PRÁCA S DATABÁZOU A ZÁKLADY SQL.

Čo je to SQL (Structured Query Language) štandardizovaná množina príkazov na manipuláciu s databázami (pôvodne SEQUEL – Structured English Query Language).

- jazyk na konštrukciu a prístup k systémom pre správu relačných databáz (RDBMS) rôzneho druhu a na mnohých hardwarových platformách.
- databázový dotazovací jazyk, prijatý v 80-90 rokoch, štandard pre prácu s db v sieťovom prostredí.
- prístup k údajom s podstatne vyššou pružnosťou ako je to u iných záznamovo orientovaných produktov. Prístup k údajom množinovo orientovaných (**set-oriented**) db na serveroch Oracle, Sybase, Informix, Ingres a InterBase. Medzi dialektami SQL v SQL databázach existujú určité rozdiely.

Množinové a záznamovo orientované databázy.

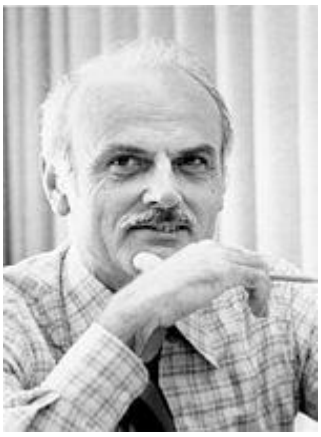
rozdiel - v organizácii tabuliek.

- v záznamovo orientovaných db (Paradox, dBase) - nástroje a práca priamo s tabuľkou.
- v množinovo orient. db sa pracuje s podmnožinou záznamov tabuľky. Ak sa pristupuje k tabuľkám na serveri cez SQL, ide na server požiadavka - SQL príkaz a on vráti skupinu, ktorá zodpovedá podmienke SQL príkazu. Užívateľ dostane podmnožinu tabuliek (data-set - množina údajov), ktorá sa štruktúrou môže líšiť od skutočných tabuliek na serveri.

Prednosti SQL

- rovnaké príkazy pre záznamovo aj pre množinovo orient. db
- zvyšuje pružnosť návrhu db aplikácií
- podstatne redukuje prevádzku na sieti - prenášané sú iba údaje, ktoré splňujú SQL podmienku na rozdiel od záznamovo orientovaných databáz.

Z histórie



Edgar F. Codd (1923-2003)

Na začiatku 70 rokov minulého storočia vyvinula firma IBM experimentálny relačný SRDB (prvý databázový systém založený na Coddovom relačnom dátovom modeli), pre ktorý bol (1974) vytvorený špeciálny jazyk SEQUEL - autori Donald D. Chamberlin (1944) a Raymond F. Boyce (1947 - 1974). SQL sa stal čoskoro štandardnou výbavou väčšiny relačných databázových systémov.

V roku 1986 inštitúcia ANSI štandardizovala jazyk SQL pod označením SQL-86. O rok neskôr vo viac-menej nezmenej forme tento štandard ratifikovala aj organizácia ISO (SQL-87). V roku 1989 a 1992 boli prijaté revízie označené ako SQL-89, resp. SQL-92 (aj SQL-2). Výrazné zmeny prišli v roku 1999, kedy sa do štandardu SQL:1999 (SQL-3)

dostali napr. rekurzívne dotazy, spúšťače (triggery), regulárne výrazy, neskalárne dátové typy a niektoré objektovo-orientované vlastnosti. Verzia SQL:2003 priniesla hlavne podporu pre XML, štandardizované postupnosti, stĺpce s automaticky generovanými hodnotami a tzv. oknové funkcie.

Year	Name	Alias	Comments
1986	SQL-86	SQL-87	Prvý variant normy publikovaný ANSI a ratifikovaný ISO 1987.
1989	SQL-89	FIPS 127-1	Malá revízia známa ako FIPS 127-1.
1992	SQL-92	SQL2, FIPS 127-2	Veľká revízia (ISO 9075), *Entry Level* SQL-92 ako FIPS 127-2.
1999	SQL:1999	SQL3	Pridaná podpora regular expression, rekurzívnych dotazov, triggers, procedurálnych a riadiacich príkazov, neskalárnych typov a niektorých objektovo-orientovaných možností.
2003	SQL:2003	SQL 2003	Zavedené rozšírenia pre prácu s XMLdátami, *window functions* (pre prácu s OLAP Online Analytical Processing db), generátory postupností a na nich založené dátové typy (vrátane identity-columns).
2006	SQL:2006	SQL 2006	ISO/IEC 9075-14:2006 definuje a rozširuje použitie SQL v spojení s XML. Definuje import a ukladanie XML dát do SQL databázy, manipuláciu s nimi v databáze a publikovanie tak XML ako aj konvenčných SQLdát v XML formáte. Pribúda možnosť integrovať do SQL kódu aj XQuery, (the XML Query Language published by the World Wide Web Consortium (W3C), na súčasný prístup k štandardným SQL-dátam a XML dokumentom.
2008	SQL:2008	SQL 2008	Vylepšenia windowing functions, vyjasnenia SQL 2003 nejednoznačností. Legalizuje ORDER BY mimo kurzor definície, pridáva INSTEAD OF triggery a príkaz TRUNCATE.
2011	SQL:2011		

Procedurálne rozšírenia SQL

Source	Common Name	Full Name
ANSI/ISO Standard	SQL/PSM	SQL/Persistent Stored Modules
Interbase/ Firebird	PSQL	Procedural SQL

IBM DB2	SQL PL	SQL Procedural Language (implements SQL/PSM)
IBM Informix	SPL	Stored Procedural Language
IBM Netezza	NZPLSQL [1]	(based on Postgres PL/pgSQL)
Microsoft / Sybase	T-SQL	Transact-SQL
Mimer SQL	SQL/PSM	SQL/Persistent Stored Module (implements SQL/PSM)
MySQL	SQL/PSM	SQL/Persistent Stored Module (implements SQL/PSM)
NuoDB	SSP	Starkey Stored Procedures
Oracle	PL/SQL	Procedural Language/SQL (based on Ada)
PostgreSQL	PL/pgSQL	Procedural Language/PostgreSQL (based on Oracle PL/SQL)
PostgreSQL	PL/PSM	Procedural Language/Persistent Stored Modules (implements SQL/PSM)
Sybase	Watcom-SQL	SQL Anywhere Watcom-SQL Dialect
Teradata	SPL	Stored Procedural Language
SAP	SAP HANA	SQL Script

Lokálne SQL

SQL - BDE umožňuje prístup k db tabuľkám lokálnym aj vzdialeným. Ide o tzv. lokálne SQL (tiež klientske SQL) - podmnožina normy ANSI-92 SQL, určené pre dBASE a Paradox konvencií pomenovania pre tabuľky a stĺpce.
Lokálne SQL - umožňuje použiť SQL na prístup k lokálnym db tabuľkám, ktoré nie sú na db serveri ako aj k vzdialeným SQL serverom. Umožňuje aj multitabuľkové dotazy nad lokálnymi a vzdialenými SQL tabuľkami.
Podporuje dve rozdielne kategórie SQL príkazov:

- DML (**Data manipulation Language**) pre výber, vkladanie, aktualizáciu a zrušenie údajov tabuľky (SELECT, INSERT, UPDATE, DELETE)
- DDL (**Data Definition Language**) pre vytváranie, doplnenie a zrušenie tabuliek a pre vytváranie a zrušenie indexov

Konvencie pomenovania

Tabuľky

ANSI norma SQL definuje meno tabuľky ako jednoduché slovo, pozostávajúce z alfanumerických znakov a podčiarknika (_). Lokálne SQL podporuje názov tabuľky v tvare plnej špecifikácie súboru – disk, cesta, názov, uzavretej v jednoduchých alebo dvojitých apostrofoch, napr.:

```
SELECT *
FROM ‘TABLE.DBF’
```

alebo

```
SELECT *
FROM “C:\CESTA\TABLE.DBF”
```

Lokálne SQL tiež podporuje aliasy (disk + cesta) pre názov tabuľky, napr.:

```
SELECT *
FROM “:ALIAS:TABLE.DBF”
```

A nakoniec lok. SQL dovoľuje použiť v názve tabuľky aj kľúčové slová SQL, ak je názov tabuľky uzavretý v jednoduchých lebo dvojitých apostrofoch, napr.:

```
SELECT *
FROM “PASSWORD”
```

Stĺpce

ANSI norma SQL definuje meno stĺpca ako jedno slovo, pozostávajúce z alfanumerických znakov a podčiarknika (_). Lokálne SQL podporuje viacslovné názvy stĺpcov tabuľky aj názvy obsahujúce kľúčové slová SQL, ak tieto názvy

- Sú uzavreté v jednoduchých alebo dvojitých apostrofoch
- Nasledujú po SQL alebo korelačnom mene tabuľky

V prvom príklade má názov stĺpca dve slová, v druhom príklade obsahuje kľúčové slovo SQL DATE:

```
SELECT T.”PRAC ID”
FROM TABLE T

SELECT TABLE.”DATE”
FROM TABLE
```

Dátumy

Lokálne SQL predpokladá dátum v tvare US – dátumu. t.j. MM/DD/YY. Medzinárodné formáty dátumu nie sú podporované.

Zoznam rezervovaných slov pre lokálne SQL:

ACTIVE, ADD, ALL, AFTER, ALTER, AND, ANY, AS, ASC, ASCENDING, AT, AUTO, AUTOINC, AVG
BASE_NAME, BEFORE, BEGIN, BETWEEN, BLOB, BOOLEAN, BOTH, BY, BYTES

CACHE, CAST, CHAR, CHARACTER, CHECK, CHECK_POINT_LENGTH, COLLATE, COLUMN, COMMIT, COMMITTED, COMPUTED, CONDITIONAL, CONSTRAINT, CONTAINING, COUNT, CREATE, CSTRING, CURRENT, CURSOR

DATABASE, DATE, DAY, DEBUG, DEC, DECIMAL, DECLARE, DEFAULT, DELETE, DESC, DESCENDING, DISTINCT, DO, DOMAIN, DOUBLE, DROP

ELSE, END, ENTRY_POINT, ESCAPE, EXCEPTION, EXECUTE, EXISTS, EXIT, EXTERNAL, EXTRACT

FILE, FILTER, FLOAT, FOR, FOREIGN, FROM, FULL, FUNCTION

GDSCODE, GENERATOR, GEN_ID, GRANT, GROUP, GROUP_COMMIT_WAIT_TIME

HAVING, HOUR

IF, IN, INT, INACTIVE, INDEX, INNER, INPUT_TYPE, INSERT, INTEGER, INTO, IS, ISOLATION

JOIN

KEY

LONG, LENGTH, LOGFILE, LOWER, LEADING, LEFT, LEVEL, LIKE, LOG_BUFFER_SIZE

MANUAL, MAX, MAXIMUM_SEGMENT, MERGE, MESSAGE, MIN, MINUTE, MODULE_NAME, MONEY, MONTH

NAMES, NATIONAL, NATURAL, NCHAR, NO, NOT, NULL, NUM_LOG_BUFFERS, NUMERIC

OF, ON, ONLY, OPTION, OR, ORDER, OUTER, OUTPUT_TYPE, OVERFLOW

PAGE_SIZE, PAGE, PAGES, PARAMETER, PASSWORD, PLAN, POSITION, POST_EVENT, PRECISION, PROCEDURE, PROTECTED, PRIMARY, PRIVILEGES

RAW_PARTITIONS, RDB$DB_KEY, READ, REAL, RECORD_VERSION, REFERENCES, RESERV, RESERVING, RETAIN, RETURNING_VALUES, RETURNS, REVOKE, RIGHT, ROLLBACK

SECOND, SEGMENT, SELECT, SET, SHARED, SHADOW, SCHEMA, SINGULAR, SIZE, SMALLINT, SNAPSHOT, SOME, SORT, SQLCODE, STABILITY, STARTING, STARTS, STATISTICS, SUB_TYPE, SUBSTRING, SUM, SUSPEND

TABLE, THEN, TIME, TIMESTAMP, TIMEZONE_HOUR, TIMEZONE_MINUTE, TO, TRAILING, TRANSACTION, TRIGGER, TRIM

UNCOMMITTED, UNION, UNIQUE, UPDATE, UPPER, USER

VALUE, VALUES, VARCHAR, VARIABLE, VARYING, VIEW

WAIT, WHEN, WHERE, WHILE, WITH, WORK, WRITE

YEAR

OPERÁTORY:
||, -, *, /, <>, <, >, ,(comma), =, <=, >=, ~=, !=, ^=, (,)

DML - Manipulácia s údajmi

SELECT	prístup – výber z existujúcich údajov
INSERT	pridanie nových údajov do tabuľky
UPDATE	modifikácia existujúcich údajov (okrem kľúča)
DELETE	zrušenie existujúcich údajov v tabuľke

Substitúcie parametrov v DML príkazoch

Premenné alebo označenia parametrov môžu byť v DML príkazoch použité namiesto konštantných hodnôt. Vždy im musí predchádzať dvojbodka, napr.:

```
SELECT PRVY, DRUHY
FROM “TABLE“
WHERE PRVY > :var1 AND DRUHY < :var2
```

Podporované súhrnné (agregačné) funkcie

- SUM() súčet všetkých numerických hodnôt v stĺpci
- AVG() priemer všetkých nenulových numerických hodnôt v stĺpci
- MIN() určenie minimálnej hodnoty v stĺpci
- MAX() určenie maximálnej hodnoty v stĺpci
- COUNT() určí počet hodnôt v stĺpci, ktoré vyhovujú danému kritériu

Komplexné agregačné výrazy sú podporované, napr.:

```
SUM( Field * 10 )
SUM( Field ) * 10
SUM( Field1 + Field2 )
```

Podporované reťazcové funkcie

manipulácia s reťazcami pre výber, vkladanie a aktualizáciu:

- UPPER() prevod reťazca na veľké znaky
- LOWER() prevod reťazca na malé znaky
- SUBSTRING() vrátenie špecifikovanej časti reťazca
- TRIM() zrušiť opakovanie znaku na ľavej, pravej či oboch stranách reťazca
- SUBSTRING() vytvoriť podreťazec z reťazca

Podporované dátumové funkcie

výber numerického poľa z poľa – stĺpca typu date/time pri výbere - syntax

```
EXTRACT (extract_field FROM field_name)
SELECT EXTRACT (YEAR FROM NAROD_DATE)
FROM TABLE
```

S použitím tejto funkcie je možné vybrať aj ďalšie položky MONTH, DAY, HOUR, MINUTE a SECOND. Extract nepodporuje klauzuly TIMEZONE_HOUR a TIMEZONE_MINUTE.

Podporované operátory

- Aritmetické +, -, *, /
- Relačné <, >, =, <>, >=, =<, IS NULL, IS NOTNULL
- Logické AND, OR, NOT
- Spojenia reťazcov ||

Použitie SELECTu

- výber údajov z jednej či viacerých tabuliek. SELECT, ktorý vyberá údaje z viacerých tabuliek, sa nazýva „**join**". Lokálne SQL podporuje SELECT:

```
SELECT [DISTINCT] column_list
FROM table_reference
[WHERE search_condition]
[ORDER BY order_list]
[GROUP BY group_list]
[HAVING having_list]
```

napr.

```
SELECT PART_NO, PART_NAME
FROM "PARTS.DB"
```

Použitie klauzuly DISTINCT - odstránenie duplikátov vo výsledku selektu.

Použitie klauzuly FROM - tabuľka alebo tabuľky, z ktorých sa vyberajú údaje, viac tabuliek oddelených čiarkou alebo vnútorný či vonkajší join v zmysle špecifikácie normy SQL-92. Výber z jednej tabuľky

```
SELECT PART_NO, PART_NAME
FROM "PARTS.DB"
```

Ľavý vonkajší join

```
SELECT *
FROM PARTS LEFT OUTER JOIN INVENTORY
ON PARTS.PART_NO = INVENTORY.PART_NO
```

Použitie klauzuly WHERE - redukuje výber na záznamy, ktoré vyhovujú podmienkam výberu, napríklad riadky s číslom súčiastky > 53:

```
SELECT *
FROM "PARTS.DB"
WHERE PART_NO > 53
```

V podmienke WHERE môže byť uvedená aj množina hodnôt pomocou slova IN, ktorým má vyhovovať príslušný atribút (stĺpec) tabuľky:

```
SELECT *
FROM "PARTS.DB"
WHERE PART_NO IN [25, 37, 53, 123, 527]
```

Dôležité: Podmienka *search_condition* nemôže obsahovať subquery – poddotaz.

Použitie klauzuly ORDER BY - poradie a smer usporiadania záznamov podľa atribútu (ASC, DESC). Prvý príklad – vzostupne, druhý zostupne usporiadané riadky:

```
SELECT *
FROM "PARTS.DB"
ORDER BY PART_NAME ASC
```

```
SELECT *
FROM “PARTS.DB”
ORDER BY PART_NO DESC
```

Vypočítané pole môže byť usporiadané podľa korelačného mena alebo ordinálnej pozície, v príklade je názov vypočítaného poľa PLNE_MENO

```
SELECT PRIEZVISKO ||’, || MENO AS PLNE_MENO, TELEFON
FROM ZAKAZNIK
ORDER BY PLNE_MENO
```

Výpočet hodnôt

Výpočet počtu zamestnancov - v príkaze Select (namiesto v položke tabuľky):

```
SELECT COUNT (*)
FROM ZAMESTNANEC
```

Počet objednávok prevzatých zamestnancom, celkovú sumu a priemer:

```
SELECT SUM (OBJEDNAVKY.SUMA_CELKOM),
            COUNT(OBJEDNAVKY.SUMA_CELKOM),
           AVG(OBJEDNAVKY.SUMA_CELKOM)
FROM OBJEDNAVKY, ZAMESTNANEC
WHERE OBJEDNAVKY.CISLO_ZAM = ZAMESTNANEC. CISLO_ZAM
AND   ZAMESTNANEC.PRIEZVISKO = “HOROVCAK”
```

Použitie klauzuly GROUP BY - ako sú vybrané riadky zoskupené pre súhrnné funkcie. Každý atribút v GROUP BY sa musí vyskytovať aj v príkaze SELECT. Pre každú skupinu je možné počítať nejakú hodnotu, napr. určiť zoznam zamestnancov a každý z nich má celkový súčet ním uskutočnených objednávok:

```
SELECT ZAMESTNANEC.PRIEZVISKO,
            SUM (OBJEDNAVKY.SUMA_CELKOM)
FROM OBJEDNAVKY, ZAMESTNANEC
WHERE OBJEDNAVKY.CISLO_ZAM = ZAMESTNANEC. CISLO_ZAM
AND  GROUP BY ZAMESTNANEC.PRIEZVISKO
```

Použitie klauzuly HAVING - doplňujúca podmienka užívaná v spojení s GROUP BY. Skupiny nevyhovujúce doplňujúcej podmienke HAVING sú z výsledného výberu vynechané.

V klauzule HAVING sú podporované poddotazy, ktoré fungujú ako vyhľadávacia podmienka vo vonkajšom alebo “rodičovskom“ dotaze. Okrem štandardných porovnávacích operátorov (=, <, > ...) sú podporované doplňujúce predikáty IN, ANY, ALL, EXISTS.

Heterogénne joiny
Lokálne SQL podporuje joiny tabuliek v rôznych databázových formátoch. Takýto join sa nazýva hetergénny join. Napríklad súčasný výber z tabuliek typu dBASE a Paradox vyzerá takto:

```
SELECT DISTINCT C.CUST_NO, C.STATE, O.ORDER_NO
FROM "CUSTOMER.DB" C, "ORDER.DBF" O
WHERE C.CUST_NO =O.CUST_NO
```

Namiesto názvov tabuliek je možné samozrejme použiť databázové aliasy.

Použitie INSERTu

Pre všetky stĺpce

```
INSERT INTO CUSTOMER
VALUES( :v_priezvisko, :v_meno, :v_telefon)
```

Pre vymenované stĺpce

```
INSERT INTO CUSTOMER(PRIEZVISKO, MENO, TELEFON)
VALUES( :v_priezvisko, :v_meno, :v_telefon)
```

V príklade sa pridáva nový riadok s hodnotami dvoch stĺpcov:

```
INSERT INTO EMP_PROJ (EMP_NO, PROJ_ID) VALUES (52, "DGPII");
```

Ďalší príklad špecifikuje vkladané údaje ako výsledok príkazu SELECT:

```
INSERT INTO PROJECTS
  SELECT * FROM NEW_PROJECTS
    WHERE NEW_PROJECTS.START_DATE > "6-JUN-1994";
```

Použitie UPDATE, DELETE

Vzhľadom na ANSI normu nie sú žiadne obmedzenia ani rozšírenia.

```
UPDATE CUSTOMER SET PRIEZVISKO='nove', MENO='novem'
WHERE Id=5
```

```
DELETE FROM CUSTOMER WHERE Id=5
```

DDL - Definícia údajov

Vytváranie, doplnenie a zrušenie tabuliek, indexov aj pohľadov (views).

Použitie CREATE TABLE – vytvorenie tabuľky

```
CREATE TABLE "employee.db"      (
        LAST_NAME CHAR(20),
        FIRST_NAME CHAR(15),
        SALARY NUMERIC(10,2),
        DEPT_NO SMALLINT,
        PRIMARY KEY(LAST_NAME, FIRST_NAME)  )
```

Použitie ALTER TABLE - Pridanie nových stĺpcov do existujúcej tabuľky

```
ALTER TABLE "employee.dbf" ADD BUILDING_NO SMALLINT
```

Nasledujúci príkaz zruší dva stĺpce v tabuľke:

```
ALTER TABLE "employee.db" DROP LAST_NAME, DROP FIRST_NAME
```

Operácie ADD a DROP môžu byť kombinované do jediného príkazu:

```
ALTER TABLE "employee.dbf" DROP LAST_NAME, ADD FULL_NAME CHAR[30]
```

Použitie DROP TABLE - ruší tabuľku

```
DROP TABLE "employee.db"
```

Použitie CREATE INDEX - vytvorenie indexu

```
CREATE INDEX index_name ON table_name (column [, column ...])
```

Použitie DROP INDEX - zrušenie indexu

```
DROP INDEX table_name.index_name | PRIMARY
```

Použitie CREATE VIEW

CREATE VIEW - pohľad na údaje založený na 1 alebo viacerých tabuľkách databázy. View nereprezentuje fyzicky uložené údaje. Je možné vykonať operácie select, project, join, a union na pohľadoch ako keby to boli tabuľky.

```
CREATE VIEW view_name [ (column_name [, column_name]...)]
```

MySQL ponúka celý rad riešení pre ukladanie a správu údajov :

Databázové stroje MySQL

Databázový stroj (alebo Storage Engine) je základná softvérová súčasť systému riadenia bázy dát (DBMS) používaná na create, read, update and delete (CRUD) dát z databázy.

Storage Engine	Description
ARCHIVE	The ARCHIVE engine is optimized for managing large amounts of data designated as stored for archived purposes. Data stored using the ARCHIVE engine can only be inserted and selected and cannot be deleted or modified.
BLACKHOLE	The BLACKHOLE storage engine accepts inserted data without error but does not store it. Instead, it deletes it upon acceptance. While seemingly useless, BLACKHOLE can actually serve several practical roles, ranging from trouble-shooting data replication processes to assisting in the identifi-cation of bottlenecks (due to the ability to use BLACKHOLE to remove the storage engine from the bottleneck candidates).
CSV	Comma-separated values (CSV) format is a common storage solution supported by many applications. MySQL's CSV storage engine manages data in this format, the data files of which can subsequently be read from and written to by applications such as Microsoft Excel.
EXAMPLE	EXAMPLE is a featureless storage engine with the sole purpose of providing a skeleton for writing your own own storage engines. It is incapable of storing data.
FEDERATED	Introduced in MySQL 5.0, the FEDERATED storage engine can pool remote MySQL databases together under the guise of a single logical database by creating pointers to these remote tables.
InnoDB	MySQL's most popular transactional storage solution, InnoDB offers complete commit, rollback, and crash recovery features alongside attractive performance capabilities. InnoDB has long served as MySQL’s default storage engine on the Windows platform. It is the default storage engine on all platforms for MySQL 5.5.
MEMORY	The MEMORY storage engine stores data within system memory (RAM), resulting in volatile, although extremely fast, data access.
MERGE	The MERGE storage engine is useful for accessing a group of identical MyISAM tables as if the data resided within a single table structure. Such a configuration might be useful when accessing large amounts of sales data, which has been separately stored by month according to an aptly named table.
MyISAM	Until MySQL 5.5.5, the MyISAM had long been MySQL's default storage engine. Although incapable of supporting transactions, MyISAM is optimized for high traffic environments and is very simple to manage