Příkaz ALTER TABLE

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlAlterC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlAlterX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"daSQLAlterS"}

Umožní upravit vzhled tabulky poté, co byla vytvořena pomocí příkazu CREATE TABLE.

Poznámka: Databázové jádro Microsoft Jet nepoužívá příkaz ALTER TABLE, nebo jakýkoli jiný příkaz DDL, pro jinou databázi než Microsoft Jet. Namísto toho můžete použít metody DAO Create.

Syntaxe

ALTER TABLE *tabulka* {ADD {COLUMN *typ pole*[(*velikost*)] [NOT NULL] [CONSTRAINT *index*] |
CONSTRAINT *vícenásobný index*} |
DROP {COLUMN *pole* I CONSTRAINT *název indexu*} }

Příkaz ALTER TABLE se skládá z těchto částí:

Část	Popis
tabulka	Název tabulky, která má být upravena.
pole	Název pole, které má být přidáno nebo odstraněno z tabulky.
typ	Datový typ *pole*.
velikost	Velikost pole ve znacích (Pouze pro textová a binární pole).
index	Index *pole*. Více informací o tom, jak vytvořit tento index naleznete u hesla CONSTRAINT.
vícenásobný index	Definice vícenásobného indexu, který má být přidán k *tabulce*. Více informací o tom, jak vytvořit tento index naleznete u hesla CONSTRAINT
název indexu	Název vícenásobného indexu, který má být odstraněn.

Poznámky

Pomocí příkazu ALTER TABLE můžete změnit existující tabulku několika způsoby. Můžete:

- Použít frázi ADD COLUMN pro přidání nového pole do tabulky. Určíte název pole, datový typ, a (pro textová a binární pole) velikost. Například následující příkaz přidá textové pole nazvané Poznámky o délce 25 znaků do tabulky Zaměstnanci:

  ```
  ALTER TABLE Zaměstnanci ADD COLUMN Poznámky TEXT(25)
  ```

  Můžete také pro toto pole definovat index. Více informací o jednoduchých indexech naleznete u hesla CONSTRAINT .

  Pokud použijete pro určité pole frázi NOT NULL, nové věty musí v tomto poli mít platná data.
- Použít frázi ADD CONSTRAINT, která vám umožní přidat vícenásobné indexy. Více informací o vícenásobných indexech naleznete u hesla CONSTRAINT .
- Použít frázi DROP COLUMN pro odstranění pole. Určíte pouze název pole, které má být odstraněno.
- Použít frázi DROP CONSTRAINT pro odstranění vícenásobného indexu. Určíte pouze název indexu, za kterým uvedete vyhrazené slovo CONSTRAINT.

Upozornění

- Nemůžete současně přidat nebo odstranit více než jedno pole nebo index.
- Pro přidání jednoduchých nebo vícenásobných indexů k tabulce, můžete použít též příkaz CREATE INDEX a pro odstranění indexu vytvořeného pomocí příkazu ALTER TABLE nebo CREATE INDEX můžete použít příkaz ALTER TABLE nebo DROP.
- Frázi NOT NULL můžete použít na jednoduchém poli, nebo uvnitř fráze CONSTRAINT, která

přísluší buď k jednoduchému nebo k vícenásobnému poli CONSTRAINT. Omezení NOT NULL můžete však použít pro určité pole pouze jednou, jinak nastane chyba při běhu programu.

Fráze CONSTRAINT

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlConstraintC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlCreateTableX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"daSQLConstraintS"}

Omezení je obdobou indexu s tím rozdílem, že může být použit pro vytvoření relace s jinou tabulkou.

Frázi CONSTRAINT je možno použít v příkazech ALTER TABLE a CREATE TABLE pro vytvoření nebo odstranění omezení. Existují dva typy fráze CONSTRAINT : jeden pro vytvoření omezení pro jedno pole a jeden pro vytvoření omezení pro více než jedno pole.

Poznámka: Databázové jádro Microsoft Jet nepoužívá frázi CONSTRAINT nebo jakýkoli jiný příkaz DDL, pro jinou databázi než Microsoft Jet. Namísto toho můžete použít metody DAO Create .

Syntaxe

Omezení pro jedno pole:

CONSTRAINT *název* {PRIMARY KEY | UNIQUE | NOT NULL |
REFERENCES *připojovaná tabulka* [(*pole připojované tabulky1, pole připojované tabulky2*)]}

Omezení pro více polí:

CONSTRAINT *název*
{PRIMARY KEY (*primární klíč1*[,*primární klíč2* [, ...]]) |
UNIQUE (*jednoznačný klíč1*[,(*jednoznačný klíč2* [, ...]]) |
NOT NULL (*platná hodnota1*[,*platná hodnota2* [, ...]]) |
FOREIGN KEY (*odkaz1*[, *odkaz2* [, ...]]) REFERENCES *připojovaná tabulka*[(*pole připojované tabulky1* [,*pole připojované tabulky2* [, ...]])]}

Fráze CONSTRAINT sestává z těchto částí:

Část	Popis
název	Název omezení, který má být vytvořen.
primární klíč1, *primární klíč2*	Název pole nebo polí, která jsou navržena jako primární klíč.
jednoznačný klíč1, *jednoznačný klíč2*	Název pole nebo polí, která jsou navržena jako jednoznačný klíč.
platná hodnota1, platná hodnota2	Název pole nebo polí vyhrazených pro hodnoty, které nejsou **prázdné**.
odkaz1, *odkaz2*	Název pole nebo polí v připojované tabulce.
připojovaná tabulka	Název připojované tabulky obsahující pole nebo několik polí určených jako *pole připojované tabulky*.
pole připojované tabulky1, *pole připojované tabulky2*	Název pole nebo polí v *připojované tabulce* určených *odkazem1*, *odkazem2*. Tuto frázi můžete vynechat, pokud je odkazem pole, které tvoří primární klíč *připojované tabulky*.

Poznámky

Syntax pro omezení pro jedno pole použijete ve frázi pro definici pole příkazu ALTER TABLE nebo CREATE TABLE bezprostředně po určení datového typu pole.

Syntax pro omezení pro více polí použijete při zadání vyhrazeného slova CONSTRAINT mimo frázi pro definici pole v příkazu ALTER TABLE nebo CREATE TABLE .

Pomocí fráze CONSTRAINT můžete označit pole jako jedno z následujících typů omezení:

- Můžete použít vyhrazené slovo UNIQUE pro označení pole jako jednoznačného klíče. To znamená, že žádné dvě věty z tabulky nesmí mít v tomto poli stejnou hodnotu. Můžete vyhradit jakékoliv pole nebo seznam polí jako jednoznačné. Pokud je jako jednoznačný klíč určeno omezení pro více polí, kombinace hodnot všech polí z indexu musí být jednoznačná, i když dvě nebo více vět mají stejnou hodnotu v jednom poli.
- Vyhrazené slovo PRIMARY KEY můžete použít pro označení jednoho pole nebo několika polí v tabulce jako primárního klíče. Všechny hodnoty primárního klíče musí být jednoznačné a nebýt **prázdné**, a jedna tabulka může mít pouze jeden primární klíč.

  Poznámka: Nenastavujte frázi PRIMARY KEY na tabulce, která již primární klíč má; pokud tak učiníte, objeví se chyba.
- Vyhrazené slovo FOREIGN KEY můžete použít pro určení klíče pomocí pole z připojované tabulky. Pokud primární klíč připojované tabulky sestává z více než jednoho pole, musíte použít definici omezení pro více polí, a vypsat všechny odkazovací pole, název připojované tabulky a názvy polí v připojované tabulce ve stejném pořadí, v jakém jsou vypsány odkazovací pole. Pokud pole nebo seznam polí v připojované tabulce jsou primárním klíčem připojované tabulky, nemusíte vypisovat pole připojované tabulky,databáze automaticky předpokládá, že primární klíč připojované tabulky jsou pole připojované tabulky.

Příkaz CREATE INDEX

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlCreateIndexC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlCreateIndexX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"daSQLCreateIndexS"}

Vytvoří nový index pro existující tabulku.

Poznámka: Databázové jádro Microsoft Jet nepoužívá příkaz CREATE INDEX(s vyjímkou vytvoření pseudo indexu na propojené tabulce ODBC), nebo jakýkoli jiný příkaz DDL, pro jinou databázi než Microsoft Jet. Namísto toho můžete použít metody DAO Create.Více informací naleznete v sekci Připomínky.

Syntaxe

CREATE [UNIQUE] INDEX *index*
ON *tabulka* (*pole* [ASC|DESC][, *pole* [ASC|DESC], ...])
[WITH { PRIMARY | DISALLOW NULL | IGNORE NULL }]

Příkaz CREATE INDEX se skládá z těchto částí:

Část	Popis
index	Název indexu, který má být vytvořen.
tabulka	Název existující tabulky, pro kterou má být index vytvořen.
pole	Název pole nebo polí pro vytvoření indexu. Pro vytvoření jednoduchého indexu vypište název pole v závorkách následovaný názvem tabulky. Chcete-li vytvořit vícenásobný index, vypište názvy všech polí, která mají být v indexu obsažena. Pokud chcete vytvořit index v sestupném pořadí použijte vyhrazené slovo DESC; pokud tak neučiníte bude index vytvořen ve vzestupném pořadí.

Připomínky

Chcete-li zamezit duplicitním hodnotám různých vět v poli nebo polích použitých v indexu, použijte vyhrazené slovo UNIQUE.

V nepovinné frázi WITH můžete připojit pravidla pro kontrolu platnosti dat. Můžete:

- Zakázat vstup **prázdné** hodnoty pro novou větu v poli nebo polích použitých pro index pomocí volby DISALLOW NULL.
- Zamezit tomu, aby byly věty s **prázdnou** hodnotou v poli nebo polích použitých pro index zahrnuty do indexu pomocí volby IGNORE NULL.
- Označit indexové pole nebo seznam polí jako primární klíč pomocí vyhrazeného slova PRIMARY. Z toho automaticky vyplývá, že klíč je jednoznačný, takže můžete vynechat vyhrazené slovo UNIQUE.

Příkaz CREATE INDEX můžete použít pro vytvoření pseudo indexu pro propojenou tabulku v ODBC datovém zdroji, jako je například SQL Server, která dosud nemá index. Pro vytvoření pseudo indexu nepotřebujete povolení nebo přístup na vzdálený server, a ve vzdálené databázi nebude proveden žádný zásah ani o této akci nebude uvědoměna. Pro vytvoření indexu na vlastní i propojené tabulce použijete stejnou syntax. To může být zvláště užitečné pro vytvoření indexu na tabulce, která je obvykle určena pouze ke čtení, kvůli tomu že nemá index.

Pro přidání jednoduchého nebo vícenásobného indexu k tabulce můžete použít také příkaz ALTER TABLE a pro odstranění indexu vytvořeného pomocí příkazu ALTER TABLE nebo CREATE INDEX můžete použít příkaz ALTER TABLE nebo DROP.

Poznámka: Nepoužívejte vyhrazené slovo PRIMARY pokud vytváříte nový index na tabulce, která již primární klíč obsahuje; pokud tak učiníte, dojde k chybě.

Příkaz CREATE TABLE

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlCreateTableC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlCreateTableX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"daSQLCreateTableS"}

Vytvoří novou tabulku

Poznámka: Databázové jádro Microsoft Jet nepoužívá příkaz CREATE TABLE, nebo jakýkoli jiný příkaz DDL, pro jinou databázi než Microsoft Jet. Namísto toho můžete použít metody DAO Create .

Syntaxe

CREATE TABLE *tabulka* (*pole1 typ* [(*velikost*)] [NOT NULL] [*index1*] [, *pole2 typ* [(*velikost*)] [NOT NULL] [*index2*] [, ...]] [, CONSTRAINT *vícenásobný index* [, ...]])

Příkaz CREATE TABLE sestává z těchto částí:

Část	Popis
tabulka	Název tabulky, která má být vytvořena.
pole1, *pole2*	Název pole nebo polí, která mají být vytvořena v nové tabulce. Musíte vytvořit alespoň jedno pole.
typ	Datový typ *pole* v nové tabulce.
velikost	Velikost pole ve znacích (Pouze pro textová a binární pole).
index1, *index2*	Fráze CONSTRAINT definující jednoduchý index. Více informací o tom, jak vytvořit tento index naleznete u heslaCONSTRAINT.
vícenásobný index	Fráze CONSTRAINT definující vícenásobný index. Více informací o tom, jak vytvořit tento index naleznete u hesla CONSTRAINT.

Poznámky

Příkaz CREATE TABLE můžete použít pro definování nové tabulky a jejích polí a omezení. Pokud je pro pole vybrána volba NOT NULL, musí nové věty obsahovat v tomto poli platné hodnoty.

Fráze CONSTRAINT vytváří různá omezení pro pole, a může být použita pro vytvoření primárního klíče. Pro vytvoření primárního klíče nebo dalších indexů k existující tabulce, můžete také použít příkaz CREATE INDEX.

Frázi NOT NULL můžete použít na jednoduchém poli, nebo uvnitř fráze CONSTRAINT, která přísluší buď k jednoduchému nebo k vícenásobnému poli CONSTRAINT. Omezení NOT NULL můžete však použít pro určitému pole pouze jednou, jinak nastane chyba při běhu programu.

Příkaz DROP

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlDropC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlDropX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"daSQLDropS"}

Odstraní existující tabulku z databáze nebo odstraní existující index z tabulky.

Poznámka: Databázové jádro Microsoft Jet nepoužívá příkaz DROP, nebo jakýkoli jiný příkaz DDL pro jinou databázi než Microsoft Jet. Namísto toho můžete použít metody DAO **Delete**.

Syntaxe

DROP {TABLE *tabulka* | INDEX *index* ON *tabulka*}

Příkaz DROP sestává z těchto částí:

Část	Popis
tabulka	Název tabulky, která má být odstraněna nebo tabulky, ze které má být odstraněn index.
index	Název indexu, který má být odstraněn z tabulky.

Připomínky

Pokud chcete odstranit tabulku nebo odstranit index z tabulky, musíte nejprve tabulku zavřít.

Chcete-li odstranit index z tabulky, můžete použít také příkaz ALTER TABLE.

Pro vytvoření tabulky použijte příkaz CREATE TABLE a pro vytvoření indexu příkaz CREATE INDEX nebo ALTER TABLE. Pro úpravu tabulky použijte příkaz ALTER TABLE.

Příkaz SELECT

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlSelectC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlSelectX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlSELECTS"}

Dá pokyn Databázové jádro Microsoft Jet pro zaslání informací z databáze v podobě sestavy vět.

Syntaxe

SELECT [*predikát*] { * | *tabulka*.* | [*tabulka*.]*pole1* [AS *alias1*] [, [*tabulka*.]*pole2* [AS *alias2*] [, ...]]}
FROM *tabulkový výraz* [, ...] [IN *externí databáze*]
[WHERE...]
[GROUP BY...]
[HAVING...]
[ORDER BY...]
[WITH OWNERACCESS OPTION]

Příkaz SELECT sestává z těchto částí:

Část	Popis
predikát	Jeden z následujících predikátů: ALL, DISTINCT, DISTINCTROW, nebo TOP. Predikáty jsou použity pro vymezení počtu vybraných vět. Pokud není určen žádný, je implicitně doplněno ALL.
*	Určuje, že budou vybrána všechna pole ze zadané tabulky nebo tabulek.
tabulka	Název tabulky obsahující pole, ze které jsou vybírány věty.
pole1, *pole2*	Názvy polí obsahující data, která chcete získat. Pokud zadáte více než jedno pole, budou vrácena v pořadí v jakém byla zadána.
alias1, *alias2*	Názvy použité v záhlaví sloupců namísto původních názvů sloupců v *tabulce*.
tabulkový výraz	Název tabulky nebo tabulek obsahující data, která chcete získat.
externí databáze	Název databáze obsahující tabulky použité v parametru *tabulkový výraz,* pokud nejsou v aktuální databázi.

Poznámky

Pro uskutečnění této operace databázové jádro Microsoft Jet prohledá určenou tabulku nebo tabulky, vyjme zvolené sloupce, vybere řádky splňující stanovená kritéria, a setřídí nebo seskupí vybrané řádky podle určeného pořadí.

Příkaz SELECT nemění data v databázi.

SELECT je obvykle první slovo v příkazu SQL . Většina příkazů SQL je buď příkaz SELECT nebo SELECT...INTO .

Minimální syntaxe pro příkaz SELECT je:

SELECT *pole* FROM *tabulka*

Můžete použít hvězdičku, (*) aby byla vybrána všechna pole z tabulky. Následující příklad vybírá všechna pole z tabulky Zaměstnanci:

```
SELECT * FROM Zaměstnanci;
```

Pokud se název pole ve frázi FROM vyskytuje ve více než jedné tabulce, umístěte před něj název tabulky a tečku (.). V následujícím příkladě, je pole Oddělení jak v tabulce Zaměstnanci tak i v tabulce Kortroloři. Příkaz SQL vybere pole Oddělení z tabulky Zaměstnanci a pole KontrJméno z tabulky Kontroloři:

```
SELECT Zaměstnanci.Oddělení, Kontroloři.KontrJméno
FROM Zaměstnanci INNER JOIN Kontroloři
WHERE Zaměstnanci.Oddělení = Kontroloři.Oddělení;
```

Při vytváření objektu **Recordset** Databázové jádro Microsoft Jet použije v objektu **Recordset** pro názvy objektu **Field** názvy polí z tabulky. Chcete-li, aby názvy polí byly odlišné nebo aby název nebyl odvozen z výrazu použitého pro vytvoření pole, použijte vyhrazené slovo AS. Následující příklad používá pro pojmenování vybraného **Filed** ve výsledném objektu **Recordset** záhlaví Narození:

```
SELECT DatumNarození
AS Narození FROM Zaměstnanci;
```

Kdykoliv použijete agregované funkce nebo dotazy, které vracejí nejednoznačné nebo duplicitní názvy **Field**, musíte použít frázi AS, aby byl objektu **Field** přiřazen náhradní název. Následující příklad používá záhlaví s názvem Počet pro pojmenování vybraného **Field** ve výsledném objektu **Recordset**:

```
SELECT COUNT(IDZaměstnance)
AS Počet FROM Zaměstnanci;
```

V příkazu SELECT můžete používat i ostatní fráze pro další omezení a organizování získaných dat. Více informací naleznete u hesla Nápovědy pro příslušnou frázi.

Fráze FROM

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlFromC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"daSQLFromX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlFROMS"}

Určuje tabulky nebo dotazy, které obsahují pole, uvedená v příkazu SELECT .

Syntaxe

SELECT *seznam polí*
FROM *tabulkový výraz* [IN *externí databáze*]

Příkaz SELECT obsahující frázi FROM sestává z těchto částí:

Část	Popis
seznam polí	Název pole nebo polí, která mají být vyhledána, spolu se všemi názvy alias, agregovanými funkcemi SQL, predikáty výběru (ALL, DISTINCT, DISTINCTROW, nebo TOP), a dalšími volbami příkazu SELECT.
tabulkový výraz	Výraz, určující jednu nebo více tabulek, ze kterých budou vybírána data. Výraz může být název jedné tabulky, uložený název dotazu, nebo výraz složený pomocí frází INNER JOIN, LEFT JOIN nebo RIGHT JOIN.
externí databáze	Úplný název externí databáze, která obsahuje všechny tabulky z parametru *tabulkový výraz*.

Poznámky

Fráze FROM je povinná a je obsažena v každém příkazu SELECT.

Pořadí názvů tabulek v parametru *tabulkový výraz* není důležité.

Pro zlepšení výkonu a snadnější použití doporučujeme při vyhledávání dat z externí databáze použít namísto fráze IN propojenou tabulku.

Následující příklad ukazuje, jak můžete získat data z tabulky Zaměstnanci :

```
SELECT Příjmení, Jméno
FROM Zaměstnanci;
```

Fráze IN

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlInC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlInX":1}
{ewc HLP95EN.DLL,DYNALINK,"Specifika":"daSQLINS"}

Určuje tabulky v jakékoliv externí databázi, ke které se může databázové jádro Microsoft Jet připojit, jako je databáze systému dBASE nebo Paradox nebo externí databáze Microsoft Jet.

Syntaxe

Určení cílové tabulky:

[SELECT | INSERT] INTO *cíl* IN
{*cesta* | ["*cesta*" "*typ*"] | ["" [*typ*; DATABASE = *cesta*]]}

Určení zdrojové tabulky:

FROM *tabulkový výraz* IN
{*cesta* | ["*cesta*" "*typ*"] | ["" [*typ*; DATABASE = *cesta*]]}

Příkaz SELECT obsahující frázi IN sestává z těchto částí:

Část	Popis
cíl	Název externí tabulky, do které jsou data vkládána.
tabulkový výraz	Název tabulky nebo tabulek, ze kterých jsou data získávána. Tento výraz může být název jedné tabulky, uložený dotaz, nebo výraz složený pomocí frází INNER JOIN, LEFT JOIN nebo RIGHT JOIN.
cesta	Úplná cesta do adresáře nebo na soubor obsahující *tabulku.*
typ	Název typu databáze použité pro vytvoření *tabulky,* pokud databáze není databází Microsoft Jet (Například dBASE III, dBASE IV, Paradox 3.x, nebo Paradox 4.x).

Poznámky

Frázi IN můžete použít pro připojení současně pouze k jedné externí databázi.

V některých případech odkazuje parametr *cesta* na adresář obsahující databázové soubory. Pokud například pracujete s databázovými tabulkami systému dBASE, FoxPro nebo Paradox, parametr *cesta* určuje adresář obsahující soubory s příponou .dbf nebo .db. Název souboru tabulky je odvozen z parametru *cíl* nebo *tabulkový výraz*.

Pokud chcete určit jinou databázi než databázi Microsoft Jet, připojte ke jménu středník (;) a uzavřete jej do jednoduchých ('') nebo dvojitých ("") uvozovek. Například 'dBASE IV;' nebo "dBASE IV;" jsou přípustné výrazy.

Pro určení externí databáze můžete také použít vyhrazené slovo DATABASE. Následující řádky určují stejnou tabulku:

```
... FROM Tabulka IN "" [dBASE IV; DATABASE=C:\DBASE\DATA\PRODEJ;];
... FROM Tabulka IN "C:\DBASE\DATA\PRODEJ" "dBASE IV;"
```

Upozornění

- Pro zlepšení výkonu a snadnější použití, doporučujeme při vyhledávání dat z externí databáze použít namísto fráze IN propojenou tabulku.
- Vyhrazené slovo IN můžete použít také jako relační operátor ve výrazu. Více informací naleznete u hesla operátor **In**.

Fráze WHERE

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlWhereC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlWhereX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlWHERES"}

Určuje, se kterými větami z tabulek uvedených ve frázi FROM bude pracovat příkaz SELECT, UPDATE nebo DELETE.

Syntaxe

SELECT *seznam polí*
FROM *tabulkový výraz*
WHERE *podmínka*

Příkaz SELECT obsahující frázi WHERE sestává z těchto částí:

Část	Popis
seznam polí	Název pole nebo polí, která mají být vybrána spolu se všemi názvy alias, výběrovými predikáty (ALL, DISTINCT, DISTINCTROW nebo TOP), a dalšími volbami příkazu SELECT.
tabulkový výraz	Název tabulky nebo tabulek, ze kterých mají být získána data.
podmínka	Výraz, který musí splňovat věty, aby byly zahrnuty do výsledku dotazu.

Poznámky

Databázové jádro Microsoft Jet vybere věty splňující podmínky uvedené ve frázi WHERE. Pokud neuvedete frázi WHERE, dotaz vrátí všechny řádky z tabulky. Pokud zadáte v dotazu více než jednu tabulku a neuvedete frázi WHERE nebo JOIN, dotaz vytvoří kartézský součin tabulek.

Fráze WHERE je nepovinná, ale pokud je uvedena, musí být umístěna za klíčové slovo FROM. Můžete například vybrat všechny zaměstnance prodejního oddělení (`WHERE Odděl = 'Prodej'`) nebo všechny zákazníky, jejichž věk se pohybuje v rozmezí 18 a 30 let (`WHERE Věk mezi 18 a 30`).

Pokud neuvedete frázi JOIN, která provádí SQL operace propojení s několika tabulkami, výsledný objekt **Recordset** nebude aktualizovatelný.

Fráze WHERE je podobná frázi HAVING. WHERE určuje, které věty budou vybrány. Podobně, pokud jsou věty seskupeny pomocí fráze GROUP BY, fráze HAVING určuje, které věty budou zobrazeny.

Použijte frázi WHERE, aby byly z výběru vyřazeny věty, které nechcete seskupit pomocí fráze GROUP BY.

Chcete-li určit, které věty má vrátit příkaz SQL, můžete pro to použít různé výrazy.Následující příkaz SQL vybírá například všechny zaměstnance, jejichž plat převyšuje 21,000 Kč:

```
SELECT Příjmení, Plat
FROM Zaměstnanci
WHERE Plat > 21000;
```

Fráze WHERE může obsahovat až 40 výrazů spojených logickými operátory **And** a **Or**.

Pokud zadáte název pole, které obsahuje mezeru nebo interpunkci, uzavřete název do hranatých závorek ([]). Například tabulka obsahující informace o zákaznících může obsahovat informace o určitých zákaznících:

```
SELECT [Zákazníkova oblíbená restaurace]
```

Když stanovujete parametr *podmínka*, datum musí být ve formátu U.S., dokonce i tehdy, pokud

nepoužíváte U.S. verzi databázového jádra Microsoft Jet. Například: 10. květen 1996, se ve Velké Británii píše 10/5/96 a ve Spojených Státech 5/10/96. Přesvědčte se, že jste datum uzavřeli znakem #, jak je uvedeno v následujícím příkladě.

Chcete-li nalézt věty s datumem 10. květen 1996 v databázi z Velké Británie, musíte použít následující příkaz SQL:

```
SELECT *
FROM Objednávky
WHERE Datum = #5/10/96#;
```

Můžete použít také funkci **DateValue,** která pracuje s mezinárodním nastavením používaným v Microsoft Windows. Následuje příklad pro Spojené Státy:

```
SELECT *
FROM Objednávky
WHERE DatumNaložení = DateValue('5/10/96');
```

A příklad pro Velkou Británii:

```
SELECT *
FROM Objednávky
WHERE Datum = DateValue('10/5/96');
```

Upozornění Pokud sloupec uvedený v řetězci podmínky je typu GUID, podmínkový výraz používá mírně odlišnou syntax:

```
WHERE IDDuplikátu = {GUID {12345678-90AB-CDEF-1234-567890ABCDEF}}
```

Ujistěte se, že jste uvedli vložené složené závorky a pomlčky, jak je uvedeno výše.

Fráze GROUP BY

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlGroupByC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlGroupByX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlGROUPBYS"}

Sloučí věty se stejnými hodnotami v zadaném seznamu polí do jediné věty. Pokud zadáte v příkazu SELECT agregovanou funkci SQL, například **Sum** nebo **Count,** vytvoří se pro každou větu souhrnná hodnota.

Syntaxe

SELECT *seznam polí*
FROM *tabulka*
WHERE *podmínka*
[GROUP BY *seznam polí skupiny*]

Příkaz SELECT obsahující frázi GROUP BY sestává z těchto částí:

Část	Popis
seznam polí	Název pole nebo polí, která mají být vybrána spolu se všemi názvy alias, výběrovými predikáty (ALL, DISTINCT, DISTINCTROW nebo TOP), a dalšími volbami příkazu SELECT.
tabulka	Název tabulky, ze které jsou získávány věty. Více informací naleznete u fráze FROM.
podmínka	Výběrová podmínka. Pokud příkaz obsahuje frázi WHERE, databázové jádro Microsoft Jet seskupí hodnoty až poté, co se pro věty uplatní podmínka uvedená ve frázi WHERE.
seznam polí skupiny	Názvy až deseti polí použitých pro seskupení vět. Pořadí názvů polí uvedených v *seznamu polí skupiny* určuje úroveň skupin od nejvyšší do nejnižší.

Poznámky

Fráze GROUP BY je nepovinná.

Souhrnné hodnoty jsou vypuštěny, pokud se v příkazu SELECT nenachází žádná agregovaná funkce SQL.

Prázdné hodnoty jsou v polích uvedených ve frázi GROUP BY seskupeny a nejsou vypuštěny. **Prázdné** hodnoty nejsou však vyhodnocovány žádnými agregovanými funkcemi SQL.

Pomocí fráze WHERE vypustíte řádky, které nechcete seskupovat a pomocí fráze HAVING vytvoříte filtr pro věty poté, co budou seskupeny.

Kromě toho, že nesmí obsahovat data typu Memo nebo OLE, může se seznam polí uvedený ve frázi GROUP BY odkazovat na jakékoliv pole v jakékoliv tabulce uvedené ve frázi FROM, dokonce i v případě, že pole není uvedeno v příkazu SELECT, za předpokladu, že příkaz SELECT obsahuje alespoň jednu agregovanou funkci SQL. Databázové jádro Microsoft Jet neumí seskupovat podle polí typu Memo nebo OLE.

Všechna pole uvedená v seznamu polí příkazu SELECT musí být buď uvedena ve frázi GROUP BY nebo obsažena v agregované funkci SQL.

Fráze HAVING

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlHavingC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlHavingX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"daSQLHAVINGS"}

Určuje, které věty seskupené pomocí fráze GROUP BY budou v příkazu SELECT zobrazeny. Poté co fráze GROUP BY sloučí věty, fráze HAVING zobrazí všechny věty seskupené frází GROUP BY, které splňují podmínku fráze HAVING.

Syntaxe

SELECT *seznam polí*
FROM *tabulka*
WHERE *výběrová podmínka*
GROUP BY *seznam polí skupiny*
[HAVING *seskupovací podmínka*]

Příkaz SELECT obsahující frázi HAVING sestává z těchto částí:

Část	Popis
seznam polí	Název pole nebo polí, která mají být vybrána spolu se všemi názvy alias, SQL agregačními funkcemi, výběrovými predikáty (ALL, DISTINCT, DISTINCTROW nebo TOP), a dalšími volbami příkazu SELECT.
tabulka	Název tabulky, ze které jsou získávány věty. Více informací naleznete u fráze FROM.
výběrová podmínka	Podmínka výběru. Pokud příkaz obsahuje frázi WHERE, databázové jádro Microsoft Jet seskupí hodnoty až poté, co se pro věty uplatní podmínka uvedená ve frázi WHERE.
seznam polí skupiny	Názvy až deseti polí použitých pro seskupení vět. Pořadí názvů polí uvedených v *seznamu polí skupiny* určuje úroveň skupin od nejvyšší do nejnižší.
seskupovací podmínka	Výraz určující, které seskupené věty mají být zobrazeny.

Poznámky

Fráze HAVING je nepovinná.

Fráze HAVING je podobná frázi WHERE, která určuje, které věty mají být vybrány. Poté, co jsou věty seskupeny pomocí fráze GROUP BY, fráze HAVING určí, které záznamy budou zobrazeny:

```
SELECT IDKategorie,
Sum(KusůVBalení)
FROM Výrobků
GROUP BY IDKategorie
HAVING Sum(KusůVBalení) > 100 And Like "BOS*";
```

Fráze HAVING může obsahovat až 40 výrazů spojených logickými operátory **And** a **Or**.

Fráze ORDER BY

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlOrderByC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlOrderByX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlORDERBYS"}

Setřídí věty výsledného dotazu podle zadaného pole nebo polí ve vzestupném nebo sestupném pořadí.

Syntaxe

SELECT *seznam polí*
FROM *tabulka*
WHERE *výběrová podmínka*
[ORDER BY *pole1* [ASC | DESC][, *pole2* [ASC | DESC]][, ...]]

Příkaz SELECT obsahující frázi ORDER BY sestává z těchto částí:

Část	Popis
seznam polí	Název pole nebo polí, která mají být vybrána spolu se všemi názvy alias, SQL agregačními funkcemi, výběrovými predikáty (ALL, DISTINCT, DISTINCTROW nebo TOP), a dalšími volbami příkazu SELECT.
tabulka	Název tabulky, ze které jsou získávány věty. Více informací naleznete u fráze FROM.
výběrová podmínka	Podmínka výběru. Pokud příkaz obsahuje frázi WHERE, databázové jádro Microsoft Jet seskupí hodnoty až poté, co se pro věty uplatní podmínka uvedená ve frázi WHERE.
pole1, *pole2*	Názvy polí, podle kterých mají být věty setříděny.

Poznámky

Fráze ORDER BY je nepovinná. Chcete-li však zobrazit data setříděná, musíte použít frázi ORDER BY.

Výchozí nastavení je vzestupné setřídění (A až Z, 0 až 9). Oba následující příklady třídí jména zaměstnanců podle příjmení:

```
SELECT Příjmení, Jméno
FROM Zaměstnanci
ORDER BY Příjmení;

SELECT Příjmení, Jméno
FROM Zaměstnanci
ORDER BY Příjmení ASC;
```

Pokud chcete věty setřídit v sestupném pořadí (Z až A, 9 až 0), připojte na konec každého pole, podle kterého se má třídit sestupně, vyhrazené slovo DESC. Následující příklad vybírá platy a třídí je v sestupném pořadí:

```
SELECT Příjmení, Plat
FROM Zaměstnanci
ORDER BY Plat DESC, Příjmení;
```

Pokud ve frázi ORDER BY zadáte pole typu Memo nebo OLE, objeví se chyba. Databázové jádro Microsoft Jet neumí třídit podle polí typu MEMO nebo OLE.

Fráze ORDER BY je obvykle poslední položka v příkazu SQL.

Ve frázi ORDER BY můžete uvést více polí. Věty budou setříděny nejprve podle prvního pole,

uvedeného za frází ORDER BY. Věty, které mají shodnou hodnotu v tomto poli budou následně setříděny podle hodnoty uvedené ve druhém poli atd.

Predikáty ALL, DISTINCT, DISTINCTROW, TOP

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlAllDistinctC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝dž˝klad":"dasqlAllDistinctX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlALLDISTINCTS"}

Blíže určují věty vybrané dotazy SQL.

Syntaxe

SELECT [ALL | DISTINCT | DISTINCTROW | [TOP *n* [PERCENT]]]
FROM *tabulka*

Příkaz SELECT obsahující tyto predikáty sestává z těchto částí:

Část	Popis

ALL	Předpokládá se, pokud neuvedete žádný predikát. Databázové jádro Microsoft Jet vybere všechny věty, které splňují podmínky uvedené v příkazu SQL. Následující dva příklady jsou rovnocenné a vrací všechny věty z tabulky Zaměstnanci:
	`SELECT ALL *`
	`FROM Zaměstnanci`
	`ORDER BY IDZaměstnance;`
	`SELECT *`
	`FROM Zaměstnanci`
	`ORDER BY IDZaměstnance;`

DISTINCT	Vypustí věty, které ve vybraných polích obsahují duplicitní data. Hodnoty všech polí, uvedených v příkazu SELECT musí být unikátní, pokud mají být zahrnuty do výsledku dotazu. Například několik zaměstanců z tabulky Zaměstnanci může mít stejné příjmení. Pokud dvě věty obsahují v poli Příjmení příjmení Novák, následující příkaz SQL vrátí pouze jednu větu, která obsahuje příjmení Novák:
	`SELECT DISTINCT`
	`Příjmení`
	`FROM Zaměstnanci;`
	Pokud vynecháte predikát DISTINCT, tento dotaz vrátí obě věty obsahující příjmení Novák.
	Pokud příkaz SELECT obsahuje více než jedno pole, musí být unikátní kombinace hodnot všech polí dané věty, aby byla věta zahrnuta do výsledku.
	Výsledek dotazu, který používá frázi DISTINCT není aktualizovatelný a nepromítají se do něho následné změny provedené jinými uživateli.

DISTINCTROW	Vypustí data na základě celé duplicitní věty, ne pouze na základě duplicitních polí. Chcete například vytvořit dotaz, který spojí tabulku Zakazníci a Objednávky na základě pole IDZákazníka. Tabulka Zákazníci neobsahuje žádné duplicitní pole IDZákazníka, ale tabulka Objednávky ano, protože každý zákazník může mít několik objednávek. Následující příkaz SQL ukazuje, jak můžete využít predikát DISTINCTROW pro vytvoření seznamu firem, které mají minimálně jednu objednávku bez toho, že by byly uvedeny další podrobnosti o objednávkách: `SELECT DISTINCTROW JménoSpolečnosti`

```
FROM Zákazníci INNER JOIN Objednávky
ON Zákazníci.IDZákazníka =
Objednávky.IDZákazníka
ORDER BY JménoSpolečnosti;
```

Pokud vynecháte predikát DISTINCTROW, tento dotaz vrátí několik řádků pro každou firmu, která má více než jednu objednávku.

Predikát DISTINCTROW má smysl uvádět pouze tehdy, pokud vybíráte pole z několika, ale ne všech tabulek použitých v dotazu. Pokud váš dotaz obsahuje pouze jednu tabulku, nebo pokud chcete výstup polí ze všech tabulek je predikát DISTINCTROW ignorován.

TOP *n* [PERCENT] Vrátí určitý počet vět, od začátku nebo konce výběru určeného frází ORDER BY. Předpokládejme, že chcete získat jména prvních 25 studentů ročníku 1994:

```
SELECT TOP 25
Jméno, Příjmení
FROM Studenti
WHERE RokUkončení = 1994
ORDER BY MaturitníPrůměr DESC;
```

Pokud neuvedete frázi ORDER BY, dotaz vrátí náhodnou podmnožinu 25 vět z tabulky STUDENTI, které splňují podmínku uvedenou ve frázi WHERE.

Predikát TOP neodděluje shodné hodnoty. Pokud by v předchozím příkladě byl studijní průměr studentů na 25. a 26. místě byl shodný, dotaz vrátí 26 vět.

Pokud chcete vrátit určité procento vět od začátku nebo konce výběru stanoveného frází ORDER BY, můžete použít také vyhrazené slovo PERCENT. Předpokládejme, že chcete namísto prvních 25 studentů získat posledních 10 procent:

```
SELECT TOP 10 PERCENT
Jméno, Příjmení
FROM Studenti
WHERE RokUkončení = 1994
ORDER BY MaturitníPrůměr ASC;
```

Predikát ASC určuje, že bude vráceny hodnoty z konce tabulky. Hodnoty uvedené za predikátem TOP musí být **celočíselné** a bez znaménka.

Predikát TOP nemá vliv na to, zda je dotaz aktualizovatelný.

tabulka Název tabulky, ze které jsou získávány věty.

Příkaz DELETE

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlDeleteC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlDeleteX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlDELETES"}

Vytvoří vymazávací dotaz, který odstraní záznamy z jedné nebo více tabulek zapsaných v klauzuli FROM, která souvisí s klauzulí WHERE.

Syntaxe

DELETE [*tabulka*.*]
FROM *tabulka*
WHERE *kritéria*

Příkaz DELETE se skládá z těchto částí:

Část	Popis
tabulka	Volitelný název tabulky, ze které se budou vymazávat záznamy.
tabulka	Název tabulky, ze které se budou vymazávat záznamy.
kritéria	Výraz, který určuje záznamy pro vymazání.

Poznámky

Příkaz DELETE je obzvlášť praktický při vymazávání většího počtu záznamů.

K vyjmutí celé tabulky z databáze můžete použít metodu **Execute** s příkazem DROP. Smažete-li tabulku, ztratí se samozřejmě i její struktura. Avšak použijete-li příkaz DELETE, vymažou se pouze data, zatímco struktura tabulky a všechny její vlastnosti jako například atributy polí a indexy zůstanou zachovány.

Příkaz DELETE můžete použít pro vymazávání záznamů z tabulek, které jsou s ostatními tabulkami v relaci 1 : mnoha. Operace kaskádového vymazávání způsobí, že ty záznamy, které jsou v tabulkách s vícenásobnou relací, budou vymazány, bude-li odpovídající záznam na jedné straně relace v dotazu vymazán. Například v relaci mezi tabulkami Zákazníci a Objednávky je tabulka Zákazníci v jednoduché a tabulka Objednávky ve vícenásobné relaci. Smažete-li nějaký záznam v tabulce Zákazníci a je-li nastavená volba kaskádového vymazávání, dojde k vymazání odpovídajících záznamů v tabulce Objednávky.

Vymazávací dotaz vymaže celé záznamy, nikoliv pouze data ve vybraných polích. Chcete-li vymazat hodnoty z vybraných polí, vytvoříte aktualizovaný dotaz, který změní vybrané hodnoty na **prázdné**.hodnoty.

Důležité

- Po vymazání záznamů pomocí vymazávacího dotazu nelze operaci vzít zpět. Chcete-li vědět které dotazy budou vymazány, nejprve použijte k vyzkoušení výběrový dotaz, který používá stejná kritéria a potom teprve spusťte vymazávací dotaz.
- Pořizujte si neustále záložní kopie. Jestliže vymažete nesprávné záznamy, můžete je nahradit ze záložní kopie.

Operace INNER JOIN

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlInnerJoinC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlInnerJoinX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlInnerJoinS"}

Slučuje záznamy ze dvou tabulek, jakmile se shodují hodnoty ve společném poli.

Syntaxe

FROM *tabulka1* INNER JOIN *tabulka2* ON *tabulka1*.*pole1 porovnávací_operátor tabulka2*.*pole2*

Operace INNER JOIN se skládá z těchto částí:

Část	Popis
tabulka1, *tabulka2*	Názvy tabulek, ze kterých budou slučovány záznamy.
pole1, *pole2*	Názvy polí, která jsou spojena. Nejsou-li číselná, pak musí být stejného datového typu a musí obsahovat stejný typ dat, ale nesmí mít stejný název.
porovnávací_ operátor	Libovolný porovnávací relační operátor: "=", "<", ">", "<=", ">=" nebo "<>".

Poznámky

Operaci INNER JOIN můžete použít v libovolné klauzuli FROM a je nejobecnějším typem sloučení. Vnitřně spojí sloučené záznamy ze dvou tabulek, jakmile se ve společném poli obou tabulek objeví shodné hodnoty.

Operaci INNER JOIN můžete použít v tabulkách Oddělení a Zaměstnanci pro výběr všech zaměstnanců v každém oddělení. Naproti tomu, pro výběr všech oddělení (dokonce ikdyž některým nejsou přiřazeni žádní zaměstnanci) nebo všech zaměstnanců (dokonce ikdyž někteří nejsou přiřazení žádnému odddělení), můžete použít operaci LEFT JOIN nebo RIGHT JOIN k vytvoření vnějšího sloučení.

Pokusíte-li se sloučit pole obsahující data typu Memo nebo Objekt OLE, dojde k chybě.

Můžete slučovat libovolná číselná pole podobných typů. Můžete například sloučit pole s datovými typy AutoNumber a Long, protože jsou podobného typu. Avšak nikoliv pole s datovými typy Single a Double.

Následující příklad ukazuje jako můžete sloučit tabulky Kategorie a Výrobky přes pole IDKategorie:

```
SELECT NázevKategorie, NázevVýrobku
FROM Kategorie INNER JOIN Výrobky
ON Kategorie.IDKategorie = Výrobky.IDKategorie;
```

V předchozím příkladu je pole IDKategorie připojené, ale není zahrnuto ve výstupi dotazu, protože není uvedeno v příkazu SELECT. Abyste jej začlenili do výstupu, přidejte jeho název do příkazu SELECT, v tomto případě `Kategorie.IDKategorie`.

Použitím následující syntaxe můžete sloučit několik klauzulí ON v příkazu JOIN:

SELECT *pole*
FROM *tabulka1* INNER JOIN *tabulka2*
ON *tabulka1*.*pole1 porovnávací_operátor tabulka2*.*pole1* AND
ON *tabulka1*.*pole2 porovnávací_operátor tabulka2*.*pole2*) OR
ON *tabulka1*.*pole3 porovnávací_operátor tabulka2*.*pole3*)];

Sadu příkazů JOIN můžete také použít v následující syntaxi:

SELECT *pole*
FROM *tabulka1* INNER JOIN

(*tabulka2* INNER JOIN [( ]*tabulka3*
[INNER JOIN [( ]*tabulkax* [INNER JOIN ...)]
ON *tabulka3*.*pole3 porovnávací_operátor tabulkax*.*polex*)]
ON *tabulka2*.*pole2 porovnávací_operátor tabulka3*.*pole3*)
ON *tabulka1*.*pole1 porovnávací_operátor tabulka2*.*pole2*;

Operace LEFT JOIN nebo RIGHT JOIN mohou být vřazeny dovnitř operace INNER JOIN, ale operace INNER JOIN nemůže být vřazena dovnitř operací LEFT JOIN nebo RIGHT JOIN.

Příkaz INSERT INTO

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlInsertIntoC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlInsertIntoX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlINSERTINTOS"}

Přidá jednoduchý nebo vícenásobný záznam do tabulky. Vztahuje se k spojovacímu dotazu.

Syntaxe

Spojovací dotaz pro vícenásobný záznam:

INSERT INTO *cílovémísto* [IN *externídatabáze*] [(*pole1*[, *pole2*[, ...]])]
SELECT [*zdrojovémísto*.]*pole1*[, *pole2*[, ...]
FROM *tabulkavýraz*

Spojovací dotaz pro jednoduchý záznam:

INSERT INTO *cílovémísto* [(*pole1*[, *pole2*[, ...]])]
VALUES (*hodnota1*[, *hodnota2*[, ...])

Příkaz INSERT INTO se skládá z těchto částí:

Část	Popis
cílovémísto	Název tabulky nebo dotazu, do kterých se mají přidat záznamy.
externídatabáze	Cesta k externí databázi. Více o popisu cesty viz klauzule IN.
zdrojovémísto	Název tabulky nebo dotazu, ze který se mají kopírovat záznamy.
pole1, *pole2*	Názvy polí, do kterých se mají vložit data, jestliže následuje argument *cílovémísto* nebo názvy polí, ze kterých se mají obdržet data, jestliže následuje argument *zdrojovémísto*.
tabulkavýraz	Název tabulky nebo tabulek, ze kterých mají být vloženy záznamy. Tento argument může být název jednotlivé tabulky nebo složené slovo vzniklé z operací INNER JOIN, LEFT JOIN nebo RIGHT JOIN nebo z uloženého dotazu.
hodnota1, *hodnota2*	Hodnoty ke vkládání do určených polí nového záznamu. Každá hodnota je vložena do určitého pole, které odpovídá pozici hodnoty v seznamu: *hodnota1* je vložena do pole *pole1* nového záznamu a *hodnota2* do *pole2*, atd. Hodnoty musíte od sebe oddělit čárkou a pole textu uzavřít do kulatých závorek (' ').

Poznámky

Příkaz INSERT INTO můžete používat k přidávání jednotlivého záznamu do tabulky s použitím syntaxe pro spojovací dotaz jednoduchého záznamu jak bylo ukázáno výše. V takovémto případě váš programový kód určuje název a hodnotu pro každé pole záznamu. Musíte určit každé pole záznamu, ke kterému má být přiřazena hodnota a hodnotu pro toto pole. Neurčíte-li každé pole, do chybějících sloupců je přiřazena výchozí hodnota nebo **prázdná** hodnota. Záznamy se přidávají na konec tabulky.

Příkaz INSERT INTO můžete také použít k připojení sady záznamů z jiné tabulky nebo dotazu použitím klauzule SELECT ... FROM jako je ukázáno výše pro syntaxi dotazu vícenásobného záznamu. V tomto případě klauzule SELECT určuje pole, která se mají připojit do tabulky *cílovémísto*.

Tabulka *zdrojovémísto* nebo *cílovémísto* může určovat tabulku nebo dotaz. Je-li určen dotaz

databázové jádro aplikace Microsoft Jet připojí záznamy do libovolné tabulky určené dotazem.

Příkaz INSERT INTO je volitelný, ale je-li začleněn, předchází příkazu SELECT.

Obsahuje-li cílová tabulka primární klíč, ujistěte se, zda pro primární klíč pole nebo polí vkládáte jedinečné hodnoty, které nejsou **prázdné**, protože databázové jádro aplikace Microsoft Jet takové záznamy jinak nepřipojí.

Jestliže připojujete záznamy do tabulky obsahující pole typu AutoNumber a chcete připojené záznamy přečíslovat, nazačleňujte do dotazu pole typu AutoNumber. Pole typu AutoNumber použijte v dotazu tehdy, když chcete zachovat původní hodnoty z tohoto pole.

Klauzuli IN použijte pro připojení záznamů do tabulky z jiné databáze.

K vytvoření nové tabulky použijte příkaz SELECT... INTO namísto vytváření dotazu pro vytváření tabulky.

K zobrazení záznamů, které budou připojeny před spuštěním spojovacího dotazu, spusťte nejprve výběrový dotaz, který používá stejná kritéria.

Spojovací dotaz kopíruje záznamy z jedné nebo více tabulek do dalších. Tabulky obsahující záznamy které budete připojovat nebudou spojovacím dotazem ovlivněny.

Namísto připojování existujících záznamů z jiné tabulky můžete určit hodnotu pro každé pole v novém záznamu pomocí klauzule VALUES. Vynecháte-li seznam polí, klauzule VALUES se pokusí začlenit hodnotu pro každé pole tabulky, což způsobí, že operace INSERT bude chybná. Pro každý přídavný záznam, který chcete vytvořit, použijte přídavný příkaz INSERT INTO s klauzulí VALUES.

Operace LEFT JOIN a RIGHT JOIN

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlLeftRightJoinC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlLeftRightJoinX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlLeftRightJoinS"}

Slučují záznamy ze zdrojové tabulky, jsou-li použity v nějaké klauzuli FROM.

Syntaxe

FROM *tabulka1* [LEFT | RIGHT] JOIN *tabulka2*
ON *tabulka1.pole1 porovnávací_operátor tabulka2.pole2*

Operace LEFT JOIN a RIGHT JOIN se skládají z těchto částí:

Část	Popis
tabulka1, *tabulka2*	Název tabulek, z nichž jsou slučovány záznamy.
pole1, *pole2*	Názvy sloučených polí. Tato pole musí být stejného datového typu a musí obsahovat stejný druh dat, ale nesmí mít stejný název.
porovnávací_operátor	Libovolný porovnávací relační operátor: "=", "<", ">", "<=", ">=" nebo "<>".

Poznámky

Operace LEFT JOIN se použije pro vytvoření vnějšího sloučení zleva. Vnější sloučení zleva zahrnuje všechny záznamy z první (levé) ze dvou tabulek a v případě, že zde nejsou obsaženy žádné vyhovující hodnoty, pak ze druhé (pravé) tabulky.

Operace RIGHT JOIN se použije pro vytvoření vnějšího sloučení zprava. Vnější sloučení zprava zahrnuje všechny záznamy ze druhé (pravé) ze dvou tabulek a v případě, že zde nejsou obsaženy žádné vyhovující hodnoty, pak z první (levé) tabulky.

Operaci LEFT JOIN můžete například použít pro tabulky Oddělení (pravá) a Zaměstnanci (levá) k výběru všech oddělení včetně těch, ke kterým nejsou přiřazeni žádní zaměstnanci. Pro výběr všech zaměstnanců včetně těch, kteří nejsou přiřazeni do žádného oddělení, použijete operaci RIGHT JOIN.

Následující příklad ukazuje jak můžete sloučit tabulky Kategorie a Výrobky přes pole IDKategorie. Dotaz vygeneruje seznam všech kategorií včetně těch, které neobsahují žádné výrobky:

```
SELECT NázevKategorie,
NázevVýrobku
FROM Kategorie LEFT JOIN Výrobky
ON Kategorie.IDKategorie = Výrobky.IDKategorie;
```

V tomto příkladu je IDKategorie připojené pole, ale není zahrnuto ve výsledcích dotazu, protože není zahrnuto v příkazu SELECT. Abyste jej do něj zahrnuli, zadejte název tohoto pole do příkazu SELECT, v tomto případě název `Kategorie.IDKategorie`.

Poznámky

- K vytvoření dotazu, který zahrnuje pouze ty záznamy, ve kterých jsou stejná data pro sloučená pole, použijete operaci INNER JOIN.
- Operace LEFT JOIN nebo RIGHT JOIN mohou být začleněny dovnitř operace INNER JOIN, ale operace INNER JOIN nemůže být začleněna dovnitř operací LEFT JOIN nebo RIGHT JOIN. Více se dozvíte v diskuzi tématu o začleňování v operaci INNER JOIN, kde můžete vidět jak se včleňují sloučení do jiných sloučení.
- Můžete slučovat vícenásobné klauzule ON. Více se dozvíte v diskuzi tématu o slučování klauzulí v operaci INNER JOIN, kde můžete vidět jak se toto provádí.

- Pokusíte-li se sloučit pole obsahující data typu Memo nebo Objekt OLE, nastane chyba.

Deklarace PARAMETERS

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlParametersC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlParametersX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlPARAMETERSS"}

Deklaruje názvy a datové typy každého parametru v parametrizovaném dotazu.

Syntaxe

PARAMETERS *název datovýtyp* [, *název datovýtyp* [, ...]]

Deklarace PARAMETERS se skládá z těchto částí:

Část	Popis
název	Název parametru. Přiřazuje vlastnost **Name** objektu **Parameter** a používá jej pro identifikaci tohoto parametru v kolekci **Parameters**. Pro *název* můžete použít řetězec, který se zobrazuje v dialogu jakmile aplikace spustí dotaz. K uzavření textu, který obsahuje mezery nebo interpunkci, použijte hranaté závorky ([]). Například [Nízká cena] a [Od kterého měsíce začít zprávu ?] jsou platné argumenty *název.*
datovýtyp	Jeden z primárních datových typů jazyka Microsoft Jet SQL nebo jejich synonyma.

Poznámky

Pro pravidelně spouštěné dotazy můžete k vytvoření parametrizovaného dotazu použít deklaraci PARAMETERS. Parametrizovaný dotaz může pomáhat tak, že zautomatizuje proces změny kritérií dotazu. Při práci s parametrizovaným dotazem bude váš programový kód při každém spuštění dotazu potřebovat zadávat parametry.

Deklarace PARAMETERS je volitelná, ale když je již obsažena v nějakém předchozím příkazu, včetně příkazu SELECT.

Obsahuje-li deklarace více než jeden parametr, oddělujte je od sebe čárkami. Následující příklad obsahuje dva parametry:

```
PARAMETERS [Sleva] Měna, [DatumOdeslání] DatumČas;
```

V klauzuli WHERE nebo HAVING můžete použít proměnnou *name*, ale nikoliv *datovýtyp*. V následujícím příkladu se předpokládá, že budou zadány dva parametry a potom budou použita kritéria do záznamů v tabulce Objednávky:

```
PARAMETERS [Sleva] Měna,
[DatumOdeslání] DatumČas;
SELECT IDObjednávky, Množství
FROM Objednávky
WHERE Množství > [Sleva]
AND DatumObjednávky >= [DatumOdeslání];
```

Klauzule PROCEDURE

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlProcedureC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlProcedureX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlPROCEDURES"}

Definuje název a volitelné parametry pro dotaz.

Syntaxe

PROCEDURE *název* [*param1 datovýtyp*[, *param2 datovýtyp*[, ...]]

Klauzule PROCEDURE se skládá z těchto částí:

Část	Popis
název	Název procedury. Musí následovat standardní pojmenovávací konvence.
param1, *param2*	Jedno nebo více polí názvů nebo parametrů. Například: `PROCEDURE ZeměPříjemce [DatumObjednávky] DatumČas, [DatumOdeslání] DatumČas;` O parametrech se více dozvíte pod heslem PARAMETERS.
datovýtyp	Jeden z primárních datových typů aplikace Microsoft Jet SQL nebo jejich synonym.

Poznámky

Procedura jazyka SQL se skládá z klauzule PROCEDURE, která určuje název procedury, z volitelného seznamu definic parametrů a jednoduchého příkazu jazyka SQL. Například procedura Dej_Číslo_Části spustí dotaz, který vyhledá číslo určité části.

Poznámky

- Obsahuje-li klauzule více než jednu definici pole (t. j. pár *param-datovýtyp*), oddělujte je od sebe čárkami.
- Klauzuli PROCEDURE musí následovat příkaz jazyka SQL jako například příkaz SELECT nebo UPDATE.

Příkaz SELECT...INTO

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlSelectIntoC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlSelectIntoX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlSelectIntoS"}

Vytváří vytvářecí tabulkový dotaz.

Syntaxe

SELECT *pole1*[, *pole2*[, ...]] INTO *novátabulka* [IN *externídatabáze*]
FROM *zdrojovémísto*

Příkaz SELECT...INTO se skládá z těchto částí:

Část	Popis
pole1, *pole2*	Názvy polí, které se mají kopírovat do nové tabulky.
novátabulka	Název tabulky, která má být vytvořena. Musí se řídit standardními pojmenovávacími konvencemi. Má-li *novátabulka* stejný název jako nějaká existující, nastane chyba přepsání tabulky.
externídatabáze	Cesta k externí databázi. O zadávání cesty se více dozvíte v popisu klauzule IN.
zdrojovémísto	Název existující tabulky, ze které jsou vybírány záznamy. Může to být jednoduchá nebo vícenásobná tabulka nebo dotaz.

Poznámky

Vytvářecí tabulkové dotazy můžete používat k archivaci záznamů, vytváření záložních kopií, vytváření kopií pro export do jiných databází nebo je používat jako základ pro zprávy, které budou zobrazovat data po jistých časových obdobích. Můžete například vytvořit zprávu Měsíční prodej podle oblastí tak, že každý měsíc spustíte stejný vytvářecí tabulkový dotaz.

Poznámky

- Pro novou tabulku můžete chtít definovat primární klíč . Když vytvoříte novou tabulku, pak její pole dědí datový typ a velikost, a to pro všechny tabulky podléhající určitému dotazu, ale žádná další pole ani vlastnosti se nepřenáší.
- K přidávání dat do existující tabulky použijte příkaz INSERT INTO namísto vytváření přidávacího dotazu.
- Chcete-li vědět které dotazy budou vybrány před tím než spustíte vytvářecí tabulkový dotaz, nejprve použijte k vyzkoušení příkaz SELECT, který používá stejná kritéria a potom teprve spusťte tabulkový vytvářecí dotaz.

Poddotazy jazyka SQL

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlSubqueriesC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝dž˝klad":"dasqlSubqueriesX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlSubqueriesS"}

Poddotaz je příkaz SELECT včleněný mezi příkazy SELECT, SELECT...INTO, INSERT...INTO, DELETE nebo UPDATE nebo mezi další poddotazy.

Syntaxe

Pro vytváření poddotazů můžete použít tři tvary syntaxe:

porovnání [ANY | ALL | SOME] (*sqlpříkaz*)
výraz [NOT] IN (*sqlpříkaz*)
[NOT] EXISTS (*sqlpříkaz*)

Poddotaz se skládá z těchto částí:

Část	Popis
porovnání	Výraz a porovnávací operátor, který porovnává výraz s výsledky poddotazu.
výraz	Výraz, pro který je vyhledávána sada výsledků poddotazu.
sqlpříkaz	Příkaz SELECT se stejným formátem a pravidly jako každý jiný příkaz SELECT. Musí být uzavřen v kulatých závorkách.

Poznámky

Poddotaz můžete použít namísto některého výrazu v seznamu polí příkazu SELECT nebo klauzule WHERE nebo HAVING. V poddotazu použijte příkaz SELECT k nastavení jedné nebo více určujících hodnot pro vyhodnocení výrazu v klauzuli WHERE nebo HAVING.

Predikáty ANY nebo SOME, která jsou synonyma, použijte pro vyhledávání těch záznamů v hlavním dotazu, které vyhovují porovnání s některými záznamy vyhledanými v poddotazu. Následující příklad vrací všechny výrobky, jejichž jednotková cena je vyšší než u některých výrobků prodaných s 25% nebo vyšší slevou:

```
SELECT * FROM Výrobky
WHERE JenotkováCena > ANY
(SELECT JenotkováCena FROM RozpisObjednávek
WHERE Sleva >= .25);
```

Predikát ALL použijte pro vyhledávání pouze těch záznamů v hlavním dotazu, které vyhovují porovnání se všemi vyhledanými záznamy v poddotazu. Jestliže by se v předešlém příkladu zaměnil predikát ANY za ALL, dotaz by vrátil pouze ty výrobky, jejichž jednotková cena je větší než u všech výrobků prodaných s 25% nebo větší slevou.

Predikát IN použijte pro vyhledávání pouze těch záznamů v hlavním dotazu, pro které některé záznamy v poddotazu obsahují stejnou hodnotu. Následující příklad vrací všechny výrobky s 25% nebo větší slevou:

```
SELECT * FROM Výrobky
WHERE IDVýrobku IN
(SELECT IDVýrobku FROM RozpisObjednávek
WHERE Sleva >= .25);
```

Jinými slovy, predikát NOT IN nemůžete použít pro vyhledávání těch záznamů v hlavním dotazu, pro které nebsahuje žádný záznam v poddotazu stejnou hodnotu.

Predikát EXISTS použijte (s volitelným rezervovaným slovem NOT) v porovnáních typu true/false k

určení skutečnosti, zda poddotaz vrátil nějaké výsledky.

V poddotazu můžete také použít zástupce názvu tabulky pro odkaz do tabulek uvedených v seznamu klauzule FROM mimo poddotaz. Následující příklad vrátí jména zaměstnanců, kteří mají větší nebo rovný plat vzhledem k průměrnému platu všech zaměstnanců majících stejnou funkci. Tabulce Zaměstnanci je dán zástupce T1:

```
SELECT Příjmení,
Jméno, Funkce, Plat
FROM Zaměstnanci AS T1
WHERE Plat >=
(SELECT Avg(Plat)
FROM Zaměstnanci
WHERE T1.Funkce = Zaměstnanci.Funkce) Order by Funkce;
```

V předchozím příkladu je rezervované slovo AS volitelné.

Některé poddotazy jsou povoleny v křížových dotazech jako predikáty (v klauzuli WHERE). Poddotazy jako výstupy (v seznamu příkazu SELECT) are v křížových dotazech povoleny nejsou.

Příkaz TRANSFORM

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlTransformC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlTransformX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlTransformS"}

Vytváří křížový dotaz.

Syntaxe

TRANSFORM *agregfunkce*
vybratpříkaz
PIVOT *kontingenčnípole* [IN (*hodnota1*[, *hodnota2*[, ...]])]

Příkaz TRANSFORM se skládá z těchto částí:

Část	Popis
agregfunkce	Libovolná agregační funkce jazyka SQL, která pracuje s vybranými daty.
vybratpříkaz	Příkaz SELECT.
kontingenčnípole	Pole nebo výraz, které chcete použít k vytvoření záhlaví sloupců ve výsledkové sadě dotazu.
hodnota1, *hodnota2*	Pevná hodnota použitá k vytvoření záhlaví sloupců.

Poznámky

Děláte-li souhrn dat pomocí křížového dotazu, vybíráte hodnoty z určitých polí nebo výrazů jako záhlaví sloupců, kde potom můžete data vidět v daleko kompaktnějším tvaru než při použití výběrového dotazu.

Příkaz TRANSFORM je volitelný tehdy, když je obsažen v prvním příkazu v řetězci jazyka SQL.Předchází příkazu SELECT, který určuje pole použitá jako záhlaví řádků a klauzuli GROUP BY, která určuje řádkove seskupení. Volitelně můžete zahrnout další klauzule jako například WHERE, která určuje přídavný výběr nebo seřazovací kritéria. Také můžete použít poddotazy jako predikáty specielně v klauzuli WHERE v křížovém dotazu.

Hodnoty vrácené v proměnné *kontingenčnípole* jsou použity jako záhlaví sloupců ve výsledkové sadě dotazu. Například kontingencí množství prodeje v měsíci prodeje v křížovém dotazu by měl křížový dotaz vytvořit 12 sloupců. Porměnnou *kontingenčnípole* můžete omezit tak, že vytvoříte záhlaví z pevně daných hodnot (*hodnota1*, *hodnota2*) uvedených v seznamu volitelné klauzule IN. Také můžete pevné hodnoty zahrnout pro neexistující data, aby se vytvořily přídavné sloupce.

Operace UNION

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlUnionC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlUnionX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlUNIONS"}

Vytváří sjednocovací dotazy, které slučují výsledky dvou či více nezávislých dotazů nebo tabulek.

Syntaxe

[TABULKA] *dotaz1* UNION [ALL] [TABULKA] *dotaz2* [UNION [ALL] [TABULKA] *dotazn* [...]]

Operace UNION se skládá z těchto částí

Část	Popis
dotaz1-n	Příkaz SELECT, název uloženého dotazu nebo tabulky, kterému předchází klíčové slovo TABLE.

Poznámky

V jednoduché operaci UNION můžete sloučit výsledky dvou dotazů, tabulek a příkazů STATEMENT, a to v libovolné kombinaci. Následující příklad ukazuje sloučení názvu existující tabulky a příkazu SELECT:

```
TABLE [NovéMnožství] UNION ALL
SELECT *
FROM Zákazníci
WHERE ObjednanéMnožství > 1000;
```

Je-li použito operace UNION, nevrací se duplicitní záznamy. Samozřejmě však můžete použít predikát ALL, k zajištění návratu všech záznamů. Toto také zrychluje práci dotazu.

Všechny dotazy v operaci UNION vyžadují stejný počet polí a samozřejmě pole nesmí být stejné velikosti a datového typu.

Použijte zástupce pouze v prvním příkazu SELECT, protože jinde by byly ignorováni. V klauzuli ORDER BY odkazujte na pole, kterými jsou voláni v prvním příkazu STATEMENT.

Poznámky

- Můžete použít klauzule GROUP BY nebo HAVING v každém v každém argumentu *dotaz* pro seskupení vracených dat.
- Můžete použít klauzuli ORDER BY na konci posledního argumentu *dotaz* pro zobrazení vracených dat v určeném pořadí.

Příkaz UPDATE

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlUpdateC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlUpdateX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlUPDATES"}

Vytváří aktualizační dotaz, který mění hodnoty v polích určené tabulky podle zadaných kritérií.

Syntaxe

UPDATE *tabulka*
SET *nová hodnota*
WHERE *kritéria*;

Příkaz UPDATE se skládá z těchto částí:

Část	Popis
tabulka	Název tabulky obsahující data, která chcete upravovat.
nová hodnota	Výraz, který určuje hodnotu, která má být vložena do určitého pole v aktualizovaném záznamu.
kritéria	Výraz, který určuje, které záznamy budou aktualizovány. Aktualizují se pouze ty záznamy, které vyhoví danému výrazu.

Poznámky

Příkaz UPDATE je zvláště výhodný, chcete-li změnit větší množství záznamů ve vícenásobných tabulkách.

Během stejného okamžiku můžete změnit několik polí. Následující příklad zvyšuje hodnotu pole Objednané množství o 10 % a hodnotu pole Dopravné o 3 % pro dodavatele ve Spojeném království:

```
UPDATE Objednávky
SET ObjednanéMnožství = ObjednanéMnožství * 1.1,
Dopravné = Dopravné * 1.03
WHERE ZeměDodavatele = 'UK';
```

Důležité

- Příkaz UPDATE negeneruje výsledkovou sadu a z toho vyplývá, že poté co aktualizujete záznamy aktualizačním dotazem, nelze vzít tuto akci zpět. Chcete-li vědět, které záznamy budou aktualizovány, použijte pro kontrolu nejdříve výběrový dotaz, který používá stejná kritéria a potom teprve spusťte aktualizační dotaz.
- Vytvářejte si neustále záložní kopie svých dat. Jestliže aktualizujete nesprávná data, můžete je obnovit ze záložní kopie.

Deklarace WITH OWNERACCESS OPTION

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlWithOwnerAccessOptionC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlWithOwnerAccessOptionX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlWITHOWNERACCESSOptionS"}

Ve víceuživatelském prostředí se zabezpečením pracovní skupiny, použijte tuto deklaraci společně s dotazem, který dává uživatelům používajícím nějaký dotaz stejná přístupová práva jako vlastníkovi dotazu.

Syntaxe

sqlpříkaz
WITH OWNERACCESS OPTION

Poznámky

Deklarace WITH OWNERACCESS OPTION je volitelná.

Následující příklad umožňuje uživateli prohlížet informace o platech (dokonce i když vlastník dotazu nemá přístupová práva pro prohlížení tabulky Payroll), což mu umožňují stejná přístupová práva jaká má vlastník dotazu:

```
SELECT Příjmení,
Jméno, Plat
FROM Zaměstnanci
ORDER BY Příjmení
WITH OWNERACCESS OPTION;
```

Jestliže je uživateli zabráněno vytvářet tabulku nebo do ní přidávat data, můžete použít deklaraci WITH OWNERACCESS OPTION, abyste uživateli umožnili spustit vytvářecí tabulkový nebo spojovací dotaz.

Jestliže chcete zvýšit nastavení bezpečnosti pracovní skupiny a uživatelská přístupová práva, nezačleňujte deklaraci WITH OWNERACCESS OPTION.

Tato volba vyžaduje, abyste měli přístup do souboru System.mdw, který je přidružen k databázi. Tato deklarace je opravdu užitečná pouze ve víceuživatelských bezpečnostních implementacích.

Agregační funkce jazyka SQL

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"daidxAggregateFunctionsC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"daidxAggregateFunctionsX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"daidxAggregateFunctionsS"}

Použitím agregačních funkcí jazyka SQL můžete určovat různé statistické údaje ze sad hodnot. Tyto funkce můžete také použít v dotazu a v agregačních výrazech ve vlastnosti **SQL** objektu **QueryDef** nebo při vytváření objektu **Recordset**, který je založen na dotazu jazyka SQL.

Funkce Avg

Funkce Count

Funkce Min a Max

Funkce StDev a StDevP

Funkce Sum

Funkce Var a VarP

Funkce Avg

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlAvgC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlAvgX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlAvgS"}

Vypočítá aritmetický průměr ze sady hodnot obsažené v určeném poli dotazu.

Syntaxe

Avg(*výraz***)**

Poziční značka *výraz* představuje řetězcový výraz určující pole obsahující číselná data, ze kterých chcete vypočítat průměr anebo výraz, který provádí výpočet použitím dat v tomto poli. Operandy v proměnné *výraz* mohou obsahovat název pole tabulky, konstantu nebo funkci, která může být buď zabudovaná nebo uživatelsky definovaná, ale nemůže to být jiná agregační funkce jazyka SQL.

Poznámky

Průměr počítaný funkcí **Avg** je aritmetický (součet hodnot dělený jejich počtem). Funkci **Avg** můžete použít například pro výpočet průměrné ceny za dopravu.

Funkce **Avg** nezahrnuje do výpočtu žádná pole typu **Null**.

Funkci **Avg** můžete také použít ve výrazu v dotazu a ve vlastnosti **SQL** objektu **QueryDef** nebo při vytváření objektu **Recordset**, založeném na dotazu jazyka SQL.

Funkce Count

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlCountC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlCountX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlCountS"}

Spočítá počet záznamů vrácený dotazem.

Syntaxe

Count(*výraz***)**

Poziční značka *výraz* představuje řetězcový výraz, který určuje pole obsahující číselná data, ze kterých chcete vypočítat průměr, anebo výraz provádějící výpočet použitím dat v tomto poli. Operandy v proměnné *výraz* mohou obsahovat název pole tabulky, konstantu nebo funkci, která může být buď zabudovaná nebo uživatelsky definovaná, ale nemůže to být další agregační funkce jazyka SQL.

Poznámky

Funkci **Count** můžete použít k výpočtu počtu záznamů v příslušném dotazu. Funkci **Count** můžete použít například k výpočtu počtu objednávek odeslaných do určité země.

Ačkoli proměnná *výraz* může provádět výpočet s polem, funkce **Count** jednoduše sečte počet záznamů, a přitom nezáleží na tom, jaké hodnoty jsou v záznamech uloženy.

Funkce **Count** nezahrnuje do počtu záznamy obsahující pole typu **Null**, jestliže proměnná *výraz* není hvězdička * představující zástupný znak. Použijete-li hvězdičku, potom funkce **Count** vypočítá celkový počet záznamů, včetně těch, které obsahují pole typu Null. Zápis funkce **Count(*)** je značně rychlejší než zápis **Count(**[*Jméno sloupce*]**)**. Hvězdičku neuzavírejte do uvozovek (' '). Následující příklad počítá počet záznamů v tabulce Objednávky:

```
SELECT Count(*)
AS ObjednávekCelkem FROM Orders;
```

Jestliže proměnná *výraz* určuje vícenásobné pole, funkce **Count** sečte pouze ty záznamy, které obsahují alespoň jedno pole, které není typu Null. Jsou-li všechna určená pole typu Null, není do počtu zahrnut žádný záznam. Názvy polí se oddělují ampersandem &. Následující příklad ukazuje, jak můžete omezit počet záznamů, ve kterých buď pole Datum odeslání nebo Dopravné není typu Null:

```
SELECT
Count('DatumOdeslání & Dopravné')
AS [Not Null] FROM Objednávky;
```

Funkci **Count** můžete použít ve výrazu dotazu. Tento výraz také můžete použít ve vlastnosti **SQL** objektu **QueryDef** nebo při vytváření objektu **Recordset**, který je založen na dotazu jazyka SQL.

Funkce First a Last

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlFirstLastC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝dž˝klad":"dasqlFirstLastX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlFirstLastS"}

Vrací hodnotu pole z prvního nebo posledního záznamu ve výsledné množině získané dotazem.

Syntaxe

First(*výraz***)**
Last(*výraz***)**

Poziční značka *výraz* představuje řetězcový výraz, který určuje pole obsahující číselná data, ze kterých chcete vypočítat průměr nebo výraz provádějící výpočet s použitím dat v tomto poli. Operandy v proměnné *výraz* mohou obsahovat název pole tabulky, konstantu nebo funkci, která může být buď zabudovaná nebo uživatelsky definovaná, ale nemůže to být další agregační funkce jazyka SQL.

Poznámky

Funkce **First** a **Last** jsou analogické s metodami **MoveFirst** a **MoveLast** objektu **Recordset** typu DAO. Obě funkce jednoduše vracejí hodnotu určeného pole v prvním nebo posledním záznamu respektive hodnotu výsledné množiny vrácené dotazem. Protože záznamy obvykle nejsou vraceny v určitém pořadí (pokud dotaz neobsahuje klauzuli ORDER BY), budou záznamy těmito funkcemi vraceny nahodile.

Funkce Min a Max

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlMinMaxC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlMinMaxX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlMinMaxS"}

Vrací minimum nebo maximum ze sady hodnot obsažené v určeném poli dotazu.

Syntaxe

Min(*výraz***)**
Max(*výraz***)**

Poziční značka *výraz* představuje řetězcový výraz, který určuje pole obsahující číselná data, ze kterých chcete vypočítat průměr nebo výraz provádějící výpočet použitím dat v tomto poli. Operandy v proměnné *výraz* mohou obsahovat název pole tabulky, konstantu nebo funkci, která může být buď zabudovaná nebo uživatelsky definovaná, ale nemůže to být další agregační funkce jazyka SQL.

Poznámky

Funkce **Min** a **Max** můžete použít k určení nejmenší a největší hodnoty v poli založeném na určité agregaci nebo seskupení. Tyto funkce můžete například použít pro návrat nejnižší a nejvyšší hodnoty dopravného. Není-li určená žádná agregace, pak funkce pracují s celou tabulkou.

Funkce **Min** a **Max** můžete použít ve výrazu dotazu a ve vlastnosti **SQL** objektu **QueryDef** nebo při vytváření objektu **Recordset**, který je založen na dotazu jazyka SQL.

Funkce StDev a StDevP

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlStDevPC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlStDevPX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlStDevPS"}

Vrací předpokládanou směrodatnou odchylku pro populaci nebo výběr z populace představovaný sadou hodnot v určeném poli dotazu.

Syntaxe

StDev(*výraz***)**
StDevP(*výraz***)**

Poziční značka *výraz* představuje řetězcový výraz, který určuje pole obsahující číselná data, ze kterých chcete vypočítat průměr nebo výraz provádějící výpočet použitím dat v tomto poli. Operandy v proměnné *výraz* mohou obsahovat název pole tabulky, konstantu nebo funkci, která může být buď zabudovaná nebo uživatelsky definovaná, ale nemůže to být další agregační funkce jazyka SQL.

Poznámky

Funkce **StDevP** vyhodnocuje populaci, zatímco funkce **StDev** vyhodnocuje výběr z populace.

Jestliže příslušný dotaz obsahuje méně než dva záznamy (nebo žádný záznam v případě funkce **StDevP**), pak tyto funkce vrátí hodnotu typu **Null**, která naznačuje, že standardní odchylku není možné počítat.

Funkce **StDev** a **StDevP** můžete použít ve výrazu dotazu a ve vlastnosti **SQL** objektu **QueryDef** nebo při vytváření objektu **Recordset**, který je založen na dotazu jazyka SQL.

Funkce Sum

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlSumC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlSumX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlSumS"}

Vrací součet ze sady hodnot obsažené v určeném poli dotazu.

Syntaxe

Sum(*výraz***)**

Poziční značka *výraz* představuje řetězcový výraz, který určuje pole obsahující číselná data, ze kterých chcete vypočítat průměr nebo výraz provádějící výpočet použitím dat v tomto poli. Operandy v proměnné *výraz* mohou obsahovat název pole tabulky, konstantu nebo funkci, která může být buď zabudovaná nebo uživatelsky definovaná, ale nemůže to být další agregační funkce jazyka SQL.

Poznámky

Funkce **Sum** zahrnuje všechny hodnoty v poli. Funkci **Sum** můžete použít například pro výpočet celkové ceny za dopravu.

Funkce **Sum** ignoruje záznamy obsahující pole typu Null. Následující příklad ukazuje jak se spočítá součet součinů polí Jednotková cena a Množství:

```
SELECT
Sum(JednotkováCena * Množství)
AS [Celkové příjmy] FROM [Rozpis objednávek];
```

Funkci **Count** můžete použít ve výrazu dotazu. Tento výraz také můžete použít ve vlastnosti **SQL** objektu **QueryDef** nebo při vytváření objektu **Recordset**, který je založen na dotazu jazyka SQL.

Funkce Var a VarP

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlVarPC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlVarPX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlVarPS"}

Vrací předpokládaný rozptyl pro populaci nebo výběr z populace představovanou sadou hodnot v určeném poli dotazu.

Syntaxe

Var(*výraz***)**
VarP(*výraz***)**

Poziční značka *výraz* představuje řetězcový výraz, který určuje pole obsahující číselná data, ze kterých chcete vypočítat průměr nebo výraz, provádějící výpočet použitím dat v tomto poli. Operandy v proměnné *výraz* mohou obsahovat název pole tabulky, konstantu nebo funkci, která může být buď zabudovaná nebo uživatelsky definovaná, ale nemůže to být další agregační funkce jazyka SQL.

Poznámky

Funkce **VarP** vyhodnocuje populaci, zatímco funkce **Var** vyhodnocuje výběr z populace.

Jestliže příslušný dotaz obsahuje méně než dva záznamy, pak funkce **Var** a **VarP** vrátí hodnotu typu Null, která naznačuje, že rozptyl nelze počítat.

Funkce **Var** a **VarP** můžete použít ve výrazu dotazu nebo v příkazu jazyka SQL.

Výpočet polí v SQL funkcích

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlCalculatingFieldsC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlCalculatingFieldsX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"daSQLCalculatingFieldsS"}

Jako argument v agregační funkci jazyka SQL můžete použít řetězcový výraz, k provedení výpočetu s hodnotami v poli. Můžete například spočítat počet procent (například penále nebo daň z přidané hodnoty) násobením hodnoty pole a zlomku.

Následující tabulka ukazuje příklad výpočtů s poli tabulek Objednávky a Rozpis objednávek databáze Northwind.mdb.

Výpočet	**Příklad**
Přičtení počtu do poli	`Dopravné + 5`
Odečtení počtu z pole	`Dopravné - 5`
Násobení pole počtem	`JednotkováCena * 2`
Dělení pole počtem	`Dopravné / 2`
Přičtení jednoho pole k druhému	`JednotekNaSkladě + ObjednánoJednotek`
Odečtení jednoho pole od druhého	`MinimálníÚroveň - JednotekNaSkladě`

Následující příklad počítá průměrnou slevu pro všechny objednávky z databáze Northwind.mdb. Násobí se hodnoty v polích Jednotková cena a Sleva, aby se určila velikost slevy pro každou objednávku zvlášť a potom se provedl výpočet průměru. Tento výraz můžete použít v příkazu jazyka SQL v programovém kódu jazyka Visual Basic:

```
SELECT Avg(JednotkováCena * Discount) AS [PrůměrnáSleva] FROM
[RozpisObjednávek];
```

Operátor Between...And

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlBETWEENC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlBetweenX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlBetweenS"}

Vyhodnocuje, zda hodnota výrazu spadá do vymezeného rozsahu.Tento operátor můžete použít v příkazech jazyka SQL.

Syntaxe

výraz [**Not**] **Between** *hodnota1* **And** *hodnota2*

Syntaxe operátoru **Between...And** obsahuje tyto tři části:

Část	Popis
výraz	Výraz, který určuje pole obsahující data, které chcete vyhodnotit.
hodnota1, *hodnota2*	Hodnoty, pro které chcete vyhodnotit *výraz*.

Poznámky

Spadá-li hodnota proměnné *výraz* do rozmezí hodnot *hodnota1* a *hodnota2* (včetně), pak operátor **Between...And** vrátí hodnotu **True**, v opačném případě hodnotu **False**. Logický operátor **Not** můžete použít k vyhodnocení opačné podmínky (t. j. když proměnná *výraz* leží mimo rozsah daný hodnotami *hodnota1* a *hodnota2)*.

Operátor **Between...And** byste měli používat k vyhodnocování skutečnosti, zda daná hodnota spadá do vymezeného rozsahu. Následující přípklad určuje, zda byla objednávka zaslána do oblasti určené jistým rozsahem PSČ. Leží-li hodnota PSČ v rozsahu 98000 až 98999, pak jej funkce **IIf** vyhodnotí jako Oblastní, v opačném případě jako Mimooblastní.

```
SELECT IIf(PSC Between 98101 And 98199, " Oblastní", " Mimoblastní")
FROM Vydavatelé
```

Je-li některá z hodnot *výraz*, *hodnota1* nebo *hodnota2* typu Null, potom operátor Between...And vrátí hodnotu Null**.**

Protože zástupné znaky jako je (*) jsou považovány za literály, nelze je používat ve spojení s operátorem **Between...And**. Nelze například použít zápisu 980* a 989* pro vyhledání všech PSČ od 980 do 989. Existují dva náhradní způsoby, kterými je to možné provést. Buď můžete přidat do dotazu výraz, který obsahuje tři znaky textového pole zleva a použít operátor **Between...And** s těmito znaky, nebo můžete vyznačit horní a dolní hodnotu se zvláštními znaky, v tomto případě 98000 až 98999, nebo 98000 až 98999-9999, používáte-li dlouhý formát PSČ. (Pro dolní hodnotu musíte vypustit část -0000, protože některá PSČ mají dlouhý formát a jiná nikoliv, a potom by bylo číslo 98000 opomenuto).

Porovnání databázových jader aplikací Microsoft Jet SQL a ANSI SQL

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlJetVANSIC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"daSQLJetVANSIX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"daSQLJetVANSIS"}

Jádro aplikace Microsoft Jet SQL obecně odpovídá standardu ANSI-89 Level 1. Ovšem určité rysy ANSI SQL v aplikaci Microsoft Jet SQL implementovány nejsou. A naopak, aplikace Microsoft Jet SQL obsahuje rezervovaná slova, která nejsou podporována v ANSI SQL.

Základní rozdíly

- Obě aplikace Microsoft Jet SQL i ANSI SQL mají různá rezervovaná slova a datové typy. Více se dozvíte pod hesly Více informací, viz Rezervovaná slova databázového jádra aplikace Microsoft Jet SQL a Ekvivalenty datových typů aplikace ANSI SQL.
- Používají různá pravidla pro konstrukce s operátorem **Between...And**, který má následující syntaxi:

  výraz1 [NOT] **Between** *hodnota1* **And** *hodnota2*

  V aplikaci Microsoft Jet SQL může být proměnná *hodnota1* větší než proměnná *hodnota2,* ale v aplikaci ANSI SQL musí být proměnná *hodnota1* rovna nebo menší než proměnná *hodnota2.*
- Ve spojení s operátorem **Like** se používají různé zástupné znaky.

Shodný znak	Microsoft Jet SQL	ANSI SQL
Libovolný jednoduchý znak	?	_ (znak podtržení)
Nula či více znaků	*	%

- Aplikace Microsoft Jet SQL má obecně méně omezení. Povoluje například seskupování a řazení výrazů.
- Aplikace Microsoft Jet SQL disponuje více výkonnými výrazy.

Vylepšené funkce aplikace Microsoft Jet SQL

Aplikace Microsoft Jet SQL nabízí následující vylepšené funkce:

- Příkaz TRANSFORM, který umožňuje práci s křížovými dotazy
- Přídavné agregační funkce jako například **StDev** a **VarP**
- Deklaraci PARAMETERS pro definici parametrizovaných dotazů

Prvky standardu ANSI SQL nepodporované aplikací Microsoft Jet SQL

Aplikace Microsoft Jet SQL nepodporuje s následujícími prvky standardu ANSI SQL:

- Příkazy pro zabezpečení jako COMMIT, GRANT a LOCK.
- Použití DISTINCT v odkazu na agregační funkce. Aplikace Microsoft Jet SQL například nepovoluje užití SUM(DISTINCT *jménosloupce*).
- Použití klauzule LIMIT TO *nn* ROWS pro vymezení počtu řádků vracených dotazem. Klauzuli WHERE můžete použít pouze k omezení rozsahu dotazu.

Ekvivalentní datové typy standardu ANSI SQL

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlEquivalentDataTypesC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlEquivalentDataTypesX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlEquivalentDataTypesS"}

Následující tabulkové seznamy zobrazují datové typy standardu ANSI SQL a jejich ekvivalenty v databázovém jádru Microsoft Jet SQL a jejich platná synonyma.

Datové typy ANSI SQL	Datové typy Microsoft Jet SQL	Synonyma
BIT, BIT VARYING	BINARY (Viz poznámky)	VARBINARY
Není podporován	BIT (Viz poznámky)	BOOLEAN, LOGICAL, LOGICAL1, YESNO
Není podporován	BYTE	INTEGER1
Není podporován	COUNTER	AUTOINCREMENT
Není podporován	CURRENCY	MONEY
DATE, TIME, TIMESTAMP	DATETIME	DATE, TIME, TIMESTAMP
Není podporován	GUID	
DECIMAL	Není podporován	
REAL	SINGLE	FLOAT4, IEEESINGLE, REAL
DOUBLE PRECISION, FLOAT	DOUBLE	FLOAT, FLOAT8, IEEEDOUBLE, NUMBER, NUMERIC
SMALLINT	SHORT	INTEGER2, SMALLINT
INTEGER	LONG	INT, INTEGER, INTEGER4
INTERVAL	Není podporován	
Není podporován	LONGBINARY	GENERAL, OLEOBJECT
Není podporován	LONGTEXT	LONGCHAR, MEMO, POZNÁMKA
CHARACTER, CHARACTER VARYING	TEXT	ALPHANUMERIC, CHAR, CHARACTER, STRING, VARCHAR
Není podporován	VALUE (Viz poznámky)	

Poznámky

- Datový typ BIT aplikace ANSI SQL neodpovídá datovému typu BIT aplikce Microsoft Jet SQL, ale odpovídá namísto toho datovému typu BINARY. Pro datový typ BIT aplikace Microsoft Jet

neexistuje žádný ekvivalent v ANSI SQL.

- Rezervované slovo VALUE nepředstavuje v jádru databáze Microsoft Jet datový typ.

Operátor In

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlInOperC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlInOperX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlInOperS"}

Určuje, zda se hodnota daného výrazu rovná některým hodnotám určeného seznamu.

Syntaxe

výraz [**Not**] **In(***hodnota1, hodnota2, . . .***)**

Poznámky

Syntaxe operátoru **In** se skládá ze tří částí:

Část	Popis
výraz	Výraz, který určuje pole obsahující data, které chcete vyhodnotit.
hodnota1, *hodnota2*	Výraz nebo seznam výrazů, pro které chcete vyhodnotit proměnnou *výraz*.

Je-li proměnná *výraz* nalezena v seznamu hodnot, operátor **In** vrátí hodnotu **True**, v opačném případě hodnotu **False**. Logický operátor **Not** můžete použít pro vyhodnocení opačné podmínky (t. j. zda proměnná není *výraz* obsažena v seznamu hodnot).

Operátor **In** můžete použít například k zjištění určitých oblastí, do kterých byly odeslány objednávky:

```
SELECT *
FROM Objednávky
WHERE OblastOdeslání In ('Praha','Brno','Ostrava')
```

Operátor Like

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlLikeC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlLikeX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlLikeS"}

Srovnává řetězcový výraz se vzorkem v SQL výrazu.

Syntaxe

výraz **Like** "*vzorek*"

Syntaxe operátoru **Like** se skládá ze tří částí:

Part	Popis
výraz	výraz aplikace SQL užitý v klauzuli WHERE.
vzorek	Řetězec nebo znakový řetězec literálů, se kterými chceme *výraz* porovnávat.

Poznámky

Operátor **Like** můžete použít k nalezení hodnot v poli, které odpovídá vzorku, který jste zadali. *Vzorek* můžete zadat buď celý (například `Like Novotný` nebo pomocí zástupných znaků pro nalezení rozsahu hodnot (například `Like No*`).

Ve výrazu můžete použít operátor **Like** pro porovnání hodnoty pole s řetězcovým výrazem. Zadáte-li například v dotaze jazyka SQL `Like C*`, dotaz vrátí všechny hodnoty polí začínajících písmenem C. V parametrizovaném dotazu můžete uživatele vyzvat k zadání vzorku pro vyhledávání.

V následujícím příkladu budou vrácena všechna data začínající písmenem P následovaným libovolným písmenem v rozsahu A až F a třemi číslicemi:

```
Like "P[A-F]###"
```

Následující tabulka ukazuje, jak můžete použít operátor Like k testování výrazů pro různé vzorky:

Typ shody	Vzorek	Shoda (vrací hodnotu True)	Neshoda (vrací hodnotu False)
Vícenásobné znaky	a*a	aa, aBa, aBBBa	aBC
	ab	abc, AABB, Xab	aZb, bac
Speciální znak	a[*]a	a*a	aaa
Vícenásobné znaky	ab*	abcdefg, abc	cab, aab
Jeden znak	a?a	aaa, a3a, aBa	aBBBa
Jedna číslice	a#a	a0a, a1a, a2a	aaa, a10a
Rozsah znaků	[a-z]	f, p, j	2, &
Mimo rozsah	[!a-z]	9, &, %	b, a
Nečíselný výraz	[!0-9]	A, a, &, ~	0, 1, 9
Kombinovaná	a[!b-m]#	An9, az0, a99	abc, aj0

Datové typy aplikace Microsoft Jet SQL

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlJetDataTypesC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"daSQLJetDataTypesX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"daSQLJetDataTypesS"}

Databázové jádro aplikace Microsoft Jet SQL sestává z třinácti základních datových typů definovaných aplikací Microsoft Jet a několika platných synonym, která jsou pro tyto datové typy rozpoznávána.

Následující tabulka zobrazuje základní datové typy. Synonyma jsou definována v rezervovaných slovech databázového jádra aplikace Microsoft Jet SQL.

Datový typ	Ukládaná velikost	Popis
BINARY	1 bajt / znak	V poli tohoto typu může být uložen libovolný datový typ. Tato data se nepřekládají. Vstupní data odpovídají výstupním.
BIT	1 bajt	Hodnoty Ano a Ne a pole, která obsahují pouze jednu z těchto hodnot.
BYTE	1 bajt	Celočíselná hodnota v rozmezí od 0 do 255.
COUNTER	4 bajty	Počet je aplikací Microsoft Jet automaticky zvyšován poté, kdy je do tabulky přidán nový záznam. V databázovém jádru aplikace Microsoft Jet je datový typ pro tuto hodnotu **Long**.
CURRENCY	8 bajtů	Celočíselné hodnoty přepočítané na rozsah – 922 337 203 685 477,5808 až 922 337 203 685 477,5807.
DATETIME (Viz DOUBLE)	8 bajtů	Datová nebo časová hodnota mezi roky 100 a 9 999.
GUID	128 bitů	Jedinečné identifikační číslo používané při dálkovém volání procedury
SINGLE	4 bajty	Hodnota s jednoduchou přesností a plovoucí řádovou čárkou v rozmezí od – 3,402823E38 do – 1,401298E-45 pro záporné hodnota a od 1,401298E-45 do 3,402823E38 pro kladné hodnoty a 0.
DOUBLE	8 bajtů	Hodnota s dvojitou přesností a plovoucí řádovou čárkou v rozmezí od – 1,79769313486232E308 do – 4,94065645841247E-324 pro záporné hodnoty a od 4,94065645841247E-324 do 1,79769313486232E308 pro kladné hodnoty a 0.
SHORT	2 bajty	Krátká celočíselná hodnota v rozmezí od – 32 768 do 32 767.
LONG	4 bajty	Dlouhá celočíselná hodnota v rozmezí od – 2 147 483 648 do 2 147 483 647.
LONGTEXT	1 bajt / znak	Nula až maximálně 1,2 Gigabajtu.
LONGBINARY	Podle potřeby	Nula až maximálně1,2 Gigabajtu. Užívá se pro objekty OLE.
TEXT	1 bajt / znak	Nula až 255 znaků.

Poznámka: Rezervované slovo VALUE můžete rovněž použít v příkazech jazyka SQL.

Rezervovaná slova databázového jádra aplikace Microsoft Jet SQL

{ewc HLP95EN.dll, DYNALINK, "Viz takďž˝":"daSQLJetSQLReservedWordsC "} {ewc HLP95EN.dll, DYNALINK, "Specifika":"daSQLJetSQLReservedWordsS "}

{button A,JI(`',`daidxSqlA')} {button B,JI(`',`daidxSqlB_C')} {button C,JI(`',`daidxSqlB_C')} {button D,JI(`',`daidxSqlD')} {button E,JI(`',`daidxSqlE_H')} {button F,JI(`',`daidxSqlE_H')} {button G,JI(`',`daidxSqlE_H')} {button H,JI(`',`daidxSqlE_H')} {button I,JI(`',`daidxSqlI')} {button J,JI(`',`daidxSqlJ_M')} {button K,JI(`',`daidxSqlJ_M')} {button L,JI(`',`daidxSqlJ_M')} {button M,JI(`',`daidxSqlJ_M')} {button N,JI(`',`daidxSqlN_P')} {button O,JI(`',`daidxSqlN_P')} {button P,JI(`',`daidxSqlN_P')} {button Q,JI(`',`daidxSqlQ_S')} {button R,JI(`',`daidxSqlQ_S')} {button S,JI(`',`daidxSqlQ_S')} {button T,JI(`',`daidxSqlT_Z')} {button U,JI(`',`daidxSqlT_Z')} {button V,JI(`',`daidxSqlT_Z')} {button W,JI(`',`daidxSqlT_Z')} {button X,JI(`',`daidxSqlT_Z')} {button Y,JI(`',`daidxSqlT_Z')} {button Z,JI(`',`daidxSqlT_Z')}

Následující seznam zobrazuje všechna rezervovaná slova aplikace Microsoft Jet, která se používají v příkazech SQL. Slova, která v tomto seznamu nejsou celá napsána velkými písmeny, jsou rovněž rezervována jinými aplikacemi. Proto jednotlivá témata nápovědy pro tato slova poskytují všeobecný popis, který není zaměřen na používání jazyka SQL.

Poznámka Slova následovaná hvězdičkou (*) jsou rezervována, ale současně nemají v kontextu příkazů jazyka SQL žádný význam (například příkazy **Level** a **TableID**).

A

ADD
ALL
Alphanumeric - Viz TEXT
ALTER
And
ANY
AS
ASC
AUTOINCREMENT - Viz COUNTER
Avg

B-C

Between
BINARY
BIT
BOOLEAN - Viz BIT
BY
BYTE
CHAR, CHARACTER - Viz TEXT
COLUMN
CONSTRAINT
Count
COUNTER
CREATE
CURRENCY

D

DATABASE
DATE - Viz DATETIME
DATETIME
DELETE
DESC
DISALLOW
DISTINCT
DISTINCTROW
DOUBLE
DROP

E-H

Eqv
EXISTS
FLOAT, FLOAT8 - Viz DOUBLE
FLOAT4 - Viz SINGLE
FOREIGN
FROM
GENERAL - Viz LONGBINARY
GROUP
GUID
HAVING

I

IEEEDOUBLE - Viz DOUBLE
IEEESINGLE - Viz SINGLE
IGNORE
Imp
In
IN
INDEX
INNER
INSERT
INT, INTEGER, INTEGER4 - Viz LONG
INTEGER1 - Viz BYTE
INTEGER2 - Viz SHORT
INTO
Is

J-M

JOIN
KEY
LEFT
Level*
Like
LOGICAL, LOGICAL1 - Viz BIT
LONG
LONGBINARY
LONGTEXT
Max
MEMO - Viz LONGTEXT
Min
Mod
MONEY - Viz CURRENCY

N-P

Not
NULL
NUMBER - Viz DOUBLE
NUMERIC - Viz DOUBLE
OLEOBJECT - Viz LONGBINARY
ON
OPTION
Or
ORDER
Outer*
OWNERACCESS
PARAMETERS
PERCENT
PIVOT
PRIMARY
PROCEDURE

Q-S

REAL - Viz SINGLE
REFERENCES
RIGHT
SELECT
SET
SHORT
SINGLE
SMALLINT - Viz SHORT
SOME
StDev
StDevP
STRING - Viz TEXT
Sum

T-Z

TABLE
TableID*
VALUE
VALUES

TEXT
TIME - Viz DATETIME
TIMESTAMP - Viz DATETIME
TOP
TRANSFORM
UNION
UNIQUE
UPDATE
Var
VARBINARY - Viz BINARY
VARCHAR - Viz TEXT
VarP
WHERE
WITH
Xor
YESNO - Viz BIT

Výrazy jazykaSQL

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlExpreessionsC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlExpreessionsX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlExpreessionsS"}

Výrazem jazyka SQL je řetězec, který tvoří část nebo celý příkaz jazyka SQL. Například metoda **FindFirst** s objektem **Recordset** používá výraz jazyka SQL, který sestává z výběrových kritérií nalezených v klauzuli WHERE jazyka SQL.

Databázové jádro aplikace Microsoft Jet používá výrazy jazyka Visual Basic pro aplikace (dále VBA) k provádění jednoduchých aritmetických výpočtů a funkcí. Všechny operátory používané ve výrazech aplikace Microsoft Jet SQL (kromě **Between**, **In** a **Like**) jsou definovány v popisu výrazů jazyka VBA. Navíc popis jazyka VBA poskytuje přes sto funkcí, které je možné ve výrazech jazyka SQL použít. Můžete například použít funkce jazyka VBA pro vytvoření dotazu SQL v aplikaci Microsoft Access pomocí návrháře dotazů a také můžete tyto funkce použít v dotazu SQL v metodě **OpenRecordset** v DAO v programových kódech Microsoft Visual C++, Microsoft Visual Basic a Microsoft Excel.

Použití zástupných znaků pro porovnávání řetězců

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dasqlUsingWildcardC"} {ewc HLP95EN.DLL,DYNALINK,"Pďž˝ďž˝klad":"dasqlUsingWildcardX":1} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dasqlUsingWildcardS"}

Vestavěné porovnávání vzorků poskytuje všestranný nástroj pro porovnávání řetězců. Následující tabulka zobrazuje zástupné znaky, které můžete použít s operátorem **Like** a počtem shodujících se číslic nebo řetězců.

Znaků ve *vzorku*	**Shoda ve** *výrazu*
?	Libovolný jednoduchý znak
*	Nula nebo více znaků
#	Libovolná jediný číslice (0 – 9)
[*seznamznaků*]	Libovolný jediný znak ze seznamu *seznamznaků*
[!*seznamznaků*]	Libovolný jediný znak, který není ze seznamu *seznamznaků*

Můžete použít skupinu jednoho či více znaků (*seznamznaků*) uzavřenou do hranatých závorek [ ] pro porovnání libovolného ve *výrazu*. *Seznamznaků* může obsahovat libovolné znaky ze znakové sady ANSI včetně číslic. Ve skutečnosti můžete použít speciální znaky jako otevírající závorku [, otazník ?, znaménko číslice # a hvězdičku * pro srovnávání pouze pokud jsou uzavřeny v hranatých závorkách. Uzavírající závorku] nelze pro srovnávání ve skupině použít , ale je možné ji použít mimo skupinu jako samostatný znak.

Kromě jednoduchého seznamu znaků uzavřených v hranatých závorkách může seznam *seznamznaků* určovat rozsah znaků užitím pomlčky - k oddělení horní a dolní meze rozsahu. Například použití [A-Z] ve *vzorku* vyústí ve shodu v případě, kdy je na odpovídající pozici znaku ve *výrazu* libovolné velké písmeno v rozmení A až Z. V hranatých závorkách můžete stanovit více rozsahů (bez oddělování). Například [a-zA-Z0-9] zahrnuje libovolný alfanumerický znak.

Ostatní důležitá pravidla pro porovnávání vzorků jsou popsána níže:

- Vykřičník (!) na začátku seznamu *seznamznaků* znamená, že shody se dosáhne, pokud bude ve *výrazu* nalezen libovolný znak kromě těch uvedených v seznamu. Je-li použit mimo závorky, porovnává se sám se sebou.
- Pomlčku (-) můžete použít buď na začátku (po vykřičníku, je-li použit) nebo na konci seznamu *seznamznaků* k porovnání sama se sebou. Je-li použita kdekoliv jinde, slouží k identifikaci rozsahu znaků ANSI.
- Při určování rozsahu znaků se znaky musí uspořádat ve vzestupném pořadí (A-Z or 0-100). [A-Z] je platný vzorek, avšak [Z-A] nikoliv.
- Znaková sekvence [ ] je ignorována, je považována za řetězec s nulovou délkou (" ").

Příklad použití příkazu ALTER TABLE

Tento příklad přidá pole Plat datového typu **Currency** do tabulky Zaměstnanci.

```
Sub AlterTableX1()

   Dim dbs As Database

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Přidání pole Plat datovéhp typu Currency
   ' do tabulky Zaměstnanci.
   dbs.Execute "ALTER TABLE Zaměstnanci " _
      & "ADD COLUMN Plat CURRENCY;"

   dbs.Close

End Sub
```

Tento příklad odstraní pole Plat z tabulky Zaměstnanci.

```
Sub AlterTableX2()

   Dim dbs As Database

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Odstranění pole Plat z tabulky Zaměstnanci.
   dbs.Execute "ALTER TABLE Zaměstnanci " _
      & "DROP COLUMN Plat;"

   dbs.Close

End Sub
```

Tento příklad přidá cizí klíč do tabulky Objednávky. Tento cizí klíč je odvozen od pole ČísloZaměstnance a odkazuje na pole ČísloZaměstnance tabulky Zaměstnanci. V tomto příkladě nesmíte vypisovat v klauzuli REFERENCES pole ČísloZaměstnance za tabulkou Zaměstnanci, protože pole Zaměstnanci je primární klíč tabulky Zaměstnanci.

```
Sub AlterTableX3()

   Dim dbs As Database

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Přidání cizího klíče do tabulky Objednávky.
   dbs.Execute "ALTER TABLE Objednávky " _
      & "ADD CONSTRAINT RelaceObjednávky " _
      & "FOREIGN KEY (Číslo zaměstnance) " _
      & "REFERENCES Zaměstnanci (Číslo zaměstnance);"
```

```
   dbs.Close

End Sub
```

Tento příklad odstraní cizí klíč z tabulky Objednávky.

```
Sub AlterTableX4()

   Dim dbs As Database

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Odstranění cizího klíče omezení RelaceObjednávky
   ' z tabulky Objednávky.
   dbs.Execute "ALTER TABLE Objednávky " _
      & "DROP CONSTRAINT RelaceObjednávky;"

   dbs.Close

End Sub
```

Příklad použití příkazu CREATE INDEX

Tento příklad vytvoří index, který sestává z polí Telefon a Linka tabulky Zaměstnanci.

```
Sub CreateIndexX1()

   Dim dbs As Database

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Vytvoří NovýIndex pro tabulku Zaměstnanci.
   dbs.Execute "CREATE INDEX NovýIndex ON Zaměstnanci " _
      & "(Telefon, Linka);"

   dbs.Close

End Sub
```

Tento příklad vytvoří index pro tabulku Zákazníci s použitím pole ČísloZákazníka. Žádné dva záznamy nemohou mít v poli ČísloZákazníka stejná data a nejsou ani povoleny hodnoty typu **Null** v tomto poli.

```
Sub CreateIndexX2()

   Dim dbs As Database

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Vytvoření jedinečného indexu KódZák pro pole
   ' Kód zákazníka.
   dbs.Execute "CREATE UNIQUE INDEX KódZák " _
      & "ON Zákazníci (Kód Zákazníka) " _
      & "WITH DISALLOW NULL;"

   dbs.Close

End Sub
```

Příklad použití příkazu CREATE TABLE a klauzule CONSTRAINT

Tento příkaz vytvoří novou tabulku se dvěma textovými poli nazvanou TatoTabulka.

```
Sub CreateTableX1()

   Dim dbs As Database

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Vytvoření tabulky se dvěma textovými poli.
   dbs.Execute "CREATE TABLE Tato tabulka " _
      & "(Jméno TEXT, Příjmení TEXT);"

   dbs.Close

End Sub
```

Tento příklad vytvoří novou tabulku nazvanou MáTabulka se dvěma textovými poli, jedním polem typu Datum/Čas a jedinečným indexem vytvořeným ze všech těchto tří polí.

```
Sub CreateTableX2()

   Dim dbs As Database

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Vytvoření tabulky se třemi poli a jedinečným
   ' indexem vytvořeným ze všech těchto tří polí.
   dbs.Execute "CREATE TABLE MáTabulka " _
      & "(Jméno TEXT, Příjmení TEXT, " _
      & "DatumNarození DATETIME, " _
      & "CONSTRAINT OmezeníMáTabulka UNIQUE " _
      & "(Jméno, Příjmení, DatumNarození));"

   dbs.Close

End Sub
```

Tento příklad vytvoří novou tabulku se dvěma textovými poli a polem typu Integer. Pole SSN je primární klíč.

```
Sub CreateTableX3()

   Dim dbs As Database

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Vytvoření tabulky se třemi poli a primárním
   ' klíčem.
   dbs.Execute "CREATE TABLE NováTabulka " _
      & "(Jméno TEXT, Příjmení TEXT, " _
```

```
        & "SSN INTEGER CONSTRAINT OmeziníMéPole " _
        & "PRIMARY KEY);"

    dbs.Close

End Sub
```

Příklad použití příkazu DROP

Následující příklad předpokládá existenci myšleného indexu NovýIndex pro tabulku Zaměstnanci databáze Northwind.

Tento příklad odstraní z tabulky Zaměstnanci index NovýIndex.

```
Sub DropX1()

   Dim dbs As Database

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Odstraní NovýIndex z tabulky Zaměstnanci.
   dbs.Execute "DROP INDEX NovýIndex ON Zaměstnanci;"

   dbs.Close

End Sub
```

Tento příklad odstraní tabulku Zaměstnanci z databáze.

```
Sub DropX2()

   Dim dbs As Database

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Odstranění tabulky Zaměstnanci.
   dbs.Execute "DROP TABLE Zaměstnanci;"

   dbs.Close

End Sub
```

Příklad použití příkazu SELECT a klauzule FROM

Některé z následujících příkladů předpokládají existenci myšleného pole Plat v tabulce Zaměstnanci. Všimněte si, že toto pole ve skutečnosti v tabulce Zaměstnanci databáze Northwind neexistuje.

Tento příklad vytvoří dynamickou sadu typu **Recordset** založenou na příkazu SQL, který vybírá pole Příjmení a Jméno všech záznamů v tabulce Zaměstnanci. Volá proceduru EnumFields, která vypíše obsah objektu **Recordset** do okna **Debug**.

```
Sub SelectX1()

   Dim dbs As Database, rst As Recordset

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Výběr hodnot příjmení a křestních jmen všech
   ' záznamů v tabulce Zaměstnanci.
   Set rst = dbs.OpenRecordset("SELECT Příjmení, " _
      & "Jméno FROM Zaměstnanci;")

   ' Osazení sady záznamů.
   rst.MoveLast

  ' Vyvolání procedury EnumFields, aby se vypsal obsah
' sady záznamů.
  EnumFields rst,12

   dbs.Close

End Sub
```

Tento příklad spočte počet záznamů, které mají vstup v poli PSČ a pojmenuje vrácené pole Počet.

```
Sub SelectX2()

   Dim dbs As Database, rst As Recordset

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Určení počtu záznamů obsahujících hodnotu PSČ
   ' a vrácení celkového počtu v poli Počet.
   Set rst = dbs.OpenRecordset("SELECT Count " _
      & "(PSČ) AS Počet FROM Zákazníci;")

   ' Osazení sady záznamů.
   rst.MoveLast

   ' Vyvolání procedury EnumFields, aby se vypsal
   ' obsah sady záznamů. Stanovení délky pole na 12.
   EnumFields rst, 12

   dbs.Close

End Sub
```

Tento příklad zobrazí počet zaměstnanců a průměrný a nejvyšší plat.

```
Sub SelectX3()

   Dim dbs As Database, rst As Recordset

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Určení počtu zaměstnanců, výpočet průměrného
   ' platu a vrácení hodnoty nejvyššího platu.
   Set rst = dbs.OpenRecordset("SELECT Count (*) " _
      & "AS CelkemZaměstnanců, Avg(Plat) " _
      & "AS PrůměrnýPlat, Max(Plat) " _
      & "AS NejvyššíPlat FROM Zaměstnanci;")

   ' Osazení sady záznamů.
   rst.MoveLast

   ' Vyvolání procedury EnumFields, aby se vypsal
   ' obsah sady záznamů. Předání objektu Recordset
   ' a požadované délky pole.
   EnumFields rst, 17

   dbs.Close

End Sub
```

Sub proceduře EnumFields je předán objekt **Recordset** z volající procedury. Tato procedura potom formátuje a vypisuje pole objektu **Recordset** do okna **Debug**. Proměnná `intFldLen` je požadovaná šířka vypsaného pole. Některá pole mohou být zkrácena.

```
Sub EnumFields(rst As Recordset, intFldLen As Integer)

   Dim lngRecords As Long, lngFields As Long
   Dim lngRecCount As Long, lngFldCount As Long
   Dim strTitle As String, strTemp As String

   ' Nastavení proměnné lngRecords na hodnotu počtu
   ' záznamů v objektu Recordset.
   lngRecords = rst.RecordCount
   ' Nastavení proměnné lngFields na hodnotu počtu
   ' polí v objektu Recordset.
   lngFields = rst.Fields.Count

   Debug.Print "Existuje " & lngRecords _
      & " záznamů obsahujících " & lngFields _
      & " polí v sadě záznamů."
   Debug.Print

   ' Vytvoření řetězce pro výpis záhlaví sloupce.
   strTitle = "Záznam  "
   For lngFldCount = 0 To lngFields - 1
      strTitle = strTitle _
      & Left(rst.Fields(lngFldCount).Name _
      & Space(intFldLen), intFldLen)
```

```
   Next lngFldCount

   ' Výpis záhlaví sloupce.
   Debug.Print strTitle
   Debug.Print

   ' Procházení objektem Recordset. Výpis počtu
   ' záznamů a hodnot polí.
   rst.MoveFirst
   For lngRecCount = 0 To lngRecords - 1
      Debug.Print Right(Space(6) & _
         Str(lngRecCount), 6) & "  ";
      For lngFldCount = 0 To lngFields - 1
         ' Provedení kontroly na hodnoty typu Null.
         If IsNull(rst.Fields(lngFldCount)) Then
            strTemp = "<null>"
         Else
            ' Nastavení strTemp na obsah pole.
            Select Case _
               rst.Fields(lngFldCount).Type
               Case 11
                  strTemp = ""
               Case dbText, dbMemo
                  strTemp = _
                     rst.Fields(lngFldCount)
               Case Else
                  strTemp = _
                     str(rst.Fields(lngFldCount))
            End Select
         End If
         Debug.Print Left(strTemp _
            & Space(intFldLen), intFldLen);
      Next lngFldCount
      Debug.Print
      rst.MoveNext
   Next lngRecCount

End Sub
```

Příklad použití klauzule IN

Následující příklad ukazuje jak můžete pomocí klauzule IN získat data z externí databáze. V každém příkladě se předpokládá existence myšlené tabulky Zaměstnanci uložené v externí databázi.

Externí databáze	SQL příkaz

databáze Microsoft Jet	`SELECT [Kód zákazníka]`
	`FROM Zákazníci`
	`IN OtherDB.mdb`
	`WHERE [Kód zákazníka] Like "A*";`

dBASE III nebo IV. Abyste získali data z tabulky dBASE III, nahraďte "dBASE IV;" výrazem "dBASE III;".	`SELECT [Kód zákazníka]`
	`FROM Zákazník`
	`IN "C:\DBASE\DATA\SALES" "dBASE IV;"`
	`WHERE [Kód zákazníka] Like "A*";`

dBASE III nebo IV s použitím syntaxe Database.	`SELECT [Kód zákazníka]`
	`FROM Zákazník`
	`IN "" [dBASE IV; Database=C:\DBASE\DATA\`
	`SALES;]`
	`WHERE [Kód zákazníka] Like "A*";`

Paradox 3.x nebo 4.x. Abyste získali data z tabulky Paradox verze 3.x, nahraďte "Paradox 3.x;" výrazem "Paradox 4.x;".	`SELECT [Kód zákazníka]`
	`FROM Zákazník`
	`IN "C:\PARADOX\DATA\SALES" "Paradox`
	`4.x;"`
	`WHERE [Kód zákazníka] Like "A*";`

Paradox 3.x nebo 4.x použitím syntaxe Database.	`SELECT [Kód zákazníka]`
	`FROM Zákazník`
	`IN "" [Paradox 4.x;Database=C:\PARADOX\`
	`DATA\SALES;]`
	`WHERE [Kód zákazníka] Like "A*";`

Sešit Microsoft Excel	`SELECT [Kód zákazníka], Firma`
	`FROM [Zákazníci$]`
	`IN "c:\documents\xldata.xls" "EXCEL`
	`5.0;"`
	`WHERE [Kód zákazníka] Like "A*"`
	`ORDER BY [Kód zákazníka];`

Pojmenovaná oblast sešitu	`SELECT [Kód zákazníka], Firma`
	`FROM [Oblast zákazníci]`
	`IN "c:\documents\xldata.xls" "EXCEL`
	`5.0;"`
	`WHERE [Kód zákazníka] Like "A*"`
	`ORDER BY [Kód zákazníka];`

Příklad použití operátoru IN

Následující příklad používá tabulku Objednávky databáze Northwind.mdb k vytvoření dotazu, který zahrne všechny objednávky s místem dodávky Lancashire a Essex a daty odeslání.

Tento příklad volá proceduru EnumFields, kterou můžete najít v příkladu k příkazu SELECT.

```
Sub InX()

   Dim dbs As Database, rst As Recordset

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Výběr záznamů z tabulky Objednávky, které mají
   ' hodnotu pole Region příjemce Lancashire nebo Essex.
   Set rst = dbs.OpenRecordset("SELECT " _
      & "[Číslo zákazníka], [Datum odeslání] FROM Objednávky " _
      & "WHERE [Region příjemce] In " _
      & "('Lancashire','Essex');")

   ' Osazení sady záznamů.
   rst.MoveLast

   ' Vyvolání funkce EnumFields, aby se vypsal obsah
   ' sady záznamů.
   EnumFields rst, 12

   dbs.Close

End Sub
```

Příklad použití klauzule WHERE

Některé z následujících příkladů předpokládají existenci myšleného pole Plat v tabulce Zaměstnanci. Všimněte si, že toto pole ve skutečnosti v tabulce Zaměstnanci databáze Northwind neexistuje.

Tento příklad vybírá pole Příjmení a Jméno všech záznamů, které mají hodnotu pole Příjmení King.

Tento příklad volá proceduru EnumFields, kterou můžete najít v příkladu k příkazu SELECT.

```
Sub WhereX()

   Dim dbs As Database, rst As Recordset

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Výběr těch záznamů z tabulky Zaměstnanci, které
   ' mají hodnotu příjmení King.
   Set rst = dbs.OpenRecordset("SELECT Příjmení, " _
      & "Jméno FROM Zaměstnanci " _
      & "WHERE Příjmení = 'King';")

   ' Osazení sady záznamů.
   rst.MoveLast

   ' Vyvolání procedury EnumFields, aby byl vypsán
   ' obsah sady záznamů.
   EnumFields rst, 12

   dbs.Close

End Sub
```

Příklad použití klauzule GROUP BY

Tento příklad vytvoří seznam jedinečných funkcí v zaměstnání a počet zaměstnanců pro každou funkci.

Tento příklad volá proceduru EnumFields, kterou můžete najít v příkladu použití příkazu SELECT.

```
Sub GroupByX1()

   Dim dbs As Database, rst As Recordset

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Výpočet počtu zaměstnanců s danou funkcí
   ' pro všechny funkce.
   Set rst = dbs.OpenRecordset("SELECT Funkce, " _
      & "Count([Funkce]) AS Počet " _
      & "FROM Zaměstnanci GROUP BY Funkce;")

   ' Osazení sady záznamů.
   rst.MoveLast

   ' Vyvolání procedury EnumFields pro výpis obsahu
   ' sady záznamů. Předání objektu Recordset
   ' a požadované délky pole.
   EnumFields rst, 25

   dbs.Close

End Sub
```

Tento příklad vypočítá pro každou jedinečnou funkci v změstnání počet zaměstnanců ve Washingtonu, kteří mají takovouto funkci.

```
Sub GroupByX2()

   Dim dbs As Database, rst As Recordset

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")


   ' Určení počtu zaměstnanců s danou funkcí
   ' pro všechny funkce. Zahrnutí pouze zaměstnanců
   ' v oblasti Washington.
   Set rst = dbs.OpenRecordset("SELECT Funkce, " _
      & "Count(Funkce) AS Počet " _
      & "FROM Zaměstnanci WHERE Region = 'WA' " _
      & "GROUP BY Funkce;")

   ' Osazení sady záznamů.
   rst.MoveLast

   ' Vyvolání procedury EnumFields pro výpis obsahu
   ' sady záznamů. Předání objektu Recordset
```

```
    ' a požadované délky pole.
    EnumFields rst, 25

    dbs.Close

End Sub
```

Příklad použití klauzule HAVING

Tento příklad vybere funkce v změstnání přiřazené více než jednomu zaměstnanci v oblasti Washington.

Tento příklad volá proceduru EnumFields, kterou můžete najít v příkladu použití příkazu SELECT.

```
Sub HavingX()

   Dim dbs As Database, rst As Recordset

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Výběr funkce přiřazené více než jednomu
   ' zaměstnanci v oblasti Washington.
   Set rst = dbs.OpenRecordset("SELECT Funkce, " _
      & "Count(Funkce) as Celkem FROM Zaměstnanci " _
      & "WHERE Region = 'WA' " _
      & "GROUP BY Funkce HAVING Count(Funkce) > 1;")

   ' Osazení sady záznamů.
   rst.MoveLast

   ' Vyvolání procedury EnumFields pro výpis obsahu
   ' sady záznamů.
   EnumFields rst, 25

   dbs.Close

End Sub
```

Příklad použití klauzule ORDER BY

Příkaz SQL použitý v následujícím příkladu používá klauzuli ORDER BY k seřazení záznamů podle příjmení v setupném pořadí (Z-A).

Tento příklad volá proceduru EnumFields, kterou můžete najít v příkladu použití příkazu SELECT.

```
Sub OrderByX()

   Dim dbs As Database, rst As Recordset

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Výběr hodnot polí Jméno a Příjmení z tabulky
   ' Zaměstnanci a jejich seřazení v sestupném pořadí.
   Set rst = dbs.OpenRecordset("SELECT Příjmení, " _
      & "Jméno FROM Zaměstnanci " _
      & "ORDER BY Příjmení DESC;")

   ' Osazení sady záznamů.
   rst.MoveLast

   ' Vyvolání procedury EnumFields pro výpis obsahu
   ' sady záznamů.
   EnumFields rst, 12

   dbs.Close

End Sub
```

Příklady použití predikátů ALL, DISTINCT, DISTINCTROW a TOP

Tento příklad vytvoří dotaz, který spojí tabulky Zákazníci a Objednávky podle pole ČísloZákazníka. Tabulka Zákazníci neobsahuje duplicitní pole ČísloZákazníka, ale tabulka Objednávky ano, protože každý zákazník může mít více objednávek. Použití predikátu DISTINCTROW vytvoří seznam společností, které mají alespoň jednu objednávku, avšak bez podrobností o těchto objednávkách.

```
Sub AllDistinctX()

   Dim dbs As Database, rst As Recordset

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Spojení tabulky Zákazníci a Objednávky podle pole
   ' ČísloZákazníka. Výběr seznamu společností, které
   ' mají alespoň jednu objednávku.
   Set rst = dbs.OpenRecordset("SELECT DISTINCTROW " _
      & "JménoSpolečnosti FROM Zákazníci" _
      & "INNER JOIN Objednávky " _
      & "ON Zákazníci.[Kód zákazníka] = " _
      & "Objednávky.[Kód zákazníka] " _
      & "ORDER BY Firma;")

   ' Osazení sady záznamů.
   rst.MoveLast

   ' Vyvolání procedury EnumFields pro výpis obsahu
   ' sady záznamů. Předání objektu Recordset
   ' a požadované délky pole.
   EnumFields rst, 25

   dbs.Close

End Sub
```

Příklad použití příkazu DELETE

Tento příklad vymaže všechny záznamy zaměstnanců, jejichž funkce je učeň. Jestliže klauzule FROM obsahuje pouze jednu tabulku, potom nemusíte uvádět název této tabulky v příkazu DELETE.

```
Sub DeleteX()

   Dim dbs As Database, rst As Recordset

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Odstranění záznamů zaměstnanců s funkcí učeň.
   dbs.Execute "DELETE * FROM " _
      & "Zaměstnanci WHERE Funkce = 'Učeň';"

   dbs.Close

End Sub
```

Příklad použití operace INNER JOIN

Tento příklad vytvoří dvě ekvivalentní spojení: jedno mezi tabulkami Rozpis objednávek a Objednávky a druhé mezi tabulkami Objednávky a Zaměstnanci. Toto je nezbytné, protože tabulka Zaměstnanci neobsahuje data o prodeji a tabulka Rozpis objednávek neobsahuje data o zaměstnancích. Uvedený dotaz vytvoří seznam zaměstnanců a jejich celkový prodej.

Tento příklad volá proceduru EnumFields, kterou můžete najít v příkladu použití příkazu SELECT.

```
Sub InnerJoinX()

   Dim dbs As Database, rst As Recordset

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Vytvoření spojení mezi tabulkami Rozpis
   ' objednávek a Objednávky a dalšího mezi tabulkami
   ' Objednávky a Zaměstnanci. Získání seznamu
   ' zaměstnanců a jejich celkového prodeje.
   Set rst = dbs.OpenRecordset("SELECT DISTINCTROW " _
      & "Sum([Jednotková cena] * Množství) AS Prodej, " _
      & "(Jméno & Chr(32) & Příjmení) AS Jméno " _
      & "FROM Zaměstnanci INNER JOIN(Objednávky " _
      & "INNER JOIN [Rozpis objednávek] " _
      & "ON [Rozpis objednávek].[Číslo objednávky] = " _
      & "Objednávky.[Číslo objednávky]) " _
      & "ON Objednávky.[Číslo zaměstnance] = " _
      & "Zaměstnanci.[Číslo zaměstnance] " _
      & "GROUP BY (Jméno & Chr(32) & Příjmení);")

   ' Osazení sady záznamů.
   rst.MoveLast

   ' Vyvolání procedury EnumFields pro výpis obsahu
   ' sady záznamů. Předání objektu Recordset
   ' a požadované délky pole.
   EnumFields rst, 20

   dbs.Close

End Sub
```

Příklad použití příkazu INSERT INTO

Tento příklad vybere všechny záznamy z myšlené tabulky Noví zákazníci a přidá je do tabulky Zákazníci. Jestliže nejsou navrženy jednotlivé sloupce, názvy sloupců tabulky v příkazu SELECT musí přesně odpovídat názvům sloupců v příkazu INSERT INTO.

```
Sub InsertIntoX1()

   Dim dbs As Database

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Výběr všech záznamů z tabulky Noví zákazníci a
   ' a jejich přidání do tabulky Zákazníci.
   dbs.Execute " INSERT INTO Zákazníci" _
      & "SELECT * " _
      & "FROM [Noví zákazníci];"

   dbs.Close

End Sub
```

Tento příklad vytvoří nový záznam v tabulce Zaměstnanci.

```
Sub InsertIntoX2()

   Dim dbs As Database

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Vytvoření nového záznamu v tabulce Zaměstnanci.
   ' Jméno je Harry, příjmení je Washington a funkce
   ' je učeň.
   dbs.Execute " INSERT INTO Zaměstnanci " _
      & "(Jméno, Příjmení, Funkce) VALUES " _
      & "('Harry', 'Washington', 'Učeň');"

   dbs.Close

End Sub
```

Příklad použití operací LEFT JOIN a RIGHT JOIN

Tento příklad předpokládá myšlená pole Jméno oddělení a Číslo oddělení v tabulce Zaměstnanci. Všimněte si, že tato pole ve skutečnosti v tabulce Zaměstnanci databáze Northwind neexistují.

Tento příklad vybere všechna oddělení včetně těch, která nemají žádné zaměstnance.

Tento příklad volá proceduru EnumFields, kterou můžete najít v příkladu použití příkazu SELECT.

```
Sub LeftRightJoinX()

   Dim dbs As Database, rst As Recordset

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Výběr všech oddělení včetně těch, která nemají
   ' zaměstnance.
   Set rst = dbs.OpenRecordset _
      ("SELECT [Jméno oddělení], " _
      & "Jméno & Chr(32) & Příjmení AS Jméno " _
      & "FROM Oddělení LEFT JOIN Zaměstnanci " _
      & "ON Oddělení.[Číslo oddělení] = " _
      & "Zaměstnanci.[Číslo oddělení] " _
      & "ORDER BY [Jméno oddělení];")

   ' Osazení sady záznamů.
   rst.MoveLast

   ' Vyvolání procedury EnumFields pro výpis obsahu
   ' sady záznamů. Předání objektu Recordset
   ' a požadované délky pole.
   EnumFields rst, 20

   dbs.Close

End Sub
```

Příklad použití deklarace PARAMETERS

Tento příklad požaduje po uživateli, aby zadal funkci, kterou poté použije jako kritérium pro dotaz. Tento příklad volá proceduru EnumFields, kterou můžete najít v příkladu použití příkazu SELECT.

```
Sub ParametersX()

   Dim dbs As Database, qdf As QueryDef
   Dim rst As Recordset
   Dim strSql As String, strParm As String
   Dim strMessage As String
   Dim intCommand As Integer

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("NorthWind.mdb")

   ' Definice parametrů klauzule.
   strParm = "PARAMETERS [Funkce zaměstnance] TEXT; "

   ' Definice SQL příkazu s parametry klauzule.
   strSql = strParm & "SELECT Příjmení, Jméno, " _
      & "ČísloZaměstnance " _
      & "FROM Zaměstnanci " _
      & "WHERE Funkce =[Funkce zaměstnance];"

   ' Vytvoření objektu QueryDef založeného na SQL
   ' příkazu.
   Set qdf = dbs.CreateQueryDef _
      ("Najít zaměstnance", strSql)

   Do While True
      strMessage = " Najít zaměstnance podle" _
         & "funkce:" & Chr(13) _
         & "  Vyberte funkci:" & Chr(13) _
         & "   1 - Obchodní ředitel" & Chr(13) _
         & "   2 - Obchodní zástupce" & Chr(13) _
         & "   3 - Referent tuzemského obchodu"

      intCommand = Val(InputBox(strMessage))

      Select Case intCommand
         Case 1
            qdf("Funkce zaměstnance") = _
               "Obchodní ředitel"
         Case 2
            qdf("Funkce zaměstnance") = _
               "Obchodní zástupce"
         Case 3
            qdf("Funkce zaměstnance") = _
               "Referent tuzemského obchodu"
         Case Else
            Exit Do
      End Select

      ' Vytvoření dočasného objektu Recordset typu
```

```
      ' snímek.
      Set rst = qdf.OpenRecordset(dbOpenSnapshot)
      ' Osazení sady záznamů.
      rst.MoveLast

   ' Vyvolání procedury EnumFields pro výpis obsahu
   ' sady záznamů. Předání objektu Recordset
   ' a požadované délky pole.
      EnumFields rst, 12
   Loop

   ' Odstranění objektu QueryDef, protože se jedná
   ' pouze o ukázku.
   dbs.QueryDefs.Delete "Najít zaměstnance"

   dbs.Close

End Sub
```

Příklad použití klauzule PROCEDURE

Tento příklad pojmenuje dotaz SeznamKategorií.

Tento příklad volá proceduru EnumFields, kterou můžete najít v příkladu použití příkazu SELECT.

```
Sub ProcedureX()

   Dim dbs As Database, rst As Recordset
   Dim qdf As QueryDef, strSql As String

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   strSql = "PROCEDURE SeznamKategorií; " _
      & "SELECT DISTINCTROW [Název kategorie], " _
      & "[Číslo kategorie] FROM Kategorie " _
      & "ORDER BY [Název kategorie];"

   ' Vytvoření pojmenovaného objektu QueryDef
   ' založeného na SQL příkazu.
   Set qdf = dbs.CreateQueryDef("NovýDotaz", strSql)

   ' Vytvoření dočasného objektu Recordset typu snímek.
   Set rst = qdf.OpenRecordset(dbOpenSnapshot)
   ' Osazení sady záznamů.
   rst.MoveLast

   ' Vyvolání procedury EnumFields pro výpis obsahu
   ' sady záznamů. Předání objektu Recordset
   ' a požadované délky pole.
   EnumFields rst, 15

   ' Odstranění objektu QueryDef, protože se jedná
   ' pouze o ukázku.
   dbs.QueryDefs.Delete "NewQry"

   dbs.Close

End Sub
```

Příklad použití příkazu SELECT...INTO

Tento příklad vybere všechny záznamy z tabulky Zaměstnanci a zkopíruje je do nové tabulky pojmenované Zaměstnanci-záloha.

```
Sub SelectIntoX()

   Dim dbs As Database
   Dim qdf As QueryDef

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Výběr všech záznamů z tabulky Zaměstnanci
   ' a jejich zkopírování do nové tabulky
   ' Zaměstnanci-záloha.
   dbs.Execute "SELECT Zaměstnanci.* INTO " _
      & "[Zaměstnanci-záloha] FROM Zaměstnanci;"

   ' Odstranění této tabulky, protože se jedná pouze
   ' o ukázku.
   dbs.Execute "DROP TABLE [Zaměstnanci-záloha];"

   dbs.Close

End Sub
```

Příklad použití příkazu TRANSFORM

Tento příklad používá klauzuli SQL TRANSFORM k vytvoření křížového dotazu, který zobrazuje počet objednávek přijatých každým zaměstnancem za každý kvartál roku 1994. Pro spuštění této procedury je třeba funkce SQLTRANSFORMOutput.

```
Sub TransformX1()

   Dim dbs As Database
   Dim strSQL As String
   Dim qdfTRANSFORM As QueryDef

   strSQL = "PARAMETERS prmYear SHORT; TRANSFORM " _
      & "Count(ČísloObjednávky) " _
      & "SELECT Jméno & "" "" & Příjmení AS " _
      & "CeléJméno FROM Zaměstnanci INNER JOIN " _
      & "Objednávky ON Zaměstnanci.[Číslo zaměstnance] = " _
      & "Objednávky.[Číslo zaměstnance] WHERE DatePart " _
      & "(""yyyy"", DatumObjednávky) = [prmYear] "

   strSQL = strSQL & "GROUP BY Jméno & " _
      & """ "" & Příjmení " _
      & "ORDER BY Jméno & "" "" & Příjmení " _
      & "PIVOT DatePart(""q"", [Datum objednávky])"

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   Set qdfTRANSFORM = dbs.CreateQueryDef _
      ("", strSQL)

   SQLTRANSFORMOutput qdfTRANSFORM, 1994

   dbs.Close

End Sub
```

Tento příkaz používá klauzuli SQL TRANSFORM k vytvoření trochu složitějšího křížového dotazu, který zobrazí celkové množství objednávek (v dolarech) přijatých zaměstnanci za každý kvartál roku 1994. Pro spuštění této procedury je třeba funkce SQLTRANSFORMOutput.

```
Sub TransformX2()

   Dim dbs As Database
   Dim strSQL As String
   Dim qdfTRANSFORM As QueryDef

   strSQL = "PARAMETERS prmYear SHORT; TRANSFORM " _
      & "Sum(Mezisoučet) SELECT Jméno & "" """ _
      & "& Příjmení AS CeléJméno " _
      & "FROM Zaměstnanci INNER JOIN " _
      & "(Objednávky INNER JOIN "_
      & "[Mezisoučty objednávek] " _
      & "ON Objednávky.[Číslo objednávky] = " _
      & "[Mezisoučty objednávek].[Číslo objednávky]) " _
      & "ON Zaměstnanci.[Číslo zaměstnance] = " _
      & "Objednávky.[Číslo zaměstnance] WHERE DatePart" _
```

```
      & "(""yyyy"", [Datum objednávky]) = [prmYear] "

   strSQL = strSQL & "GROUP BY Jméno & "" """ _
      & "& Příjmení " _
      & "ORDER BY Jméno & "" "" & Příjmení " _
      & "PIVOT DatePart(""q"",[Datum objednávky])"

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   Set qdfTRANSFORM = dbs.CreateQueryDef _
      ("", strSQL)

   SQLTRANSFORMOutput qdfTRANSFORM, 1994

   dbs.Close

End Sub

Function SQLTRANSFORMOutput(qdfTemp As QueryDef, _
   intYear As Integer)

   Dim rstTRANSFORM As Recordset
   Dim fldLoop As Field
   Dim booFirst As Boolean

   qdfTemp.PARAMETERS!prmYear = intYear
   Set rstTRANSFORM = qdfTemp.OpenRecordset()

   Debug.Print qdfTemp.SQL
   Debug.Print
   Debug.Print , , "Kvartál"

   With rstTRANSFORM
      booFirst = True
      For Each fldLoop In .Fields
         If booFirst = True Then
            Debug.Print fldLoop.Name
            Debug.Print , ;
            booFirst = False
         Else
            Debug.Print , fldLoop.Name;
         End If
      Next fldLoop
      Debug.Print

      Do While Not .EOF
         booFirst = True
         For Each fldLoop In .Fields
            If booFirst = True Then
               Debug.Print fldLoop
               Debug.Print , ;
               booFirst = False
            Else
               Debug.Print , fldLoop;
            End If
```

```
            Next fldLoop
            Debug.Print
            .MoveNext
        Loop
    End With

End Function
```

Příklad použití příkazu UPDATE

Tento příklad mění hodnoty v poli Nadřízený na hodnotu 5 pro všechny zaměstnance, jejichž pole Nadřízený má hodnotu 2.

```
Sub UpdateX()

   Dim dbs As Database
   Dim qdf As QueryDef

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Změna hodnoty v poli Nadřízený na hodnotu 5 pro
   ' všechny zaměstnance, jejichž pole Nadřízený má
   ' hodnotu 2.
   dbs.Execute "UPDATE Zaměstnanci " _
      & "SET Nadřízený = 5 " _
      & "WHERE Nadřízený = 2;"

   dbs.Close

End Sub
```

Příklad použití operace UNION

Tento příklad vyhledá jména a města všech dodavatelů a zákazníků z Brazílie.

Tento příklad volá proceduru EnumFields, kterou můžete najít v příkladu použití příkazu SELECT.

```
Sub UnionX()

   Dim dbs As Database, rst As Recordset

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Zajisti jména a města všech dodavatelů
   ' a zákazníků z Brazílie.
   Set rst = dbs.OpenRecordset("SELECT Firma," _
      & " Město FROM Dodavatelé" _
      & " WHERE Země = 'Brazílie' UNION" _
      & " SELECT Firma, Město FROM Zákazníci" _
      & " WHERE Země = 'Brazílie';")

   ' Osazení sady záznamů.
   rst.MoveLast

   ' Vyvolání procedury EnumFields pro výpis obsahu
   ' sady záznamů. Předání objektu Recordset
   ' a požadované délky pole.
   EnumFields rst, 12

   dbs.Close

End Sub
```

Příklad použití poddotazu SQL

Tento příklad zobrazí jméno a kontakt na každého zákazníka, který podal nějakou objednávku ve druhém kvartálu roku 1995.

Tento příklad volá proceduru EnumFields, kterou můžete najít v příkladu použití příkazu SELECT.

```
Sub SubQueryX()

   Dim dbs As Database, rst As Recordset

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Zobrazení jména a kontaktu na každého zákazníka,
   ' který podal nějakou objednávku ve druhém kvartálu
   ' roku 1995.
   Set rst = dbs.OpenRecordset("SELECT " _
      & " [Kontaktní osoba], Firma," _
      & " Funkce, Telefon FROM Zákazníci " _
      & " WHERE [Kód zákazníka]" _
      & " IN (SELECT ČísloZákazníka FROM Objednávky" _
      & " WHERE DatumObjednávky Between #1.4.1995#" _
      & " And #1.7.1995#);")

   ' Osazení sady záznamů.
   rst.MoveLast

   ' Vyvolání procedury EnumFields pro výpis obsahu
   ' sady záznamů. Předání objektu Recordset
   ' a požadované délky pole.
   EnumFields rst, 25

   dbs.Close

End Sub
```

Příklad použití funkce Avg

Tento příklad používá tabulku Objednávky k výpočtu průměrné ceny za přepravné u objednávek, jejichž cena za dopravné překročila 100 USD.

Tento příklad volá proceduru EnumFields, kterou můžete najít v příkladu použití příkazu SELECT.

```
Sub AvgX()

   Dim dbs As Database, rst As Recordset

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Výpočet průměrné ceny za přepravné u objednávek,
   ' jejichž cena za dopravné překročila 100 USD.
   Set rst = dbs.OpenRecordset("SELECT Avg(Dopravné)" _
      & " AS [Průměr dopravného]" _
      & " FROM Objednávky WHERE Dopravné > 100;")

   ' Osazení sady záznamů.
   rst.MoveLast

   ' Vyvolání procedury EnumFields pro výpis obsahu
   ' sady záznamů. Předání objektu Recordset
   ' a požadované délky pole.
   EnumFields rst, 25

   dbs.Close

End Sub
```

Příklad použití funkce Count

Tento příklad používá tabulku Objednávky k výpočtu množství objednávek dodaných do Velké Británie.

Tento příklad volá proceduru EnumFields, kterou můžete najít v příkladu použití příkazu SELECT.

```
Sub CountX()

   Dim dbs As Database, rst As Recordset

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Výpočet množství objednávek dodaných do Velké
   ' Británie.
   Set rst = dbs.OpenRecordset("SELECT" _
      & " Count ([Země příjemce])" _
      & " AS [Objednávky UK] FROM Objednávky" _
      & " WHERE [Země příjemce] = 'UK';")

   ' Osazení sady záznamů.
   rst.MoveLast

   ' Vyvolání procedury EnumFields pro výpis obsahu
   ' sady záznamů. Předání objektu Recordset
   ' a požadované délky pole.
   EnumFields rst, 25

   dbs.Close

End Sub
```

Příklad použití funkcí First a Last

Tento příklad používá tabulku Zaměstnanci k vrácení hodnot z pole Příjmení prvního a posledního vráceného záznamu z této tabulky.

Tento příklad volá proceduru EnumFields, kterou můžete najít v příkladu použití příkazu SELECT.

```
Sub FirstLastX1()

   Dim dbs As Database, rst As Recordset

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Vrácení hodnoty z pole Příjmení prvního
   ' a posledního vráceného záznamu z tabulky.
   Set rst = dbs.OpenRecordset("SELECT " _
      & "First(Příjmení) as První, " _
      & "Last(Příjmení) as Poslední FROM " _
      & " Zaměstnanci;")

   ' Osazení sady záznamů.
   rst.MoveLast

   ' Vyvolání procedury EnumFields pro výpis obsahu
   ' sady záznamů. Předání objektu Recordset
   ' a požadované délky pole.
   EnumFields rst, 12

   dbs.Close

End Sub
```

Další příklad porovnává použití funkcí **First** a **Last** s jednoduchým použitím funkcí **Min** a **Max** k nalezení nejstaršího a nejmladšího data narození zaměstnanců.

```
Sub FirstLastX2()

   Dim dbs As Database, rst As Recordset

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Nalezení nejstaršího a nejmladšího data narození
   ' zaměstnanců.
   Set rst = dbs.OpenRecordset("SELECT " _
      & "First([Datum narození]) as PrvníDN, " _
      & "Last([DatumNarození]) as PosledníDN FROM " _
      & "Zaměstnanci;")

   ' Osazení sady záznamů.
   rst.MoveLast

   ' Vyvolání procedury EnumFields pro výpis obsahu
   ' sady záznamů. Předání objektu Recordset
   ' a požadované délky pole.
```

```
    EnumFields rst, 12

    Debug.Print

    ' Nalezení nejstaršího a nejmladšího data narození
    ' zaměstnanců.
    Set rst = dbs.OpenRecordset("SELECT " _
       & "Min([Datum narození]) as NejnižšíDN," _
       & "Max([Datum narození]) as NejvyššíDN FROM " _
       & "Zaměstnanci;")

    ' Osazení sady záznamů.
    rst.MoveLast

    ' Vyvolání procedury EnumFields pro výpis obsahu
    ' sady záznamů. Předání objektu Recordset
    ' a požadované délky pole.
    EnumFields rst, 12

    dbs.Close

End Sub
```

Příklad použití funkcí Min a Max

Tento příklad používá tabulku Objednávky k vrácení nejnižší a nejvyšší ceny dopravného za objednávky dodané do Velké Británie.

Tento příklad volá proceduru EnumFields, kterou můžete najít v příkladu použití příkazu SELECT.

```
Sub MinMaxX()

   Dim dbs As Database, rst As Recordset

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Vrácení nejnižší a nejvyšší ceny dopravného
   ' za objednávky dodané do Velké Británie.
   Set rst = dbs.OpenRecordset("SELECT " _
      & "Min(Dopravné) AS [Nejnižší dopravné], " _
      & "Max(Dopravné)AS [Nejvyšší dopravné] " _
      & "FROM Objednávky WHERE [Země příjemce] = 'UK';")

   ' Osazení sady záznamů.
   rst.MoveLast

   ' Vyvolání procedury EnumFields pro výpis obsahu
   ' sady záznamů. Předání objektu Recordset
   ' a požadované délky pole.
   EnumFields rst, 12

   dbs.Close

End Sub
```

Příklad použití funkcí StDev a StDevP

Tento příklad používá tabulku Objednávky k odhadu směrodatné odchylky ceny dopravného za objednávky dodané do Velké Británie.

Tento příklad volá proceduru EnumFields, kterou můžete najít v příkladu použití příkazu SELECT.

```
Sub StDevX()

   Dim dbs As Database, rst As Recordset

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Výpočet směrodatné odchylky ceny dopravného
   ' za objednávky dodané do Velké Británie.
   Set rst = dbs.OpenRecordset("SELECT " _
      & "StDev(Dopravné) " _
      & "AS [Odchylka dopravného] FROM Objednávky " _
      & "WHERE ZeměPříjemce = 'UK';")

   ' Osazení sady záznamů.
   rst.MoveLast

   ' Vyvolání procedury EnumFields pro výpis obsahu
   ' sady záznamů. Předání objektu Recordset
   ' a požadované délky pole.
   EnumFields rst, 15

   Debug.Print

   Set rst = dbs.OpenRecordset("SELECT " _
      & "StDevP(Dopravné) " _
      & "AS [Odhad odchylky dopravného] FROM Objednávky " _
      & "WHERE ZeměPříjemce = 'UK';")

   ' Osazení sady záznamů.
   rst.MoveLast

   ' Vyvolání procedury EnumFields pro výpis obsahu
   ' sady záznamů. Předání objektu Recordset
   ' a požadované délky pole.
   EnumFields rst, 15

   dbs.Close

End Sub
```

Příklad použití funkce Sum

Tento příklad používá tabulku Objednávky k výpočtu celkového odbytu za objednávky dodané do Velké Británie.

Tento příklad volá proceduru EnumFields, kterou můžete najít v příkladu použití příkazu SELECT.

```
Sub SumX()

   Dim dbs As Database, rst As Recordset

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Výpočet celkového odbytu za objednávky dodané
   ' do Velké Británie.
   Set rst = dbs.OpenRecordset("SELECT" _
      & " Sum(Jednotková cena*Množství)" _
      & " AS [Celkový odbyt UK] FROM Objednávky" _
      & " INNER JOIN [Rozpis objednávek] ON" _
      & " Objednávky.Číslo objednávky = " _
      & " [Rozpis objednávek].[Číslo objednávky]" _
      & " WHERE (Země příjemce = 'UK');")

   ' Osazení sady záznamů.
   rst.MoveLast

   ' Vyvolání procedury EnumFields pro výpis obsahu
   ' sady záznamů. Předání objektu Recordset
   ' a požadované délky pole.
   EnumFields rst, 15

   dbs.Close

End Sub
```

Příklad použití funkcí Var a VarP

Tento příklad používá tabulku Objednávky k odhadu rozptylu cen dopravného za objednávky dodané do Velké Británie.

Tento příklad volá proceduru EnumFields, kterou můžete najít v příkladu použití příkazu SELECT.

```
Sub VarX()

   Dim dbs As Database, rst As Recordset

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Výpočet rozptylu cen dopravného za objednávky
   ' dodané do Velké Británie.
   Set rst = dbs.OpenRecordset("SELECT " _
      & "Var(Dopravné) " _
      & "AS [Rozptyl dopravného UK] " _
      & "FROM Objednávky WHERE [Země příjemce] = 'UK';")

   ' Osazení sady záznamů.
   rst.MoveLast

   ' Vyvolání procedury EnumFields pro výpis obsahu
   ' sady záznamů. Předání objektu Recordset
   ' a požadované délky pole.
   EnumFields rst, 20

   Debug.Print

   Set rst = dbs.OpenRecordset("SELECT " _
      & "VarP(Dopravné) " _
      & "AS [Odhad rozptylu dopravného UK] " _
      & "FROM Objednávky WHERE [Země příjemce] = 'UK';")

   ' Osazení sady záznamů.
   rst.MoveLast

   ' Vyvolání procedury EnumFields pro výpis obsahu
   ' sady záznamů. Předání objektu Recordset
   ' a požadované délky pole.
   EnumFields rst, 20

   dbs.Close

End Sub
```

Příklad použití operátoru Like

Tento příklad vrací seznam zaměstnanců, jejichž jména začínají písmeny v rozsahu A až D.

Tento příklad volá proceduru EnumFields, kterou můžete najít v příkladu použití příkazu SELECT.

```
Sub LikeX()

   Dim dbs As Database, rst As Recordset

   ' Upravte tuto řádku, aby zahrnovala cestu
   ' k databázi Northwind na daném počítači.
   Set dbs = OpenDatabase("Northwind.mdb")

   ' Vrácení seznamu zaměstnanců, jejichž jména
   ' začínají písmeny v rozsahu A až D.
   Set rst = dbs.OpenRecordset("SELECT Příjmení," _
      & " Jméno FROM Zaměstnanci" _
      & " WHERE Příjmení Like '[A-D]*';")

   ' Osazení sady záznamů.
   rst.MoveLast

   ' Vyvolání procedury EnumFields pro výpis obsahu
   ' sady záznamů. Předání objektu Recordset
   ' a požadované délky pole.
   EnumFields rst, 15

   dbs.Close

End Sub
```

Přizpůsobení registru systému Windows pro objekty Data Access

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dahowCustomizingDataAccessC"} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dahowCustomizingDataAccessS"}

Jestliže vaše aplikace nemůže správně pracovat ve výchozím nastavením databázového jádra Microsoft Jet, budete možná muset změnit nastavení registru systému Windows, aby vyhovoval vašim potřebám. Registr systému Windows se může také použít k odladění operací instalovatelného ISAM a ovladače ODBC.

Nastavení registru systému Windows můžete upravit třemi rozlišnými způsoby. Za prvé můžete jednoduše použít program Regedit.exe k přepsání výchozího nastavení, které bylo zavedeno při registraci databázového jádra Microsoft Jet. Tato metoda je nejméně pružná, protože všechny aplikace, které používají databázové jádro Microsoft Jet mají tato nová výchozí nastavení.

Druhá metoda jak upravit nastavení registru systému Windows je vytvoření částí aplikace Microsoft Jet ve stromu registru vaší aplikace pro řízení nastavení databázového jádra Microsoft Jet. Nejsnažší způsob jak toho dosáhnout je export existujícího klíče aplikace Microsoft Jet a jeho import do stromu aplikace pomocí vstupů Export a Import programu Regedit.exe. Potom můžete zaměňovat hodnoty, které chcete určit ve vašem novém stromu registru. Jestliže máte nějaké hodnoty zadané ve vnořené složce Jádra, aplikace Microsoft Jet tato nastavení načte při spuštění aplikace. Žádné hodnoty zadané ve stromu registru vaší aplikace typu klient nebudou načteny ze stínového nastavení.

Pořadí, v jakém se budou načítat příslušné části klíče registru systému Windows vaší aplikace, musíte určit umístění pomocí DAO vlastnosti **INIPath**. Aplikace musí nastavit vlastnost **INIPath** ještě před spuštěním dalšího DAO kódu. Rozsah tohoto nastavení pro aplikaci je omezen a nemůže být změněn, aniž by byla aplikace znovu spuštěna.

Poznámka Ačkoli je vytváření částí aplikace Microsoft Jet pružnější než přepisování výchozích položek jádra Microsoft Jet, vyžaduje stálé udržování stromu registru. Ve výchozím nastavení se vyžaduje záznam každé změny, a tak budete muset pokaždé registr upravovat. V předchozí verzi jádra Microsoft Jet existovaly pouze dvě strategie úpravy registru.

DAO 3.5 umožňuje nové způsoby jak upravovat výchozí nastavení. Nastavení registru se nyní může upravovat pomocí metody **SetOption** za běhu aplikace. Metoda **SetOption** dovoluje uživateli určovat nové nastavení pro jakékoliv z následujících výchozích nastavení:

- Klíč PageTimeout
- Klíč SharedAsyncDelay
- Klíč ExclusiveAsyncDelay
- Klíč LockRetry
- Klíč UserCommitSync
- Klíč ImplicitCommitSync
- Klíč MaxBufferSize
- Klíč MaxLocksPerFile
- Klíč LockDelay
- RecycleLVs
- Klíč FlushTransactionTimeout

Použitím metody **SetOption** vaše aplikace získá nejvyšší pružnost a schopnost řízení. Umožní vám vytvářet aplikace, které jsou nenáročné na udržování a odlaďování a mají největší výkonnost.

Registr systému Windows můžete také upravovat, a to z následujících důvodů:

- Nastavení použité pro vzájemnou spolupráci databází Microsoft FoxPro, Paradox a dBASE. Více se dozvíte pod hesly Inicializace ovladače databáze Paradox, Inicializace ovladače databáze Microsoft FoxPro Inicializace ovladače databáze Microsoft Excel, Inicializace ovladače databáze Lotusa Inicializace ovladače databáze dBASE.

- Nastavení použité databází Microsoft ODBC pro vzájemnou spolupráci s databázemi SQL. Více se dozvíte pod heslem Inicializace ovladače databáze Microsoft ODBC.
- Nastavení, které ovlivňuje jak bude databázové jádro Microsoft Jet načítat a ukládat data. Více se dozvíte pod heslem Inicializace ovladače jádra databáze Microsoft Jet Engine a Inicializace ovladače databázového jádra Microsoft Jet 3.5.
- Nastavení, které řídí jak bude databázové jádro Microsoft Jet vzájemně spolupracovat se sítí Internet a aplikací Microsoft Exchange. Více se dozvíte pod hesly Inicializace ovladače zdroje dat HTML (Internet) a Inicializace ovladače zdroje dat aplikace Microsoft Exchange.
- Nastavení určující, jak bude databázové jádro Microsoft Jet zacházet s daty importovanými jako text. Viz Inicializace ovladače databáze textového zdroje dat.

Inicializace ovladače databáze dBASE

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dahowChangingdBASEC"} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dahowChangingdBASES"}

Při instalaci ovladače databáze dBASE zapíše program SETUP sadu výchozích hodnot do registru systému Windows v podklíčích Jádra a Formáty ISAM. Tato nastavení byste neměli přímo upravovat, raději použijte program Setup pro přidávání, odstraňování nebo pozměňování nastavení vaší aplikace. Následující oddíly popisují nastavení inicializace a formátů ISAM ovladače databáze dBASE.

Nastaveni inicializace databáze dBASE

Složka Jet\3.5\Engines\Xbase obsahuje nastavení inicializace pro ovladač Msxbse35.dll, který se používá pro přístup k externím zdrojům dat. Typická nastavení pro vstupy v této složce jsou znázorněny v následujícím příkladu.

```
win32=<path>\MSXBSE35.dll
NetworkAccess=On
PageTimeout=600
INFPath=C:\DBASE\SYSTEM
CollatingSequence=ASCII
DataCodePage=OEM
Deleted=On
Century=Off
Date=MDY
Mark=47
Exact=Off
```

Databázové jádro Microsoft Jet používá vstupy složky Xbase jak je znázorněno dále:

Vstup	Popis
win32	Umístění ovladače Msxbse35.dll. Plná cesta je určena během instalace. Hodnoty vstupu pro systémy Windows 95 a Windows NT 4.0 jsou datového typu String a pro systém Windows NT 3.51 typu REG_SZ.
NetworkAccess	Binární indikátor pro předvolbu uzamykání souboru. Je-li hodnota vstupu NetworkAccess nastavena na 00, tabulky jsou otevřeny s výhradním přístupem, a to nehledě na nastavení argumentu *exclusive* metod **OpenDatabase** a **OpenRecordset**. Výchozí hodnota je 01. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 typu Binary a pro systém Windows NT 3.51 datového typu REG_BINARY.
PageTimeout	Časový úsek mezi uložením dat do vnitřní mezipaměti a jejich znehodnocením. Tato hodnota se zadává v jednotkách dlouhých 100 milisekund. Výchozí hodnota je buď 600 jednotek nebo 60 sekund. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu DWORD a pro systém Windows NT 3.51 typu REG_DWORD.
INFPath	Plná cesta k adresáři souboru .inf. Databázové jádro Microsoft Jet nejprve hledá soubor .inf v adresáři obsahujícím tabulku. Nenajde-li jej tam, potom prohledá vstup INFPath. Jestliže žádný vstup INFPath neexistuje, použije jakýkoliv indexový soubor (.cdx či .mdx), který najde v adresáři databáze. Hodnoty

	vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu STRING a pro systém Windows NT 3.51 datového typu REG_SZ. Tento vstup není zapisován inicializační procedurou.
CollatingSequence	Řadící sekvence pro všechny tabulky dBASE, vytvořená nabo otevřená databázovým jádrem Microsoft Jet. Přípustné hodnoty vstupu jsou typu ASCII a International. Výchozí hodnota je ASCII. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 typu String a pro systém Windows NT 3.51 typu REG_SZ.
DataCodePage	Indikátor způsobu jakým se ukládají stránky textu. Možná nastavení jsou: • OEM - Konverze OemToAnsi a AnsiToOem jsou dokončeny. • ANSI - Konverze OemToAnsi a AnsiToOem nejsou dokončeny. Výchozí hodnota vstupu je OEM. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu String a pro systém Windows NT 3.51 typu REG_SZ.
Deleted	Binární indikátor, který určuje jak jsou záznamy označené k vymazání ošetřeny databázovým jádrem Microsoft Jet. Hodnota 01 odpovídá vstupu SET DELETED ON programu dBASE a naznačuje, že se vymazaný záznam už nikdy nebude obnovovat nebo se na něj nebude nastavovat. Hodnota 00 odpovídá vstupu SET DELETED OFF programu dBASE a naznačuje, že se s vymazaným záznamem má zacházet jako s jakýmkoliv jiným. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 typu Binary a pro systém Windows NT 3.51 datového typu REG_BINARY.
Century	Binární indikátor pro formátování částí století datumů v případech, kdy jsou v indexových výrazech použity funkce, které převádí data do řetězců. Hodnota 01 odpovídá vstupu SET CENTURY ON programu dBASE a hodnota 00 vstupu SET CENTURY OFF. Výchozí hodnota je 00. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 typu Binary a pro systém Windows NT 3.51 datového typu REG_BINARY.
Date	Způsob formátování datumu, který se používá v případech, kdy jsou v indexových výrazech použity funkce, které převádí data do řetězců. Přípustné hodnoty tohoto vstupu, který odpovídá vstupu SET DATE programu dBASE, jsou: American, ANSI, British, French, DMY, German, Italian, Japan, MDY, USA a YMD. Výchozí hodnoty je MDY. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu String a pro systém Windows NT 3.51 typu REG_SZ.
Mark	Dekadická hodnota znaku ASCII, která je použita

	k oddělování částí datumu. Výchozí hodnoty nastavení vstupu Date jsou následující: • "/" (American, MDY) • "." (ANSI) • "/" (British, French, DMY) • "." (German) • "-" (Italian) • "/" (Japan, YMD) • "-" (USA) Hodnota 0 určuje, že by systém měl použít oddělovač, který se obvykle používá v souvislosti s vybraným datumovým formátem. Výchozí hodnota je 0. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu DWORD a pro systém Windows NT 3.51 typu REG_DWORD.
Exact	Binární indikátor pro porovnávání řetězců. Hodnota 01 odpovídá vstupu SET EXACT ON programu dBASE a hodnota 00 vstupu SET EXACT OFF. Výchozí hodnota je 00. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 typu Binary a pro systém Windows NT 3.51 datového typu REG_BINARY.

Formáty ISAM databáze dBASE

Složka Jet\3.5\ISAM Formats\dBASE III obsahuje následující vstupy.

Název vstupu	Datový typ pro systém Windows NT 3.51	Datový typ pro systémy Windows 95 a Windows NT 4.0	Hodnota
Engine	REG_SZ	String	Xbase
ExportFilter	REG_SZ	String	dBASE III (*.dbf)
ImportFilter	REG_SZ	String	dBASE III (*.dbf)
CanLink	REG_BINARY	Binary	01
OneTablePerFile	REG_BINARY	Binary	01
IsamType	REG_DWORD	DWORD	0
IndexDialog	REG_BINARY	Binary	01
IndexFilter	REG_SZ	String	dBASE Index (*.ndx)
CreateDBOnExport	REG_BINARY	Binary	00
ResultTextImport	REG_SZ	String	Import dat z externího souboru do aktuální databáze. Změna dat v aktuální databázi nebude mít vliv na data v externím souboru.
ResultTextLink	REG_SZ	String	Vytvoří tabulku v aktuální databázi, která je propojena s externím souborem. Změna dat v aktuální databázi změní

			data v externím souboru.
ResultTextExport	REG_SZ	String	Export dat z aktuální databáze do souboru ve formátu dBASE III. Tento proces přepíše data, jestliže jsou exportována do existujícího souboru.

Složka Jet\3.5\ISAM Formats\dBASE IV obsahuje následující vstupy.

Název vstupu	Datový typ pro systém Windows NT 3.51	Datový typ pro systémy Windows 95 a Windows NT 4.0	Hodnota
Engine	REG_SZ	String	Xbase
ExportFilter	REG_SZ	String	dBASE IV (*.dbf)
ImportFilter	REG_SZ	String	dBASE IV (*.dbf)
CanLink	REG_BINARY	Binary	01
OneTablePerFile	REG_BINARY	Binary	01
IsamType	REG_DWORD	DWORD	0
IndexDialog	REG_BINARY	Binary	01
IndexFilter	REG_SZ	String	dBASE Index (*.ndx; *.mdx)
CreateDBOnExport	REG_BINARY	Binary	00
ResultTextImport	REG_SZ	String	Import dat z externího souboru do aktuální databáze. Změna dat v aktuální databázi nebude mít vliv na data v externím souboru
ResultTextLink	REG_SZ	String	Vytvoří tabulku v aktuální databázi, která je propojena s externím souborem. Změna dat v aktuální databázi změní data v externím souboru.
ResultTextExport	REG_SZ	String	Export dat z aktuální databáze do souboru ve formátu dBASE IV. Tento proces přepíše data, jestliže jsou exportována do existujícího souboru.

Složka Jet\3.5\ISAM Formats\dBASE 5.x obsahuje následující vstupy.

Název vstupu	Datový typ pro systém Windows NT 3.51	Datový typ pro systémy Windows 95 a Windows NT 4.0	Hodnota
Engine	REG_SZ	String	Xbase

ExportFilter	REG_SZ	String	dBASE 5 (*.dbf)
ImportFilter	REG_SZ	String	dBASE 5 (*.dbf)
CanLink	REG_BINARY	Binary	01
OneTablePerFile	REG_BINARY	Binary	01
IsamType	REG_DWORD	DWORD	0
IndexDialog	REG_BINARY	Binary	01
IndexFilter	REG_SZ	String	dBASE Index (*.ndx; *.mdx)
CreateDBOnExport	REG_BINARY	Binary	00
ResultTextImport	REG_SZ	String	Import dat z externího souboru do aktuální databáze. Změna dat v aktuální databázi nebude mít vliv na data v externím souboru.
ResultTextLink	REG_SZ	String	Vytvoří tabulku v aktuální databázi, která je propojena s externím souborem. Změna dat v aktuální databázi změní data v externím souboru.
ResultTextExport	REG_SZ	String	Export dat z aktuální databáze do souboru ve formátu dBASE 5. Tento proces přepíše data, jestliže jsou exportována do existujícího souboru.

Poznámka Aby se projevily změny nastavení registru systému Windows, musíte opustit a potom znovu spustit databázové jádro.

Inicializace ovladače databáze Lotus

{ewc HLP95EN.dll, DYNALINK, "Viz takďž˝":"dahowChangingLotusC "} {ewc HLP95EN.dll, DYNALINK, "Specifika":"dahowChangingLotusS "}

Při instalaci ovladače aplikace Lotus zapíše program SETUP sadu výchozích hodnot do registru systému Windows v podklíčích Jádra a Formáty ISAM. Tato nastavení byste neměli přímo upravovat, raději použijte program Setup pro přidávání, odstraňování nebo pozměňování nastavení vaší aplikace. Následující oddíly popisují nastavení inicializace a formátů ISAM ovladače databáze Lotus.

Nastavení inicializace databáze Lotus

Složka Jet\3.5\Engines\Lotus obsahuje nastavení inicializace pro ovladač Msltus35.dll, který se používá pro externí přístup k tabulkám formátu Lotus. Typická nastavení pro vstupy pod touto hlavičkou jsou znázorněny v následujícím příkladu.

```
win32=<path>\MSLTUS35.dll
TypeGuessRows=8
ImportMixedTypes=Text
AppendBlankRows=4
FirstRowHasNames=Yes
```

Databázové jádro Microsoft Jet používá vstupy složky Lotus jak je ukázáno dále.

Vstup	Popis
win32	Umístění ovladače Msltus35.dll. Plná cesta je určena během instalace. Hodnoty vstupu pro systémy Windows 95 a Windows NT 4.0 jsou datového typu String a pro systém Windows NT 3.51 typu REG_SZ.
TypeGuessRows	Počet řádků, které mají být zkontrolovány na datový typ. Určení datového typu je založeno na vyhodnocení nejčastěji se vyskytujícího datového typu ve výběru. Jestliže žádný datový typ nepřevažuje, pak jsou typy uspořádány v následujícím pořadí: Number, Currency, Date, Text, Long Text. Jestliže zpracovávaná data neodpovídají datovému typu, který se pro sloupec předpokládá, je vrácena hodnota **Null**. Jestliže v případě importu sloupec obsahuje smíšená data, pak bude celý sloupec převeden podle nastavení vstupu ImportMixedTypes. Výchozí počet řádků ke kontrole je nastaven na hodnotu 8. Hodnoty vstupu pro systémy Windows 95 a Windows NT 4.0 jsou datového typu DWORD a pro systém Windows NT 3.51 typu REG_DWORD.
ImportMixedTypes	Hodnota vstupu může být nastavena na MajorityType nebo Text. Jestliže bude hodnota vstupu nastavena na MajorityType, sloupce smíšených dat na import budou převedeny do převládajícího datového typu. Bude-li hodnota nastavena na Text, sloupce smíšených dat na import budou převedeny na datový typ Text. Výchozí hodnota nastavení je Text. Hodnoty vstupu pro systémy Windows 95 a Windows NT 4.0 jsou datového typu String a pro systém Windows NT 3.51 typu REG_SZ.
AppendBlankRows	Počet prázdných řádků, které se mají připojit na konec sešitu formátu WK1 před přidáváním nových

	dat. Jestliže je například hodnota vstupu AppendBlankRows nastavena na 4, aplikace Microsoft Jet připojí čtyři prázdné řádky na konec sešitu před připojením řádků obsahujících data. Hodnoty pro toto nastavení jsou celočíselné a mohou se zadávat v rozmezí 0 až 16. Výchozí hodnota je 01 (připojí se jeden přídavný řádek). Hodnoty vstupu pro systémy Windows 95 a Windows NT 4.0 jsou datového typu DWORD a pro systém Windows NT 3.51 typu REG_DWORD.
FirstRowHasNames	Binární hodnota, která naznačuje, zda první řádek tabulky obsahuje názvy sloupců. Hodnota nastavená na 01 znamená, že během importu dat jsou názvy sloupců brány z prvního řádku. Hodnota nastavená na 00 znamená, že v prvním řádku nejsou žádné názvy sloupců a názvy slouců se objevují ve tvaru F1, F2, F3, atd. Výchozí hodnota nastavení je 01. Hodnoty vstupu pro systémy Windows 95 a Windows NT 4.0 jsou typu Binary a pro systém Windows NT 3.51 datového typu REG_ BINARY.

Formáty ISAM databáze Lotus

Složka Jet\3.5\ISAM Formats\Lotus WK1 obsahuje následující vstupy.

Název vstupu	Datový typ pro systém Windows NT 3.51	Datový typ pro systémy Windows 95 a Windows NT 4.0	Hodnota
Engine	REG_SZ	String	Lotus
ExportFilter	REG_SZ	String	Lotus 1-2-3 WK1 (*.wk1)
ImportFilter	REG_SZ	String	Lotus 1-2-3 (*.wk*;*.wj*)
CanLink	REG_BINARY	Binary	00
OneTablePerFile	REG_BINARY	Binary	00
IsamType	REG_DWORD	DWORD	1
IndexDialog	REG_BINARY	Binary	00
CreateDBOnExport	REG_BINARY	Binary	01
ResultTextImport	REG_SZ	String	Import dat z externího souboru do aktuální databáze. Změna dat v aktuální databázi nebude mít vliv na data v externím souboru.
ResultTextExport	REG_SZ	String	Export dat z aktuální databáze do souboru ve formátu Lotus 1-2-3 verze 2. Tento proces přepíše data, jestliže

			jsou exportována do existujícího souboru.

Složka Jet\3.5\ISAM Formats\Lotus WK3 obsahuje následující vstupy.

Název vstupu	Datový typ pro systém Windows NT 3.51	Datový typ pro systémy Windows 95 a Windows NT 4.0	Hodnota
Engine	REG_SZ	String	Lotus
ExportFilter	REG_SZ	String	Lotus 1-2-3 WK3 (*.wk3)
CanLink	REG_BINARY	Binary	00
OneTablePerFile	REG_BINARY	Binary	00
IsamType	REG_DWORD	DWORD	1
IndexDialog	REG_BINARY	Binary	00
CreateDBOnExport	REG_BINARY	Binary	01
ResultTextExport	REG_SZ	String	Export dat z aktuální databáze do souboru ve formátu Lotus 1-2-3 verze 3. Tento proces přepíše data, jestliže jsou exportována do existujícího souboru.

Složka Jet\3.5\ISAM Formats\Lotus WK4 obsahuje následující vstupy.

Název vstupu	Datový typ pro systém Windows NT 3.51	Datový typ pro systémy Windows 95 a Windows NT 4.0	Hodnota
Engine	REG_SZ	String	Lotus
CanLink	REG_BINARY	Binary	00
OneTablePerFile	REG_BINARY	Binary	00
IsamType	REG_DWORD	DWORD	1
IndexDialog	REG_BINARY	Binary	00
CreateDBOnExport	REG_BINARY	Binary	01

Poznámka Aby se projevily změny nastavení registru systému Windows, musíte opustit a potom znovu spustit databázové jádro.

Inicializace ovladače databáze Microsoft Excel

{ewc HLP95EN.dll, DYNALINK, "Viz takďž˝":"dahowChangingExcelC "} {ewc HLP95EN.dll, DYNALINK, "Specifika":"dahowChangingExcelS "}

Při instalaci ovladače aplikace Microsoft Excel zapíše program SETUP sadu výchozích hodnot do registru systému Windows v podklíčích Jádra a Formáty ISAM. Tato nastavení byste neměli přímo upravovat, raději použijte program Setup pro přidávání, odstraňování nebo pozměňování nastavení vaší aplikace. Následující oddíly popisují nastavení inicializace a formátů ISAM ovladače databáze Microsoft Excel.

Nastavení inicializace databáze Microsoft Excel

Složka Jet\3.5\Engines\Excel obsahuje nastavení inicializace pro ovladač Msexcl35.dll, který se používá pro externí přístup k sešitům formátu Excel. Typická nastavení pro vstupy v této složce jsou znázorněny v následujícím příkladu.

```
win32=<path>\MSEXCL35.dll
TypeGuessRows=8
ImportMixedTypes=Text
AppendBlankRows=1
FirstRowHasNames=Yes
```

Databázové jádro Microsoft Jet používá vstupy složky Excel jak je ukázáno dále.

Vstup	Popis
win32	Umístění ovladače Msexcl35.dll. Plná cesta je určena během instalace. Hodnoty vstupu pro systémy Windows 95 a Windows NT 4.0 jsou datového typu String a pro systém Windows NT 3.51 typu REG_SZ.
TypeGuessRows	Počet řádků, které mají být zkontrolovány na datový typ. Určení datového typu je založeno na vyhodnocení nejvyššího počtu druhů nalezených datových typů. Jestliže žádný datový typ nepřevažuje, pak jsou typy uspořádány v následujícím pořadí: Number, Currency, Date, Text, Boolean. Jestliže zpracovávaná data neodpovídají datovému typu, který se pro sloupec předpokládá, je vrácena hodnota **Null**. Jestliže při importu sloupec obsahuje smíšená data, pak bude celý sloupec převeden podle nastavení vstupu ImportMixedTypes. Výchozí počet řádků ke kontrole je nastaven na hodnotu 8. Hodnoty vstupu pro systémy Windows 95 a Windows NT 4.0 jsou datového typu DWORD a pro systém Windows NT 3.51 typu REG_DWORD.
ImportMixedTypes	Hodnota vstupu může být nastavena na MajorityType nebo Text. Jestliže bude hodnota vstupu nastavena na MajorityType, sloupce smíšených dat na import budou převedeny do převládajícího datového typu. Bude-li hodnota nastavena na Text, sloupce smíšených dat na import budou převedeny na datový typ Text. Výchozí hodnota nastavení je Text. Hodnoty vstupu pro systémy Windows 95 a Windows NT 4.0 jsou datového typu String a pro systém Windows NT 3.51 typu REG_SZ.
AppendBlankRows	Počet prázdných řádků, které se mají připojit na konec sešitu verze 3.5 nebo 4.0 před přidáváním nových dat.

	Jestliže je například hodnota vstupu AppendBlankRows nastavena na 4, aplikace Microsoft Jet připojí čtyři prázdné řádky na konec sešitu před připojením řádků obsahujících data. Hodnoty pro toto nastavení jsou celočíselné a mohou se zadávat v rozmezí 0 až 16. Výchozí hodnota je 01 (připojí se jeden přídavný řádek). Hodnoty vstupu pro systémy Windows 95 a Windows NT 4.0 jsou datového typu DWORD a pro systém Windows NT 3.51 typu REG_DWORD.
FirstRowHasNames	Binární hodnota, která naznačuje, zda první řádek tabulky obsahuje názvy sloupců. Hodnota nastavená na 01 znamená, že během importu dat jsou názvy sloupců brány z prvního řádku. Hodnota nastavená na 00 znamená, že v prvním řádku nejsou žádné názvy sloupců a názvy slouců se objevují ve tvaru F1, F2, F3, atd. Výchozí hodnota nastavení je 01. Hodnoty vstupu pro systémy Windows 95 a Windows NT 4.0 jsou typu Binary a pro systém Windows NT 3.51 datového typu REG_ BINARY.

Formáty ISAM databáze Excel

Složka Jet\3.5\ISAM Formats\ Excel 3.0 obsahuje následující vstupy.

Název vstupu	Datový typ pro systém Windows NT 3.51	Datový typ pro systémy Windows 95 a Windows NT 4.0	Hodnota
Engine	REG_SZ	String	Excel
ExportFilter	REG_SZ	String	Microsoft Excel 3 (*.xls)
CanLink	REG_BINARY	Binary	01
OneTablePerFile	REG_BINARY	Binary	00
IsamType	REG_DWORD	DWORD	1
IndexDialog	REG_BINARY	Binary	00
CreateDBOnExport	REG_BINARY	Binary	01
ResultTextExport	REG_SZ	String	Export dat z aktuální databáze do souboru ve formátu Excel 3.0. Tento proces přepíše data, jestliže jsou exportována do existujícího souboru.

Složka Jet\3.5\ISAM Formats\ Excel 4.0 obsahuje následující vstupy.

Název vstupu	Datový typ pro systém Windows NT 3.51	Datový typ pro systémy Windows 95 a Windows	Hodnota

		NT 4.0	
Engine	REG_SZ	String	Excel
ExportFilter	REG_SZ	String	Microsoft Excel 4 (*.xls)
CanLink	REG_BINARY	Binary	01
OneTablePerFile	REG_BINARY	Binary	00
IsamType	REG_DWORD	DWORD	1
IndexDialog	REG_BINARY	Binary	00
CreateDBOnExport	REG_BINARY	Binary	01
ResultTextExport	REG_SZ	String	Export dat z aktuální databáze do souboru ve formátu Excel 4.0. Tento proces přepíše data, jestliže jsou exportována do existujícího souboru.

Složka Jet\3.5\ISAM Formats\ Excel 5.0 obsahuje následující vstupy, které se používají i pro aplikaci Microsoft Excel ve verzích 5.0 a 7.0.

Název vstupu	**Datový typ pro systém Windows NT 3.51**	**Datový typ pro systémy Windows 95 a Windows NT 4.0**	**Hodnota**
Engine	REG_SZ	String	Excel
ExportFilter	REG_SZ	String	Microsoft Excel 5 až 7 (*.xls)
ImportFilter	REG_SZ	String	Microsoft Excel (*.xls)
CanLink	REG_BINARY	Binary	01
OneTablePerFile	REG_BINARY	Binary	00
IsamType	REG_DWORD	DWORD	1
IndexDialog	REG_BINARY	Binary	00
CreateDBOnExport	REG_BINARY	Binary	01
ResultTextImport	REG_SZ	String	Import dat z externího souboru do aktuální databáze. Změna dat v aktuální databázi nebude mít vliv na data v externím souboru.
ResultTextLink	REG_SZ	String	Vytvoří tabulku v aktuální databázi, která je propojena s externím souborem. Změna dat v aktuální databázi změní data v externím souboru.
ResultTextExport	REG_SZ	String	Export dat z aktuální

			databáze do souboru ve formátu Excel 5.0. Tento proces přepíše data, jestliže jsou exportována do existujícího souboru.

Složka Jet\3.5\ISAM Formats\ Excel 8.0 obsahuje následující vstupy, které se používají i pro aplikaci Microsoft Excel 97.

Název vstupu	**Datový typ pro systém Windows NT 3.51**	**Datový typ pro systémy Windows 95 a Windows NT 4.0**	**Hodnota**
Engine	REG_SZ	String	Excel
ExportFilter	REG_SZ	String	Microsoft Excel 97
CanLink	REG_BINARY	Binary	01
OneTablePerFile	REG_BINARY	Binary	00
IsamType	REG_DWORD	DWORD	1
IndexDialog	REG_BINARY	Binary	00
CreateDBOnExport	REG_BINARY	Binary	01
ResultTextExport	REG_SZ	String	Export dat z aktuální databáze do souboru ve formátu Excel 97. Tento proces přepíše data, jestliže jsou exportována do existujícího souboru.

Poznámka Aby se projevily změny nastavení registru systému Windows, musíte opustit a potom znovu spustit databázové jádro.

Inicializace ovladače databáze Microsoft FoxPro

{ewc HLP95EN.dll, DYNALINK, "Viz takďž˝":"dahowChangingFoxProC "} {ewc HLP95EN.dll, DYNALINK, "Specifika":"dahowChangingFoxProS "}

Při instalaci ovladače databáze Microsoft FoxPro zapíše program SETUP sadu výchozích hodnot do registru systému Windows v podklíčích Jádra a Formáty ISAM. Tato nastavení byste neměli přímo upravovat, raději použijte program Setup pro přidávání, odstraňování nebo pozměňování nastavení vaší aplikace. Následující oddíly popisují nastavení inicializace a formátů ISAM ovladače databáze Microsoft FoxPro.

Nastavení inicializace databáze Microsoft FoxPro

Složka Jet\3.5\Engines\ Xbase obsahuje nastavení inicializace pro ovladač Msxbse35.dll, který se používá pro přístup k externím zdrojům dat formátu FoxPro. Typická nastavení pro vstupy v této složce jsou znázorněny v následujícím příkladu.

```
win32=<path>\MSXBSE35.dll
NetworkAccess=On
PageTimeout=600
INFPath=C:\DBASE\SYSTEM
CollatingSequence=ASCII
DataCodePage=OEM
Deleted=Off
Century=Off
Date=MDY
Mark=47
Exact=Off
```

Databázové jádro Microsoft Jet používá vstupy složky Xbase jak je dále ukázáno.

Vstup	Popis
win32	Umístění ovladače Msxbse35.dll. Plná cesta je určena během instalace. Hodnoty vstupu pro systémy Windows 95 a Windows NT 4.0 jsou datového typu String a pro systém Windows NT 3.51 typu REG_SZ.
NetworkAccess	Binární indikátor pro předvolbu uzamykání souboru. Je-li hodnota vstupu NetworkAccess nastavena na 00, tabulky jsou otevřeny s výhradním přístupem, a to nehledě na nastavení argumentu *exclusive* metod **OpenDatabase** a **OpenRecordset**. Výchozí hodnota je 01. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 typu Binary a pro systém Windows NT 3.51 datového typu REG_BINARY.
PageTimeout	Časový úsek mezi uložením dat do vnitřní mezipaměti a jejich znehodnocením. Tato hodnota se zadává v jednotkách dlouhých 100 milisekund. Výchozí hodnota je buď 600 jednotek nebo 60 sekund. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu DWORD a pro systém Windows NT 3.51 typu REG_DWORD.
INFPath	Plná cesta k adresáři souboru .inf. Databázové jádro Microsoft Jet nejprve hledá soubor .inf

	v adresáři obsahujícím tabulku. Nenajde-li jej tam, potom prohledá vstup INFPath. Jestliže žádný vstup INFPath neexistuje, použije jakýkoliv indexový soubor (.cdx či .mdx), který najde v adresáři databáze. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu String a pro systém Windows NT 3.51 datového typu REG_SZ. Tento vstup není zapisován inicializační procedurou.
CollatingSequence	Řadící sekvence pro všechny tabulky FoxPro, vytvořená nabo otevřená databázovým jádrem Microsoft Jet. Přípustné hodnoty vstupu jsou typu ASCII a International. Výchozí hodnota je ASCII. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 typu String a pro systém Windows NT 3.51 typu REG_SZ.
DataCodePage	Indikátor způsobu jakým se ukládají stránky textu. Možná nastavení jsou: • OEM - Konverze OemToAnsi a AnsiToOem jsou dokončeny. • ANSI - Konverze OemToAnsi a AnsiToOem nejsou dokončeny. Výchozí hodnota vstupu je OEM. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu String a pro systém Windows NT 3.51 typu REG_SZ.
Deleted	Binární indikátor, který určuje jak jsou záznamy označené k vymazání ošetřeny databázovým jádrem Microsoft Jet. Hodnota 01 odpovídá vstupu SET DELETED ON programu Microsoft FoxPro a naznačuje, že se vymazaný záznam už nikdy nebude obnovovat nebo se na něj nebude nastavovat. Hodnota 00 odpovídá vstupu SET DELETED OFF programu Microsoft FoxPro a naznačuje, že se s vymazaným záznamem má zacházet jako s jakýmkoliv jiným. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 typu Binary a pro systém Windows NT 3.51 datového typu REG_BINARY.
Century	Binární indikátor pro formátování částí století datumů v případech, kdy jsou v indexových výrazech použity funkce, které převádí data do řetězců. Hodnota 01 odpovídá vstupu SET CENTURY ON programu Microsoft FoxPro a hodnota 00 vstupu SET CENTURY OFF. Výchozí hodnota je 00. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 typu Binary a pro systém Windows NT 3.51 datového typu REG_BINARY.
Date	Způsob formátování datumu, který se používá v případech, kdy jsou v indexových výrazech použity funkce, které převádí data do řetězců. Přípustné hodnoty tohoto vstupu, který odpovídá

	vstupu SET DATE programu Microsoft FoxPro, jsou: American, ANSI, British, French, DMY, German, Italian, Japan, MDY, USA a YMD. Výchozí hodnoty je MDY. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu String a pro systém Windows NT 3.51 typu REG_SZ.
Mark	Dekadická hodnota znaku ASCII, která je použita k oddělování částí datumu. Výchozí hodnoty nastavení vstupu Date jsou následující: • "/" (American, MDY) • "." (ANSI) • "/" (British, French, DMY) • "." (German) • "-" (Italian) • "/" (Japan, YMD) • "-" (USA) Hodnota 0 určuje, že by systém měl použít oddělovač, který se obvykle používá v souvislosti s vybraným datumovým formátem. Výchozí hodnota je 0. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu DWORD a pro systém Windows NT 3.51 typu REG_DWORD.
Exact	Binární indikátor pro porovnávání řetězců. Hodnota 01 odpovídá vstupu SET EXACT ON programu Microsoft FoxPro a hodnota 00 vstupu SET EXACT OFF. Výchozí hodnota je 00. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 typu Binary a pro systém Windows NT 3.51 datového typu REG_BINARY.

Formáty ISAM databáze Microsoft FoxPro

Složka Jet\3.5\ISAM Formats\ FoxPro 2.0 obsahuje následující vstupy.

Název vstupu	Datový typ pro systém Windows NT 3.51	Datový typ pro systémy Windows 95 a Windows NT 4.0	Hodnota
Engine	REG_SZ	String	Xbase
ExportFilter	REG_SZ	String	Microsoft FoxPro 2.0 (*.dbf)
ImportFilter	REG_SZ	String	Microsoft FoxPro (*.dbf)
CanLink	REG_BINARY	Binary	01
OneTablePerFile	REG_BINARY	Binary	01
IsamType	REG_DWORD	DWORD	0
IndexDialog	REG_BINARY	Binary	01
IndexFilter	REG_SZ	String	FoxPro Index (*.idx; *.cdx)

CreateDBOnExport	REG_BINARY	Binary	00
ResultTextImport	REG_SZ	String	Import dat z externího souboru do aktuální databáze. Změna dat v aktuální databázi nebude mít vliv na data v externím souboru.
ResultTextLink	REG_SZ	String	Vytvoří tabulku v aktuální databázi, která je propojena s externím souborem. Změna dat v aktuální databázi změní data v externím souboru.
ResultTextExport	REG_SZ	String	Export dat z aktuální databáze do souboru ve formátu Microsoft FoxPro 2.0. Tento proces přepíše data, jestliže jsou exportována do existujícího souboru.

Složka Jet\3.5\ISAM Formats\ FoxPro 2.5 obsahuje následující vstupy.

Název vstupu	**Datový typ pro systém Windows NT 3.51**	**Datový typ pro systémy Windows 95 a Windows NT 4.0**	**Hodnota**
Engine	REG_SZ	String	Xbase
ExportFilter	REG_SZ	String	Microsoft FoxPro 2.5 (*.dbf)
CanLink	REG_BINARY	Binary	01
OneTablePerFile	REG_BINARY	Binary	01
IsamType	REG_DWORD	DWORD	0
IndexDialog	REG_BINARY	Binary	01
IndexFilter	REG_SZ	String	FoxPro Index (*.idx; *.cdx)
CreateDBOnExport	REG_BINARY	Binary	00
ResultTextExport	REG_SZ	String	Export dat z aktuální databáze do souboru ve formátu Microsoft FoxPro 2.5. Tento proces přepíše data, jestliže jsou exportována do existujícího souboru.

Složka Jet\3.5\ISAM Formats\ FoxPro 2.6 obsahuje následující vstupy.

Název vstupu	**Datový typ pro systém**	**Datový typ pro systémy**	**Hodnota**

	Windows NT 3.51	Windows 95 a Windows NT 4.0	
Engine	REG_SZ	String	Xbase
ExportFilter	REG_SZ	String	Microsoft FoxPro 2.6 (*.dbf)
CanLink	REG_BINARY	Binary	01
OneTablePerFile	REG_BINARY	Binary	01
IsamType	REG_DWORD	DWORD	0
IndexDialog	REG_BINARY	Binary	01
IndexFilter	REG_SZ	String	FoxPro Index (*.idx; *.cdx)
CreateDBOnExport	REG_BINARY	Binary	00
ResultTextExport	REG_SZ	String	Export dat z aktuální databáze do souboru ve formátu Microsoft FoxPro 2.6. Tento proces přepíše data, jestliže jsou exportována do existujícího souboru.

Složka Jet\3.5\ISAM Formats\ FoxPro 3.0 obsahuje následující vstupy.

Název vstupu	Datový typ pro systém Windows NT 3.51	Datový typ pro systémy Windows 95 a Windows NT 4.0	Hodnota
Engine	REG_SZ	String	Xbase
ExportFilter	REG_SZ	String	Microsoft FoxPro 3.0 (*.dbf)
CanLink	REG_BINARY	Binary	00
OneTablePerFile	REG_BINARY	Binary	01
IsamType	REG_DWORD	DWORD	0
IndexDialog	REG_BINARY	Binary	01
CreateDBOnExport	REG_BINARY	Binary	00
ResultTextExport	REG_SZ	String	Export dat z aktuální databáze do souboru ve formátu Microsoft FoxPro 3.0. Tento proces přepíše data, jestliže jsou exportována do existujícího souboru.

Složka Jet\3.5\ISAM Formats\ FoxPro 3.0 DBC obsahuje následující vstupy.

Název vstupu	Datový typ pro systém Windows NT 3.51	Datový typ pro systémy Windows 95 a	Hodnota

		Windows NT 4.0	
Engine	REG_SZ	String	Xbase
ExportFilter	REG_SZ	String	Microsoft FoxPro 3.0 (*.dbc)
CanLink	REG_BINARY	Binary	00
OneTablePerFile	REG_BINARY	Binary	00
IsamType	REG_DWORD	DWORD	0
IndexDialog	REG_BINARY	Binary	00
CreateDBOnExport	REG_BINARY	Binary	00
ResultTextImport	REG_SZ	String	Import dat z externího souboru do aktuální databáze. Změna dat v aktuální databázi nebude mít vliv na data v externím souboru.

Poznámka: Aby se projevily změny nastavení registru systému Windows, musíte opustit a potom znovu spustit databázové jádro.

Inicializace ovladače databázového jádra Microsoft Jet 2.5

{ewc HLP95EN.dll, DYNALINK, "Viz takďž˝":"dahowChangingJetISAMC "} {ewc HLP95EN.dll, DYNALINK, "Specifika":"dahowChangingJetISAMS "}

Při instalaci ovladače databázového jádra Microsoft Jet 2.x zapíše program Setup sadu výchozích hodnot do registru systému Windows v podklíčích Jádra a Formáty ISAM. K přidávání, odstraňování nebo pozměňování tohoto nastavení musíte použít program Registry Editor. Následující oddíly popisují nastavení inicializace a formátů ISAM ovladače databázového jádra Microsoft Jet 2.x.

Poznámka Instalační procedura zapíše do složky Jet\3.5\Engines\Jet 2.x pouze jedno nastavení (win32), které je jediným platným nastavením v této složce.

Nastavení inicializace databázového jádra Microsoft Jet

Složka Jet\3.5\Engines\Jet 2.x obsahuje následující vstupy.

Název vstupu	Datový typ pro systém Windows NT 3.51	Datový typ pro systémy Windows 95 a Windows NT 4.0	Výchozí hodnota
PageTimeOut	REG_DWORD	Integer	5
LockedPageTimeOut	REG_DWORD	Integer	5
LockRetry	REG_DWORD	Integer	20
CommitLockRetry	REG_DWORD	Integer	20
CursorTimeout	REG_DWORD	Integer	5
IdleFrequency	REG_DWORD	Integer	10
ForceOSFlush	REG_DWORD	Integer	0
MaxBufferSize	REG_DWORD	Integer	512
ReadAheadPages	REG_DWORD	Integer	8

Databázové jádro Microsoft Jet používá k určení cesty k ovladači databázového jádra následující vstup.

Vstup	Popis
win32	Umístění ovladače databázového jádra (soubor .dll). Cesta je určena během instalace. Hodnoty vstupu pro systémy Windows 95 a Windows NT 4.0 jsou datového typu String a pro systém Windows NT 3.51 typu REG_SZ.

Přídavné nastavení inicializace pro databázové jádro Microsoft Jet je obsaženo ve složce Jet\3.5\Engines\Jet 2.x.

Typická nastavení inicializace pro vstupy ve složce Jet\3.5\Engines\Jet 2.x\ISAM jsou znázorněny v následujícím příkladu.

```
PageTimeout=5
LockedPageTimeout=5
CursorTimeout=5
LockRetry=20
CommitLockRetry=20
MaxBufferSize=512
ReadAheadPages=16
IdleFrequency=10
ForceOsFlush = 0
```

Následující vstupy jsou použity pro konfiguraci databázového jádra Microsoft Jet.

Vstup	Popis
PageTimeout	Časový úsek mezi uložením dat, která nejsou uzamčena pro čtení, do vnitřní mezipaměti a jejich znehodnocením, vyjádřený v jednotkách dlouhých 100 milisekund. Výchozí hodnota je buď 5 jednotek nebo 0,5 sekundy. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu DWORD a pro systém Windows NT 3.51 typu REG_DWORD.
LockedPageTimeout	Časový úsek mezi uložením dat, která jsou uzamčena pro čtení, do vnitřní mezipaměti a jejich znehodnocením, vyjádřený v jednotkách dlouhých 100 milisekund. Výchozí hodnota je buď 5 jednotek nebo 0,5 sekundy. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu DWORD a pro systém Windows NT 3.51 typu REG_DWORD.
CursorTimeout	Doba trvání odkazu na stránku, po kterou zůstává touto stránkou, vyjádřená v jednotkách dlouhých 100 milisekund. Výchozí hodnota je buď 5 jednotek nebo 0,5 sekundy. Toto nastavení platí pouze pro databáze vytvořené databázovým jádrem Microsoft Jet verze 1.x. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu DWORD a pro systém Windows NT 3.51 typu REG_DWORD.
LockRetry	Počet pokusů o opakování přístupu k uzamčené stránce před zobrazením zprávy o konfliktu uzamčení. Výchozí hodnota je 20 opakování. Vstup LockRetry odvolává na vstup CommitLockRetry. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu DWORD a pro systém Windows NT 3.51 typu REG_DWORD.
CommitLockRetry	Počet pokusů databázového jádra Microsoft Jet o uzamčení propojených dat z důvodu jejich změny. Jestliže se databázovému jádru Microsoft Jet nepodaří uzamknout propojení, změny dat budou neúspěšné. Počet pokusů databázového jádra Microsoft Jet o získání uzamčení propojených dat se přímo vztahuje na hodnotu vstupu LockRetry. Po každé, když databázové jádro Microsoft Jet bude vyžadovat uzamčení propojení, bude provedeno tolik pokusů kolik je je jich určeno hodnotou vstupu LockRetry. Jestliže je například hodnota vstupu CommitLockRetry nastavena na 20 a hodnota vstupu LockRetry na 20, potom se bude databázové jádro Microsoft Jet pokoušet o získání uzamčení spojení dvacetkrát a při každém tomto pokusu se bude pokoušet o uzamčení také až dvacetkrát, bude tedy učiněno celkem 400 pokusů.

	Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu DWORD a pro systém Windows NT 3.51 typu REG_DWORD.
MaxBufferSize	Velikost vnitřní mezipaměti databázového jádra vyjádřená v kilobajtech (kB). Hodnota vstupu MaxBufferSize musí být celé číslo v rozmezí 9 až 4096 včetně. Výchozí hodnota je 512. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu DWORD a pro systém Windows NT 3.51 typu REG_DWORD.
ReadAheadPages	Počet stránek k načtení při procesu prohledávání vpřed. Výchozí hodnota je 16. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu DWORD a pro systém Windows NT 3.51 typu REG_DWORD.
ForceOSFlush	Jakékoliv jiné nastavení než 0 znamená, že propojení nebo zápis provede vyprázdnění obsahu mezipaměti operačního systému na disk. Nastavení 0 (výchozí nastavení) znamená, že žádné vyprázdnění mezipaměti nebude provedeno. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu DWORD a pro systém Windows NT 3.51 typu REG_DWORD.
IdleFrequency	Časový úsek vyjádřený jednotkách 100 milisekund, po který bude jádro Microsoft Jet čekat před ukončením uzamčení pro čtení. Výchozí hodnota je 10 jednotek nebo jedna sekunda. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu DWORD a pro systém Windows NT 3.51 typu REG_DWORD.

Formáty ISAM databáze Microsoft Jet

Složka Jet\3.5\ISAM Formats\ Jet 2.x obsahuje následující vstupy.

Název vstupu	Datový typ pro systém Windows NT 3.51	Datový typ pro systémy Windows 95 a Windows NT 4.0	Hodnota
Engine	REG_SZ	String	Jet 2.x
OneTablePerFile	REG_BINARY	Binary	00
IndexDialog	REG_BINARY	Binary	00
CreateDBOnExport	REG_BINARY	Binary	00
IsamType	REG_DWORD	DWORD	0

Poznámka Aby se projevily změny nastavení registru systému Windows, musíte opustit a potom znovu spustit databázové jádro.

Inicializace ovladače databázového jádra Microsoft Jet 3.5

{ewc HLP95EN.dll, DYNALINK, "Viz takďž˝":"dahowChangingJetEngine30C "} {ewc HLP95EN.dll, DYNALINK, "Specifika":"dahowChangingJetEngine30S "}

Následující oddíly popisují nastavení inicializace a formátů ISAM ovladače databázového jádra Microsoft Jet 3.5.

Při registraci souborů MSJET35.DLL a MSRD2X35.DLL jsou do složky \HKEY_LOCAL_MACHINES\Software\Microsoft\Jet\3.5\Engines zapsány dva vstupy. Toto je provedeno automaticky při instalaci programu Microsoft Access 97. Jsou to tyto dva následující vstupy:

```
SystemDB = <path>\System.mdb
CompactBYPkey = 1
```

Databázové jádro Microsoft Jet používá následující vstupy:

Vstup	Popis
SystemDB	Určřuje plnou cestu a název souboru informací o pracovní skupině. Výchozí nastavení je příslušná cesta následovaná názvem souboru System.mdb. Hodnoty vstupu pro systémy Windows 95 a Windows NT 4.0 jsou datového typu String a pro systém Windows NT 3.51 typu REG_SZ.
CompactByPKey	Určuje umístění komprimovaných tabulek, které jsou kopírovány v pořadí podle primárního klíče, jestliže primární klíč existuje. Jestliže pro tabulku neexistuje primární klíč, pak jsou kopírovány v pořadí základní tabulky. Hodnota 0 naznačuje, že by tabulky měly být komprimovány v pořadí základní tabulky a nenulová hodnota znamená, že by měly být komprimovány v pořadí primárního klíče, jestliže primární klíč existuje. Výchozí hodnota je nenulová. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu DWORD a pro systém Windows NT 3.51 typu REG_DWORD. **Poznámka** Toto nastavení se týká pouze komprimace databází vytvořených databázovým jádrem Microsoft Jet verze 3.0 nebo vyšším. Jestliže komprimujete databáze vytvořené databázovým jádrem Microsoft Jet verze 2.x, data jsou vždy kopírována v pořadí základní tabulky.

Nastavení inicializace databázového jádra Microsoft Jet

Registrace programu Microsoft Jet zároveň vytváří ve složce \HKEY_LOCAL_MACHINES\Software\Microsoft\Jet\3.5\Engines\Jet 3.5 následující vstupy.

Typická nastavení pro vstupy ve složce Jet\3.5\Engines\Jet 3.5 jsou ukázána v následujícím příkladu.

```
FlushTransactionTimeout=500
LockDelay=100
LockRetry=20
MaxBufferSize= 0
MaxLocksPerFile= 9500
PageTimeout=5000
Threads=3
```

```
UserCommitSync=Yes
ImplicitCommitSync=No
ExclusiveAsyncDelay=2000
SharedAsyncDelay=0
```

Databázové jádro Microsoft Jet používá následující vstupy.

Vstup	Popis
PageTimeout	Časový úsek mezi uložením dat, která nejsou uzamčena pro čtení, do vnitřní mezipaměti a jejich znehodnocením vyjádřený v milisekundách. Výchozí hodnota je buď 5000 jednotek nebo 5 sekund. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu DWORD a pro systém Windows NT 3.51 typu REG_DWORD.
FlushTransactionTimeout	Tento vstup znemožňuje provedení registrových vstupů ExclusiveAsyncDelay a SharedAsyncDelay. Pro umožnění provedení těchto vstupů se musí zadat hodnota 0. Vstup FlushTransactionTimeout mění metodu asynchronního zápisu databázového jádra Microsoft Jet do databázového souboru. V předešlém případě by databázové jádro Microsoft Jet použilo k určení doby čekání před provedením asynchronního zápisu buď vstup ExclusiveAsyncDelay nebo SharedAsyncDelay. Vstup FlushTransactionTimeout mění tento proces tak, že má nyní zadanou hodnotu, která určuje dobu, po jejímž vypršení se započne asynchronní zápis v případě, že nebyly do mezipaměti přidány žádné stránky. Jediná výjimka nastává v případě, že obsah mezipaměti překročil hodnotu zadanou vstupem MaxBufferSize, a tak mezipaměť spustí asynchronní zápis bez ohledu na to, jestli zadaný čas již vypršel. Databázové jádro Microsoft Jet 3.5 bude tedy setrvávat v nečinnosti buď 500 milisekund nebo do doby přeplnění mezipaměti před spuštěním asynchronního zápisu.
LockDelay	Nastavení tohoto vstupu spolupracuje se vstupem LockRetry a způsobuje, že každý tento vstup vyčká 100 milisekund před provedením dalšího požadavku uzamčení. Nastavení vstupu LockDelay bylo přidáno, aby zabránilo „předávkování", které by nastalo ve spolupráci s některými síťovými operačními systémy.
MaxLocksPerFile	Toto nastavení chrání transakce jádra Microsoft Jet před překročením zadané hodnoty. Jestliže se uzamčení v transakci pokusí překročit tuto hodnotu, pak je transakce rozdělena do dvou či více částí a po těchto částech je také propojena. Toto nastavení bylo přidáno, aby ochraňovalo server Netware 3.1 před kolizemi v případě, že by zadaný limit uzamčení serveru Netware byl překročen a aby zvýšilo výkon obou serverů, Netware i Windows NT.
LockRetry	Počet pokusů o opakování přístupu k uzamčené stránce před zobrazením zprávy o konfliktu

	uzamčení. Výchozí hodnota je 20 opakování. Vstup LockRetry odvolává na vstup CommitLockRetry. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu DWORD a pro systém Windows NT 3.51 typu REG_DWORD.
RecycleLVs	Je-li zadáno toto nastavení, jádro Microsoft Jet bude obnovovat dlouhé názvy (LV) stránek (Memo, Long Binary [objekt OLE], a datové typy Binary). Program Microsoft Jet 3.0 tyto typy stránek neuloží, dokud poslední uživatel neuzavře databázi. Jestliže je zadáno nastavení RecyleLVs, jádro Microsoft Jet 3.5 začne obnovovat stránky LV v okamžiku, kdy je databáze rozšiřována, tj. když se přidávají nové skupiny stránek. **Poznámka** Při použití této funkce si uživatelé během práce s daty s dlouhými jmény povšimnou snížení výkonu aplikace. Program Microsoft Access 97 tuto funkci při práci s moduly, formuláři a sestavami automaticky nastaví nebo potlačí, čímž odpadne nutnost jejich nastavování během úprav těchto objektů. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu DWORD a pro systém Windows NT 3.51 typu REG_DWORD.
SortMemorySource	Toto nastavení určuje jakým způsobem bude databázové jádro Microsoft Jet přidělovat paměť během operací řazení. Výchozí hodnota je použití nativního volání vstupu Win32. Potlačení nastavení vstupu SortMemorySource vyvolá metodu překladače pro přidělování paměti během operací třídění. Testy využití paměti prokázaly, že výchozí nastavení vede k menší spotřebě paměti. Výchozí nastavení má hodnotu 0, které znamená, že databázové jádro Microsoft Jet použije nativní volání vstupu Win32. Změna nastavení na hodnotu 1 bude znamenat, že databázové jádro Microsoft Jet použije k přidělování paměti metodu VisualC. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu DWORD a pro systém Windows NT 3.51 typu REG_DWORD.
MaxBufferSize	Velikost vnitřní mezipaměti databázového jádra vyjádřená v kilobajtech (kB). Hodnota vstupu MaxBufferSize musí být celé číslo větší nebo rovné 512. Výchozí hodnota se získá z následujícího vzorce: ((Celkem paměti RAM v MB - 12 MB) / 4) + 512 KB Například v systému s pamětí RAM o velikosti 32 MB je výchozí nastavení velikosti vyrovnávací paměti ((32 MB - 12 MB) / 4) + 512 kB nebo 5632 kB. Aby se tato hodnota nastavila do klíče registru jako výchozí, zadá se do vstupu: `MaxBufferSize=` Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu DWORD a pro

	systém Windows NT 3.51 typu REG_DWORD.
Threads	Nastavení tohoto vstupu udává počet podprocesů na pozadí, které jsou dostupné databázovému jádru Microsoft Jet. Hodnota výchozího nastavení je 3. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu DWORD a pro systém Windows NT 3.51 typu REG_DWORD.
UserCommitSync	Nastavení tohoto vstupu určuje, zda má systém čekat na ukončení propojení. Hodnota Ano dává systému instrukci aby čekal a hodnota Ne aby systém provedl propojení asynchronně. Hodnoty vstupu pro systémy Windows 95 a Windows NT 4.0 jsou datového typu String a pro systém Windows NT 3.51 typu REG_SZ.
ImplicitCommitSync	Nastavení tohoto vstupu určuje, zda má systém čekat na ukončení propojení. Hodnota Ne dává systému instrukci aby pkračoval v činnosti bez čekání na ukončení propojení a hodnota Ano aby čekal na ukončení propojení. Hodnoty vstupu pro systémy Windows 95 a Windows NT 4.0 jsou datového typu String a pro systém Windows NT 3.51 typu REG_SZ.
ExclusiveAsyncDelay	Určuje časový úsek v milisekundách, po který se odkládá asynchronní zápis do databáze otevřené s výhradním přístupem. Výchozí hodnota je buď 2000 nebo 2 sekundy. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu DWORD a pro systém Windows NT 3.51 typu REG_DWORD.
SharedAsyncDelay	Určuje časový úsek v milisekundách, po který se odkládá asynchronní zápis do sdílené databáze. Hodnota výchozího nastavení je 0. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu DWORD a pro systém Windows NT 3.51 typu REG_DWORD.

Formáty ISAM databázového jádra Microsoft Jet

Složka Jet\3.5\ISAM Formats obsahuje následující vstupy.

Název vstupu	Datový typ pro systém Windows NT 3.51	Datový typ pro systémy Windows 95 a Windows NT 4.0	Hodnota
Engine	REG_SZ	String	Jet 3.5
ExportFilter	REG_SZ	String	Microsoft Access (*.mdb)
ImportFilter	REG_SZ	String	Microsoft Access (*.mdb)
CanLink	REG_BINARY	Binary	00
OneTablePerFile	REG_BINARY	Binary	00
IndexDialog	REG_BINARY	Binary	00

CreateDBOnExport	REG_BINARY	Binary	00
ResultTextImport	REG_SZ	String	Import dat z externího souboru do aktuální databáze. Změna dat v aktuální databázi nebude mít vliv na data v externím souboru.
ResultTextLink	REG_SZ	String	Vytvoří tabulku v aktuální databázi, která je propojena s externím souborem. Změna dat v aktuální databázi změní data v externím souboru.
ResultTextExport	REG_SZ	String	Export dat z aktuální databáze do databáze programu Microsoft Access. Tento proces přepíše data, jestliže jsou exportována do existujícího souboru.

Poznámka Aby se projevily změny nastavení registru systému Windows, musíte opustit a potom znovu spustit databázové jádro.

Konfigurace databázového jádra Microsoft Jet pro objekty ODBC Access

{ewc HLP95EN.dll, DYNALINK, "Viz takďž˝":"dahowChangingMicrosoftODBCC "} {ewc HLP95EN.dll, DYNALINK, "Specifika":"dahowChangingMicrosoftODBCS "}

Následující oddíly popisují nastavení registru systému Windows pro databázové jádro Microsoft Jet pro spojení s databázemi ODBC.

Nastavení inicializace datábázového jádra Microsoft Jet pro spojení s databázemi ODBC

Složka \HKEY_LOCAL_MACHINE\Software\Microsoft\Jet\3.5\Engines\ODBC pbsahuje nastavení inicializace pro databázové jádro Microsoft Jet.

Poznámka Typická nastavení pro vstupy ve složce Jet\3.5\Engines\ODBC jsou ukázány v následujícím příkladu.

```
LoginTimeout=20
QueryTimeout=60
ConnectionTimeout=600
AsyncRetryInterval=500
AttachCaseSensitive=0
AttachableObjects='TABLE','VIEW','SYSTEM TABLE','ALIAS','SYNONYM'
SnapshotOnly=0
TraceSQLMode=0
TraceODBCAPI=0
DisableAsync=1
JetTryAuth=1
PreparedInsert=0
PreparedUpdate=0
FastRequery=0
```

Databázové jádro Microsoft Jet používá ODBC vstupy jak je ukázáno dále.

Vstup	Popis
LoginTimeout	Počet sekund, po který může pokračovat pokus o přihlášení před vypršením času. Výchozí hodnota nastavení je 20 (hodnota datového typu REG_DWORD).
QueryTimeout	Počet sekund, po který může být spuštěný dotaz (celkový čas procesu) před vypršením času. Je-li nastavení vstupu DisableAsync=0 (výchozí nastavení), potom nastavení vstupu QueryTimeout je počet sekund, po který se má čekat na odezvu serveru výzvami pro dokončení dotazu. Výchozí hodnota nastavení je 60 (hodnoty jsou datového typu REG_DWORD).
ConnectionTimeout	Počet sekund, po který může zůstat spojení spravované v mezipaměti v nečinnosti před vypršením času. Výchozí hodnota nastavení je 600 (hodnoty jsou datového typu REG_DWORD).
AsyncRetryInterval	Počet milisekund mezi výzvami, které informují, zda server dokončil proces zpracování dotazu. Tento vstup se používá pouze pro asynchronní procesy. Výchozí hodnota nastavení je 500 (hodnoty jsou datového typu REG_DWORD).

AttachCaseSensitive	Indikátor přesné shody jmen tabulek během propojení. Může nabývat hodnoty 0 (propojí první tabulku odpovídající zadanému jménu bez ohledu na další aspekty) a hodnoty 1 (propojí tabulku pouze v případě, že se jména shodují přesně). Hodnota výchozího nastavení je 0 (hodnoty jsou datového typu REG_DWORD).
AttachableObjects	Seznam objektů typu server, se kterými bude dostupné propojení. Výchozí nastavení je: 'TABLE', 'VIEW', 'SYSTEM TABLE', 'ALIAS', 'SYNONYM' (hodnoty jsou datového typu REG_SZ).
SnapshotOnly	Indikátor stavu, zda se vynucuje, aby se objekty **Recordset** staly typem snímek. Nastavení může nabývat hodnoty 0 (umožňuje dynamické sady) a hodnoty 1 (vynucuje pouze typ snímek). Hodnota výchozího nastavení je 0 (hodnoty jsou datového typu REG_DWORD).
TraceSQLMode	Indikátor stavu, zda bude databázové jádro Microsoft Jet sledovat SQL vstupy vyslané zdroji dat ODBC v souboru SQLOUT.txt. Nastavení může nabývat hodnoty 0 (nebude) a 1 (bude). Hodnota výchozího nastavení je 0 (hodnoty jsou datového typu REG_DWORD).Tento vstup je zaměnitelný se vstupem SQLTraceMode.
TraceODBCAPI	Indikátor stavu, zda se bude trasovat ODBC API volané v souboru ODBCAPI.txt. Nastavení může nabývat hodnoty 0 (nebude) a 1 (bude). Hodnota výchozího nastavení je 0 (hodnoty jsou datového typu REG_DWORD).
DisableAsync	Indikátor stavu, zda se bude vynucovat synchronní spuštění dotazu. Nastavení může nabývat hodnoty 0 (použití asynchronního spuštění dotazu je možné) a 0 (vynutí synchronní spuštění dotazu). Hodnota výchozího nastavení je 1 (hodnoty jsou datového typu REG_DWORD).
JetTryAuth	Indikátor stavu, zda se bude pokoušet použití uživatelského jména programu Microsoft Access k přihlášení k serveru ještě před výzvou. Nastavení může nabývat hodnoty 0 (nebude) a 1 (bude). Hodnota výchozího nastavení je 0 (hodnoty jsou datového typu REG_DWORD).
PreparedInsert	Indikátor stavu, zda se bude používat připravený vstup INSERT, který vkládá data do všech sloupců. Nastavení může nabývat hodnoty 0 (bude se používat uživatelský vstup INSERT, který bude vkládat pouze hodnoty, které nejsou typu **Null**) a 1 (bude se používat připravený vstup INSERT). Hodnota výchozího nastavení je 0 (hodnoty jsou datového typu REG_DWORD). Použití připravených vstupů INSERT může způsobit výskyt hodnot **Nulls**, které přepíší výchozí nastavení serveru a mohou způsobit nepředvídatelné chyby při práci se sloupci, které

	nebyly vloženy explicitně.
PreparedUpdate	Indikátor stavu, zda se bude používat připravený vstup UPDATE, který aktualizuje data ve všech sloupcích. statement that updates data in all columns. Nastavení může nabývat hodnoty 0 (bude se používat uživatelský vstup UPDATE, který nastavuje pouze sloupce, které byly změněny) a 1 (bude se používat připravený vstup UPDATE). Hodnota výchozího nastavení je 0 (hodnoty jsou datového typu REG_DWORD). Použití připravených vstupů INSERT může způsobit nepředvídatelné chyby při práci se sloupci, které nebyly změněny.
FastRequery	Indikátor stavu, zda se bude používat připravený vstup SELECT pro parametrizované dotazy. Nastavení může nabývat hodnoty 0 (nebude) a 1 (bude). Hodnota výchozího nastavení je 0 (hodnoty jsou datového typu REG_DWORD).

Poznámka Aby se projevily změny nastavení registru systému Windows, musíte opustit a potom znovu spustit databázové jádro.

Inicializace ovladače databáze Paradox

{ewc HLP95EN.dll, DYNALINK, "Viz takďž˝":"dahowChangingParadoxC "} {ewc HLP95EN.dll, DYNALINK, "Specifika":"dahowChangingParadoxS "}

Při instalaci ovladače databáze Paradox zapíše program Setup sadu výchozích hodnot do registru systému Windows v podklíčích Jádra a Formáty ISAM. Pokud to nebude nutné, byste neměli tato nastavení přímo upravovat, raději použijte program Setup pro přidávání, odstraňování nebo pozměňování nastavení vaší aplikace. Následující oddíly popisují nastavení inicializace a formátů ISAM ovladače databáze Microsoft Paradox.

Nastavení inicializace databáze Paradox

Složka Jet\3.5\Engines\ Paradox obsahuje nastavení inicializace pro ovladač Mspdox35.dll, který se používá pro přístup k externím datům ve formátu Paradox. Typická nastavení pro vstupy v této složce jsou znázorněny v následujícím příkladu.

```
win32=<path>\MSPDOX35.dll
PageTimeout=600
CollatingSequence=ASCII
DataCodePage=OEM
ParadoxUserName=Kimberly
ParadoxNetPath=P:\PDOXDB
ParadoxNetStyle=4.X
```

Databázové jádro Microsoft Jet používá vstupy složky Paradox jak je ukázáno dále.

Vstup	Popis
win32	Umístění ovladače Mspdox35.dll. Plná cesta je určena během instalace. Hodnoty vstupu pro systémy Windows 95 a Windows NT 4.0 jsou datového typu String a pro systém Windows NT 3.51 typu REG_SZ.
PageTimeout	Časový úsek mezi uložením dat do vnitřní mezipaměti a jejich znehodnocením. Tato hodnota se zadává v jednotkách dlouhých 100 milisekund. Výchozí hodnota je buď 600 jednotek nebo 60 sekund. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu DWORD a pro systém Windows NT 3.51 typu REG_DWORD.
CollatingSequence	Řadící sekvence pro všechny tabulky Paradox, vytvořená nabo otevřená databázovým jádrem Microsoft Jet. Přípustné hodnoty vstupu jsou typu ASCII, International, Norwegian-Danish a Swedish-Finnish. Výchozí hodnota je ASCII. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 typu String a pro systém Windows NT 3.51 typu REG_SZ.
DataCodePage	Indikátor způsobu jakým se ukládají stránky textu. Možná nastavení jsou: • OEM - Konverze OemToAnsi a AnsiToOem jsou dokončeny. • ANSI - Konverze OemToAnsi a AnsiToOem nejsou dokončeny. Výchozí hodnota vstupu je OEM. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu String a pro systém Windows NT 3.51 typu REG_SZ.

ParadoxUserName

Jméno, které má být aplikací Paradox zobrazeno, jestliže je tabulka uzamčena ovladačem Paradox ISAM a interaktivní uživatel, který má přístup k datům aplikace Paradox (spíše než ovladač ISAM), se pokouší o jejich nekompatibilní uzamčení. Jestliže počítač není připojen k síti, tento vstup se nepřidává. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu String a pro systém Windows NT 3.51 typu REG_SZ.

Poznámka Jestliže zadáváte vstup ParadoxUserName, pak také musíte zadat vstupy ParadoxNetPath a ParadoxNetStyle, jinak obdržíte chybové hlášení, jakmile se pokusíte o přístup k externím datům aplikace Paradox. Jestliže přistupujete k databázi Paradox ve víceuživatelském režimu prostřednictvím sítě, musíte tento vstup přidat nebo jej v registru ručně upravit.

ParadoxNetPath

Plná cesta do adresáře obsahujícího soubor PARADOX.NET (pro Paradox verze 3.x) nebo PDOXUSRS.NET (pro Paradox verze 4.x). Jestliže počítač není připojen k síti, tento vstup se nepřidává. Obvykle budete potřebovat změnit inicializační nastavení (přidané programem Setup), které nejlépe určí, kde by se soubor měl nacházet. Plná cesta určená vstupem ParadoxNetPath (včetně označení jednotky) musí být shodná pro všechny uživatele, kteří jednotlivě sdílí databázi (adresář). Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu String a pro systém Windows NT 3.51 typu REG_SZ.

Poznámka Jestliže zadáte vstup ParadoxNetPath, pak také musíte zadat vstupy ParadoxUserName a ParadoxNetStyle, jinak obdržíte chybové hlášení, jakmile se pokusíte o přístup k externím datům aplikace Paradox. Jestliže přistupujete k databázi Paradox ve víceuživatelském režimu prostřednictvím sítě, musíte tento vstup přidat nebo jej v registru ručně upravit.

ParadoxNetStyle

Způsob síťového přístupu, který se má použít pro přístup k datům aplikace Paradox. Přípustné hodnoty jsou:

- 3.x
- 4.x

Poznámka Tento vstup nemohou uživatelé programu Paradox verze 3.x nastavovat na hodnotu 4.x, protože by ovladač používal nesprávnou uzamykací metodu. Uživatelé programu Paradox verze 5.0 musí hodnotu vstupu ParadoxNetStyle nastavit na 4.x, aby zajistili správnou funkci uzamykání.

Jestliže počítač není připojen k síti, tento vstup se nepřidává. Hodnota tohoto vstupu by měla odpovídat verzi programu Paradox, kterou používají uživatelé dané skupiny a musí být shodná pro všechny

uživatele, kteří jednotlivě sdílí databázi (adresář). Výchozí nastavení je hodnota 4.x. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu String a pro systém Windows NT 3.51 typu REG_SZ.

Poznámka Jestliže zadáte vstup ParadoxNetStyle, pak také musíte zadat vstupy ParadoxUserName a ParadoxNetPath, jinak obdržíte chybové hlášení, jakmile se pokusíte o přístup k externím datům aplikace Paradox.

Formáty ISAM databáze Paradox

Složka Jet\3.5\ISAM Formats\Paradox 3.x obsahuje následující vstupy.

Název vstupu	Datový typ pro systém Windows NT 3.51	Datový typ pro systémy Windows 95 a Windows NT 4.0	Hodnota
Engine	REG_SZ	String	Paradox
ExportFilter	REG_SZ	String	Paradox 3 (*.db)
ImportFilter	REG_SZ	String	Paradox (*.db)
CanLink	REG_BINARY	Binary	01
OneTablePerFile	REG_BINARY	Binary	01
IsamType	REG_DWORD	DWORD	0
IndexDialog	REG_BINARY	Binary	00
CreateDBOnExport	REG_BINARY	Binary	00
ResultTextImport	REG_SZ	String	Import dat z externího souboru do aktuální databáze. Změna dat v aktuální databázi nebude mít vliv na data v externím souboru.
ResultTextLink	REG_SZ	String	Vytvoří tabulku v aktuální databázi, která je propojena s externím souborem. Změna dat v aktuální databázi změní data v externím souboru.
ResultTextExport	REG_SZ	String	Export dat z aktuální databáze do souboru ve formátu Paradox 3. Tento proces přepíše data, jestliže jsou exportována do existujícího souboru.

Složka Jet\3.5\ISAM Formats\Paradox 4.x obsahuje následující vstupy.

Název vstupu	Datový typ pro systém Windows NT 3.51	Datový typ pro systémy Windows 95 a Windows NT 4.0	Hodnota
Engine	REG_SZ	String	Paradox
ExportFilter	REG_SZ	String	Paradox 4 (*.db)

CanLink	REG_BINARY	Binary	01
OneTablePerFile	REG_BINARY	Binary	01
IsamType	REG_DWORD	DWORD	0
IndexDialog	REG_BINARY	Binary	00
CreateDBOnExport	REG_BINARY	Binary	00
ResultTextExport	REG_SZ	String	Export dat z aktuální databáze do souboru ve formátu Paradox 4. Tento proces přepíše data, jestliže jsou exportována do existujícího souboru.

Složka Jet\3.5\ISAM Formats\Paradox 5.x obsahuje následující vstupy.

Název vstupu	**Datový typ pro systém Windows NT 3.51**	**Datový typ pro systémy Windows 95 a Windows NT 4.0**	**Hodnota**
Engine	REG_SZ	String	Paradox
ExportFilter	REG_SZ	String	Paradox 5 (*.db)
CanLink	REG_BINARY	Binary	01
OneTablePerFile	REG_BINARY	Binary	01
IsamType	REG_DWORD	DWORD	0
IndexDialog	REG_BINARY	Binary	00
CreateDBOnExport	REG_BINARY	Binary	00
ResultTextExport	REG_SZ	String	Export dat z aktuální databáze do souboru ve formátu Paradox 4. Tento proces přepíše data, jestliže jsou exportována do existujícího souboru.

Poznámka Aby se projevily změny nastavení registru systému Windows, musíte opustit a potom znovu spustit databázové jádro.

Inicializace ovladače databáze textového zdroje dat

{ewc HLP95EN.dll, DYNALINK, "Viz takďž˝":"dahowChangingTextC "} {ewc HLP95EN.dll, DYNALINK, "Specifika":"dahowChangingTextS "}

Při instalaci ovladače textového zdroje dat zapíše program Setup sadu výchozích hodnot do registru systému Windows v podklíčích Jádra a Formáty ISAM. Pokud to nebude nutné, byste neměli tato nastavení přímo upravovat, raději použijte program Setup pro přidávání, odstraňování nebo pozměňování nastavení vaší aplikace. Následující oddíly popisují nastavení inicializace a formátů ISAM ovladače databáze textového zdroje dat.

Nastavení inicializace databáze textového zdroje dat

Složka Jet\3.5\Engines\Text obsahuje nastavení inicializace pro ovladač Mstext35.dll, který se používá pro přístup k textovým datovým souborům. Typická nastavení pro vstupy v této složce jsou znázorněny v následujícím příkladu.

```
win32=<path>\MSTEXT35.dll
MaxScanRows=25
FirstRowHasNames=True
CharacterSet=OEM
Format=CSVDelimited
Extensions=none,asc,csv,tab,txt
ExportCurrencySymbols=Yes
```

Databázové jádro Microsoft Jet používá vstupy složky Text jak je ukázáno dále.

Vstup	Popis
win32	Umístění ovladače Mstext35.dll. Plná cesta je určena během instalace. Hodnoty vstupu pro systémy Windows 95 a Windows NT 4.0 jsou datového typu String a pro systém Windows NT 3.51 typu REG_SZ.
MaxScanRows	Počet řádků, které mají být zkontrolovány na datový typ. Je-li nastavena hodnota 0, bude se prohledávat celý soubor. Hodnota výchozího nastavení je 25. Hodnoty vstupu pro systémy Windows 95 a Windows NT 4.0 jsou datového typu DWORD a pro systém Windows NT 3.51 typu REG_DWORD.
FirstRowHasNames	Binární hodnota, která naznačuje, zda první řádek tabulky obsahuje názvy sloupců. Hodnota nastavená na 01 znamená, že během importu dat jsou názvy sloupců brány z prvního řádku. Hodnota nastavená na 00 znamená, že v prvním řádku nejsou žádné názvy sloupců. Výchozí hodnota nastavení je 01. Hodnoty vstupu pro systémy Windows 95 a Windows NT 4.0 jsou typu Binary a pro systém Windows NT 3.51 datového typu REG_ BINARY.
CharacterSet	Indikátor způsobu jakým se ukládají stránky textu. Možná nastavení jsou: • OEM - Konverze OemToAnsi a AnsiToOem jsou dokončeny. • ANSI - Konverze OemToAnsi a AnsiToOem nejsou dokončeny. Výchozí hodnota vstupu je OEM. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0

	datového typu String a pro systém Windows NT 3.51 typu REG_SZ.
Format	Hodnota může být jedna z následujících: TabDelimited, CSVDelimited a Delimited (<jednoduchý znak>). Jednoznakový oddělovač v hodnotě Delimited může být libovolný jednoduchý znak kromě uvozovek " . Hodnota výchozího nastavení je CSVDelimited. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu String a pro systém Windows NT 3.51 typu REG_SZ.
Extensions	Přípona, podle níž je soubor vyhledáván při hledání textových dat. Výchozí hodnoty jsou txt, csv, tab a asc. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu String a pro systém Windows NT 3.51 typu REG_SZ.
ExportCurrencySymbols	Binární hodnota, která označuje, zda je příslušný symbol měny zahrnut při exportu aktuálních polí. Hodnota 01 označuje, že symbol je zahrnut a hodnota 00 označuje, že jsou exportována pouze číselná data. Hodnota výchozího nastavení je 01. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu Binary a pro systém Windows NT 3.51 typu REG_BINARY.

Formáty ISAM databáze textového zdroje dat

Složka Jet\3.5\ISAM Formats\ Text obsahuje následující vstupy.

Název vstupu	**Datový typ pro systém Windows NT 3.51**	**Datový typ pro systémy Windows 95 a Windows NT 4.0**	**Hodnota**
Engine	REG_SZ	String	Text
ExportFilter	REG_SZ	String	Text Files (*.txt; *.csv; *.tab; *.asc)
ImportFilter	REG_SZ	String	Text Files (*.txt; *.csv; *.tab; *.asc)
CanLink	REG_BINARY	Binary	01
OneTablePerFile	REG_BINARY	Binary	01
IsamType	REG_DWORD	DWORD	2
IndexDialog	REG_BINARY	Binary	00
CreateDBOnExport	REG_BINARY	Binary	00
ResultTextImport	REG_SZ	String	Import dat z externího souboru do aktuální databáze. Změna dat v aktuální databázi nebude mít vliv na data v externím souboru.
ResultTextLink	REG_SZ	String	Vytvoří tabulku v aktuální databázi, která je propojena

			s externím souborem. Změna dat v aktuální databázi změní data v externím souboru.
ResultTextExport	REG_SZ	String	Export dat z aktuální databáze do textového souboru. Tento proces přepíše data, jestliže jsou exportována do existujícího souboru.

Poznámka Aby se projevily změny nastavení registru systému Windows, musíte opustit a potom znovu spustit databázové jádro.

Přizpůsobení souboru Schema.ini

Pro čtení, import nebo export textových dat musíte kromě přidání informací ovladače Text ISAM do souboru .ini vytvořit navíc soubor schema.ini. Soubor schema.ini obsahuje specifické informace o textovém zdroji dat jako jsou: formát textového souboru, způsob načítání během importu a výchozí exportní formát souborů. Následující příklad ukazuje uspořádání pro soubor s pevnou šířkou nazvaný Filename.txt:

```
[Filename.txt]
ColNameHeader=False
Format=FixedLength
MaxScanRows=25
CharacterSet=OEM
Col1=columnname Char Width 24
Col2=columnname2 Date Width 9
Col3=columnname7 Float Width 10
Col4=columnname8 Integer Width 10
Col5=columnname9 LongChar Width 10
```

Podobně vypadá uspořádání pro soubor bez omezení:

```
[Delimit.txt]
ColNameHeader=True
Format=Delimited(!)
MaxScanRows=0
CharacterSet=OEM
Col1=username char width 50
Col2=dateofbirth Date width 9
```

Jestliže exportujete data do souboru bez omezení, zadejte formát souboru jak je ukázáno dále:

```
[Export: My Special Export]
ColNameHeader=True
Format=TabDelimited
MaxScanRows=25
CharacterSet=OEM
DateTimeFormat=mm.dd.yy.hh.mm.ss
CurrencySymbol=Dm
CurrencyPosFormat=0
CurrencyDigits=2
CurrencyNegFormat=0
CurrencyThousandSymbol=,
CurrencyDecimalSymbol=.
```

```
DecimalSymbol=,
NumberDigits=2
NumberLeadingZeros=0
```

Příklad My Special Export odkazuje na určité volby pro export: během doby spojení si můžete určit libovolnou variantu voleb pro export. Poslední příklad také odpovídá jménu zdroje dat (DSN), který může být v době spojení volitelně předáván. Všechny tři oddíly formátů mohou být začleněny do stejného souboru .ini.

Databázové jádro Microsoft Jet používá vstupy souboru Schema.ini jak je ukázáno dále.

Vstup	**Popis**
ColNameHeader	Hodnota může být nastavena buď na **True** (ukazuje, že data v prvním záznamu znamenají názvy sloupců) nebo na **False**.
Format	Hodnota může nastavena do jednoho z následujících tvarů: TabDelimited, CSVDelimited, Delimited (<jednoduchý znak>) nebo FixedLength. Oddělovač v hodnotě Delimited může být libovolný jednoduchý znak kromě uvozovek " .
MaxScanRows	Označuje počet řádků, které mají být zkontrolovány na datový typ. Je-li nastavena hodnota 0, bude se prohledávat celý soubor.
CharacterSet	Hodnota může být nastavena buď na OEM nebo na ANSI, čímž naznačuje, zda je zdrojový soubor zapisován pomocí kódové stránky OEM nebo ANSI.
DateTimeFormat	Hodnota může být nastavena na formátovací řetězec označující datum a čas. Tento vstup by měl být zadán v případě, že všechna pole typu datum/čas pro import a export jsou ošetřována ve stejném formátu. Databázové jádro Microsoft Jet pracuje se všemi formáty kromě AM a PM. V případě, že formátovací řetězec schází, je užito zkráceného tvaru data a času podle nastavení v ovládacím panelu systému Windows.
CurrencySymbol	Označuje symbol měny, který se má použít jako aktuální hodnota v textovém souboru. Příklady obsahují znaménko dolaru $ a německé marky DM. Jestliže tento vstup chybí, použije se hodnoty nastavené ovládacím panelu systému Windows.
CurrencyPosFormat	Může být nastaveno do libovolné z následujích hodnot: Předpona symbolu měny bez oddělení ($1) Předpona symbolu měny bez oddělení(1$) Předpona symbolu měny s jednoznakovým oddělením ($ 1) Předpona symbolu měny s jednoznakovým oddělením (1 $) Jestliže tento vstup chybí, použije se hodnoty nastavené ovládacím panelu systému

	Windows.
CurrencyDigits	Určuje počet číslic použitý ve zlomkové části hodnoty měny. Jestliže tento vstup chybí, použije se hodnoty nastavené ovládacím panelu systému Windows.
CurrencyNegFormat	Může být jednou z následujích hodnot: ($1) –$1 $–1 $1– (1$) –1$ 1–$ 1$– –1 $ –$ 1 1 $– $ 1– $ –1 1– $ ($ 1) (1 $) Znaménko dolaru je zde ukázáno z důvodu ukázky, ale v aktuálním programu by mělo být nahrazeno příslušnou hodnotou ve vstupu CurrencySymbol. Jestliže tento vstup chybí, použije se hodnoty nastavené ovládacím panelu systému Windows.
CurrencyThousandSymbol	Označuje jednoznakový symbol, který se použije pro oddělování tisíců v měnových hodnotách v textovém souboru. Jestliže tento vstup chybí, použije se hodnoty nastavené ovládacím panelu systému Windows.
CurrencyDecimalSymbol	Označuje jednoznakový symbol, který se použije pro oddělování celku od zlomkové části měnové hodnoty. Jestliže tento vstup chybí, použije se hodnoty nastavené ovládacím panelu systému Windows.
DecimalSymbol	Může být nastaven libovolný jednoznakový symbol, který se použije pro oddělování celku od desetinné části čísla. Jestliže tento vstup chybí, použije se hodnoty nastavené ovládacím panelu systému Windows.
NumberDigits	Označuje počet počet desetinných míst ve zlomkové části čísla. Jestliže tento vstup chybí, použije se hodnoty nastavené ovládacím panelu systému Windows.
NumberLeadingZeros	Určuje, zda by desetinná hodnota v rozsahu -1 až 1 mohla obsahovat počáteční nulu. Tato hodnota může být buď Ano (neobsahuje

	počáteční nulu) nebo Ne.
Col1, Col2, ...	Seznam sloupců, které mají být v textovém souboru čteny. Formát tohoto vstupu by měl vypadat následovně: Col*n*=*Jméno sloupce* type [Width #] *Jméno sloupce*: Jména sloupců s vloženými mezerami by měla být uzavřena v uvozovkách. *type*: Může být Bit, Byte, Short, Long, Currency, Single, Double, DateTime, Text nebo Memo. Kromě těchto ovladač ODBC Text pracuje ještě s následujícími typy: Char (stejný jako Text) Float (stejný jako Double) Integer (stejný jako Short) LongChar (stejný jako Memo) Date *formát datumu*

Poznámka Aby se projevily změny nastavení registru systému Windows, musíte opustit a potom znovu spustit databázové jádro.

Inicializace ovladače zdroje dat HTML (Internet)

{ewc HLP95EN.dll, DYNALINK, "Viz takďž˝":"dahowChangingHTMLC "} {ewc HLP95EN.dll, DYNALINK, "Specifika":"dahowChangingHTMLS "}

Při instalaci ovladače zdroje dat HTML zapíše program Setup sadu výchozích hodnot do registru systému Windows v podklíčích Jádra a Formáty ISAM. Pokud to nebude nutné, byste neměli tato nastavení přímo upravovat, raději použijte program Setup pro přidávání, odstraňování nebo pozměňování nastavení vaší aplikace. Následující oddíly popisují nastavení inicializace a formátů ISAM ovladače zdroje dat HTML.

Nastavení inicializace zdroje dat HTML

Složka Jet\3.5\Engines\Text obsahuje nastavení inicializace pro ovladač Mstext35.dll, který se používá pro přístup k tabulkám uloženým v souborech HTML. Typická nastavení pro vstupy v této složce jsou znázorněny v následujícím příkladu.

```
win32=<path>\MSTEXT35.dll
MaxScanRows=25
FirstRowHasNames=False
CharacterSet=OEM
Format=TabDelimited
Extensions=none,asc,csv,tab,txt
ExportCurrencySymbols=Yes
```

Databázové jádro Microsoft Jet používá vstupy složky Text jak je ukázáno dále.

Vstup	Popis
win32	Umístění ovladače Mstext35.dll. Plná cesta je určena během instalace. Hodnoty vstupu pro systémy Windows 95 a Windows NT 4.0 jsou datového typu String a pro systém Windows NT 3.51 typu REG_SZ.
MaxScanRows	Počet řádků, které mají být zkontrolovány na datový typ. Je-li nastavena hodnota 0, bude se prohledávat celý soubor. Hodnota výchozího nastavení je 25. Hodnoty vstupu pro systémy Windows 95 a Windows NT 4.0 jsou datového typu DWORD a pro systém Windows NT 3.51 typu REG_DWORD.
FirstRowHas Names	Binární hodnota, která naznačuje, zda první řádek tabulky obsahuje názvy sloupců. Hodnota nastavená na 01 znamená, že během importu dat jsou názvy sloupců brány z prvního řádku. Hodnota nastavená na 00 znamená, že v prvním řádku nejsou žádné názvy sloupců. Výchozí hodnota nastavení je 01. Hodnoty vstupu pro systémy Windows 95 a Windows NT 4.0 jsou typu Binary a pro systém Windows NT 3.51 datového typu REG_ BINARY.
CharacterSet	Indikátor způsobu jakým se ukládají stránky textu. Možná nastavení jsou: • OEM - Konverze OemToAnsi a AnsiToOem jsou dokončeny. • ANSI - Konverze OemToAnsi a AnsiToOem nejsou dokončeny.

	Výchozí hodnota vstupu je OEM. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu String a pro systém Windows NT 3.51 typu REG_SZ.
Format	Hodnota může být jedna z následujících: TabDelimited, CSVDelimited a Delimited (<jednoduchý znak>). Jednoznakový oddělovač v hodnotě Delimited může být libovolný jednoduchý znak kromě uvozovek " . Hodnota výchozího nastavení je CSVDelimited. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu String a pro systém Windows NT 3.51 typu REG_SZ.
Extensions	Přípona, podle níž je soubor vyhledáván při hledání textových dat. Výchozí hodnoty jsou txt, csv, tab a asc. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu String a pro systém Windows NT 3.51 typu REG_SZ.
ExportCurrencySymbols	Binární hodnota, která označuje, zda je příslušný symbol měny zahrnut při exportu aktuálních polí. Hodnota 01 označuje, že symbol je zahrnut a hodnota 00 označuje, že jsou exportována pouze číselná data. Hodnota výchozího nastavení je 01. Hodnoty vstupu jsou pro systémy Windows 95 a Windows NT 4.0 datového typu Binary a pro systém Windows NT 3.51 typu REG_BINARY.

Formáty ISAM pro import zdroje dat HTML

Složka Jet\3.5\ISAM Formats\HTML Import obsahuje následující vstupy.

Název vstupu	**Datový typ pro systém Windows NT 3.51**	**Datový typ pro systémy Windows 95 a Windows NT 4.0**	**Hodnota**
Engine	REG_SZ	String	Text
ImportFilter	REG_SZ	String	HTML Files (*.ht*)
CanLink	REG_BINARY	Binary	01
OneTablePerFile	REG_BINARY	Binary	00
IsamType	REG_DWORD	DWORD	2
IndexDialog	REG_BINARY	Binary	00
CreateDBOnExport	REG_BINARY	Binary	00
ResultTextImport	REG_SZ	String	Import dat z externího souboru do aktuální databáze. Změna dat v aktuální databázi nebude mít vliv na data v externím souboru.
ResultTextLink	REG_SZ	String	Vytvoří tabulku

			v aktuální databázi, která je propojena s externím souborem. Změna dat v aktuální databázi změní data v externím souboru.

Poznámka Aby se projevily změny nastavení registru systému Windows, musíte opustit a potom znovu spustit databázové jádro.

Formáty ISAM pro export zdroje dat HTML

Složka Jet\3.5\ISAM Formats\HTML Export obsahuje následující vstupy.

Název vstupu	Datový typ pro systém Windows NT 3.51	Datový typ pro systémy Windows 95 a Windows NT 4.0	Hodnota
Engine	REG_SZ	String	Text
ExportFilter	REG_SZ	String	HTML Files (*.htm)
CanLink	REG_BINARY	Binary	00
OneTablePerFile	REG_BINARY	Binary	01
IsamType	REG_DWORD	DWORD	2
IndexDialog	REG_BINARY	Binary	00
CreateDBOnExport	REG_BINARY	Binary	00
ResultTextExport	REG_SZ	String	Export dat z aktuální databáze do textového souboru. Tento proces přepíše data, jestliže jsou exportována do existujícího souboru.

Poznámka Aby se projevily změny nastavení registru systému Windows, musíte opustit a potom znovu spustit databázové jádro.

Inicializace ovladače zdroje dat Microsoft Exchange

{ewc HLP95EN.DLL,DYNALINK,"Viz takďž˝":"dahowInitializingMicrosoftExchangeDataSourceDriverC"} {ewc HLP95EN.DLL,DYNALINK,"Specifika":"dahowInitializingMicrosoftExchangeDataSourceDriverS"}

Při instalaci ovladače zdroje dat Microsoft Exchange zapíše program Setup sadu výchozích hodnot do registru systému Windows v podklíčích Jádra a Formáty ISAM. Tato nastavení byste neměli přímo upravovat, raději použijte program Setup pro přidávání, odstraňování nebo pozměňování nastavení vaší aplikace. Následující oddíly popisují nastavení inicializace a formátů ISAM ovladače zdroje dat Microsoft Exchange.

Nastavení inicializace zdroje dat Microsoft Exchange

Složka Jet\3.5\Engines\ Exchange obsahuje nastavení inicializace pro ovladač Msexch35.dll, který se používá pro externí přístup ke složkám aplikace Microsoft Exchange. V této složce je jediný vstup:

```
win32=<path>\MSEXCH35.dll
```

Databázové jádro Microsoft Jet database používá složku Exchange pro umístění ovladače Mstext35.dll. Plná cesta je určena během instalace. Hodnoty vstupu pro systémy Windows 95 a Windows NT 4.0 jsou datového typu String a pro systém Windows NT 3.51 typu REG_SZ.

Microsoft Exchange ISAM Formats

Složka Jet\3.5\ISAM Formats\Exchange 4.0 obsahuje následující vstupy.

Název vstupu	Datový typ pro systém Windows NT 3.51	Datový typ pro systémy Windows 95 a Windows NT 4.0	Hodnota
Engine	REG_SZ	String	Exchange
CanLink	REG_BINARY	Binary	01
OneTablePerFile	REG_BINARY	Binary	00
IsamType	REG_DWORD	DWORD	0
IndexDialog	REG_BINARY	Binary	00
CreateDBOnExport	REG_BINARY	Binary	00

Poznámka Aby se projevily změny nastavení registru systému Windows, musíte opustit a potom znovu spustit databázové jádro.

IN

Klíčové slovo IN se používá v těchto kontextech:

Operátor **In**

Klauzule IN

Vnořené dotazy SQL

Příkaz TRANSFORM

ON

Klíčové slovo ON se používá v těchto kontextech:

Operace INNER JOIN

Operace LEFT JOIN a RIGHT JOIN

ALL

Klíčové slovo ALL se používá v těchto kontextech:

Predikáty ALL, DISTINCT, DISTINCTROW a TOP

Vnořené dotazy SQL

Operace UNION

ASC/DESC

Klíčová slova ASC a DESC se používají v těchto kontextech:

Příkaz CREATE INDEX

Klauzule ORDER BY

BY

Klíčové BY slovo se používá v těchto kontextech:

Klauzule GROUP BY

Klauzule ORDER BY

CREATE

Klíčové slovo CREATE se používá v těchto kontextech:

Příkaz CREATE INDEX

Příkaz CREATE TABLE

DROP

Klíčové slovo DROP se používá v těchto kontextech:

Příkaz ALTER TABLE

Příkaz DROP

INDEX

Klíčové slovo INDEX se používá v těchto kontextech:

Příkaz CREATE INDEX

Příkaz DROP

INTO

Klíčové slovo INTO se používá v těchto kontextech:

Příkaz INSERT INTO

Příkaz SELECT ... INTO

JOIN

Klíčové slovo JOIN se používá v těchto kontextech:

Operace INNER JOIN

Operace LEFT JOIN a RIGHT JOIN

SELECT

Klíčové slovo SELECT se používá v těchto kontextech:

Příkaz SELECT

Příkaz SELECT ... INTO

TABLE

Klíčové slovo TABLE se používá v těchto kontextech:

Příkaz ALTER TABLE

Příkaz CREATE INDEX

Příkaz CREATE TABLE

Příkaz DROP

Operace UNION

WITH

Klíčové slovo WITH se používá v těchto kontextech:

Příkaz CREATE INDEX

Declarace WITH OWNERACCESS OPTION