**Základy SQL (Structured Query Language)**

MS Access je najvhodnejším jednoduchým prostredím pre aplikáciu základných prvkov programovacieho jazyka SQL.

Vytvorte jednoduchý dotaz z tabuliek ako na obrázku1.



Zmeňte zobrazenie na SQL zobrazenie. Access používa pri tvorbe dotazov SQL jazyk bez toho, aby bolo potrebné niečo špeciálne programovať.



***Príkaz SELECT***

Príkaz SELECT, je základným prvkom SQL. Umožňuje získať záznamy z množiny údajov. Základná syntax príkazu SELECT je:

 SELECT column\_name(s)

 FROM table\_name

Príkaz SELECT sa vždy používa s klauzulou FROM. Klauzula FROM identifikuje tabuľku(y), ktoré tvoria zdroj údajov.

Vytvorte nový dotaz v návrhovom zobrazení. Zatvorte dialógové okno pre pridanie tabuliek (ak je otvorené). Zmeňte zobrazenie na SQL zobrazenie. V tomto zobrazení napíšte príkazy:

SELECT FullName

 FROM Dim\_AccountManagers;

 ***Výber konkrétnych stĺpcov***

Konkrétne stĺpce môžeme získať z množiny údajov explicitným definovaním stĺpcov v príkaze SELECT takto:

 

Stĺpce v databáze s názvom, ktorý obsahuje medzery alebo špecifické znaky musí byť uzatvorený v hranatých zátvorkách – [].

 ***Výber všetkých stĺpcov***

Použitie \* umožňuje vybrať všetky stĺpce naraz.

 SELECT \* FROM Dim\_AccountManagers

***Príkaz WHERE***

Príkaz WHERE sa používa v spojitosti s príkazom SELECT pre filtrovanie údajov a výber údajov so špecifickými podmienkami. Príkaz WHERE je vždy spojený s operátormi ako =, <>, >, <, >=, <= a BETWEEN.

Vytvorte dotaz pomocou nasledovných príkazov SQL:



Dotaz spustite a potom zmeňte posledný riadok príkazu nasledovne:



Pri odkazovaní na textové hodnoty v kritériách sa používajú úvodzovky. Pri odkazovaní na dátumy sa používa mriežka.

***Význam spojení***

Často je nutné vytvoriť dotazy, ktoré vyžadujú spojenie dvoch alebo viacerých súvisiacich tabuliek, aby boli dosiahnuté požadované výsledky. Typ použitého spojenia určí záznamy, ktoré budú výstupom.

 ***Vnútorné spojenia***

Vytvorenie vnútorného spojenia povie Accessu, aby vybral iba tie záznamy z oboch tabuliek, ktoré majú zhodné hodnoty. Záznamy s hodnotami v spojenom poli, ktoré sa nezobrazujú v oboch tabuľkách, sú z výsledkov dotazu vynechané. Nasledovné SQL príkazy vyberú iba tie záznamy, v ktorých čísla zamestnancov v poli AccountManagerID sú v oboch tabuľkách – Dim\_AccountManagers a Dim\_Territory.

 SELECT Region, Market,

 Dim\_AccountManagers.AccountManagerID, FullName

 FROM Dim\_AccountManagers INNER JOIN Dim\_Territory ON

 Dim\_AccountManagers.AccountManagerID = Dim\_Territory.AccountManagerID

 ***Vonkajšie spojenia***

Vytvorenie vonkajšieho spojenia povie Accessu, aby vybral všetky záznamy z jednej tabuľky a iba záznamy z druhej tabuľky so zodpovedajúcimi hodnotami v spojenom poli. Existujú dva typy vonkajších spojení: ľavé spojenia a pravé spojenia.

Operácia ľavého spojenia (niekedy nazývaná vonkajšie ľavé spojenie) informuje Access, aby vybral všetky záznamy z prvej tabuľky bez ohľadu na zhodu a iba tie záznamy z druhej tabuľky, ktoré majú v spojenom poli zhodné hodnoty.

Nasledovné SQL príkazy vyberú všetky záznamy z tabuľky Dim\_AccountManagers a iba tie záznamy z tabuľky Dim\_Territory kde hodnoty pre pole AccountManagerID existujú v tabuľke Dim\_AccountManagers:

 SELECT Region, Market,

 Dim\_AccountManagers.AccountManagerID, FullName

 FROM Dim\_AccountManagers LEFT JOIN Dim\_Territory ON

 Dim\_AccountManagers.AccountManagerID = Dim\_Territory.AccountManagerID

Operácia pravého spojenia (niekedy nazývaná vonkajšie pravé spojenie) povie Accessu, aby vybral všetky záznamy z druhej tabuľky bez ohľadu na zhodu a iba tie záznamy z prvej tabuľky, ktoré majú v spojenom poli zhodné hodnoty.

Nasledovné SQL príkazy vyberú všetky záznamy z tabuľky Dim\_Territory a iba tie záznamy v tabuľke Dim\_AccountManagers kde hodnoty pre pole AccountManagerID existujú v tabuľke Dim\_Territory:

 SELECT Region, Market,

 Dim\_AccountManagers.AccountManagerID, FullName

 FROM Dim\_AccountManagers RIGHT JOIN Dim\_Territory ON

 Dim\_AccountManagers.AccountManagerID = Dim\_Territory.AccountManagerID

***Rozšírenie vyhľadávania pomocou operátora Like***

Výsledok filtrovania pomocou operátora Like nie je rozdielny od znamienka =. Vyskúšajte nasledovné výrazy:



Operátor Like sa zvyčajne používa so zástupnými znakmi na rozšírenie rozsahu nášho vyhľadávania tak, aby zahŕňal akýkoľvek záznam, ktorý sa zhoduje so vzorom. Zástupné znaky, ktoré sú platné v Accesse, sú nasledovné:

 \* - predstavuje ľubovoľný počet a typ znakov

 ? – predstavuje akýkoľvek jeden znak

 # - predstavuje akékoľvek číslo

[] - Zátvorky umožňujú odovzdať jeden znak alebo pole znakov operátorovi Like. Všetky hodnoty zodpovedajúce hodnotám znakov v zátvorkách budú zahrnuté do výsledkov.

[!] - Zátvorky s vloženým výkričníkom vám umožňujú odovzdať jeden znak alebo pole znakov operátorovi Like. Všetky hodnoty zodpovedajúce hodnotám znakov po výkričníku budú z výsledkov vylúčené.

|  |  |  |
| --- | --- | --- |
| **Zástupný znak** | **SQL výraz** | **Výsledok** |
| \* | SELECT Field1FROM Table1WHERE Field1 Like "A\*" | Vyber všetky záznamy kde Field1 začína písmenom A |
| \* | SELECT Field1FROM Table1WHERE Field1 Like "A\*" | Vyber všetky záznamy kde Field1 zahŕňa písmeno A |
| ? | SELECT Field1FROM Table1WHERE Field1 Like "???" | Vyber všetky záznamy kde dĺžka Field1 je tri znaky |
| ? | SELECT Field1FROM Table1WHERE Field1 Like "B??" | Vyber všetky záznamy kde Field1 je trojznakový reťazec začínajúcu písmenom B |
| # | SELECT Field1FROM Table1WHERE Field1 Like "###" | Vyber všetky záznamy kde Field1 je číslo, ktoré má dĺžku troch číslic |
| # | SELECT Field1FROM Table1WHERE Field1 Like "A#A" | Vyber všetky záznamy kde hodnota vo Field1 je trojznaková hodnota, ktorá začína písmenom A, obsahuje jednu číslicu a končí písmenom A |
| #\* | SELECT Field1FROM Table1WHERE Field1 Like "A#\*" | Vyber všetky záznamy kde Field1 začína písmenom A a akýmkoľvek číslom |
| []\* | SELECT Field1FROM Table1WHERE Field1 Like "\*[$%!\*/]\*" | Vyber všetky záznamy kde Field1 zahŕňa niektorý zo špeciálnych znakov  |
| [!]\* | SELECT Field1FROM Table1WHERE Field1 Like "\*[!a-z]\*" | Vyber všetky záznamy kde hodnota vo Field1 nie je textová hodnota ale číselná hodnota alebo špeciálny znak ako @ |
| [!]\* | SELECT Field1FROM Table1WHERE Field1 Like "\*[!0-9]\*" | Vyber všetky záznamy kde hodnota vo Field1 nie je číselná hodnota ale textová hodnota alebo špeciálny znak ako @ |

***Výber jedinečných hodnôt a riadkov bez zoskupovania***

Príkaz DISTINCT umožňuje získať iba jedinečné hodnoty z vybratých polí v množine údajov. Nasledujúci príklad SQL príkazov vyberie iba názvy trhov z tabuľky PvTblFeed. Výsledkom bude 14 záznamov:

 SELECT DISTINCT Market

 FROM PvTblFeed

Ak príkaz SQL vyberie viac ako jedno pole, kombinácia hodnôt zo všetkých polí musí byť jedinečná, aby sa daný záznam zahrnul do výsledkov. Ak požadujeme aby bol celý riadok unikátny, je možné použiť príkaz DISTINCTROW. Príkaz DISTINCTROW umožňuje získať len tie záznamy, pre ktoré je jedinečný celý riadok. To znamená, že kombinácia všetkých hodnôt vo vybratých poliach sa nezhoduje so žiadnym iným záznamom vo vrátenom súbore údajov. Príkaz DISTINCTROW môžeme použiť rovnako ako v klauzule SELECT DISTINCT:

 SELECT DISTINCTROW AccountManagerID

 FROM Dim\_AccountManagers

***Zoskupovanie a agregovanie pomocou klauzuly GROUP BY***

Klauzula GROUP BY umožňuje agregovať záznamy v množine údajov podľa hodnôt stĺpcov. Keď vytvárame súhrnný dotaz v návrhovom zobrazení, v podstate používame klauzulu GROUP BY. Nasledujúci príkaz SQL zoskupí pole Market a poskytne nám počet štátov na každom trhu:

 SELECT Market, Count (State)

 FROM Dim\_Territory

 GROUP BY Market

Keď používame klauzulu GROUP BY, pred agregáciou sa vyhodnotí každá klauzula WHERE zahrnutá v dotaze. Môžeme však mať scenáre, keď po použití zoskupenia potrebujeme použiť podmienku WHERE. V týchto prípadoch môžeme použiť klauzulu HAVING. Napríklad tento príkaz SQL zoskupí záznamy, ktorých hodnota v poli Market je Dallas, a potom vráti len tie záznamy zákazníkov, ktorých súčet výnosov je menší ako 100. Zoskupenie sa vykonáva pred kontrolou, či je súčet výnosov menší ako 100:

 SELECT Customer\_Name, Sum(Revenue) AS Sales

 FROM PvTblFeed

 Where Market = "Dallas"

 GROUP BY Customer\_Name

 HAVING (Sum(Revenue)<100)

***Nastavenie zoraďovania pomocou príkazu ORDER BY***

Príkaz ORDER BY umožňuje zoraďovať údaje pomocou špecifického poľa. Predvolené zoraďovanie je nastavené vzostupne, nie je potrebné pridávať na to špecifické inštrukcie. Nasledujúce SQL príkazy zoradia záznamy podľa poľa Last Name vzostupne a potom podľa poľa First Name tiež vzostupne:

 SELECT AccountManagerID, [Last Name], [First Name]

 FROM Dim\_AccountManagers

 ORDER BY [Last Name], [First Name]

Pre zoraďovanie v zostupnom poradí, je potrebné použiť príkaz DESC za každým poľom, ktoré chceme zoradiť v zostupnom poradí. Nasledovná skupina príkazov zoradí záznamy podľa poľa Last Name zostupne a potom podľa First Name vzostupne:

SELECT AccountManagerID, [Last Name], [First Name]

 FROM Dim\_AccountManagers

 ORDER BY [Last Name] DESC, [First Name]

***Vytvorenie aliasov pomocou klauzuly AS***

Klauzula AS nám umožňuje priradiť aliasy k stĺpcom a tabuľkám. Vo všeobecnosti existujú dva dôvody, prečo by sme chceli používať aliasy: buď chceme skrátiť názvy stĺpcov alebo tabuliek a skrátiť ich na čítanie, alebo pracujeme s viacerými inštanciami tej istej tabuľky a potrebujeme spôsob, ako odkazovať na jednu inštanciu.

*Vytvorenie aliasu pre stĺpec*

Nasledovný SQL príkaz zoskupí pole Market a dá informáciu o počte štátov na každom trhu. Okrem toho bol do stĺpca obsahujúceho State Count priradený alias počet štátov zahrnutím klauzuly AS:

 SELECT Market, Count(State) AS [State Count]

 FROM Dim\_Territory

GROUP BY Market

HAVING Market = "Dallas"

*Vytvorenie aliasu pre tabuľku*

Tento SQL príkaz dá tabuľke Dim\_AccountManagers alias MyTable:

 SELECT AccountManagerID, [Last Name], [First Name]

 FROM Dim\_AccountManagers AS MyTable

***Zobrazenie hodnôt SELECT TOP alebo SELECT TOP PERCENT***

Pri spustení dotazu pomocou príkazu SELECT, získame všetky záznamy, ktoré spĺňajú dané definície a kritériá. Keď spustíme príkaz SELECT TOP alebo podmienku pre zobrazenie najvyšších hodnôt, hovoríme Accessu, aby vrátil množinu údajov zobrazujúcu iba určitý počet záznamov.

*Vysvetlenie top hodnôt pomocou dotazu*

Ak chceme jasne pochopiť, čo robí príkaz SELECT TOP, vytvoríme najskôr súhrnný dotaz. Vytvorte dotaz podľa nasledujúceho obrázku:



Dotaz je súhrnný a pre pole LineTotal je nastavená suma a zoradenie je nastavené zostupne. Aktivujte hárok vlastností. V riadku Najvyššie hodnoty nastavte hodnotu na 25. Po spustení dotazu sa zobrazia iba hodnoty, ktoré patria do zoznamu 25 najvyšších hodnôt podľa príjmov. Ak chceme zobraziť spodných 25 hodnôt, je potrebné zmeniť zoradenie na vzostupné. Ak sú niektoré hodnoty presne zhodné, Access zobrazí viac hodnôt ako 25.

*Príkaz SELECT TOP*

Toto je SQL zobrazenie predošlého príkazu:

 SELECT TOP 25 Customer\_Name, Sum(LineTotal) AS SumOfLineTotal

 FROM Dim\_Customers INNER JOIN Dim\_Transactions ON

 Dim\_Customers.CustomerID = DimTransactions.CustomerID

 GROUP BY Customer\_Name

ORDER BY Sum(LineTotal) DESC

Majme na pamäti, že na použitie dotazu s najvyššími hodnotami nemusíme pracovať so súčtom. V nasledujúcom príkaze SQL získame ako výsledok desať účtovných manažérov, ktorí majú v spoločnosti najskorší dátum prijatia, čím efektívne vytvárame správu o seniorskom postavení:

 SELECT Top 10 AccountManagerID, [Last Name], [First Name]

 FROM Dim\_AccountManagers

 ORDER BY HireDate ASC

Pri používaní príkazu SELECT TOP je dôležité špecifikovať spôsob zoradenia – DESC, ASC.

*Príkaz SELECT TOP PERCENT*

Príkaz SELECT TOP PERCENT funguje presne rovnakým spôsobom ako SELECT TOP okrem toho, že záznamy vrátené v príkaze SELECT TOP PERCENT predstavujú n-té percento celkových záznamov a nie n-tý počet záznamov. Napríklad nasledujúci príkaz SQL vráti prvých 25 percent záznamov podľa výnosov:

SELECT TOP 25 PERCENT Customer\_Name, Sum(LineTotal) AS SumOfLineTotal

FROM Dim\_Customers INNER JOIN Dim\_Transactions ON

Dim\_Customers.CustomerID = Dim\_Transactions.CustomerID

GROUP BY Customer\_Name

ORDER BY Sum(LineTotal) DESC

***Spúšťanie akčných dotazov pomocou SQL***

Keď vytvárame akčný dotaz, vytvárame príkaz SQL, ktorý je špecifický pre danú akciu. Tieto príkazy SQL nám umožňujú ísť nad rámec jednoduchého výberu záznamov.

*Dotaz na vytvorenie tabuľky*

Vytváracie dotazy používajú príkaz SELECT...INTO na vytvorenie pevne zakódovanej tabuľky, ktorá obsahuje výsledky dotazu. Nasledujúci príklad najprv vyberie číslo manažéra, priezvisko a meno; a potom vytvorí novú tabuľku s názvom Employees:

 SELECT AccountManagerID, [Last Name], [First Name] INTO Employees

 FROM Dim\_AccountManagers

*Dotaz pre pridanie údajov*

Pridávacie dotazy používajú príkazy INSERT INTO pre vloženie nových riadkov v špecifickej tabuľke. Nasledovný príklad vloží nové riadky do tabuľky Employees z tabuľky Dim\_AccountManagers:

 INSERT INTO Employees (AccountManagerID, [Last Name], [First Name])

 SELECT AccountManagerID, [Last Name], [First Name]

 FROM Dim\_AccountManagers

*Aktualizačný dotaz*

Aktualizačné dotazy používajú príkaz UPDATE v spojení s príkazom SET pre úpravu údajov v tabuľkách. Nasledovný príkaz aktualizuje pole List\_Price v tabuľke Dim\_Products – zvýši sa cena o 10%:

 UPDATE Dim\_Products SET List\_Price = [List\_Price]\*1.1

*Vymazávací dotaz*

Tento typ dotazov používa príkaz DELETE pre vymazávanie riadkov v tabuľkách. V nasledujúcom príklade vymažeme všetky riadky z tabuľky Employees:

 DELETE \* FROM Emplyees

***Vytvorenie krížového dotazu pomocou príkazu TRANSFORM***

Príkaz TRANSFORM umožňuje vytvorenie množiny údajov krížového dotazu, ktorý zobrazuje údaje v kompaktnom zobrazení. Príkaz TRANSFORM vyžaduje na fungovanie tri hlavné komponenty:

* Pole, ktoré bude tvoriť súhrn.
* Príkaz SELECT, ktorý bude tvoriť obsah riadkov pre krížový dotaz.
* Pole, ktoré bude tvoriť stĺpec krížového dotazu.

Syntax je nasledovný:

 TRANSFORM Aggregated\_Field

 SELECT Field1, Field2

 FROM Table1

 GROUP BY Select Field1, Field2

 PIVOT Pivot\_Field

Nasledovné príkazy vytvoria krížový dotaz, ktorý zobrazí polia Region a Market v riadkoch a produkty v stĺpcoch, so ziskami v hodnotách:

 TRANSFORM Sum(Revenue) AS SumOfRevenue

 SELECT Region, Market

 FROM PvTblFeed

 GROUP BY Region, Market

 PIVOT Product\_Description

***Použitie špecifických SQL dotazov***

Špecifické SQL dotazy sú v podstate akčné dotazy, ktoré nemožno spustiť cez návrhové zobrazenie dotazov programu Access. Tieto dotazy musia byť spustené buď v zobrazení SQL alebo prostredníctvom kódu (makro alebo VBA). Existuje niekoľko typov dotazov špecifických pre SQL, z ktorých každý vykonáva špecifickú akciu.

**Zlučovanie údajov pomocou príkazu UNION**

Operátor UNION sa používa na zlúčenie dvoch kompatibilných príkazov SQL na vytvorenie jednej množiny údajov iba na čítanie. Nasledujúci príklad pomocou SELECT zobrazí zisky za Region a Market:

 SELECT Region, Market, Sum(Revenue) AS [Sales]

 FROM PvTblFeed

 GROUP BY Region, Market

Druhý príkaz SELECT vytvorí ďalší set údajov, ktorý ukazuje celkové zisky za Region:

 SELECT Region, "Total " AS [Market], Sum(Revenue) AS [Sales]

 FROM PvTblFeed

 GROUP BY Region

Myšlienkou je spojiť tieto dva údajové sety pre vytvorenie analýzy, ktorá zobrazí podrobnosti a celkové hodnoty spolu v jednej tabuľke. Toto je možné docieliť pomocou príkazu UNION. Pre použitie vytvoríme nový dotaz nasledovne:

SELECT Region, Market, Sum(Revenue) AS [Sales]

 FROM PvTblFeed

 GROUP BY Region, Market

 UNION

 SELECT Region, "Total " AS [Market], Sum(Revenue) AS [Sales]

 FROM PvTblFeed

 GROUP BY Region

Keď sa spustí zjednocovací dotaz, Access priradí stĺpce z oboch množín údajov podľa ich pozície v príkaze SELECT. To znamená dve veci: vaše príkazy SELECT musia mať rovnaký počet stĺpcov a poradie stĺpcov v oboch príkazoch SELECT je dôležité. Access sa nepokúsi o zhodu s názvami stĺpcov. V skutočnosti sa názvy stĺpcov v dvoch množinách údajov nemusia zhodovať. Pokiaľ majú stĺpce zhodné typy údajov, Access vytvorí spojenie dvoch tabuliek pomocou pozície každého stĺpca. Názvy stĺpcov z prvého príkazu SELECT sa zobrazia vo výslednej množine údajov.

Všimnite si, že operátor UNION efektívne vykonáva SELECT DISTINCT na výsledných súboroch údajov. To znamená, že príkaz UNION by mohol veľmi dobre eliminovať duplicitné riadky, v ktorých sú všetky hodnoty v každom poli identické medzi dvoma množinami údajov. Ak pri spustení dotazu UNION zistíte, že vám chýbajú záznamy, zvážte použitie operátora UNION ALL. UNION ALL vykonáva rovnakú funkciu ako UNION, okrem toho, že nepoužíva SELECT DISTINCT, a teda neodstraňuje duplicitné riadky.

**Vytvorenie tabuľky pomocou príkazu CREATE TABLE**

Vo svojich analytických procesoch budete často musieť vytvoriť dočasnú tabuľku, aby ste mohli zoskupovať, manipulovať alebo jednoducho uchovávať údaje. Príkaz CREATE TABLE vám to umožňuje urobiť pomocou jedného dotazu špecifického pre SQL.

Na rozdiel od vytváracieho dotazu je príkaz CREATE TABLE navrhnutý tak, aby vytvoril iba štruktúru alebo schému tabuľky. S príkazom CREATE TABLE sa nikdy nevracajú žiadne záznamy. Tento príkaz nám umožňuje strategicky vytvoriť prázdnu tabuľku v ktoromkoľvek bode nášho analytického procesu.

Základná syntax je nasledovná:

 CREATE TABLE TableName

 (Field1Name> Type(<Field Size>), <Field2Name> Type(<Field Size>))

Pre použitie príkazu CREATE TABLE jednoducho stačí spustiť nový dotaz v SQL zobrazení a definovať štruktúru tabuľky. Vytvorte tabuľku s názvom TempLog s tromi poľami. Prvé pole bude textové pole s veľkosťou 50 znakov, druhé pole je tiež textové a môže mať 255 znakov a tretie pole bude dátumové pole:

 CREATE TABLE TempLog

 ([User] Text(50), [Description] Text, [LogDate] Date)

Ak nie je špecifikovaná veľkosť poľa ako v predošlom príklade pre druhé pole, Access automaticky nastaví preddefinovanú dĺžku poľa.

**Práca so stĺpcami pomocou príkazu ALTER TABLE**

Príkaz ALTER TABLE poskytuje niektoré ďalšie metódy na zmenu štruktúry tabuľky v zákulisí. Existuje niekoľko klauzúl, ktoré môžeme použiť s príkazom ALTER TABLE, z ktorých štyri sú celkom užitočné pri analýze údajov programu Access: ADD, ALTER COLUMN, DROP COLUMN a ADD CONSTRAINT.

Príkaz ALTER TABLE spolu s jeho rôznymi podpríkazmi sa používa oveľa menej často ako príkazy SQL uvedené vyššie. Príkaz ALTER TABLE sa však hodí, keď naše analytické procesy vyžadujú, aby sme za behu zmenili štruktúru tabuliek, čo nám pomôže vyhnúť sa akýmkoľvek manuálnym manipuláciám, ktoré je potrebné vykonať. Treba poznamenať, že neexistuje spôsob, ako vrátiť späť žiadne akcie vykonané pomocou príkazu ALTER TABLE. To si samozrejme vyžaduje určitú opatrnosť pri používaní tohto príkazu.

**Pridanie stĺpca pomocou príkazu ADD**

Príkaz ADD umožňuje pridať pole do existujúcej tabuľky. Základná syntax je nasledovná:

 ALTER TABLE <TableName>

 ADD <ColumnName> Type (<Field Size>)

Nasledovný príklad pridá nové pole s názvom SupervisorPhone, ktoré sa pridá do tabuľky TempLog:

 ALTER TABLE TempLog

 ADD SupervisorPhone Text(10)

**Zmena stĺpca pomocou klauzuly ALTER COLUMN**

Pri použití klauzuly ALTER COLUMN zadáme existujúci stĺpec v existujúcej tabuľke. Táto klauzula sa používa predovšetkým na zmenu typu údajov a veľkosti poľa daného stĺpca. Základná syntax je nasledovná:

 ALTER TABLE <TableName>

 ALTER COLUMN <ColumnName> Type (<Field Size>)

Nasledovný príkaz zmení veľkosť poľa SupervisorPhone:

ALTER TABLE TempLog

 ALTER COLUMN SupervisorPhone Text(13)

**Vymazanie poľa pomocou príkazu DROP COLUMN**

Príkaz DROP COLUMN nám umožňuje vymazať daný stĺpec z existujúcej tabuľky. Syntax je nasledovná:

 ALTER TABLE <TableName>

 DROP COLUMN <ColumnName>

Nasledovným príkladom vymažeme pole SupervisorPhone z tabuľky TempLog:

ALTER TABLE TempLog

 DROP COLUMN SupervisorPhone

**Dynamické pridávanie hlavného kľúča pomocou príkazu ADD CONSTRAINT**

Pre mnohých analytikov Access slúži ako ľahko použiteľný nástroj na extrahovanie (Extract), transformáciu (Transform), načítanie (Load) - ETL. To znamená, že Access nám umožňuje extrahovať údaje z mnohých zdrojov a potom tieto údaje preformátovať a vyčistiť do konsolidovaných tabuliek. Mnoho analytikov tiež automatizuje procesy ETL pomocou makier, ktoré spúšťajú sériu dopytov. Vo väčšine prípadov to funguje celkom dobre.

Existujú však prípady, v ktorých proces ETL vyžaduje pridanie hlavných kľúčov do dočasných tabuliek, aby sa údaje počas spracovania udržali v normalizácii. V týchto situáciách väčšina ľudí robí jednu z dvoch vecí: zastavia makro uprostred spracovania, aby manuálne pridali požadované hlavné kľúče, alebo vytvoria trvalú tabuľku výlučne na účely držania tabuľky, kde sú hlavné kľúče už nastavené.

Existuje však aj tretia možnosť: klauzula ADD CONSTRAINT nám umožňuje dynamicky vytvárať hlavné kľúče. Základná syntax je nasledovná:

 ALTER TABLE <TableName>

 ADD CONSTRAINT ConstraintName PRIMARY KEY (<FieldName>)

Nasledovný príklad vytvorí zložený kľúč z troch polí v tabuľke TempLog:

 ALTER TABLE TempLog

 ADD CONSTRAINT ConstraintName PRIMARY KEY (ID, Name, Email)