار است، تنها اعداد كامل (هم مثبت و هم منفي).

اعداد صحيح معتبر عبارتند از :

1
0
-1
99
-99

دسيمالهاي با نقطه مميز شناور: دسيمالهاي با نقطه مميز شناور، مقادير دسيما لي هستند كه طول دقت و مقياس آنها متغيير و عملاً نا محدود است. هر دقت و مقياس قابل پذيرش است. داده هاي نوع REAL ،مشخص كنندة ستوني با دقت معمولي براي اعداد با نقطه مميز شناور هستند. داده هاي نوع DOUBLE PRECISION ، مشخص كننده ستوني با دقت مضاعف براي اعداد با نقطه مميز شناور هستند. براي اينكه دقت مورد نظر از نوع معمولي به شمار آيد، مي بايست عددي در بازة بسته 1و21 باشد.براي اينكه دقت مورد نظر از نوع مضاعف به شمار آيد، مي بايستدعددي در بازة بستة 22و53 باشد.

FLOAT
FLOAT (15)
FLOAT (50)

تاريخ ها وزمان: داده هاي نوع تاريخ وزمان مسلمآ براي حفظ اطلاعات مربوط به زمان و تاريخ هستند. SQL استاندارد از داده هاي نوع DATETIME ، كه شامل موارد زير هستند، پشتيباني مي كند:

DATE
TIME
INTERVAL
TIMESTAMP

عناصر داده هاي نوع DATETIME عبارتند از:

YEAR
MONTH
DAY
HOUR
MINUTE
SECOND

رشته هاي ليترال: هر رشته ليترال مجموعه اي از كاراكترهايي چون نام يا شمارة تلفن است كه صريحآ توسط يك كاربر يا برنامه مشخص مي شود. رشته هاي ليترال از داده هايي با صفات مشخصه انواع داده هاي مورد بررسي قسمتهاي پيشين تشكيل مي شوند، اما مقدار رشته معلوم است؛ مقدار يك ستون معمولآ معلوم نيست، چرا كه مقدار يك ستون در سطرهاي مختلف از يك جدول متفاوت است.

نوع رشته هاي ليترال مشخص نمي شود، بلكه تنها خود رشته مشخص مي شود. چند مثال از رشته هاي ليترال در ذيل آمده است:

‘HELLO’
45000

‘45000’
3.14
‘November 1,1997’

رشته هاي حرفي- عددي در بين علائم نقل قول تكي قرار ميگيرند، در صورتي كه عدد 45000 اين گونه نيست. همچنين توجه كنيد كه 45000 دوم بين علائم نقل قول قرار گرفته است. عموماً رشته هاي كاراكتري به علائم نقل قول نياز ندارد، در صورتي كه اعداد به اين علائم نياز ندارند. در آينده خواهيد ديد كه چگونه رشته هاي ليترال با پرس و جوهاي بانك اطلاعاتي مورد استفاده قرار مي گيرند.

انواع داده هاي تهي: هر مقدار تهي، مقداري است كه مشخص نيست،يا ستوني از يك سطر است كه مقدار به آن تخصيص نيافته است. مقادير تهي در تقريباً تمامي بخشهاي SQL مورد استفاده قرار ميگيرند، از جمله ايجاد جداول، شرطهاي جستجو در پرس و جوها، حتي در رشته هاي ليترال. دو روش براي ارجاع به يك مقدار تهي در ذيل نشان داده شده است:

1. NULL (خود كلمة كليدي NULL)
2. NULL (علائم نقل قول تكي بدون فاصله)

مثال ذيل نشان دهنده يك مقدار تهي نيست، بلكه نشان دهنده رشته ليترال است كه شامل كاراكترهاي N-U-L-L است.
‘NULL’

<u>مديريت شيءهاي بانك اطلاعاتي</u>

منظور از هر شيء بانك اطلاعاتي، شيء تعريف شده اي است كه براي ذخيره سازي يا ارجاع به داده ها مورد استفاده قرار مي گيرد. جداول، ويوها، كلاسترها، Sequence ها، شاخص ها، Synonym ها، مثالهايي از شيء هاي بانك اطلاعاتي هستند. جداول ساده ترين فرم ذخيره سازي در يك بانك اطلاعاتي رابطه اي به شمار مي آيد.

تعريف Schema: به مجموعه اي از شيء هاي بانك اطلاعاتي كه با يك نام كاربري خاص مرتبط هستند، Schema گفته مي شود. اين نام كاربري، صاحب Schema ناميده مي شود؛ و يا صاحب مجموعه شيء هاي مرتبط به هم مي شود. ممكن است يك يا چند Schema در يك بانك اطلاعاتي داشته باشيد. اساساً، هر كاربري كه شيئي را ايجاد مي كند، در واقع Schema خاص خود را ايجاد نموده است. يك Schema مي تواند از تنها يك جدول تشكيل شود و هيچ محدوديتي دربارة تعداد شيء هايي كه مي تواند داشته باشد وجود ندارد، مگر اينكه محدوديتي از جانب يك نسخة خاص اعمال شده باشد.

فرض كنيد مدير بانك اطلا عاتي، يك نام كاربري و كلمه عبور به شما اختصاص داده است. نام كاربري، USER1 است. فرض كنيد با بانك اطلاعاتي ارتباط برقرار مي كنيد، و سپس جدولي بنام EMPLOYEE_TBL ايجاد مي كنيد. نام واقعي جدول شما، USER1.EMPLOYEE_TBL

است. نام Schema ي آن جدول، USER1 است كه صاحب آن جدول نيز هست.

نكته جالب دربارة Schema ها آن است كه وقتي با جدول خود(در Schema خاص خودتان) سرو كار داريد، نيازي به مشخص كردن نام Schema نداريد. به عنوان مثال، با استفاده از يكي از روشهاي ذيل مي توانيد به جدول خود رجوع كنيد:

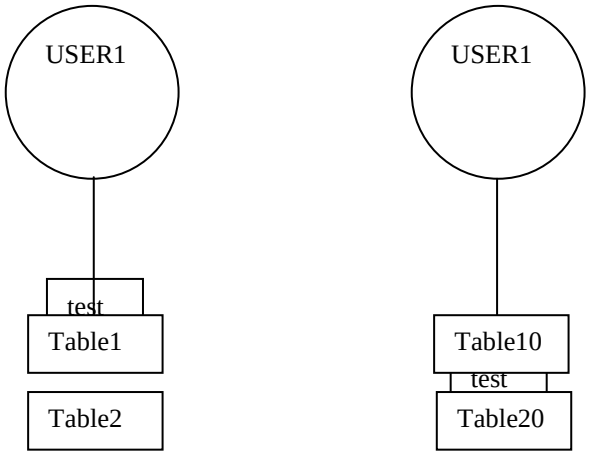
EMPLOYEE_TBL

USER1.EMPLOYEE_TBL

روش اول ترجيح داده مي شود، چرا كه تعداد كاراكترها كمتر است. چنانچه كاربر ديگري بخواهد با يكي از جداولتان كار كند، در آن صورت مي بايست Schema را به صورت ذيل مشخص كند:

USER1.EMPLOYEE_TBL

شكل صفحة بعد Schema را در يك بانك اطلاعاتي نشان مي دهد:



در شكل بالا دو account كاربري در بانك وجود دارد كه جداولي به آنها تعلق دارد: USER1 و USER2 . هر account كاربري Schema خاص خودش را دارد. هر دو كاربر جدولي بنام TEST دارند. تا زماني كه جدول يك بانك اطلاعاتي به Schema هاي مختلفي تعلق داشته باشند،مي توانند نام هاي يكسان داشته باشند. اگر اين مطلب را به همين صورت در نظر بگيريم، نام جداول يك بانك اطلاعاتي هميشه منحصر به فرد هستند، چرا كه صاحب Schema در واقع بخشي از نام جدول است. به عنوان مثال USER1.TEST با USER2.TEST تفاوت دارد. اگر هنگام دستيابي به جداول يك بانك اطلاعاتي نام يك Schema را مشخص نكنيد، سرويس دهندة

بانك اطلاعاتي طبق پيش فرض جدولي را جستجو مي كند كه به خودتان تعلق دارد.

جدول: جدول، محل اصلي ذخيره سازي داده ها در يك بانك اطلاعاتي است. هر جدول از يك يا چند سطر وستون، كه هر دو داده را نگه مي دارند، تشكيل مي شود. هر جدول فضايي را به طور فيزيكي در بانك اطلاعاتي اشغال نموده و مي تواند دائمي يا موقت باشد.

فيلدها و ستونها: هر فيلد كه در بانك اطلاعاتي رابطه اي ستون ناميده مي شود، بخشي از يك جدول است كه نوع خاصي براي داده هاي آن در نظر گرفته ميشود؛ هر فيلد بايد به گونه اي نامگذاري شود تا با نوع داده هايي كه در آن ستون وارد مي شوند متناسب باشد. ستونها را مي توان بصورت NULL يا NULL NOT تعريف نمود، بدين معنا كه اگر ستوني NOTNULL باشد، در آن صورت چيزي بايد در آن وارد شود. اگر ستوني بصورت NULL تعريف شود، نيازي نيست كه چيزي در آن وارد شود. هر جدول بانك اطلاعاتي بايد از حداقل يك ستون تشكيل شود.ستونها عناصري از يك جدول هستند كه داده هايي از يك نوع خاصي را نگه مي دارند،مثلاً نام يا نشاني يك شخص. بعنوان مثال، يك ستون معتبر در جدول مشتريان مي تواند نام آنها باشد. عموماً، هر نام بايد يك رشته پيوسته باشد. نام هر شيء بايد يك رشته پيوسته باشد و بسته به نسخه پياده سازي شده ممكن است به تعدادي كاراكتر محدود باشد. استفاده از ‘‘-‘‘ در نامها براي جداسازي كاراكترها متداول است. بايد توجه شود كه پيش از نامگذاري شيء ها و ساير عناصر بانك اطلاعاتي، حتماً نسخه مورد استفاده خود را براي آشنايي با قوانين نامگذاري بررسي كنيد.

سطرها: هر سطر، ركوردي از داده ها در يك جدول است. بعنوان مثال، هر سطر از داده ها در جدول مشتريان ممكن است از شماره شناسايي،نام،نشاني،شماره تلفن،شماره نمابر، و غيره مشتريان تشكيل شود. هر سطر از فيلدهايي تشكيل مي شود كه حاوي داده هاي يك ركورد است. هر جدول مي تواند حداقل داراي يك سطر از داده ها و چندين ميليون سطر يا ركورد باشد.

دستور CREATE TABLE :از دستور CREATE TABLE براي ايجاد جدول استفاده ميشود. اگر چه عمل ايجادجدول كاملاً ساده است، اما پيش از اجراي اين دستور بايد زمان زيادي صرف طراحي ساختار آن شود.

هنگام ايجاد يك جدول، به پرسش هاي اساسي بايد پاسخ داد:

1.چه نوع داده هايي جدول را تشكيل ميدهند؟
2.نام جدول چه خواهد بود؟
3.كدام ستون(ها) كليد اصلي را تشكيل ميدهند؟
4.نام ستونها(فيلدها) چه خواهد بود؟
5.چه نوع داده اي براي هر ستون تعريف خواهد شد؟
6.طول هر ستون چه خواهد بود؟

قالب اصلي ايجاد جدول به صورت زير است:

```
CREATE TABLE table_name
( field1   datatype   [ not  null ],
  field2   datatype   [ not  null ],
  field3   datatype   [ not  null ],
  field4   datatype   [ not  null ],
  field5   datatype   [ not  null ])
```

در مثال ذيل جدولي بنام EMPLOYEE_TBL ايجاد خواهيم نمود:

```
CREATE TABLE EMPLOYEE_TBL
( EMP_ID      CHAR (9)      NOT NULL,
  EMP_NAME    VARCHAR (40)  NOT NULL,
  EMP_ST_ADR  VARCHAR (20)  NOT NULL,
  EMP_CITY    VARCHAR (15)  NOT NULL,
  EMP_PHONE   VARCHAR (10)  NOT NULL);
```

ًۀۀً آخرين كاراكتر دستور بالاست. بيشتر نسخه هاي پيادهسازي شدة SQL كاراكتري دارند كه پايان يك دستور را مشخص نموده و آنرا به سرويس دهنده بانك اطلاعاتي ارسال مي كنند. اراكل از ًۀۀً استفاده مي كند. Transact_SQL از عبارت GO استفاده مي كند.

توجه شود كه NULL مقدار پيش فرض يك ستون است؛ بنابراين، نيازي نيست كه در دستور CREATE TABLE وارد شود.

دستور ALTER TABLE : هر جدول را مي توان پس از ايجاد ازطريق كاربرد دستور ALTER TABLE اصلاح نمود. ميتوان ستون يا ستونهايي را اضافه كرد، ستون يا ستونهايي را حذف نمود، تعريف ستونها را تغيير داد، محدوديتهايي را اضافه يا حذف نمود، ودر برخي از نسخه هاي پياده سازي شده SQL مقادير ذخيره شده را تغيير داد. قا لب استاندارد اين دستور به شكل زير است:

```
alter table table_name [modify] [column column_name] [data type null not | null] [restorict |cascade ]
                                [drop] [constraint constraint_name]
                                [add] [column] column definition
```

اصلاح عناصر يك جدول

صفات مشخصه يك ستون را مي توانيد با دستور ALTER TABLE تغيير دهيد. در اينجا منظور از كلمة صفت مشخصه يكي از موارد ذيل است:

1. نوع داده هاي يك ستون

2. طول،دقت،يا مقياس يك ستون
3. اينكه ستون مي تواند مقادير تهي داشته باشد يا خير

در مثالهاي ذيل از دستور ALTER TABLE براي تغيير صفات مشخصه ستون EMP_ID جدول EMPLOYEE_TBL استفاده شده است:

```
ALTER TABLE EMPLOYEE_TBL MODIFY
(EMP_ID VARCHAR2 (10));
Table altered.
```

اين دستور پيشتر به صورت VARCHAR2 (كاراكتري با طول متغيير) تعريف شده بود، اما بيشينه طول آن در اينجا از 9 به 10 تغيير يافته است.

افزودن ستونهاي اجباري به جدول

يكي از قوانين افزودن ستونها به يك جدول آن است كه اگر داده هايي در آن جدول موجود باشند، ستون مورد نظر را نمي توان بصورت NOT NULL تعريف نمود. NOT NULL بدين معنا كه يك ستون بايد مقداري را در هر يك از سطرهاي جدول داشته باشد. بنابراين اگر ستوني را بصورت NOT NULL به جدولي مي افزاييد، در حقيقت محدوديت NOY NULL را نقيض مي كنيد، چراكه سطرهاي پيشين جدول مقاديري براي ستون جديد ندارند.

با اين وجود يك روش براي افزودن اين گونه ستونها به يك جدول وجود دارد:

1. ستون را اضافه كنيد و به صورت NULL تعريف كنيد
2. مقداري را بريا هر يك از سطرها در اين ستون جديد وارد كنيد
3. پس از حصول اطمينان از وجود يك مقدار در تمامي سطرهاي جدول، صفت مشخصه ستون را به NOT NULL تغيير دهيد.

تغيير ستونها

وقتي ستونهاي يك جدول تغيير داده مي شوند، مسائل زيادي را بايد در نظر گرفت:

1. طول يك ستون را مي توان به بيشينه طول يك نوع داده معين افزايش داد.
2. طول يك ستون را تنها زماني مي توان كاهش داد كه بزرگترين مقدار آن ستون در كل جدول، كوچكتر يا مساوي طول جديد ستون باشد.
3. تعداد ارقام داده هاي عددي را هميشه مي توان افزايش داد.
4. تعداد ارقام داده هايي را تنها زماني مي توان كاهش داد كه تعداد ارقام بزرگترين مقدار آن ستون، كوچكتر يا مساوي تعداد ارقام جديد باشد.
5. تعداد ارقام اعشاري داده هاي عددي را مي توان افزايش يا كاهش داد.
6. نوع داده هاي يك ستون را عموماً مي توان تغيير داد.

ايجاد يك جدول از روي يكي از جدولهاي موجود

با استفاده از تركيب دستور CREATE TABLE و SELECT مي توان يك نسخه از يكي از جداول موجود ايجاد نمود. تعريف ستونهاي جدول جديد همچون جدول موجود خواهد بود. تمام يا برخي از ستونهاي جدول را مي توان انتخاب نمود. اندازه ستونهاي جديدي كه از طريق توابع يا تركيبي از ستونها ايجاد مي شوند، متناسب با طول داده هايي خواهد بود كه ذخيره خواهند نمود.

قالب دستور براي ايجاد يك جدول از روي يك جدول ديگر:

```
create table  new_table_name as
select [* | column, column2]
from table_name
[ where ]
```

توجه داشته باشيد كه كلمه كليدي SELECT براي پرس و جو از بانكهاي اطلاعاتي است و به تفصيل در قسمتهاي آينده مورد بررسي قرار خواهد گرفت. با اين وجود، مهم است بدانيد كه يك جدول را مي توان بر اساس نتايج يك پرس وجو ايجاد نمود.

حذف جداول

حذف يك جدول در حقيقت يكي از آسانترين كارهاست. زماني كه از گزينه restrict استفاده مي شود و ارجاع به جدول توسط يك ويو يا محدوديت صورت مي گيرد، دستور drop با خطا كار خود را به پايان مي رساند. زماني كه از گزينه Cascade استفاده مي شود، يقخح كار خود را با موفقيت انجام داده و تمام ويوها و محدوديتها حذف مي شوند.

قالب دستور drop :

```
drop table  table_name  [restrict | cascade]
```

توجه:

هرگاه خواستيد جدولي را حذف كنيد، حتماً پيش از صدور دستور، نام Schema يا صاحب جدول را مشخص كنيد. ممكن است جدول ديكري را حذف كنيد. اگر به account چندين كاربر دستيابي داريد، پيشاز حذف جداول اطمينان حاصل كنيد كه از طريق account مربوطه به يك بانك اطلاعاتي متصل شده ايد.

<u>فرايند متعارف سازي</u>

متعارف سازي، فرايند كاهش افزونگيها در يك بانك اطلاعاتي است. علاوه بر داده ها،نام ها،نام شيء ها،و فرم ها نيز در يك بانك اطلاعاتي متعارف مي شوند.

بانك اطلاعاتي خام: بانك اطلاعاتي كه متعارف نشده باشد، ممكن است داده هايي داشته باشد كه بدون هر گونه دليل آشكار در يك يا چند جدول تكرار شده اند.اين امر بنا به دلائل امنيتي، مصرف فضايي ديسك، سرعت پرس

وجوها، بازدهي به روزرساني بانك اطلاعاتي،و شايد مهمتر از همه، جامعيت داده ها، نامناسب خواهد بود. هر بانك اطلاعاتي پيش از متعارف سازي، بانك اطلاعاتي است كه به طور منطقي به جداول كوچكتر قابل وديريت تجزيه نشده است. شكل زير يك بانك اطلاعاتي خام را نشان مي دهد:

COMPANY_DATABASE

Emp_id

cust_id

Last_name

cust_name

First_name

st_address

middle_name

ust_city

address

cust_state

city cust_zip

state cust_fax

data_hire prod_id

pay_rate

prod_desc

data_last_raise cost

طراحي منطقي بانك اطلاعاتي

هر بانك اطلاعاتي مي بايست با در نظر گرفتن كاربر نهايي طراحي شود. طراحي منطقي بانك اطلاعاتي، كه تحت عنوان مدل منطقي نيز شناخته شده است، فرايند آرايش داده ها در گروههاي منطقي و سازمان يافته اي از شيء هايي است كه به آساني قابل نگهداشت هستند. طراحي منطقي يك بانك اطلاعاتي مي بايست تكرار داده را كاهش دهد، و يا آنقدر پيش رود تا آنرا كاملا حذف نمايد.از اينها گذشته، چرا بايد داده هاي مشابه را دومرتبه ذخيره نمود؟ قراردادهاي نامگذاري مورد استفاده در يك بانك اطلاعاتي نيز بايد استاندارد و منطقي باشند.

فرم هاي متعارف: فرم متعارف، روشي براي اندازه گيري سطوح يا عمقي است كه بانك اطلاعاتي تا بدان حد متعارف شده است. سطح متعرف سازي يك بانك اطلاعاتي توسط فرم متعارف تعيين مي شود.

متداولترين فرمهاي متعرف در فرايند متعارف سازي عبارتند از:

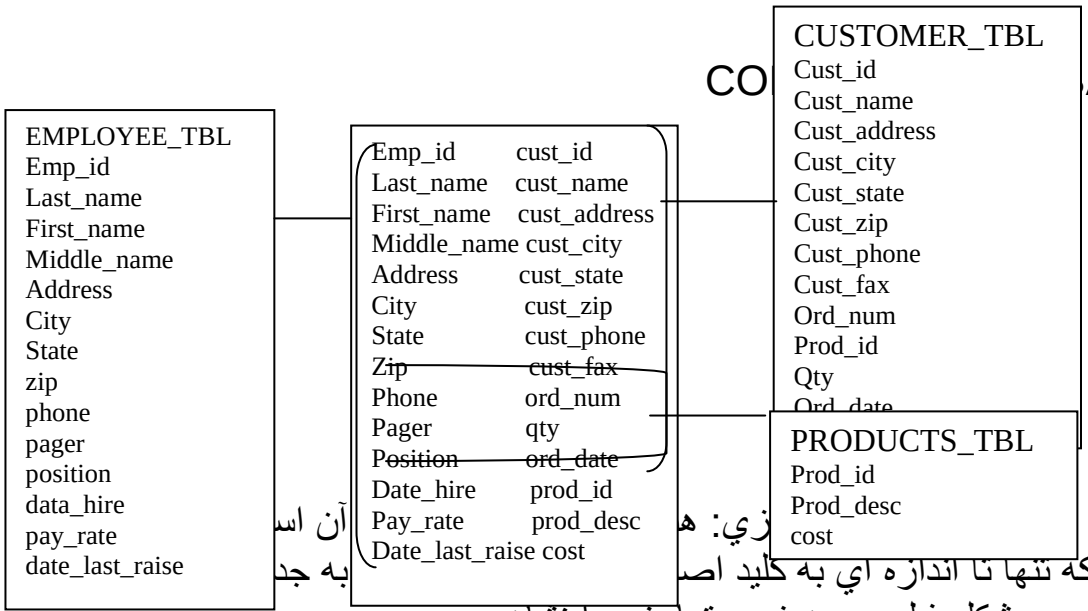
- نخستين فرم متعارف
- دومين فرم متعارف
- سومين فرم متعرف

برخي ازمزاياي متعارف سازي

1. كاهش داده هاي تكراري
2. سازگاري داده ها درون بانك اطلاعاتي
3. سازماندهي كلي بهتر براي بانك اطلاعاتي
4. طراحي انعطاف پذيرتر بانك اطلاعاتي
5. مديريت بهتر امنيت بانك اطلاعاتي

نخستين فرم متعارف: هدف نخستسن فرم متعارف، تقسيم داده ها به واحدهاي منطقي است كه جدول ناميده مي شوند. پس از طراحي هر جدول، يك كليد اصلي به تمام يا اكثر جداول اختصاص مي يابد. شكل ذيل نشان مي دهد كه بانك اطلاعاتي خام شكل مذكور، چگونه با استفاده از نخستين فرم متعارف از نو طراحي شده است. براي رسيدن به نخستين فرم متعارف، داده ها به واحدهاي منطقي تجزيه شدهاند و هر واحد با يك كليد اصلي تضمين مي كند كه گروه تكراريي در هيچ يك از واحدها وجود ندارد. اينك بجاي يك جدول بزرگ، جداول كوچكتر قابل مديريت تري وجود دارد: EMPLOYEE_TBL , PRIDUCTS_TBL , CUSTOMER_TBL . كليدهاي عمومي معمولا نخستين ستونهاي يك جدول هستند؛ در اين حا لت، CUST_ID , PROD_ID

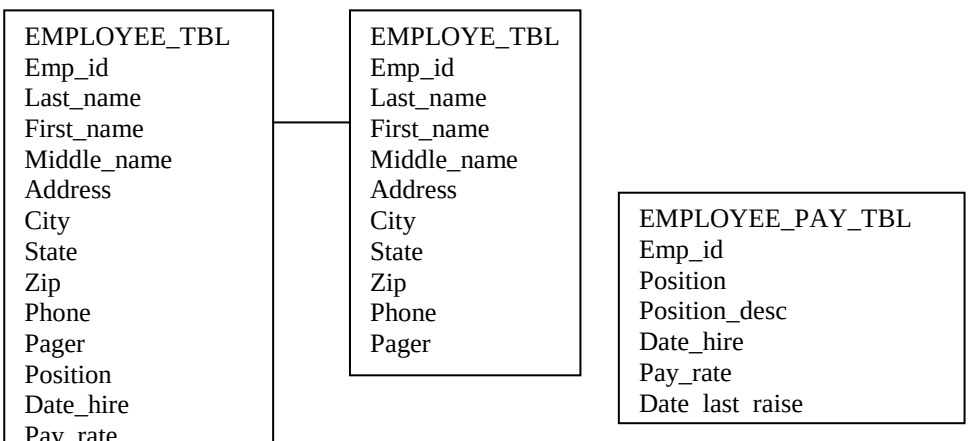
CO ASE



زي: ه آن اس

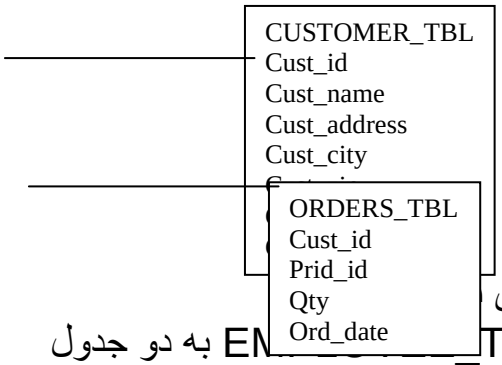
هايي كه تنها تا اندازه اي به كليد اص به جد

وارد نمود. شكل ذيل دومين فرم متعارف را نشان مي دهد:



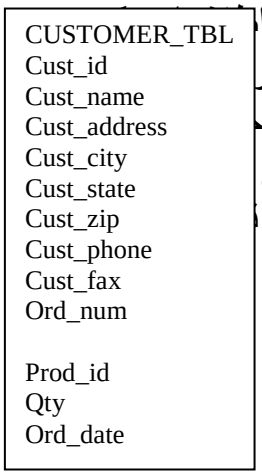
دومين فرم متعار ف

نخست فرم متعارف

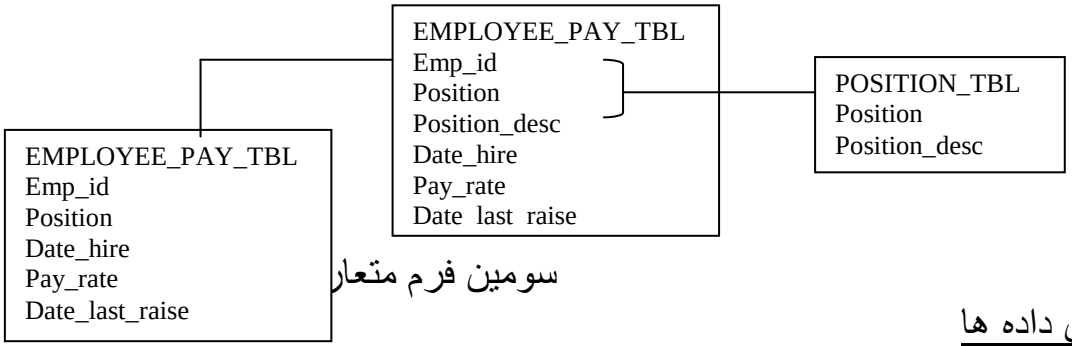


دومين فرم متعارف نخستين

طبق شكل بالا جدول TBL به دو جدول EMPLOYEE_PAY_TBL وEMPLOYEE_TBL تجزيه شده است. اطلاعات شخصي كارمندان به كليد اصلي وابسته است بنابراين آن اطلاعات در جدول EMPLOYEE_TBL باقي مانده اند. از طرف ديگر، ا تنها تا اندازه اي به EMP_ID وابسته هستند براي پر كر EMPLOYEE_PAY_TBL مورد استفاده قرار گرفته اند. تو هر دو جدول فيلد CUST_ID را دارند. اين همان كليد اصلي است و براي تطبيق داده هاي متناضر بين دو جدول مورد استفاده گيرد.



سومين فرم متعارف: هدف سومين فرم متعارف آن است كه داده هايي را كه به كليد اصلي وابسته نيستند، از يك جدول خارج كند. شكل ذيل سومين فرم متعارف را نمايش مي دهد:



سومين فرم متعار

<u>پردازش داده ها</u>

زبان پردازش داده ها (DML) بخشي از SQL است كه به كاربر يك بانك اطلاعاتي امكان مي دهد تا داده هاي بانك اطلاعاتي را بطه اي را تغيير دهيد. با وجود DML ،كاربر مي تواند جداول را با داده هاي جديد پر كند، داده هاي موجود جداول را به روز برساند، و داده ها را از جدول حذف كند. پرس و جوهاي ساده را نيز مي توان با يك دستور DML انجام داد.

سه دستور DML اصلي در SQL وجود دارد:

INSERT
UPDATE
DELETE

دستور SELECT، كه با دستورات DML قابل استفاده است، با تفصيل بيشتر در قسمت ‹‹مقدمه اي بر پرس وجواز بانك اطلاعاتي›› بررسي خواهد شد.

پر كردن جداول با داده هاي جديد: پر كردن يك جدول با داده ها صرفآ به فرايندي گفته مي شود كهدر طي آن داده هاي جديد، اعم از طريق يك فرايند دستي با استفاده از دستورات مجزا، يا از طريق فرايندي دسته اي (batch) با استفاده از برنامه ها يا ساير نرم افزارهاي مربوطه به اين كار، وارد يك جدول مي شوند. هنگامي كه جداول را با داده هاي جديد پر ميكنيد،عوامل زيادي ممكن است بر نوع و مقدار داده هايي كمه مي توان در يك جدول وارد كرد، تآثير مي گذارند. برخي از عوامل اصلي عبارتند ازمحدوديتهاي موجود جدول، اندازه فيزيكي جدول، نوع داده ستونها، طول ستونها، و محدوديتهاي جامعيت ديگري چون كليدهاي اصلي و خارجي. قسمتهاي ذيل علاوه بر مشخص كردن كارهايي كه بايد انجام شوند و انجام نشوند، شما را در فراگيري اصول درج داده هاي جديد در يك جدول ياري مي كنند.

نكته: دستورات SQL را مي توان با حروف بزرگ ويا كوچك نوشت. بزرگ يا كوچك بودن داده ها با توجه به شيوة ذخيره سازي آنها در بانك اطلاعاتي تعيين مي شود.

درج داده ها در يك جدول:

از دستور INSERT براي درج داده ها در يك جدول استفاده مي شود.اين دستور چندين گزينه دارد. براي شروع به قالب اصلي زير توجه كنيد:

```
insert into schema. Table_name
VALUES (‘Value1’ ‘Value2’ , [ NULL ]);
```

در صورت استفاده از اين قالب، مي بايست تمامي ستونهاي جدول مشخص شده را در فهرست VALUES بگنجانيد. داده هاي كاراكتري و تاريخهايي كه در جدول درج ميشوند بايد بين علائم نقل قول قرار گيرند. علائم نقل قول براي داده هاي عددي يا مقادير تهي كه با كلمه كليدي NULL وارد مي

شوند لازم نيستند. يك مقدار بايد براي هر يك از ستونهاي جدول وجود داشته باشد.

در مثال ذيل ركورد جديد در جدول PRODUCTS_TBL درج مي شود.

ساختار جدول:

products_tbl

نام ستون	نوع داده تهي؟
PROD_ID	NOT NULL VARCHAR2 (10)
PROD_DESC	NOT NULL VARCHAR2 (25)
COST	NOT NULL NUMBER (6,2)

دستور INSERT نمونه

```
insert into product_tbl
values (“7725” m “leather gloves”,24.99)
```

خروجي	a	تايپ	1

در اين مثال سه مقدار در جدولي كه سه ستون دارد وارد مي شود. ترتيب مقادير درج شده با ستونهاي جدول يكسان است. دو مقدار نخست با استفاده از علائم نقل قول درج شده اند، چرا كه نوع داده هاي ستونهاي متناظرشان كاراكتري است. مقدار سومين ستون، COST، عددي بوده و نيازي به علائم نقل قول نيست، گرچه وجود آنها اختياري است.

نكته: همانطور كه در قالب دستور INSERT نيز مشخص شده، نام Schema، يا صاحب جدول، به عنوان بخشي از نام جدول ضروري نيست. اگر به عنوان كاربري كه صاحب جدول است به بانك اطلاعاتي متصل شده باشيد، نيازي به نام Schema نيست.

درج داده ها در برخي از ستونهاي جدول:

روشي وجود دارد كه با استفاده از آن مي توانيد داده ها را در برخي از ستونهاي يك جدول درج كنيد. به عنوان مثال، فرض كنيد مي خواخيد تمام مقاديرذ مربوطه به يك كارمند، به غير از شماره Pager، را در جدولي وارد كنيد. در اين حالت بايد علاوه بر فهرستهاي VALUES، فهرست ستونها را نيز در دستور INSERT مشخص كنيد. قالب اين دستور براي درج مقادير برخي از ستونهاي يك جدول به شكل زير است:

قـالـب

```
insert into schema.table_name
('column1','column2')
values('value1','value2')
```

در مثال زير از جدول ORDERS_TBL استفاده شده و مقاديري در برخي از ستونهاي خاص درج شده اند.

ساختار جدول:

ORDERS_TBL

نام ستونها	نوع داده تهي؟
ORD_NUM	NOT NULL VARCHAR2 (10)
CUST_ID	NOT NULL VARCHAR 2(10)
PROD_ID	NOT NULL VARCHAR2 (10)
QTY	NOT NULL NUMBER (4)
ORD_DATE	DATE

Sample INSERT statement

تـاي

```
Insert into orders_tbl (ord_num,cust_id,prod_qty)
Values ('23a16','109','7725',2);
```

خـروجـي

1 row created.

در اين مثال فهرستي از ستونها مشخص شده و پس از نام جدول در دستور در بين پرانتز نوشته شده است. فهرست تمام ستونهايي كه در آنها درج شده است. ORD_DATE تنها ستوني است كه مشخص نشده است. اگر به تعريف جدول نگاه كنيد، خواهيد ديد كه ORD_DATE نيازي به داده ندارد، چراكه NOT NULL در تعريف جدول مشخص نشده است. NOT NULL مشخص مي كند كه مقادير تهي براي اين ستون مجاز نيستند. بعلاوه، فهرست مقادير بايد با ترتيبي كه مي خواهيد آنها را در ستونها درج كنيد يكسان باشد.

درج داده ها ا ز يك جدول ديگر:

ممكن است با استفاده از تركيب دو دستور INSERT و SELECT داده هايي را بر اساس نتائج يك پرس و جو از يك جدول ديگر در يك جدول درج كنيد. بطور مختصر، هر پرس و جو، استعلام از يك بانك اطلاعاتي است. هر پرس و جو، سئو الي است كه كاربر از بانك اطلاعاتي مي پرسد، و داده

هايي كه بازگردانده مي شوند پاسخ آن هستند. در حالتي كه دو دستور INSERT و SELECT با يكديگر تركيب مي شوند، مي توانيد داده هاي با[illegible] ر يك جدول درج كنيد.

قالب

قالب دستور INSERT براي درج داده ها از يك جدول ديگر عبارتست از:

```
insert into schema.table_name
[('column1','column2')]
select [* | ('column1','column2')]
from table_name
[where condition(s)];
```

سه كلمه كليدي در اين قالب دستور وئجود دارد كه بطور مختصر بررسي شده اند. اين كلمات كليدي SELECT,FROM,WHERE هستند. SELECT دستور اصلي مورد استفاده براي آغاز يك پرس و جو در SQL است. FROM عبارتي در پرس و جو است كه نام جدولي را مشخص مي كند كهخ داده هاي مورد نظر بايد در آنها پيدا شوند. عبارت WHERE نيز كه بخشي از پرس و جو است.، براي تعيين شرطهاي آن است. يك شرط نمونه ممكن است به صورت "WHERE NAME="SMITH باشد.

مثال زير از يك پرس و جوي ساده براي مشاهدة تمامي داده هاي جدول PRODUCTS_TBL استفاده مي كند. * SELECT به سرويس دهنده باتنك اطلاعاتي فرمان مي دهد كه اطلاعات تمامي ستونهاي جدول مد نظر هستند. چون از عبارت WHERE استفاده نشده است، تمام ركوردهاي جدول را مشاهده خواهيد كرد.

تاي

```
Select * from products_t
```

خروجي

```
PROD_ID                    PROD_DESC                    COST
11235                      WITCHES COSTUME              29.99
222                        PLASTIC PUMPKIN 18 INCH      7.75
13                         FALSE PARAFFIN TEETH         1.1
```

90 LIGHTED LANTERNG 14.5

ASSORTED COSTUMES 15 10

1.35 CANDY CORD 9

PUPMKIN CANDY 6 1.45

PLASTIC SPIDERS 87 1.05

ASSORTED MASKS 119 4.95

KEY CHAIN 1234 5.95

اينك مقادير را بر اساس پرس و جوي بالا در جدولPRODUCTS_TBL درج مي كنيم.

تـــايـپ

```
Insert into products_t
```

خـروجـي

```
Select * from products_
```

پرس و جوي ذيل تمام داده هايي را كه در جدول PRODUCTS_TBL درج شده اند نشان مي دهد.

```
Select * from products_tbl
```

درج مقادير تهي:

درج مقادير تهي در يك ستون از يك جدول كار ساده اي است. چنانچه مقدار ستون مورد نظر نامعلوم باشد، بهتر است مقدار تهي در آن درج شود. به عنوان مثال، تمامي اشخاص داراي pager نيستند، بنابراين وارد كردن يك شماره نادرست كار درستي نيست ـ بعلاوه، فضا نيز مصرف مي شود. مقدار تهي با ايتفاده از كلمه كليدي NULL در يك ستون از يك جدول درج نمود.

قـالـب

قالب دستور INSERT براي درج يك مقدار تهي بصورت ذيل است:

```
insert into schema.table_name values
('column1',NULL,'column2');
```

كلمه كليدي NULL را بايد در صورت وجود ستون مربوطه در جدول به كار برد. اگر NULL را وارد كنيد، آن دستور در آن سطر مقداري نخواهد داشت. در قالب دستور بالا، يك مقدار تهي(null) در COLUMN2 وارد شده است.

تـــاي

```
Insert into orders_tbl
(ord_num,cust_id,prod_id,qty,ord_DATE)
Values (‘23A16’,’109’,’7725’,2,NULL);
```

خروجي

```
1 row created.
```

در اين مثال تمام ستونهايي كه بايد مقداري در آنها درج شود فهرست شده اند، يعني تمام ستونهاي جدول ORDERS_TBL. مقدار تهي در ستون ORD_DATE وارد مي شود، يعني يا تاريخ سفارش معلوم نيست، يا تاريخي در اين زمان براي درج وجود ندارد.

تـاي

```
Insert into oreders_t
Values (‘23A16’,’109’,’7725’,2,’’);
```

خروجي

```
1 row created.
```

مثال دوم از دو جهت با مثال اول تفاوت دارد، اما نتايج يكسان هستند. نخست اينكه، فهرستي از ستونها مشخص نشده است. به خاطر داشته باشيد كه اگر بخواهيد مقاديري را در تمام ستونها درج كنيد، نيازي به فهرست ستونها نيست. دوم اينكه، بجاي درج مقدار تهي در ستون ORD_DATE، ‘ ’(دو علامت نقل قول) درج ميشود كه نماد مقدار تهي است(چون چيزي در بين آنها نيست)

به روز رساني داده ها: داده هاي موجود در يك جدول را مي توان با دستور UPDATE اصلاح نمود. اين دستور ركوردهاي جديدي را به جدول نمي افزايد، و ركوردي را نيز حذف نمي كند، بلكه صرفآ داده هاي موجود را به روز مي رساند. اين دستور عمومآ براي به روز رساني يك جدول از يك بانك اطلاعاتي در هر زمان است، ام مي توان آنرا براي به روزرساني همزمان چند ستون از يك جدول نيز به كار مي رود. يك سطر از داده هاي يك جدول را ميتوان به روز رساند، بسته به نيازهاي موجود، چندين سطر از داده ها را مي توان با يك دستور به روز رساند.

به روز رساني مقدار يك ستون: ساده ترين فرم دستور UPDATE، كاربرد آن براي به روز رساني يك ستون منفرد يك جدول است. زماني كه مقدار تنها يك ستون به روز رسانده ميشود، مي توان تنها يك سطر يا چندين ركورد را به روز رساند.

قالب دستور براي به روز رساني يك ستون به شكل زير است:

قـالـب

```
update table
```

set column_name='value'
[where codition];

مثال زير مقدار ستون QTY جدول ORDERS را براي ركوردهايي كه مقدار ORD_NUM آنها 23A16 است، به 1 تغيير مي دهد.

تـــاي

```
Update orders_tb
Set qty = 1
Where ord_num = '23A16';
```

خـروجـي

```
1 row updated.
```

مثال ذيل مشابه مثال پيش است، با اين تفاوت كه از عبارت WHERE استفاده نشده است.

تـــاي

```
Update prders_t
Set qty = 1
```

خـروجـي

```
11 row upda
```

توجه كنيد كه در اين مثال 11 سطر از داده ها به روز رسانده شده اند. مقدار يك براي ستون QTY در نظر گرفته شده و ستون quantity در جدول ORDERS براي تمام سطرها به روز رسانده مي شود. آيا اين واقعآ همان كاري است كه مي خواهيد انجام دهيد. شايد در برخي موارد اين گونه باشد، اما دستور UPDATE بندرت بدون عبارت WHERE به كار برده مي شود.

به روز رساني چند ستون در يك يا چند ركورد: در اينجا خواهيد ديد كه چگونه مي توان چند ستون را با تنها يك دستور UPDATE به روز رساند. قالب دستور را در اين حالت مطالعه كنيد:

قـالـب

```
update table
set column1 = 'value',
[column2 = 'value']
[column3 = 'value']
[where condition];
```

به كاربرد SET در اين حالت توجه كنيد. تنها يك SET وجود دارد، اما چندين ستون. ستونها بوسيله كاما از يكديگر جدا شده اند. معمولآ از كاما براي جداسازي انواع مختلف آرگومانها در يك دستور SQL استفاده مي شود.

تـــاي

```
Update orders_t
   Set qty =  1
  Cust_id =  ‘221’
Where ord_num = ‘23A16’;
```

خـروجـي

```
1 row upda
```

از كاما براي جداسازي دو ستوني كه به روز رسانده مي شوند استفاده مي شود. در اينجا نيز عبارت WHERE اختياري است، اما معمولآ ضروري است.

نكته: كلمه كليدي SET تنها يك بار در هر دستور UPDATE به كار برده مي شود. اگر قرار باشد بيش از يك ستون به روز رسانده شود، از كاما براي جداسازي ستونهايي كه بايد به روز رسانده شوند استفاده مي شود.

حذف داده ها از جداول:

از دستور DELETE براي حذف كل سطرها از يك جدول استفاده مي شود. ذستور DELETE براي حذف مقادير از برخي از ستونهاي خاص نيست؛ كل ركورد، از جمله تمام ستونها، حذف مي شود. دستور DELETE را بايد با احتياط به كار برد، اين دستور بيش از اندازه خوب كار مي كند. دو قسمت آتي روشهاي حذف داده ها از جداول را بررسي مي كنند.

قـالـب

```
Delete from schema.table
[where condition];
```

تـــاي

```
delete from orders_t
```

خـروج

```
where ord_num =’23A
```

به كاربرد عبارت WHERE توجه كنيد. اگر بخواهيد سطرهاي مورد نظر خود را از يك جدول حذف كنيد، عبارت WHERE بخش ضروري DELETE خواهد بود. اين دستور بندرت بدون عبارت WHERE به كار مي رود. اگر اين كار را انجام دهيد، نتائج كسب شده مشابه مثال صفحه بعدي خواهند بود:

```
delete from orders_tbl;
11 row deleted.
```

نكته: اگر عبارت WHERE از دستور DELETE حذف شود، در آن صورت تمام سطرهاي داده ها از جدول حذف مي شوند. به عنوان يك قانون كلي، هميشه عبارت WHERE را با دستور DELETE به كار مي بريد.

مديريت تراكنشها ي بانك اطلاعاتي

تعريف تراكنش: تراكنشها، واحدهاي كاري يا يك سري كار هستند كه بطور دستي توسط نوعي برنامه بانك اطلاعاتي با يك ترتيب منطقي خاص انجام مي شوند. در بانك اطلاعاتي رابطه اي كه از SQL استفاده مي شود، تركنشها با استفاد از دستورات DMLاي انجام ميشوند هر تراكنش، اعمال يك يا چند تغيير در بانك اطلااعاتي است. به عنوان مثال اگر دستور UPDATE را براي تغيير يك نام در يك جدول اجرا كنيد، در آن صورت يك تراكنش را اجرا مي كنيد.

هر تراكنش مي تواند يك يا گروهي از دستورات ي DML باشد. به هنگام مديريت چندين گروه از تراكنش ها، هر يك از گروههاي مورد نظر يا بايد به طور موفقيت آميز باشد، يا هيچ يك از آنها موفقيت آميز نخواهد بود.

فهرست ذيل ماهيت تراكنشها را شرح مي دهد:

- تمام تراكنشها يك شروع و يك پايان دارند.
- هر تراكنش را مي توان ذخيره كرد يا لغو نمود.
- اگر تراكنشي در نيمه راه با شكست مواجه شود، هيچ قسمتي از آن را نمي توان در بانك اطلاعاتي ذخيره نمود.

كنترل تراكنشي: منظور از كنترل تراكنشي، قابليت مديريت تراكنشهاي گوناگوني است كه ممكن است در يك سيستم مديريت بانك اطلاعاتي رابطه اي رخ دهد. زماني كه درباره تراكنشها صحبت مي شود، منظور دستورات INSERT,UPDATE,DELETE است.

زماني كه تراكنشي اجرا مي شود و با موفقيت به پايان مي رسد، جدول مورد نظر فوراً تغيير نمي يابد، اگرچه ممكن است با توجه به خروجي اينگونه به نظر رسد. زماني كه تراكنشي با موفقيت به پايان ميرسد، دستورات كنترلي وجود دارند كه مي توان براي نهايي كردن تراكنش به كار برد. در اين حالت يا تغييرات حاصل از تراكنش ذخيره مي شوند، و يا لغو مي شوند.

سه دستور براي كنترل تراكنش وجود دارد:

- COMMIT
- ROLLBACK
- SAVEPOINT

نكته: دستورات كنترل تراكنش ها فقط با دستورات DML (DELETE,UPDATE, INSERT) مورد استفاده قرار مي گيرند.

پس از تكميل يك تراكنش، اطلاعات تراكنشي يا در يك ناحيه اختصاص يافته ذخيره مي شوند، و يا در يك ناحيه rollback موقت در بانك اطلاعاتي. تمام تغييرات در اين ناحيه موقت نگه داشته مي شوند تا يك دستور كنترل تراكنش صادر شود. زماني كه يك دستور كنترل تراكنشي صادر مي شود، تغييرات يا اعمال مي شوند و يا ناديده انگاشته مي شوند؛ سپس ناحيه rollback موقت تخليه مي شود.

دستورCOMMIT :

دستور COMMIT دستور تراكنشي است كه براي ذخيره سازي تغييرات اعمال شده به يك بانك اطلاعاتي مورد استفاده قرا مي گيرد. دستور COMMIT تمامي تراكنشهاي پس از آخرين دستور ROLLBACK ياCOMMIT پيشين را ذخيره ميكند.

قالب

قالب اين دستور بصورت مقابل است:

commit [work]

كلمه كليدي COMMIT تنها بخش بخش اجباري قالب اين دستور است. كاراكتري كه براي نشان دادن پايان دستور به كار مي رود و به نسخه پياده سازي شدة مورد استفاده بستگي دارد نيز اجباري است. WORK يك كلمه كليدي است كه كاملاً اختياري است؛ تنها هدف آن اين است كه دستور كاربر پسند شود. در مثال صفحة بعد كار با تمام داده هاي جدول PRODUCT_TBL آغاز مي شود:

select * from product_tmp;

PROD_ID	PROD_DESC	COST
11235	WITCHES COSTUME	29.99
222	PLASTIC PUMPKIN 18 INCH	7.75
13	FALSE PARAFFIN TEETH	1.1
90	LIGHTED LANTERNS	14.5
15	ASSORTED COSTUMES	10
9	CANDY CORN	1.35

11 rows selected.

سپس، تمام ركوردهايي كه قيمت آنها كمتر از $ 14.00 است حذف مي شوند:

```
delete from product_tmp
where cost < 14;

8 rows deleted.
```

دستور COMMIT براي ذخيره تغييرات در بانك اطلاعاتي و تكميل تراكنش صادر مي شود.

```
Commit;
Commit complete.
```

توجه : اگر حجم اطلاعات به هنگام unload و load كردن زياد است، توصيه مي شود كه دستور COMMIT را زود به زود به كار بريد ؛ اما،كاربرد بيش از اندازه آن سبب مي شد كه كار در دست اجرا زمان زيادي طول بكشد. به خاطر داشته باشيد كه تمام تغييرات در ابتدا به ناحيه rollback موقت ارسال مي شوند. اگر اين ناحيه موقت پر شود، و امكان ذخيره تغييرات در بانك اطلاعاتي فراهم نباشد، احتمالاً بانك اطلاعاتي ديگر امكان فعاليتهاي تراكنشي را فراهم نخواهد ساخت.

دستور ROLLBACK: اين دستور ، دستور كنترل تراكنشي است كه براي لغو تراكنشهايي كه هنوز در بانك اطلاعاتي ذخيره نشده اند به كار مي رود. اين دستور را تنها براي لغو تراكنشهايي كه از زمان آخرين دستور COMMIT يا ROLLBACK به اين سو اجرا شده اند به كار برد.

قـا لـب

قالب اين دستور بصورت مقابل است:

```
rollback [work]
```

در اينجا همچون دستور COMMIT، كلمه كليدي WORK يك بخش اختياري در قالب دستور است.

در مثال ذيل، كار با انتخاب تمام ركوردها از جدول PRODUCTS_TMP پس از حذف 14 ركورد آخر آغاز مي شود:

```
select * from products_tmp;
```

تـا ي

PROD_ID	PROD_DESC	COST
11235	WITCHES COSTUME	29.99
90	LIGHTED LANTERNS	14.5

2345 OAK
BOOKSHELF 59.99

3 rows selected.

سپس، جدول به روز رسانده شده و قيمت كالايي كه شماره شناسايي آن 11235 است به $39.99 تغيير مي يابد:

تايپ

```
update products_tmp
set cost = 39.99
where prod_id = '11235'
```

1 row updated.

اگر پرس و جويي را براي اين جدول اجرا كنيد، خواهيد ديد كه تغيير اعمال شده است:

تايپ

```
select * from products_tmp;
```

PROD_ID	PROD_DESC	COST
11235	WITCHES COSTUME	39.99
90	LIGHTED LANTERNS	14.5
2345	OAK BOOKHELF	59.99

3 rows selected.

اينك دستور ROLLBACK را براي لغو آخرين تغيير صادر كنيد:

```
rollback;
rollback copmlete.
```

دستور SAVEPOINT: هر savepoint نقطه اي در يك تراكنش است كه مي توانيد تراكنش ها را بدون لغو كلي تا بدان نقطه لغو كنيد.

قالب

دستور savepoint به شكل مقابل است:

```
savepoint savepoint_name
```

اين دستور تنها بريا ايجاد يك SAVEPOINT در بين دستورات تراكنشي است. دستور ROLLBACK براي لغو گروهي از تراكنشها مورد استفاده قرار مي گيرد. SAVEPOINT روشي براي مديريت تراكنشها از طريق تجزية تعداد زيادي از آنها به گروههاي كوچكتر قابل مديريت تر است.

قـالـب

قالب دستور ROLLBACK براي لغو تغييرات تا يك نقطه SAVEPOINT به شكل مقابل است:

```
ROLLBACK TO SAVEPOINT_NAME;
```

مقدمه اي بر پرس وجو از بانك اطلاعاتي

هر پرس و جو، استعلام از يك بانك با استفاده از دستور SELECT است. هر پرس وجو براي استخراج داده ها از بانك اطلاعاتي، با يك فرمت خوانا مطابق با درخواست كاربر، مورد استفاده قرار مي گيرد. به عنوان مثال، اگر جدولي براي كارمندي داشته باشيد، ممكن است دستور SQLاي را صادر كنيد كه مشخص كننده نام كارمندي است كه بيشترين دريافتي را دارد. اين درخواست از بانك اطلاعاتي يك پرس و جوي نوعي است كه ممكن است در يك بانك اطلاعاتي رابطه اي مورد استفاده قرار گيرد.

دستور SELECT: اين دستور، دستوري است كه نمايانگر زبان پرس و جوي داده ها(DQL) در SQL است، بريا ساختن پرس و جوي بانك اطلاعاتي مورد استفاده قرار گيرد. دستور SELECT يك دستور مستقل نيست، بدين معنا كه به عبارتهايي نياز است. علاوه بر عبارتهاي مورد نياز، عبارتهاي اختياري وجود دارد كه قابليت كلي اين دستوئر را افزايش مي دهند.

دستور SELECT بدون شك يكي از قدرتمندترين دستورات SQL است.

عبارت FROM عبارت اجباري است كه بايد هميشه با SELECTبه كار برده شود.

چهار كلمه كليدي، يا عبارت وجود دارد كه بخشهاي ارزشمند هر دستور SELECT هستند. اين كلمات كليدي عبارتند از:

1. SELECT
2. FROM
3. WHERE
4. ORDER BY

دستور SELECT: دستور SELECT به همراه عبارت FROM به كار برده مي شود تا داده هاي خوانا و سازمان يافته از بانك اطلاعاتي استخراج كند. بخش SELECT پرس و جو براي انتخاب داده هايي است كه قرار است مطابق با ستونهايي كه در جدول ذخيره شده اند مشاهده شوند.

قـالـب

قالب يك دستور SELECT ساده به شكل ذيل است:

```
select [ * | all | distinct column1,column2]
from table [ , table2];
```

پس از كلمه كليدي SELECT فهرست ستونهايي مي آيد كه قرار است به عنوان بخشي از خروجي پرس و جو نمايش داده شوند. پس از كلمه كليدي

FROM نيز فهرست يك يا چند مورد از جدولي مي آيد كه داده ها از آن(ها) انتخاب مي شوند. (*) نشان دهندة آن است كه تمام ستونهاي يك جدول بايد به عنوان بخشي از خروجي نمايش داده شوند. براي آشنايي با كاربرد آن، نسخه مورد استفاده خود را بررسي كنيد. گزينه ALL براي نمايش تمام مقادير يك ستون مورد استفاده قرار مي گيرد، از جمله موارد تكراري. گزينه DISTINCT براي حذف سطرهاي تكراري است. پيش فرض بين دو گزينه مذكور، ALL است كه نيازي به نوشتن آن در دستور نيست. توجه كنيد كه ستونهاي پس از كلمه كليدي SELECT بوسيله كاما از يكديگر جدا مي شوند، درست همان گونه كه نام جداول از يكديگر جدا مي شوند.

با مطا لعه مثالهاي ذيل قابليتهاي پايه دستور SELECT را بررسي كنيد. ابتدا يك پرس و جوي ساده را براي جدول PRIDUCTS_TBL اجرا كنيد.

تـــاي

```
SELECT * FROM PRODUCTS_TB
```

خروج

```
PROD_ID                    PROD_DES
                                                    COST
-----------------------------------
11235                      WITCHES  COSTUME
                                                   29.99
222                        PLASTIC  PUMPKIN  18  INCH
                                                    7.75
13                         FALSE  PARAFFIN  TEETH
                                                     1.1
90                         LIGHTED  LANTERNS
                                                    14.5
15                         ASSORTED  COSTUMES
                                                      10

9 rows selected.
```

(*) نمايشگر تمام ستونهاي جدول است، و همانگونه كه مشاهده مي كنيد، به فرم PROD_ID,PROD_DESC,COST نمايش داده شده اند. هر ستون در خروجي با همان ترتيبي كه در جدول ظاهر مي شود، نمايش داده مي شود. 9 ركورد در اين جدول وجود دارد كه پيام 9 rows selected در انتهاي خروجي نمايانگر همين امر است. اين پيام در نسخه هاي مختلف متفاوت است؛ به عنوان مثال، يك پيام ديگر براي پرس وجو ممكن است به صورت 9 rows selected. باشد.

اينك داده ها از جدول ديگري به نام CANDY_TBL انتخاب مي شوند. اين جدول را با استفاده از جدول PRODUCTS_TBL براي مثالهاي ذيل ايجاد كنيد. تنها نام يك ستون را پس از كلمه كليدي SELECT بياوريد.

```
SELECT PROD_DESC FROM CANDY_TBL;
```

تـــاي

```
PROD_DESC
----------------------------------------
CANDY CORN
CANDY CORN
CANDY CORN
HERSHEYSE KISS
SMARTIES

4 rows selected.
```

خروجي

چهار ركورد در جدول CANDY_TBLموجود است. گزينه ALL در دستور آتي به كار برده شده تا نشان داده شود كه ALL اختياري و اضافه است. هيچ گاه نيازي به مشخص كردن ALL نيست، يك گزينه پيش فرض است.

تـــاي

```
SELECT ALL PROD_DESC
FROM CANDY_TBL;

PROD_DESC
--------------------------
CANDY CORN
CANDY CORN
CANDY CORN
HERSHEYS KISS
SMARTIES

4 rows selected.
```

خروجي

گزينه DISTINCT در مثال ذيل به كار برده شده تا از نمايش ركوردهاي تكراري پيشگيري شود. توجه كنيد كه مقدار CANDY CORN تنها يك بار در اين مثال نمايش داده مي شود.

تـــاي

```
SELECT DISTINCT PROD_DESC
FROM CANDY_TBL;

PROD_DESC
--------------------------
```

خروجي

```
CANDY CORN
HERSHEYS KISS
SMARTIES

3 rows selected.
```

دو گزينه DISTINCT و ALL را مي توان با ستون مورد نظر به كار برد. در اين حالت نام ستون بايد بين پرانتز نوشته شود. كاربرد پرانتزها در SQL ، و بسياري از زبانهاي ديگر، اغلب براي بهبود خوانايي است.

عبارت FROM : اين عبارت همراه با دستور SELECT به كاربرده مي شود. اين عبارت يك عنصر ضروري براي هر پرس و جو است. اين عبارت براي بانك اطلاعاتي مشخص مي كند كه كدام جداول را براي بازيابي داده هاي مورد نياز پرس و ججو مورد نظر دستيابي قرار دهد. يك يا چند جدول را مي توان در عبارت FROM مشخص نمود.

قالب

قالب عبارت FROM به شكل زير است:

```
from  table1 [ , table2]
```

استفاده از شرطها براي تمايز داده ها: هر شرط بخشي از يك پرس و جو است كه براي نمايش اطلاعات مورد نظر كاربر مورد استفاده قرار مي گيرد. مقدار هر شرط «درست» يا «نادرست» است، و در نتيجه داده هاي دريافتي از پرس و جو را محدود مي كند. عبارت WHERE براي تعيين شرطهاي يك پرس و جو مورد استفاده قرار گرفته و بدين ترتيب سطر هايي حذف مي شوند كه در غير اين صورت بدون شرطهاي پرس و جو بازگردانده مي شوند.

بيش از يك شرط را مي توان در عبارت WHERE به كار برد. اگر بيش از ثيك شرط وجود داشته باشد، اين شرطها با استفاده از عملگرهاي AND و OR با يكديگر تركيب مي شوند.

قالب

قالب عبارت WHERE به شكل ذيل است:

```
select [ all | * | distinct column1, column2 ]
from table [ , table2 ]
where [ condition1 | expression1 ]
[ and condition2 | expression2 ]
```

مثال صفحه بعدي نشان دهنده يك SELECT ساده بدون شرط است:

تاي

```
SELECT
FROM  PRODUCTS_TBL
```

خروجي

PROD_ID PROD_DESC COST

PROD_ID	PROD_DESC	COST
11235	WHITCHES COSTUME	29.99
222	PLASTIC PUMPKIN 18 INCH	7.75
13	FALSE PRAFFIN TEETH	1.1
90	LIGHTED LANTERNS	14.5
15	ASSORTED COSTUMES	10

5 rows selected.

اينك شرطي را به پرس و جوي مثال قبلي مي افزاييم:

```
SELECT * FROM PRODUCTS_TE
WHERE COST < 5;
```

تـا ي

PROD_ID PROD_DESC COST

PROD_ID	PROD_DESC	COST
13	FALSE PARAFFIN TEETH	1.1
9	CANDY CORN	1.35
6	PUMP KIN SPIDERS	1.45
87	PLASTIC SPIDERS	1.05

4 rows selected.

5 تنها ركوردهايي نمايش داده مي شوند كه قيمت كالا در آنها كمتر از دلار است. در مثال ذيل مي خواهيم شرح كالاهايي را نمايش دهيم كه شماره شناسايی آن 119 است.

```
SELECT PROD_DESC, COS
```

تـا ي

```
FROM  PRODUCTS_TBL
WHERE PROD_ID = ‘ 119 ’ ;
```

خروجي

```
PROD_DESC                CO
ASSORTED MASKS    4.95

1 row selected.
```

مرتب سازي خروجي ها: معمولآ خروجيها بايد به شكلي مرتب شوند. داده ها را مي توان با استفاده از عبارت ORDER BY مرتب نمود. اين عبارت نتائج يك پرس و جو را با فرمي كه شما مشخص مي كنيد مرتب مي كند. ترتيب پيش فرض عبارت ORDER BY صعودي است؛ چنانچه نامهايي بر اساس حروف افبا مرتب شوند، ترتيب آنها از A تا Z خواهد بود. ترتيب نزولي خروجي هاي الفبايي از Z تا A خواهد بود. مرتب سازي مقادير عددي 1 تا 9 به گونه اي خواهد بود كه ارقام از يك تا 9 نمايش داده مي شوند، براي ترتيب نزولي نيز از 9 تا 1 نمايش داده مي شوند.

قالب

قالب دستوري ORDER BY به شكل ذيل خواهد بود:

```
select [ all | *  | distinct column1, column2 ]
from taable1 [ , table2 ]
where [ condition1 | expression1 ]
[ and condition2 | expression2 ]
ORDER BY COLUMN1 | integer [ ASC | DESC ]
```

بررسي عبارت ORDER BY را با بسط يكي از دستورات پيشين آغاز مي كنيم. شرط كالاها را به طور صعودي بر اساس حروف الفبا مرتب خواهيم كرد. به كاربرد گزينه ASC توجه كنيد. اين گزينه را مي توان در عبارت ORDER BY پس از هر ستوني به كار برد.

تاي

```
SELECT  PROD_DESC,  PROD_ID,  COS
FROM  PRODUCTS_TB
WHERE  COST < 20
ORDER  BY  PROD_DESC  DE
```

خروجي

```
PROD_DESC                                PROD_ID
                                            COST
-------------------------------------------------
ASSORTED  COSTUMES                       15
                                            10
```

ASSORTED MASKS	119	4.95
CANDY CORN	9	1.35
FALSE PARAFFIN TEETH	13	1.1
LIGHTED LANTERNS	90	14.5
PLASTIC PUMPKIN 18 INCH	222	7.75

6 rows selected.

نام مستعار ستونها: نام مستعار ستونها را مي توان براي تغيير نام ستونهاي يك جدول براي يك پرس وجو خاص به كار برد. جدول employee_tbl كاربرد نام مستعار ستونها را نشان ميدهد.

قالب

```
SELECT COLUMN_NAME ALLAS_[illegible]
FROM TABLE_NAME;
```

مثال ذيل شرح كالا را دو مرتبه نمايش مي دهد و نام مستعار PRODUCT را به ستون دوم اختصاص مي دهد. به هدر ستونها در خروجي توجه كنيد.

تاي

```
Select prod_desc,
Prod_desc product
From products_
```

خروجي

PROD_DESC	PRODUCT
WHITCHES COSTUME	WHITCHES COSTUME
PLASTIC PUMPKIN 18 INCH	PLASTIC PUMPKIN 18 INCH

PROD_DESC	PRODUCT
FALSE PARAFFIN TEETH	FALSE PARAFFIN TEETH

```
LIGHTED LANTERNS                    LIGHTED
                                      LANTERNS
ASSORTED COSTUMES                   ASSORTED
                                      COSTUMES
CANDY CORN                            CANDY
                                      CORN
PUMPKIN CANDY                       PUMPKIN
                                      CANDY

7 rows selected.
```

نام مستعار ستونها را مي توان براي تغيير نام هدر ستونها با توجه به نيازهاي شخصي به كار برد. آنها را همچنين مي توان براي ارجاع به يك ستون با يك نام كوتاهتر در برخي از نسخه هاي پياده سازي شدة SQL به كار برد.

استفاده از عملگرها براي دسته بندي داده ها

هر عملگر، كلمه رزرو شده يا كاراكتري است كه در عبارت WHERE يك دستور SQL به منظور انجام عملياتي چون مقايسه و عمليات حسابي مورد استفاده قرار مي گيرد. عملگرها براي مشخص كردن شرط ها در يك دستور SQL و براي تركيب چند شرط در يك دستور مورد استفاده قرار مي گيرند. برخي از اين عملگرها در ذيل آمده اند:

- عملگرهاي مقايسه
- عملگرهاي منطقي
- عملگرهاي مورد استفاده براي نفي شرط ها
- عملگرهاي حسابي

عملگرهاي مقايسه: عملگرهاي مقايسه براي آزمايش مقادير منفرد در يك دستور SQL مورد استفاده قرار مي گيرد. عملگرهاي مقايسة مورد بررسي عبارتند از: <> , > , < , =

در اين مورد مثالهايي در قسمتهاي ذيل آورده شده اند.

تساوي

عملگر تساوي مقادير را در يك دستور SQL با يكديگر مقايسه مي كند. علامت (=) نماد عملگر تساوي است. زماني كه تساوي دو مقدار آزمايش مي شود، مقادير بايد دقيقآ همخواني داشته باشند، در غير اينصورت هيچ داده اي بازگردانده نمي شود. اگر دو مقدار در طي يك مقايسه مساوي باشند، مقداري كه براي مقايسه بازگردانده مي شود، «درست» است؛ اگر دو مقدار مساوي نباشند، مقدار حاصل «نادرست» خواهد بود. از اين مقدار بولي(درست/نادرست) براي تعيين اينكه داده ها مطابق با شرط هستند يا خير استفاده مي شود.

عملگر = را مي توان به تنهايي يا با عملگرهاي ديگر به كار برد. مثالي از كاربرد و مفهوم اين عملگر در ذيل نشان داده شده است.

كاربرد	مفهوم	
' 20000 ' = SALARY WHERE		حقوق

مساوي 20000

پرس و جوي ذيل تمام سطر هايي را بر مي گرداند كه مقدار PROD_ID مساوي 2345 است.

```
SELECT *
FROM PRODUCTS_TBL
WHERE PROD_ID = ' 2345 ';
```

نامساوي

بريا هر تساوي، يك نامساوئي وجود دارد. عملگري كه در SQL براي نامساوي مورد استفاده قرار مي گيرد، < > است. چنانچه شرط مورد بررسي نامساوي باشد، حاصل «درست» خواهد بود، چنانچه شرط مورد بررسي مساوي باشد، حاصل «نادرست» خواهد بود؛

نكته: گزينه ديگري كه با < > قابل استفاده است، =! است. در بسياري از نسخه هاي پياده سازي شده SQL از =! براي نشان دادن نامساوي استفاده شده است.

كاربرد	مفهوم	
' 20000 ' < > SALARY WHERE		حقوق

مخالف 20000

كوچكتر، بزرگتر

نمادهاي > (كوچكتر) و < (بزرگتر) را مي توان به تنهايي يا با هر يك از عملگرهاي ديگر به كار برد.

كاربرد	مفهوم	
' 20000 ' < SALARY WHERE		حقوق
كمتر از 20000		
' 20000 ' > SALARY WHERE		حقوق
بيشتر از 20000		

در مثال نخست، هر چيزي كه كمتر از و مخالف 20000 باشد سبب بازگرداندن «درست» ميشود. مقدار 20000 يا بيشتر از آن سبب بازگرداندن «نادرست» ميشود. عملگر «بزرگتر» بر خلاف «كوچكتر» عمل مي كند.

تـــاي

```
SELECT
FROM PRODUCTS_TBL
WHERE PROD_ID > ' 23
```

خـروجي

```
PROD_ID                                 PROD_DESC
                                                                COST
-----------------------------------------
11235                                   WITCHES  COSTUME
                                                                29.99
2345                                    OAK  BOOKSHEFKL
                                                                59.99
7725                                    LEATHER  GLOVES
                                                                24.99

                                                3 rows selected.
```

عملگرهاي منطقي: عملگرهاي منطقي، آن عملگرهايي هستند كه به جاي نمادها از كلمات كليدي SQL براي مقايسه استفاده مي كنند. عملگرهاي منطقي مورد بررسي ذيل عبارتند از:

- IS NULL
- BETWEEN
- IN
- LIKE
- EXISTS
- UNIQUE
- ANY و ALL

IS NULL : از عملگر NULL براي مقايسه يك مقدار تهي استفاده مي شود.

BETWEEN : عملگر between براي جستجوي مقاديري كه بين كمينه و بيشينه مقادير مشخص شده قرار دارند مورد استفاده قرار مي گيرد. بيشينه و كمينه مقادير به عنوان بخشي از مجموعه شرطي مورد نظردر نظرگرفته مي شوند.

IN : اين عملگر براي مقايسه يك مقدار با فهرستي از مقادير ليترال مشخص شده مورد استفاده قرار مي گيرد. براي اينكه حاصل كار ‹‹درست›› باشد، مقدار مقايسه بايد با حداقل يكي از مقادير درون فهرست يكسان باشد.

LIKE : اين عملگر براي مقايسه يك مقدار با مقادير مشابه با استفاده از عملگرهاي جانشين مورد استفاده قرار مي گيرد. دو عملگر جانشين در رابطه با عملگر L IKE وجود دارد، علامت درصد(%) و (-)

علامت % نماينگر هيچ، يك، يا چند كاراكتر است. (-) نمايانگر يك عدد يا كاراكتر است. اين نمادها را مي توان با هم به كار برد.

EXISTS : عملگر EXISTS براي جستجوي وجود يك سطر با شرط معيين در يك جدول مورد استفاده قرار مي گيرد.

WHERE EXISTS (SELECT EMPLOYEE_ID
جستجو براي وجود كارمندي با
FROM EMPLOYEE_TBL
شماره 33333
WHERE EMPLOYEE_ID = ‘33333’);

UNIQUE : اين عملگر تمامي سطرهاي جدول مشخص شده را براي منحصر بفرد بودن جستجو مي كند.

عملگرهاي ALL و ANY : عملگر ALL براي مقايسه يك مقدار با تمامي مقادير يك مجموعه مورد استفاده قرار مي گيرد. عملگر ANY براي مقايسه يك مقدار با هر يك از مقادير حائز شرط در يك فهرست مورد استفاده قرار مي گيرد.

عملگرهاي منطقي : اگر بخواهيد با استفاده از چند شرط محدوده داده هاي يك دستور SQL را تنگ تر كنيد چه خواهيد كرد؟ مي بايست بتوانيد چند شرط را تركيب كنيد، و اين كار با عملگرهايي كه عملگرهاي منطقي تركيب ناميده مي شوند انجام مي شود. اين عملگرها عبارتند از:

- AND
- OR

اين عملگرها امكان انجام چند مقايسه را با عملگرهاي مختلف در يك دستور SQL را فراهم مي سازند. قسمتهاي ذيل عملكرد هر يك از اين عملگرها را شرح مي دهند.

AND : اين عملگر وجود چند شرط را در عبارت WHERE يك دستور SQL فراهم مي كند. براي اينكه عملي توسط دستور SQL انجام شود، چه يك تراكنش و چه يك پرس و جو، تمام شرطهايي كه توسط AND از يكديگر جدا شده اند بايد «درست» باشند.

OR : اين عملگر براي تركيب چند شرط در عبارت WHERE يك دستور SQL مورد استفاده قرار مي گيرد. براي اينكه عملي توسط دستور SQL انجام شود، چه يك تراكنش چه يك پرس و جو، حداقل يكي از شرط هايي كه توسط OR از يكديگر جدا شده اند بايد درست باشد.

عملگرهاي حسابي: عملگرهاي حسابي براي انجام عمليات رياضي در SQL مورد استفاده قرار مي گيرند، همچون بسياري از زبانهاي ديگر. چهار عملگر مرسوم براي عمليات رياضي وجود دارد.

+(جمع)
-(تفريق)
*(ضرب)
/(تقسيم)

چند مثال در ذيل نشان داده شده است:

```
SELECT SALARY * 10 + 1000
FROM EMPLOYEE_PAY_TBL
WHERE SALSRY > 20000

SELECT (SALARY / 52 ) + BONUS
FROM EMPLOYEE_PAY_TBL

SELECT (SALSRY –100 + 1000 + BONUS) / 52 *
1.1
FROM EMPLOYee_PAY_TBL

SELECT SALARY
FROM EMPLOYEE_PAY_TBL
WHERE SALARY < BONUS * 3 + 10 / 2 –50
```

خلاصه سازي نتائج يك پرس و جو

توابع جمعي: توابع كلمات كليدي در SQL هستند كه براي پردازش مقادير موجود در ستونها براي خروجي ها مورد استفاده قرار مي گيرند. هر تابع دستوري است كه هميشه با نام يك ستون يا عبارت به كار برده مي شود. انواع مختلفي از توابع در SQL وجود دارد.

توابع جمعي، توابعي هستند كه داده ها را شمارش مي كنند، خلاصه مي كنند، حد آنها را نشان مي دهند. MIN,AVG,COUNT,SUM و MAX توابع جمعي هستند.

تابع COUNT : اين تابع براي شمارش سطرها يا مقادير ستوني به كار مي رود كه داراي مقدار تهي نيستند. زماني كه اين تابع در يك پرس و جو به كار برده مي شود، يك مقدار عددي را بر مي گرداند. زماني كه اين تابع با DISTINCT به كار برده مي شود، تنها سطرهاي غير تكراري شمارش مي شوند. ALL(نقطه مقابل DISTINCT) پيش فرض اين كار است؛ نيازي به كاربرد ALL در قالب دستور نيست. چنانچه DISTINCT به كار برده نشود، سطرهاي تكراري شمارش مي شوند. يك روش ديگر براي استفاده از اين تابع، كاربرد آن با يك * است. زماني كه اين تابع با يك * به كار برده مي شود، تمام سطرهاي يك جدول، از جمله سطرهاي تكراري، اعم از اينكه يك ستون داراي مقدار تهي باشد يا نباشد، شمارش ميشوند. قالب تابع COUNT به صورت ذيل است:

```
count [ (*) | (DISTINCT | ALL) ]
```

بايد توجه داشت كه DISTINCT را نمي توان با (*) به كار برد، تنها با (نام ستون) COUNT

با تايپ دستور SELECT COUNT (*) FROM EMPLOYEE_TBL; خروجي بصورت ذيل خواهد شد:

```
COUNT(*)
--------------
6
```

تابع SUM : اين تابع براي به دست آوردن مجموع مقادير يك ستون از گروهي از سطرها مورد استفاده قرار مي گيرد. اين تابع را مي توان با DISTINCT نيز به كار برد. زماني كه تابع SUM با DISTINCT به كار برده مي شود، تنها حاصل جمع سطرهاي غير تكراري محاسبه مي شود كه ممكن است چندان مفيد نباشد. حاصل جمع در اين صورت دقيق نيست، چرا كه سطرهايي از داده هايي كنار گذاشته شده اند. قالب تابع SUM بصورت ذيل است:

SUM ([DISTINCT] COLUMN NAME)

كاربرد	مفهوم
SELECT SUM(SALARY) FROM EMPLOYEE_PAY_TBL;	حاصل مجموع را بدست مي آورد.
SELECT SUM(DISTINCT SALARY) FROM EPLOYEE_PAY_TBL;	حاصل جمع حقوق در سطرهاي غير تكراري را به دست مي آورد.

تابع AVG : اين تابع براي پيدا كردن ميانگين گروهي از سطرها مورد استفاده قرار مي گيرد. زماني كه اين تابع با DISTINCT به كار برده مي شود، ميانگين سطرهاي غير تكراري را به دست مي آورد. قالب تابع AVG بصورت ذيل است:

AVG ([DISTINCT] COLUMN NAME)

كاربرد	مفهوم
SELECT AVG(SALARY) FROM EMPLOYEE_PAY_TBL;	متوسط حقوق را بر مي گرداند.
SELECT AVG(DISTINCT SALARY) FROM EMPLOYEE_PAY_TBL;	ميانگين حقوق سطرهاي غير تكراري را بر مي گرداند.

تابع MAX : اين تابع براي به دست آوردن بيشينه مقدار مقادير يك ستون در گروهي از سطرها مورد استفاده قرار مي گيرد. مقادير تهي به

هنگام استفاده از تابع MAX نائيده انگاشته مي شوند. كاربرد DISTINCT اختياري است. اما چون بيشينه مقدار تمام سطرها با سطرهاي غير تكراري يكسان است، از اين رو كاربرد اين گزينه بي فايده است. قالب دستور MAX بصورت ذيل است:

MAX ([DISTINCT] COLUMN NAME)

كاربرد	مفهوم
SELECT MAX(SALARY) FROM EMPLOYEE_PAY_TBL;	بيشترين حقوق را بر مي گرداند.
SELECT MAX(DISTINCT SALARY) FROM EMPLOYEE_PAY_TBL;	بيشترين حقوق را بر مي گرداند.

تابع MNI : اين تابع،گمينه مقدار يك ستون را براي گروهي از سطرها برمي گرداند. مقادير تهي به هنگام استفاده از تابع MIN ناديده انگاشته مي شوند. كاربرد DISTINCT اختياري است. اما چون كمينه مقدار تمام سطرها با سطرهاي غير تكراري يكسان است،از اين رو كاربرد اين گزينه بي فايده است. قالب اين دستور بصورت ذيل است:

MIN([DISTINCT] COLUMN NAME)

كاربرد	مفهوم
SELECT MIN(SALSRY) FROM EMPLOYEE_PAY_TBL;	كمترين حقوق را بر مي گرداند.
SELECT MIN(DISTINCT SALARY) FROM EMPLOYEE_PAY_TBL;	كمترين حقوق را بر مي گرداند.

<u>موضوعات پيشرفته SQL :</u>

در اين قسمت برخي از موضوعات پيشرفته SQL مورد بررسي قرار گرفته است. در پايان بحث مفاهيم مربوط به Cursor ها، رويه هاي ذخيره شده، triggerها، SQL پويا و SQL مستقيم در مقابل SQL گنجانده شده را خواهيد دانست.

CURSOR ها:

از نظر بيشتر اشخاص، cursor يك نقطه يا يك خط تيره چشمك زن است كه در صفحه نمايشگر ظاهر شده و محل شما را در يك فايل يا برنامه كاربردي نشان مي دهد؛ اما cursor هاي مورد بررسي در اين قسمت از

اين نوع نيستند. Cursor در SQL، ناحيه اي در حافظه بانك اطلاعاتي است كه آخرين دستور SQL در آنجا ذخيره مي شود. اگر دستور جاري يك پرس و جو باشد، در آن صورت سطري از پرس و جو در حافظه نيز ذخيره ميشود. اين سطر، مقدار جاري cursorيا سطر جاري است. ناحيه موجود در حافظه نامگذاري شده و در دسترس تمام برنامه هاست.

هر cursor عموماً براي بازيابي زير مجموعه اي از داده ها از بانك اطلاعاتي مورد استفاده قرار مي گيرد. از اين رو، هر سطر در cursor توسط يك برنامه قابل ارزيابي است. Cursorها عموماً در دستورات SQL مورد استفاده قرار مي گيرند كه در برنامه هاي پروسيجرال گونه گنجانده مي شوند. برخي از cursor ها بطور ضمني توسط سسرويس دهنده بانك اطلاعاتي ايجاد مي شوند، در صورتي كه برخي ديگر توسط برنامه سازان ايجاد مي شوند. كاربرد cursor ها در نسخه هاي SQL ممكن است متفاوت باشد.

در اين قسمت قالبهاي مورد استفاده در دو نسخه مشهور را نشان مي دهد: ميكروسافتSQL Server و اراكل.

قالب دستور تعريف يك cursor براي ميكروسافت به شكل زير است:

DECLARE CURSOR_NAME CURSOR
FOR SELECTدستور
[FOR [READ ONLY | UPDATE [COLUMN]]]

قالب مورد استفاده در اراكل به شكل زير است:

DECLARE CURSOR CURSOR_NAME
IS {SELECTدستور}

طبق استاندارد ANSI، عمليات ذيل پس از تعريف يك cursor براي دستيابي به آن مورد استفاده قرار مي گيرند:

OPEN : cursor تعريف شده اي را باز مي كند

FETCH : سطرهايي را از يك cursor بازيابي و در متغييري از يك برنامه ذخيره مي كند

CLOSE : پس از انجام عمليات، cursor را مي بندد.

باز كردن يك cursor :

پس از باز كردن يك cursor، دستور SELECT مربوط به cursor مورد نظر اجرا شده و نتائج پرس و جو در ناحيه اي از حافظه ذخيره مي شوند. قالب دستور استفاده شده در BASE d به شكل زير است:

OPEN CURSOR_NAME

قالب دستور در اراكل بصورت زير است:

OPEN CURSOR [PARAMETER1 [PARAMETER2]]

براي باز كردن EMP_CURSOR :

```
OPEN EMP_CURSOR
```

واكشي داده ها از يك CURSOR

محتواي يك cursor را مي توان با استفاده از دستور FETCH بازيابي نمود، البته پس از باز كردن آن. قالب دستور FETCH در ميكروسافت SQL SERVER به شكل زير است:

```
FETCH CURSOR_NAME [ INTO FETCH_LIST ]
```

قالب دستور در اراكل به شكل زير است:

```
FETCH CURSOR_NAME {INTO : HOST_VARIABLE
[[ INDICATOR ] : INDICATOR_VARIABLE ]
[ , : HOST_VARIABLE
[[ INDICATOR ] : INDICATOR_VARIABLE ] ]
| USING DESCRPTOR DESCRIPTOR }
```

قالب دستور در d BASE به شكل زير است:

```
FETCH CURSOR_NAME into MEMORY_VARIABLES
```

براي واكشي محتواي EMP_CURSOR و ذخيره آنها در متغييري به نام EMP_RECORD، دستور FETCH به شكل زير خواهد بود:

```
FETCH EMP_CURSOR INTO EMP_RECORD
```

بستن يك CURSOR

چون cursor ها قابل بازكردن هستند، قابل بستن نيز مي باشند. بستن يك cursor كاملاً ساده است. Cursor ها پس از بستن ديگر در دسترس برنامه ها نيستند.

قالب دستور مورد استفاده براي بستن يك cursor و آزاد كردن حافظه آن در ميكروسافت SQL Server به شكل زير است:

```
CLOSE CURSOR_NAME

DEALLOCATE CURSOR CURSOR_NAME
```

زماني cursor كه در اراكل بسته مي شود، منابع و نام آن بدون دستور deallocate آزاد ميشوند

قالب دستور در اراكل به شكل زير است:

```
CLOSE CURSOR_NAME
```

براي آزاد سازي منابع در d BASE، جدول بايد پيش از آزاد شدن منابع و نام، بسته و از نو باز شود. قالب دستور در d BASE به شكل زير است:

CLOSE CURSOR_NAME

رويه هاي ذخيره شده

رويه هاي ذخيره شده، گروه هايي از دستورات SQL مرتبط به هم هستند. عموماً تحت عنوان توابع و زير برنامه شناخته شده هستند، كه سهولت و انعطاف پذيري را براي يك برنامه ساز به همراه دارند. اين سهولت و انعطاف پذيري به دليل آنست كه اجراي يك رويه ذخيره شده اغلب آسانتر از اجراي دستورات S QL مجزا است. رويه هاي ذخيره شده ممكن است رويه هاي ذخيره شده ديگري در خود داشته باشند. يعني، يك رويه ممكن است رويه ديگري را فرا بخواند، و آن رويه نيز به نوبه خود رويه ديگري را فرا بخواند. زماني كه يك رويه ذخيره شده ايجاد مي شود، زير برنامه ها و توابع گوناگون تشكيل دهندة آن بانك اطلاعاتي ذخيره مي شوند. اين رويه ها از پيش كامپايل شده، و به محض فراخواني توسط يك كاربر اجرا مي شوند. قالب دستور ايجاد يك رويه در ميكروسافت SQL Server به شكل زير است:

```
CREATE PROCEDURE PROCEDURE_NAME
[ [ ( ] @ PARAMETER_NAME
DATATYPE [ (LENGTH) | (PRECISION [ , SCALE
) ]
[ =DEFAULT ][ OUTPUT ]
[ , @ PARAMETE_NAME
DATATYPE [ (LENGTH) | (PRECISION [ , SCALE
])
] ] [ = DEFAULT ][ OUTPUT ]][ )
[ WITH RECOMPLIE ]
AS SQL_STATEMENTS
```

قالب دستور در اراكل به شكل زير است:

```
CREATE [ OR REPLACE ] PROCEDURE
PROCEDURE_NAME
[ (ARGUMENT) [ {IN | OUT | IN OUT} ] TYPE,
ARGUMENT [ {IN | OUT | IN OUT} ] TYPE) ] {IS |
AS}
PROCEDURE_BODY
```

مثالي لز يك رويه ذخيره شده ساده در زير نشان داده شده است:

تـــاي

```
CREATE PROCEDURE NEW_PRODUCT
(PROD_ID IN VARCHAR2, PROD_DESC IN
VARCHAR2, COST IN NUMBER)
AS
BEGIN
INSERT INTO PRODUCTS_TBL
VALUSE (PROD_ID, PROD_DESC, COST);
COMIT;
END;
```

خروجي

Procedure cre

اين رويه براي درج سطرهاي جديد در جدول products_tbl مورد استفاده قرار مي گيرد.

قالب دستور مورد استفاده براي اجراي يك رويه ذخيره شده در ميكروسافت SQL Server به شكل زير است:

```
EXECUTE [ @RETURN_STATUS = ]
PROCEDURE_NAME
@PARAMETER_NAME = ] VALUE | [ [
[ @PARAMETER_NAME = ] @VARIABLE [
OUTPUT ] ]
[ WITH RECOMPIL]
```

قالب دستور در اراكل به شكل زير است:

```
EXECUTE [ @RETURN STATUS =]
PROCEDURE NAME
[[ @PARAMETER NAME = ] VALUE | [
@PARAMETER NAME = ] @VARIABLE [ OUTPUT ] ]
]
[ WITH REOMPILE ]
```

با اجراي دستور EXECUTE NEW_PRODUCT (‘9999’,’INDIAN CORN’,1.99) ; خروجي بصورت ذيل خواهد بود:

PL/SQL procedure successfully completed.

مزاياي رويه هاي ذخيره شده: رويه هاي ذخيره شده نسبت به اجراي مجزاي دستورات SQL چندين مزيت دارند، برخي از اين مزايا در ذيل آورده شده است:

- دستورات هميشه در بانك اطلاعاتي موجود هستند.
- دستورات از پيش آماده شده و قابل اجرا هستند.

- رويه هاي ذخيره شده از برنامه سازي پيمانه اي پشتيباني مي كنند.
- رويه هاي ذخيره شده مي توانند رويه ها و توابع ديگري را فرابخوانند.
- زمان پاسخگويي رويه هاي ذخيره شده بهتر است.
- كتربرد آنها بهتر است.

Trigger ها

هر trigger نوعي رويه ذخيره شده است كه به هنگام انجام يك عمل مشخص (DML) در يك جدول اجرا مي شود. Trigger ها ممكن است پيش يا پس از هر عمل درج، حذف يا به روزرساني اجرا شوند. Trigger ها ممكن است براي بررسي جامعيت داده ها پيش از هر عمل درج، حذف يا به روزرساني نيز مورد استفاده قرار گيرند. Trrigger ها مي توانند سبب اعمال تراكنشها شوند؛ آنها مي توانند داده ها را در يك جدول تغيير دهند و آنها را از جدولي از يك بانك اطلاعاتي ديگر بخوانند.

Trigger ها معمولاً توابع بسيار خوبي براي استفاده هستند؛ اما مي توانند سبب افزايش I/ / O شوند. در شرايطي كه يك رويه ذخيره شده يا يك برنامه قادر به انجام كارها با I / O كمتر است، نبايد از trigger ها استفاده كرد. قالب دستور ايجاد trigger در ميكروسافت SQL به شكل زير است:

```
CREATE TRIGGER TRIGGER_NAME
ON TABLE_NAME
FOR { INSERT | UPDATE | DELETE [ , .. ]}
AS
SQLدستورات
[ RETURN ]
```

قالب دستور در اراكل به شكل زير است:

```
CREATE [ OR REPLACE ] TRIGGER
TRIGGER_NAME
[ BEFORE | AFTER ]
[ DELETE | INSERT | UPDATE [ OF
COLUMN_NAME ]
ON [ USER.TABLE_NAME ]
[ FOR EACH ROW ]
[ PL/SQL BLOCK ]
```

قالب دستور حذف يك trigger به شكل DROP TRIGGER TRIGGER_NAME مي باشد.

SQL پويا

SQL پويا به يك برنامه ساز يا كاربر امكان مي دهد كه يك دستور را در زمان اجرا ايجاد نمايد و به بانك اطلاعاتي ارسال نمايد. بانك اطلاعاتي سپس داده هايي را برگردانده و در متغييرهاي برنامه ذخيره مي كند.

براي درك SQL پويا، SQL ايستا را بررسي كنيد. هر دستور SQL ايستا ازپيش نوشته شده و براي تغيير آماده نمي شود. اگرچه اين گونه دستورات را مي توان در يك فيل ذخيره نمود و براي اجراي آتي آماده نمود، اما SQL ايستا انعطاف پذيري SQL پويا را ندارد.

مشكل SQL ايستا آن است كه اگر چه پرس و جوهاي زيادي ممكن است در اختيار كاربر باشد، اما باز هم اين احتمال وجود دارد كه اين ‹‹پرس و جوهاي كنسرو شده›› نيازهاي كاربران را در تمام موارد بر طرف نكنند.

SQL پويا اغلب توسط ابزارهاي ويژه مورد استفاده قرار مي گيرد، ابزارهايي كه به كاربران امكان مي دهند كه يم دستور را در كسري از ثانيه براي بر طرف كردن نيازهاي يك پرس و جوي خاص در شرايط مورد نظر ايجاد كنند. پس از تغيير دستور مطابق با نيازهاي شخصي كاربرؤ، دستور به بانك اطلاعاتي ارسال مي شود، اگر امر و حقوق دستيابي مورد نياز براي اجراي آن بررسي مي شود، و در پايان در همان بانك اطلاعاتي، يعني محلي كه توسط سرويس دهنده بانك اطلاعاتي اجرا خواهد شد، كامپايل ميشود. SQL پويا را مي توان با استفاده از رابط سطح فرخواني ايجاد نمود.

رابط سطح فراخواني: برنامه سازان برنامه هاي كاربردي مي بايست با مفهوم رابط سطح فراخواني بسيار آشنا باشند. اين رابط يكي از روشهايي است كه به يك برنامه ساز امكان ميدهد تا دستورات SQL را در زبانهاي برنامه سازي پروسيجرال بگنجاند. زماني كه از CLI استفاده مي كنيد، صرفآ متن يك دستور SQL را با استفاده از قوانين زبان برنامه سازي ميزبان به يك متغيير ارسال مي كنيد. با استفاده از متغييري كه متن به آن ارسال مي شود مي توانيد دستور SQL را در زبان ميزبان اجرا كنيد.

از رابط سطح فراخواني براي گنجاندن دستورات S Q L در يك زبان ميزبان استفاده مي شود؛ مثلآ ANSI C .

EXEC SQL يك دستور متداول در زبانهاي برنامه سازي ميزبان است كه امكان فراخواني يك دستور SQL از برنامه را فراهم مي كند.

EXEC SQL

مثالهايي از زبانهاي برنامه سازيي كه از CLI پشتيباني مي كنند:

- كوبول
- ANSI C
- پاسكال
- فرترن
- Ada

SQL مستقيم در مقابل SQL گنجانده شده : منظور از SQL مستقيم، اجراي دستورات SQL از نوعي ترمينال محاوره اي است. نتائج SQL مستقيماً به ترمينال بازگردانده مي شوند كه دستور از آنجا اجرا شده است.
SQL مستقيم، تحت عنوان درخواست محاوره اي يا درخواست مستقيم نيز شناخته شده است.

منظور از SQL شناخته شده، دستورات SQL ي است كه در برنامه هاي ديگري چون پاسكال، فرترن، كوبول، و C به كار برده مي شوند. دستورات SQL با رابط سطح فراخواني در يك زبان برنامه سازي ميزبان گنجانده مي شوند. دستورات SQL در بسياري از موارد در برنامه هاي زبان ميزبان با SQL EXEC آغاز و به يك ‹‹ ; ›› ختم مي شوند. ساير كاراكترهاي كه مي توان بجاي ‹‹ ; ›› به كار برد عبارتند از : END_EXEC و " (" .

مثالي از SQL گنجانده شده در يك برنامه ميزبان، همچون زبان ANSI C ، در ذيل نشان داده شده است:

}دستورات زبان ميزبان{
SQL} SQL EXECدستور; {
}ساير دستورات زبان ميزبان{

<u>گسترش استفاده از SQL در سازمانها، اينترنت و اينترانت</u>

در اين قسمت خواهيد ديد كه SQL واقعاً چگونه در سازمانها و اينترنت يك شركت مورد استفاده قرار گرفته و كاربرد آن چگونه تا به اينترنت گسترش يافته است.

SQL و كارهاي سازماني : بسياري از سازمانهاي بازرگاني داده هايي ويژه اي دارند كه در اختيار سازمانهاي ديگر، مشتريان و فروشندگان قرار مي دهند. به عنوان مثال، يك سازمان ممكن است اطلاعات مفصلي درباره محصولات خود داشته باشد وآنها را به اميد خريد بيشتر در اختيار مشتريان قرار دهد. نيازهاي كاركنان سازمان نيز در اين اطلاعات موجود هستند. داده هاي خاصي چون برگه هاي خاصي گزارش كاري، زمانبندي مرخصي ها، زمانبندي آموزشها، سياستهاي شركت، ئ غيره نيز نگهداري مي شوند. مي توان يك بانك اطلاعاتي ايجاد نمود، و دستيابي آسان به داده هاي مهم را از طريق S QL و زبان اينترنت براي مشتريانو كارمندان فراهم نمود.

برنامه هاي كاربردي back-end : قلب هر كاري در برنامه كاربردي back_end است. اين همان جايي است كه كاها در پشت پرده رخ مي دهند و در معرض ديد كاربر بانك اطلاعاتي نيستند. برنامه كاربردي back_end از سرويس دهنده بانك اطلاعاتي، منابع داده اي و يك رابط مناسب براي مرتبط كردن يك برنامه كاربردي به وب يا يك بانك اطلاعاتي راه دور در شبكه محلي تشكيل مي شود. بد نيست بدانيد كه برخي از سرويس دهنده هاي بانك اطلاعاتي عبارتند از: اراكل، اينفورميكس، Sybase ، ميكروسافت Server SQL و

بورلند InterBase . اين نخستين گام براي انتقال هر برنامه كاربردي به يك سازمان، از طريق يك شبكه محلي (LAN)، به اينترنت خاص يا سازمان و يا به اينترنت است. سرويس دهنده بانك اطلاعاتي مي بايست توسط مدير بانك اطلاعاتي كه نيازهاي شركت و برنامة كاربردي را درك مي كند، تعيين شود.

رابط برنامه كاربردي شامل يك سرويس دهنده وب و ابزاري با قابليت برقراري ارتباط بين سرويس دهنده وب و سرويس دهنده بانك اطلاعاتي است. هدف اصلي داشتن يك برنامه كاربردي در وب است كه بتواند با يك بانك اطلاعاتي مجتمع كار كند.

برنامه كاربردي front-end : اين برنامه كاربردي، بخشي از يك برنامه كاربردي است كه كاربر با آن ارتباط برقرار مي كند. اين برنامه يا يك محصول نرم افزاري متفرقه است كه توسط شركت خريداري مي شود، ويا يك برنامه كاربردي است كه توسط خود اشخاص با ابزارهاي متفرقه آماده مي شود. ابزارهاي متفرقه در پاراگرافهاي ذيل تشريح شده اند.

پيش از ارائه ابزارهاي front-end جديد امروزي، كاربران مي بايست با شيوه برنامه سازي با زبانهايي چون C++ ، HTML يا يكي از برنامه هاي پروسيجرال مورد استفاده در برنامه هاي كاربردي مبتني بر وب، آشنا مي شدند. زبانهاي ديگر، همچون ANSI C، كوبول، فرترن و پاسكال، براي توليد برنامه هاي كاربردي front-end، كه پيشتر صرفآ مبتني بر متن بودند، مورد استفاده قرار مي گيرند. امروزه بيشتر برنامه هاي front-end جديد، GUI هستند، يعني يك رابط گرافيكي دارند. ابزارهاي امروزي به واسطه داشتن نمادهاي گرافيكي، ويزاردها، و تكنيك كشيدن و رها كردن با ماوس، كاربر پسند و شيءگرا هستند. برخي از ابزارهاي مشهور انتقال برنامه هاي كاربردي به وب عبارتند از C++ Builder و Intra Builder (محصول بورلند) و ويژوال J++ و C++ (محصول ميكروسافت). ساير برنامه هاي كاربردي مشهوري كه براي توليد برنامه هاي كاربردي مجتمع در شبكه هاي محلي به كار برده مي شوند عباتند از Power Builder (محصول Powersoft)، Developer/2000 (محصول Oracle Corporation)، ويژوال بيسيك (محصول ميكروسافت) و دلفي (محصول بورلند). برنامه كاربردي front-end ، سادگي را براي بانك اطلاعاتي و كاربران به همراه دارد. بانك اطلاعاتي مربوطه، دستورات، و رويدادهايي كه در بانك اطلاعاتي روي مي دهند از ديد كاربر پنهان هستند. برنامه كاربردي front-end براي آن طراحي مي شود تا كاربر از حدس و سردرگمي راحت شود. تكنولوژيهاي جديد اين امكان را فراهم آورده اند كه برنامه هاي كاربردي هوشمندتر باشند، و در نتيجه به كاربران امكان مي دهند كه بيشتر بر جنبه هاي حقيقي كارهاي خاص خود تمركز كنند، و از اين رو بهرهوري كلي افزايش يابد.

دستيابي به يك بانك اطلاعاتي راه دور: گاهي اوقات بانك اطلاعاتي كه به آن دستيابي داريد يك بانك اطلاعاتي محلي است، يعني بانك اطلاعاتي كه

مستقيماً به آن متصل مي شويد. به طور كلي، احتمالاً به نوعي بانك اطلاعاتي راه دور دسترسي خواهيد داشت. يك بانك اطلاعاتي راه دور، بانك اطلاعاتي محلي است، بدين معنا كه براي برقراري ارتباط با آن مي بايست از يك شبكه و نوعي پروتكل شبكه استفاده شود. بطور كلي يك بانك اطلاعاتي راه دور،بانكي است كه بر روي سرويس دهندهاي به غير از سرويس دهنده اي كه به آن متصل هستيد قرار گرفته است.

چندين روش براي دستيابي به بانك اطلاعاتي راه دور وجود دارد. اگر به طور كلي به مسآله بنگريم، هر بانك اطلاعاتي راه دور از طريق شبكه يا اتصال اينترنت و با استفاده از يك رابط (ODBC، يك رابط استاندارد) مورد دستيابي قرار مي گيرد.

ODBC : ODBC (Open Data Base Cconnectivity)

برقراري ارتباط با بانك هاي اطلاعاتي راه دور را از طريق يك نرم افزار راه اندازي فراهم مي سازد. ممكن است براي برقراري ارتباط با يك بانك اطلاعاتي راه دور، يك نرم افزار راه اندازي شبكه نيز لازم باشد. يك برنامه كاربردي توابع ODBC را فرا مي خواند، و يك مدير نيز نرم افزار راه اندازي ODBC را بارگذاري مي كند. نرم افزار راه اندازي ODBC پردازش فراخواني را بر عهده مي گيرد، درخواست S QL را ارائه مي دهد، ونتائج را از بانك اطلاعاتي بر مي گرداند. O DBC اينك يك استاندارد به شمار آمده و توسط محصولاتي چون Power Builder، فاكس پرو، ويژوال C++ ، ويژوال بيسيك، دلفي بورلند، ميكروسافت Access و محصولات ديگر مورد استفاده قرار مي گيرد.

هر نرم افزار راه اندازي O DBC براي برقراري ارتباط با يك بانك اطلاعاتي back-end توسط يك برنامه كاربردي front-end مورد استفاده قرار مي گيرد.

تمام فروشندگان RDBMS يك API (Application Programmatic Interface) به عنوان بخشي از ODBC به همراه بانك اطلاعاتي خود ارائه مي دهند. OCI (Open Call Interface) اراكل و SQL Gateway و SQL Router شركت Centura چند نمونه از اين محصولات هستند.

دستيابي به يك بانك اطلاعاتي راه دور از طريق وب : دسيابي به يك بانك اطلاعاتي راه دور از طريق يك رابط وب شباهت بسيار زيادي به انجام اين كار از طريق يك شبكه محلي دارد. تفاوت اصلي آنست كه تمام درخواستهاي كاربر از بانك اطلاعاتي از طريق سرويس دهنده وب هدايت مي شوند.

كاربر با فعال سازي يك مرورگر وب، از طريق يك رابط وب با يك بانك اطلاعاتي راه دور ارتباط برقرار مي كند. مرورگر وب براي برقراري ارتباط با يك URL خاص يا نشاني IP اينترنت، كه توسط محل سرويس دهنده

وب تعيين مي شود، مورد استفاده قرار مي گيرد. سرويس دهنده وب مجوز دستيابي كاربر را بررسي و درخواست وي (مثلآ يك پرس وجو) را به بانك اطلاعاتي راه دور ارسال مي كند، كه آن نيز ممكن است مجوز دستيابي كاربر را بررسي كند. سرويس دهنده بانك اطلاعاتي سپس نتائج را به سرويس دهنده وب باز مي گرداند، و سرويس دهنده وب نيز نتائج را بر روي مرورگر وب نمايش مي دهد. دستيابي غير مجاز به يك سرويس دهنده خاص را مي توان با استفاده از يك fireball كنترل نمود. Fireball يك مكانيزم امنيتي براي پيشگيري ارتباطهاي غير مجاز با يك سرويس دهنده است. با استفاده از يك يا چند مورد از آنها مي توان دستيابي به يك بانك اطلاعاتي يا يك سرويس دهنده را كنترل نمود.

بايد توجه داشت كه چه اطلاعاتي را در وب قابل دسترسي مي كنيد. هميشه دقت كنيد كه تمام جوانب لازم براي پياده سازي درست امنيت در تمام سطوح در نظر گرفته شوند؛ اين امر ممكن است شامل سرويس دهنده وب، سرويس دهنده ميزبان و بانك اطلاعاتي راه دور باشد. از سرقت داده ها، مثلآ شماره تآمين اجتماعي اشخاص، بايد هميشه پيشگيري نمود و داده ها را در تمام وب منتشر نكرد.

SQL و اينترنت : SQL را مي توان با زبانهاي برنامه سازيي چون C يا كوبول به كار برد يا در برنامه هاي اين زبانها گنجاند. به عنوان مثال، دستورات S QL را مي توان در برنامه هاي زبانهاي برنامه سازي اينترنت، همچون Java نيز گنجاند. براي ارسال يك پرس و جو از يك برنامه front-end وب به يك بانك اطلاعاتي راه دور مي توان متون HTML، يك زبان اينترنت ديگر، را به SQL ترجمه نمود. پس از اجراي پرس و جو توسط بانك اطلاعاتي، خروجي به HTML ترجمه شده و مرورگر وب ماشين اجرا كنندة پرس و جو نمايش داده مي شود. كاربرد SQL در اينترنت در قسمتهاي ذيل بررسي شده است.

كاربرد اطلاعات به مشتريان در سرتاسر جهان : با ظهور اينترنت در سرتاسر جهان در دسترس مشتريان و فروشندگان قرا گرفتند. عموماً از طريق يك ابزار front-end به طور «فقط خواندني» در دسترس قرار مي گيرند.

داده هايي كه در دسترس مشتريان قرار مي گيرند ممكن است درباره اطلاعات عمومي مشتريان، اطلاعات محصولات، اطلاعات صورتحسابها، سفارشهاي جاري، سفارشهاي پيشين، وساير اطلاعات مرتبط با مشتريان باشند. اطلاعات خصوصي، همچون استراتژيهاي يك شركت و اطلاعات كارمندان، نبايد در دسترس باشند.

صفحات خانگي در اينترنت براي شركتهايي كه ميخواهند رقابت خود را حفظ كنند تقريباً يك ضرورت است. هر صفخه وب يك ابزار بسيار قدرتمند است كه مي تواند اطلاعات زيادي درباره يك شركت در اختيار مشتريان

بگذارد، خدمات، محصولات و ساير اطلاعات شركت، صفحه هاي وب اين كار را با بهاي كمي انجام مي دهند.

ارائه اطلاعات به كارمندان و مشتريان : يك بانك اطلاعاتي را مي توان از طريق اينترنت يا اينترانت يك شركت در اختيار كارمندان يا مشتريان آن قرار داد. استفاده از تكنولوژيهاي اينترنت براي آگاه ساختن كارمندان نسبت به سايتها، سود، آموزش و غيره شركت، دارايي ارتباطي ارزشمند به شمار مي آيد.

ابزارهاي front-end وب با استفاده از SQL : با ابزارهاي زيادي مي توان دستيابي به بانك هاي اطلاعاتي را فراهم نمود. بسياري از آنها يك رابط گرافيكي دارند و نيازي نيست كه كاربران SQL را براي پرس و جو از يك بانك اطلاعاتي بدانند. اين ابزارها به كاربران امكان مي دهند كه براي انتخاب شيء هاي نمايانگر جداول، پردازش داده ها در شيء، مشخص كردن معيارهاي انتخاب داده ها و غيره، از اشاره به وسيله ماوس و فشردن دكمه آن استفاده كنند. اين ابزارها اغلب با توجه به نيازهاي يك شركت آماده مي شوند.

SQL و اينترانت : IBM در آغاز SQL را براي استفاده در بين بانكهاي اطلاعاتي موجود در كامپوترهاي بزرگ و در بين كاربران ماشينهاي سرويس گيرنده پياده سازي نمود. كاربران از طريق شبكه محلي به كامپيوترهاي بزرگ متصل مي شوند. SQL به عنوان زبان استاندارد برقراري ارتباط بين بانكهاي اطلاعاتي و كاربران پذيرفته شد. هر اينترانت اساساً يك اينترنت كوچك است. تنها تفاوت اصلي آنست كه هر اينترانت تنها براي استفاده يك سازمان است، در صورتي كه اينترنت در اختيار همگان است. رابط كاربر(سرويس گيرنده) در يك اينترانت همچون يم محيط سرويس گيرنده/سرويس دهنده است. درخواستهاي S QL پيش از هدايت به بانك اطلاعاتي (براي اجرا)، از طريق سرويس دهنده وب و زبانها (همچون HTML) هدايت مي شوند. بايد توجه داشت كه امنيت بانك اطلاعاتي بسيار پايدارتر از امنيت در اينترنت است. هميشه از ويژگيهاي امنيتي كه از طريق سرويس دهنده بانك اطلاعاتي در اختيارتان گذارده مي شوند استفاده كنيد.