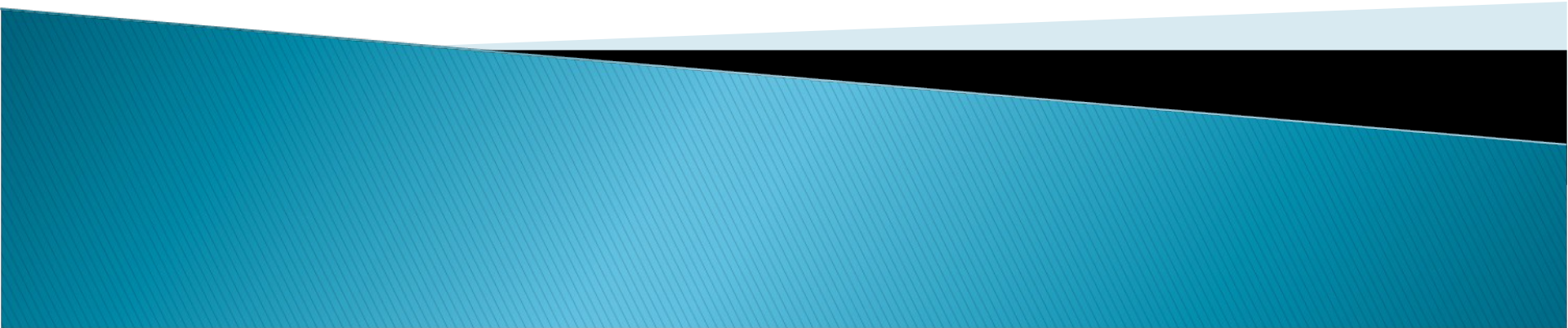
Structured Query Language (SQL) 06

By: M Shuaib Zarinkhail 2010



Data Types (MS SQL Server)

Data Type	Description
Binary *applicable in MySQL*	.Binary, length 0 to 800 bytes
Char *applicable in MySQL*	.Character, length 0 to 800 bytes
Datetime *applicable in MySQL*	byte datetime. Range from January 1,-8 1753, through December 31, 9999, with an .accuracy of three-hundredths of a second

Data Types (MS SQL Server)

Data Type	Description
Image	Variable length binary data. Maximum length 2,147,483,647
Integer *applicable in MySQL*	– byte integer. Value range from-4 through 2,147,483,647 2,147,483,648-

Data Types (MS SQL Server)

Data Type	Description
Money	byte money. Range from-8 -922,337,203,685,477.5808 through +922,337,203,685,477.5807, with accuracy .to a ten-thousandth of a monetary unit
Numeric *applicable in MySQL*	Decimal – can set precision and scale. Range $-10^{38} + 1$ through $10^{38} - 1$

Data Types (MS SQL Server)

Data Type	Description
Text *applicable in MySQL*	Variable length text, maximum length .2,147,483,647 characters
Tinyint *applicable in MySQL*	byte integer. Range from 0 through 255-1
(Varchar(n *applicable in MySQL*	Variable-length character, length 0 to 8000 bytes

Data Types (Oracle)

Data Type	Description
BLOB *applicable in MySQL*	Binary Large OBject. Up to 4 gigabytes in length
(CHAR(n *applicable in MySQL*	Fixed length character field of length n. Maximum 2,000 characters
DATE *applicable in MySQL*	byte field containing both date and time-7

Data Types (Oracle)

Data Type	Description
INTEGER *applicable in MySQL*	Whole number of length 38
(NUMBER(n, d	Numeric field of length n, d places to the right of the decimal
VARCHAR(n (Or (NVARCHAR(n *applicable in MySQL*	Variable length character field up .to n characters long Maximum value of n = 4,000

Data Types (MySQL Server)

Data Type	Description
BIGINT	A large integer. The signed range is: -9+E18 to 9+E18 The unsigned range is: 0 to 2+E19
SERIAL	Is an alias for BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE

Data Types (MySQL Server)

Data Type	Description
AUTO_INCREMENT	Generates a unique identity for new records in a table
(CHAR(M	A fixed-length string. M represents the column length which is: 0 to 255
BINARY	The BINARY type is similar to the CHAR type, but stores binary byte strings rather than non-binary character strings

Data Types (MySQL Server)

Data Type	Description
VARCH (AR (M	A variable-length string. M represents the maximum column length. In MySQL 5.0, the range of M is 0 to 255 before MySQL 5.0.3, and 0 to 65,535 in MySQL 5.0.3 and later.
VARBIN ARY ((M	The VARBINARY type is similar to the VARCHAR type, but stores binary byte strings rather than non-binary character strings.

Data Types (MySQL Server)

Data Type	Description
CHAR BYTE	This is an alias for the BINARY data type
TINYIN (T(M	A very small integer. The signed range is -128 to 127. The unsigned range is 0 to 255
BOOLEAN or BOOL	These types are synonyms for TINYINT(1). A value of zero is considered false. Non-zero values are considered true

Data Types (MySQL Server)

Data Type	Description
(BIT(M	A bit-field type. M indicates the number of bits per value, from 1 to 64. The default is 1 if M is omitted.
BLOB Data	Has four types: TINYBLOB, BLOB, MEDIUMBLOB, and LONGBLOB.
TINYBLOB	A BLOB column with a maximum length of 255 (28 - 1) bytes.

Data Types (MySQL Server)

Data Type	Description
(BLOB(M	Stands for Binary Large Object. The maximum length is 65,535 (216 - 1) bytes.
MEDIUMBLOB	A BLOB column with a maximum length .of 16,777,215 (224 - 1) bytes
LONGBLOB	A BLOB column with a maximum length of 4,294,967,295 or 4GB (232 - 1) bytes.

Data Types (MySQL Server)

Data Type	Description
TEXT Data Types	Has four types: TINYTEXT, TEXT, MEDIUMTEXT, and LONGTEXT.
TINYTEXT	A TEXT column with a maximum length .of 255 (28 - 1) characters
(TEXT(M	A TEXT column with a maximum length of 65,535 (216 - 1) bytes.

Data Types (MySQL Server)

Data Type	Description
MEDIUM TEXT	A TEXT column with a maximum length of 16,777,215 (224 - 1) characters.
LONGTEXT	A TEXT column with a maximum length of 4,294,967,295 or 4GB (232 - 1) bytes.
SET	A set. A string object that can have zero or more values, each of which must be chosen from the list of values 'value1', 'value2', ...

Data Types (MySQL Server)

Data Type	Description
DATE	A date. The supported range is '1000-01-01' to '9999-12-31'. MySQL displays DATE values in 'YYYY-MM-DD'
DATETIME	A date and time combination. The supported range is '1000-01-010:00:00‘ to '9999-12-31 23:59:59'. MySQL displays DATETIME values in 'YYYY-MM-DD HH:MM:SS' format.

Data Types (MySQL Server)

Data Type	Description
TIME	A time. The range is '-838:59:59' to '838:59:59'. MySQL displays TIME values in 'HH:MM:SS' format.
TIMESTAMP	A timestamp. The range is '1970-01-01 00:00:01' UTC to partway through the year 2038. TIMESTAMP values are stored as the number of seconds since the epoch ('1970-01-01 00:00:00' UTC).

Data Types (MySQL Server)

Data Type	Description
(YEAR(2 | 4	A year in two-digit or four-digit format. The default is four-digit format. In four-digit format, the allowable values are 1901 to 2155. In two-digit format, the allowable values are 70 to 69, representing years from 1970 to 2069.
(INT(M	A normal-size integer. The signed range is -2147483648 to 2147483647. The unsigned range is 0 to 4294967295.

Data Types (MySQL Server)

Data Type	Description
INTEGER ((M	.This type is a synonym for INT
MEDIUMI (NT(M	A medium-sized integer. The signed range is -8388608 to 8388607. The unsigned range is 0 to 16777215.
SMALLIN (T(M	A small integer. The signed range is -32768 to 32767. The unsigned range is 0 to 65535.

Data Types (MySQL Server)

Data Type	Description
DOUBLE((M, D	A normal-size (double-precision) floating-point number. Allowable values are -1.7976931348623157E+308 to -2.2250738585072014E-308, 0, and 2.2250738585072014E-308 to 1.7976931348623157E+308.
DOUBLE PRECISION((M, D	Synonym for DOUBLE(M,D)

Data Types (MySQL Server)

Data Type	Description
REAL(M,(D	Synonym for DOUBLE. Exception: If the REAL_AS_FLOAT SQL mode is enabled, REAL is a synonym for FLOAT rather than DOUBLE.
FLOAT((M,D	A small (single-precision) floating-point number. Allowable values are -3.402823466E+38 to -1.175494351E-38, 0, and 1.175494351E-38 to 3.402823466E+38.

Data Types (MySQL Server)

Data Type	Description
ENUM (value1, value2, (… ,value3	An enumeration. A string object that can have only one value, chosen from the list of values. Can have a maximum of 65,535 distinct values.

Data Types (MySQL Server)

Data	Description
DECIMA (L(M, D	A packed "exact" fixed-point number. M is the total number of digits (the precision) and D is the number of digits after the decimal point (the scale). The maximum number of digits (M) for DECIMAL is 65. The maximum number of supported decimals (D) is 30. If D is omitted, the default is 0. If M is omitted, the default is 10.

Data Types (MySQL Server)

Data Type	Description
DEC(M, (D	Synonym for DECIMAL.
FIXED(M, (D	Synonym for DECIMAL. This data type is available for compatibility with other database systems.
NUMERI (C(M, D	.Synonym for DECIMAL

Choosing the Right Type for a Column

- For the most efficient use of storage, try to use the most precise type in all cases
  - For example, if an integer column is used for values in the range from 1 to 99999, MEDIUMINT UNSIGNED is the best type

Using Data Types from other DBMSs

- Databases are used through different DBMSs
- A number of DBMSs do not support some of data types whereas other DBMSs do
- This may cause minor and sometimes major problems
- Customers may shift from one DBMS to another

Using Data Types from other DBMSs

- Users can shift from one DBMS to a second DBMS
- In such cases, data type support for the second DBMS is very important
- MySQL, as a second DBMS in such cases, has some privileges
- It can use data types from other DBMS by changing those types to its supported data type

Using Data Types from other DBMSs

- To make it easier to use code written for SQL implementations from other vendors, MySQL maps data types as shown in the coming slides
- These mappings make it easier to import table definitions from other database systems into MySQL
- The following slides include external data types and their equal data types in MySQL

Using Data Types from other Database Engines

Other Vendor Type	MySQL Type
BINARY(*M*)	CHAR(*M*) BINARY (before MySQL 4.1.2)
BOOL	TINYINT
BOOLEAN	TINYINT
CHARACTER VARYING(*M*)	VARCHAR(*M*)
FIXED	DECIMAL (MySQL 4.1.0)
FLOAT4	FLOAT

Using Data Types from other Database Engines

Other Vendor Type	MySQL Type
FLOAT8	DOUBLE
INT1	TINYINT
INT2	SMALLINT
INT3	MEDIUMINT
INT4	INT
INT8	BIGINT
LONG VARBINARY	MEDIUMBLOB

Using Data Types from other Database Engines

Other Vendor Type	MySQL Type
LONG VARCHAR	MEDIUMTEXT
LONG	MEDIUMTEXT (MySQL 4.1.0 on)
MIDDLEINT	MEDIUMINT
NUMERIC	DECIMAL
VARBINARY(*M*)	VARCHAR(*M*) BINARY (before MySQL 4.1.2)

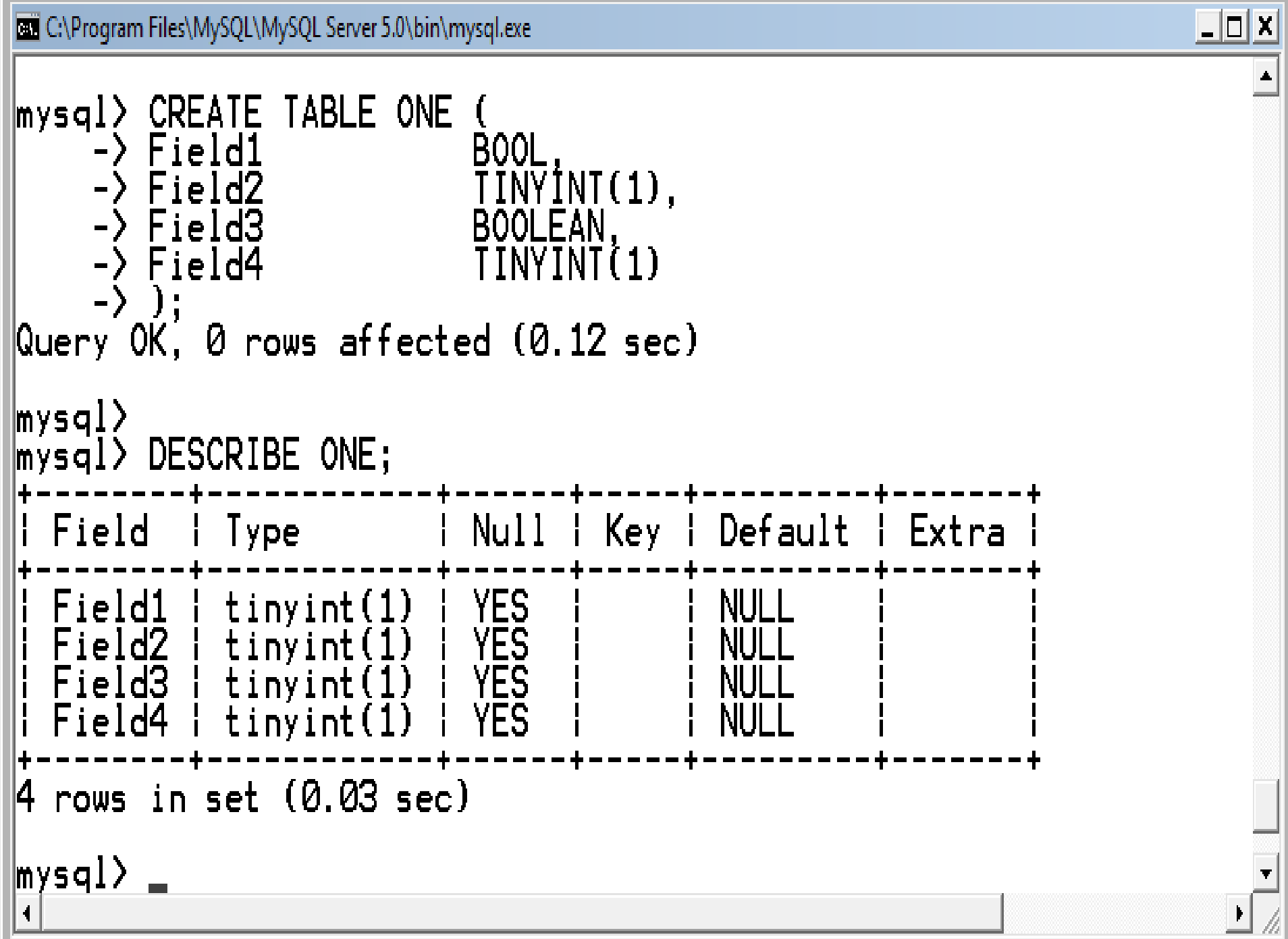
More on Data Types

- As of MySQL 4.1.2, BINARY and VARBINARY are distinct data types and are not converted to CHAR BINARY and VARCHAR BINARY
- Data type mapping occurs at table creation time
  - after which the original type specifications are discarded

- Moving a DB from one DBMS to another can be done by two ways
  1. Load a database from its backup file
  2. Develop a database and recreate all the components of that DB in the new DBMS
- In both cases, MySQL automatically changes data types to its supported ones
  - Table columns can be defined in any data type, but the results will be recorded and shown in MySQL supported data types

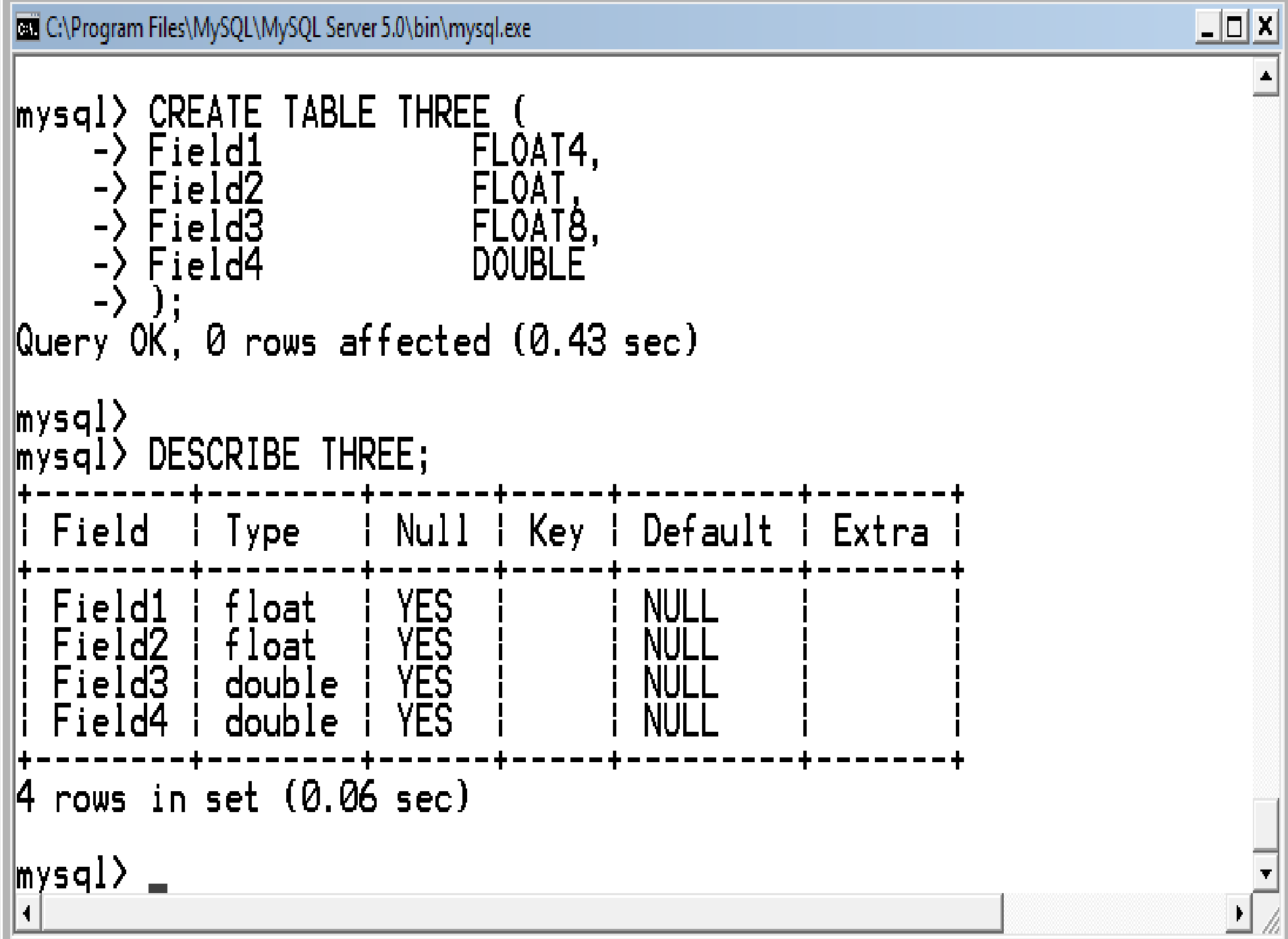
More on Data Types

- If you create a table with types used by other vendors and then issue a DESCRIBE *tbl_name* statement, MySQL reports the table structure using the equivalent MySQL types
- The following slides show adopting data types from other vendors into MySQL


C:\Program Files\MySQL\MySQL Server 5.0\bin\mysql.exe
mysql> CREATE TABLE ONE (
-> Field1 BOOL,
-> Field2 TINYINT(1),
-> Field3 BOOLEAN,
-> Field4 TINYINT(1)
->);
Query OK, 0 rows affected (0.12 sec)
mysql>
mysql> DESCRIBE ONE;
+--------+------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+--------+------------+------+-----+---------+-------+
Field1	tinyint(1)	YES		NULL	
Field2	tinyint(1)	YES		NULL	
Field3	tinyint(1)	YES		NULL	
Field4	tinyint(1)	YES		NULL	
+--------+------------+------+-----+---------+-------+
4 rows in set (0.03 sec)
mysql> _

C:\Program Files\MySQL\MySQL Server 5.0\bin\mysql.exe

```
mysql> CREATE TABLE TWO (
    -> Field1           CHARACTER VARYING(22),
    -> Field2           VARCHAR(22),
    -> Field3           FIXED,
    -> Field4           DECIMAL(10,0)
    -> );
Query OK, 0 rows affected (0.26 sec)

mysql>
mysql> DESCRIBE TWO;
+--------+--------------+------+-----+---------+-------+
| Field  | Type         | Null | Key | Default | Extra |
+--------+--------------+------+-----+---------+-------+
| Field1 | varchar(22)  | YES  |     | NULL    |       |
| Field2 | varchar(22)  | YES  |     | NULL    |       |
| Field3 | decimal(10,0)| YES  |     | NULL    |       |
| Field4 | decimal(10,0)| YES  |     | NULL    |       |
+--------+--------------+------+-----+---------+-------+
4 rows in set (0.06 sec)

mysql>
```

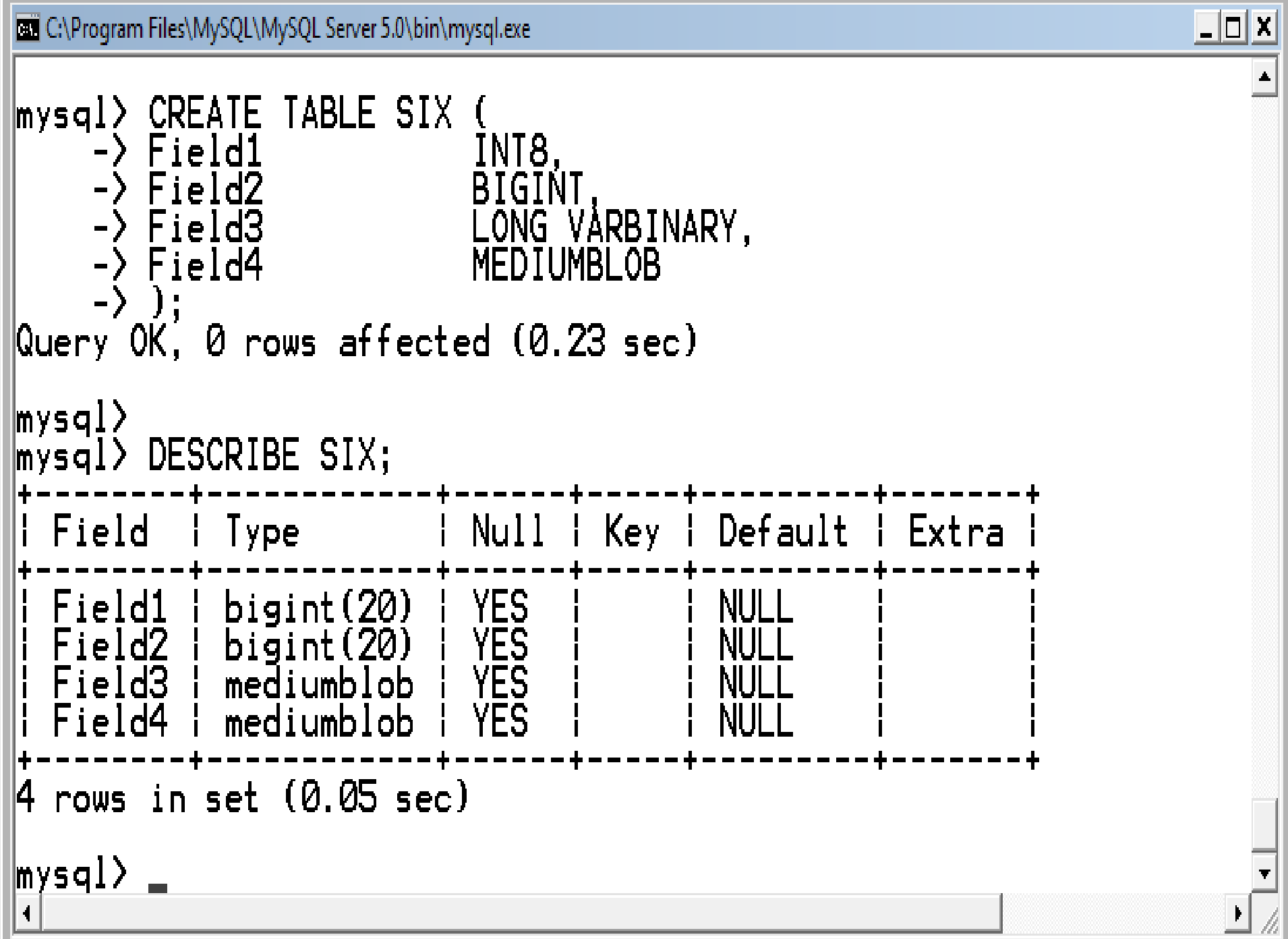

C:\Program Files\MySQL\MySQL Server 5.0\bin\mysql.exe
mysql> CREATE TABLE THREE (
-> Field1 FLOAT4,
-> Field2 FLOAT,
-> Field3 FLOAT8,
-> Field4 DOUBLE
->);
Query OK, 0 rows affected (0.43 sec)
mysql>
mysql> DESCRIBE THREE;
+--------+--------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+--------+--------+------+-----+---------+-------+
Field1	float	YES		NULL	
Field2	float	YES		NULL	
Field3	double	YES		NULL	
Field4	double	YES		NULL	
+--------+--------+------+-----+---------+-------+
4 rows in set (0.06 sec)
mysql> _

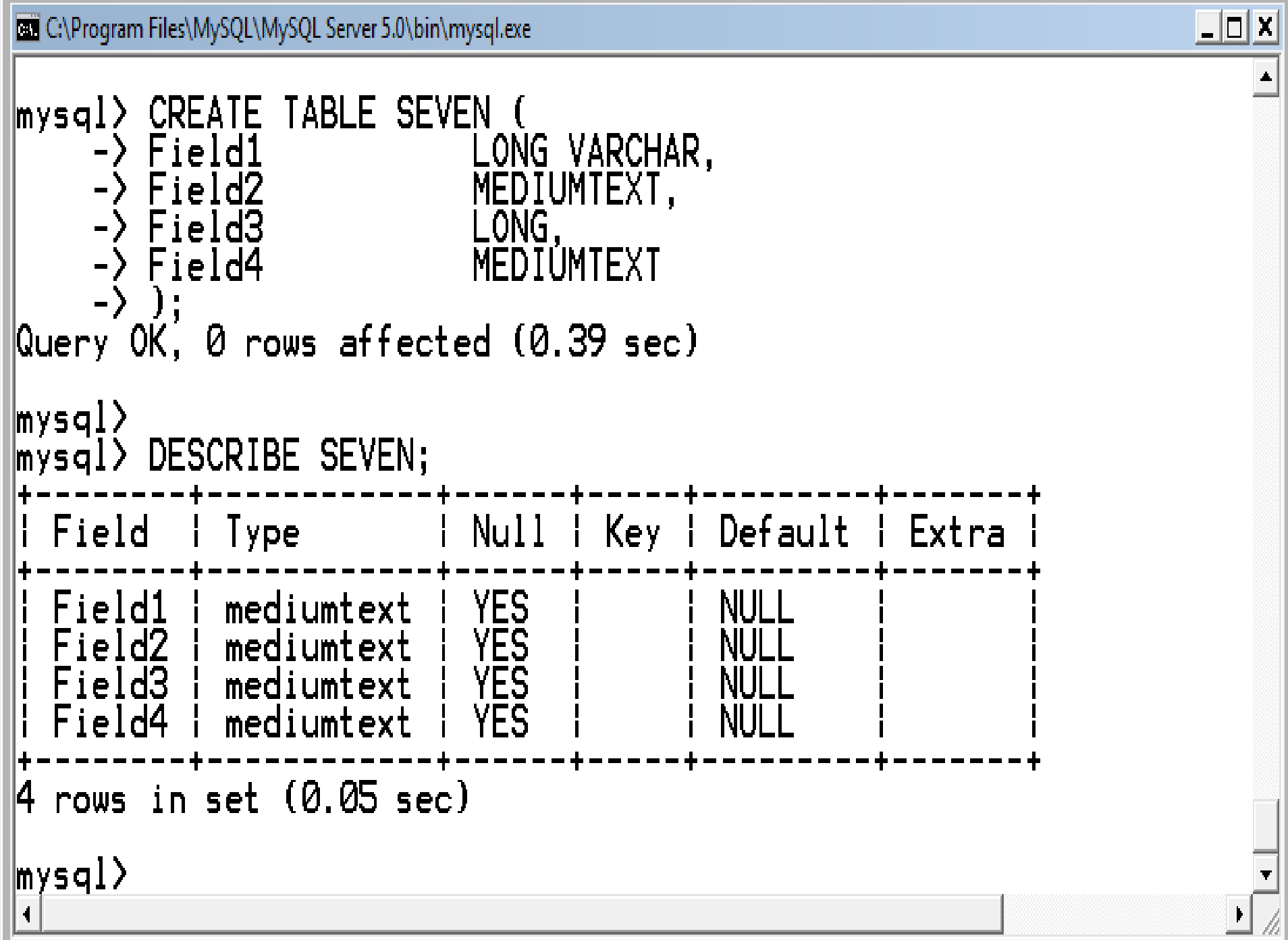
C:\Program Files\MySQL\MySQL Server 5.0\bin\mysql.exe

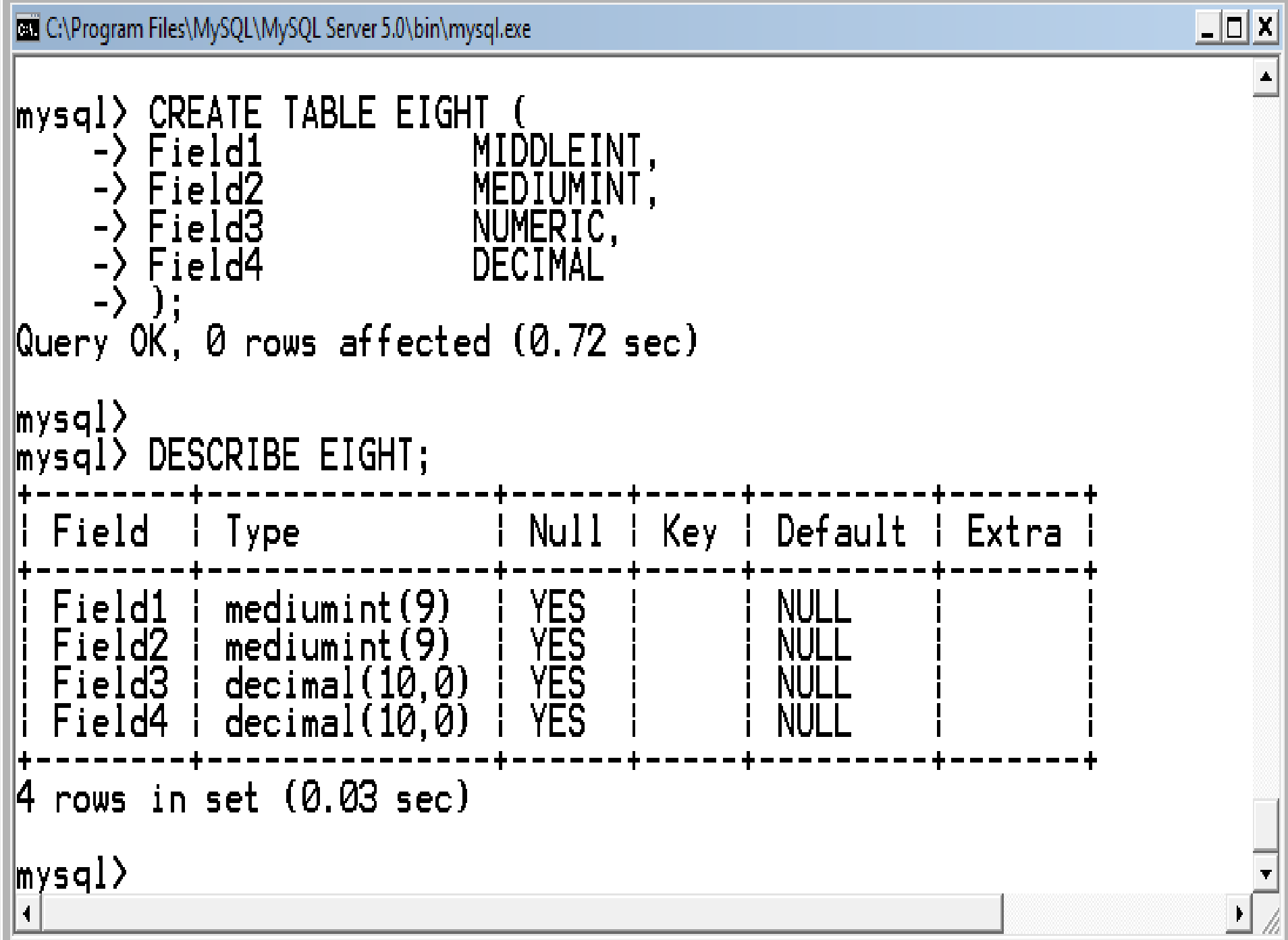
```
mysql> CREATE TABLE FOUR (
    -> Field1           INT1,
    -> Field2           TINYINT,
    -> Field3           INT2,
    -> Field4           SMALLINT
    -> );
Query OK, 0 rows affected (0.07 sec)

mysql>
mysql> DESCRIBE FOUR;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| Field1 | tinyint(4)  | YES  |     | NULL    |       |
| Field2 | tinyint(4)  | YES  |     | NULL    |       |
| Field3 | smallint(6) | YES  |     | NULL    |       |
| Field4 | smallint(6) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
4 rows in set (0.03 sec)

mysql>
```

C:\Program Files\MySQL\MySQL Server 5.0\bin\mysql.exe

```
mysql> CREATE TABLE FIVE (
    -> Field1           INT3,
    -> Field2           MEDIUMINT,
    -> Field3           INT4,
    -> Field4           INT
    -> );
Query OK, 0 rows affected (0.23 sec)

mysql>
mysql> DESCRIBE FIVE;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| Field1 | mediumint(9) | YES  |     | NULL    |       |
| Field2 | mediumint(9) | YES  |     | NULL    |       |
| Field3 | int(11)     | YES  |     | NULL    |       |
| Field4 | int(11)     | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
4 rows in set (0.05 sec)

mysql> _
```

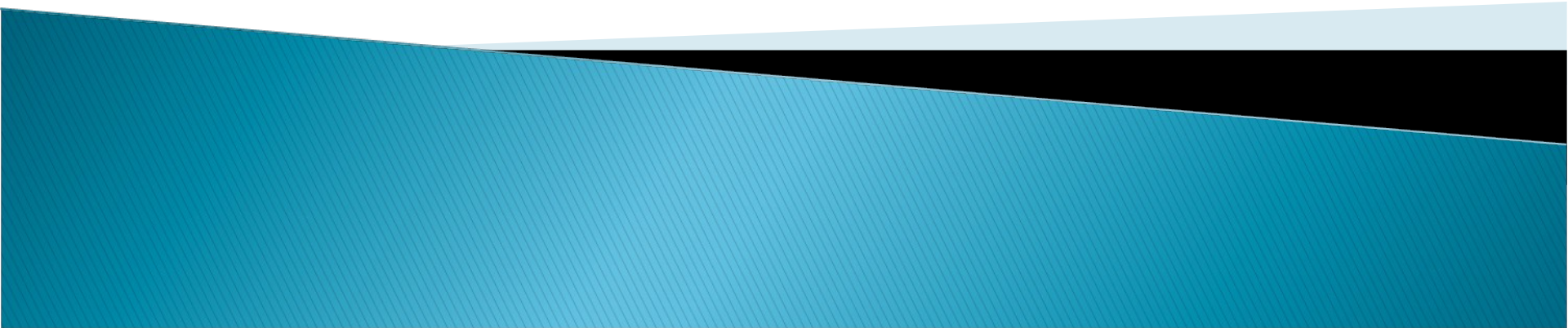

C:\Program Files\MySQL\MySQL Server 5.0\bin\mysql.exe
mysql> CREATE TABLE SIX (
-> Field1 INT8,
-> Field2 BIGINT,
-> Field3 LONG VARBINARY,
-> Field4 MEDIUMBLOB
->);
Query OK, 0 rows affected (0.23 sec)
mysql>
mysql> DESCRIBE SIX;
+--------+-----------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+--------+-----------+------+-----+---------+-------+
Field1	bigint(20)	YES		NULL	
Field2	bigint(20)	YES		NULL	
Field3	mediumblob	YES		NULL	
Field4	mediumblob	YES		NULL	
+--------+-----------+------+-----+---------+-------+
4 rows in set (0.05 sec)
mysql> _


C:\Program Files\MySQL\MySQL Server 5.0\bin\mysql.exe
mysql> CREATE TABLE SEVEN (
-> Field1 LONG VARCHAR,
-> Field2 MEDIUMTEXT,
-> Field3 LONG,
-> Field4 MEDIUMTEXT
->);
Query OK, 0 rows affected (0.39 sec)
mysql>
mysql> DESCRIBE SEVEN;
+--------+------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+--------+------------+------+-----+---------+-------+
Field1	mediumtext	YES		NULL	
Field2	mediumtext	YES		NULL	
Field3	mediumtext	YES		NULL	
Field4	mediumtext	YES		NULL	
+--------+------------+------+-----+---------+-------+
4 rows in set (0.05 sec)
mysql>


C:\Program Files\MySQL\MySQL Server 5.0\bin\mysql.exe
mysql> CREATE TABLE EIGHT (
-> Field1 MIDDLEINT,
-> Field2 MEDIUMINT,
-> Field3 NUMERIC,
-> Field4 DECIMAL
->);
Query OK, 0 rows affected (0.72 sec)
mysql>
mysql> DESCRIBE EIGHT;
+--------+--------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+--------+--------------+------+-----+---------+-------+
Field1	mediumint(9)	YES		NULL	
Field2	mediumint(9)	YES		NULL	
Field3	decimal(10,0)	YES		NULL	
Field4	decimal(10,0)	YES		NULL	
+--------+--------------+------+-----+---------+-------+
4 rows in set (0.03 sec)
mysql>

Structured Query Language (SQL) 07

By: M Shuaib Zarinkhail 2010



Database Engines

- A database engine is the underlying software component that a DBMS uses to create, retrieve, update and delete (CRUD) data from a DB
- MySQL supports several storage engines that act as handlers for different table types including:
  - Transaction-safe (transactional) tables
  - Nontransaction-safe (nontransactional) tables

Transactional Vs Non-transactional Tables

- Transactional tables in comparing with nontransactional tables, need:
  - significantly higher memory
  - more disk space
  - more CPU overhead
- On the other hand, transactional storage engines such as InnoDB offer many significant features than nontransactional storage engines like MyISAM

Database Engines

- The original storage engine for MySQL was ISAM
  - ISAM managed nontransactional tables
  - This engine has been replaced by MyISAM and should no longer be used
  - It is deprecated in MySQL 4.1
- The ISAM storage engine is no longer be distributed from MySQL 5.0

Database Engines

- In MySQL 3.23.0, the MyISAM and HEAP storage engines were introduced
  - MyISAM is an improved replacement for ISAM
  - MyISAM also manages nontransactional tables
  - The HEAP storage engine provides in-memory tables
    - The HEAP storage engine has been renamed the MEMORY engine
  - The MERGE storage engine was added in MySQL 3.23.25
    - It allows a collection of identical MyISAM tables to be handled as a single table

Database Engines

- The InnoDB storage engine that handles transaction-safe tables were introduced in MySQL 3.23
  - The InnoDB is included by default in all MySQL binary distributions
- In source distributions, you can enable or disable either engine by configuring MySQL as you like

Database Engines

- The EXAMPLE storage engine was added in MySQL 4.1.3
  - It is a “stub” engine that does nothing
  - You can create tables with this engine, but no data can be stored in them or retrieved from them
  - The purpose of this engine is to serve as an example in the MySQL source code that illustrates how to begin writing new storage engines
  - As such, it is primarily of interest to developers

Database Engines

- NDBCLUSTER is the storage engine used by MySQL Cluster to implement tables that are partitioned over many computers
  - It is available in source code distributions as of MySQL 4.1.2 and binary distributions as of MySQL 4.1.3
- The ARCHIVE storage engine was added in MySQL 4.1.3
  - It is used for storing large amounts of data without indexes in a very small space

Database Engines

- The CSV storage engine was added in MySQL 4.1.4
  - This engine stores data in text files using comma-separated values format
- The BLACKHOLE storage engine was added in MySQL 4.1.11
  - This engine accepts but does not store data and retrievals always return an empty set

Database Engines

- SHOW ENGINES command displays status information about the server's storage engines
- This is particularly useful for checking whether a storage engine is supported, or to see what the default engine is
- SHOW TABLE TYPES is a synonym for this command

Database Engines

- e.g. → SHOW ENGINES;
- The following slides show the database engines that are supported or not-supported by the MySQL 5.0.45

Database Engines

Engine	Support	Comment
MyISAM	YES	Default engine as of MySQL 3.23 with great performance
MEMORY	YES	Hash based, stored in memory, useful for temporary tables
InnoDB	DEFAULT	Supports transactions, row-level locking, and foreign keys
Berkeley DB	NO	Supports transactions and page-level locking

Database Engines

Engine	Support	Comment
BLACKHOLE	YES	/dev/null storage engine (anything you write to it disappears)
EXAMPLE	NO	Example storage engine
ARCHIVE	YES	Archive storage engine
CSV	NO	CSV storage engine

Database Engines

Engine	Support	Comment
ndbcluster	NO	Clustered, fault-tolerant, memory-based tables
FEDERATED	YES	Federated MySQL storage engine
MRG_MYISAM	YES	Collection of identical MyISAM tables
ISAM	NO	Obsolete storage engine

Database Engines

- When you create a new table, you can specify which storage engine to use
  - You can do this by adding an ENGINE or TYPE table option to the CREATE TABLE statement:

```
CREATE TABLE tOne (f1 INT) ENGINE =
  INNODB;
CREATE TABLE tOne (f1 INT) TYPE =
  MEMORY;
```

Database Engines

- ENGINE is the preferred term, but cannot be used before MySQL 4.0.18
- TYPE is available beginning with MySQL 3.23.0
  - This is the first version of MySQL for which multiple storage engines were available
  - TYPE is supported for backward compatibility but is deprecated

Database Engines

- If you omit the ENGINE or TYPE option, the default storage engine is used
- You can set the default storage engine to be used during the current session by setting the storage_engine or table_type variable:
  - SET storage_engine = MYISAM;
  - SET table_type = BDB;

Database Engines

- If you try to use a storage engine that is not compiled in or that is compiled in but deactivated
  - MySQL does not make error, instead it creates a table using the default storage engine

SQL-DDL

- Column definitions include three parts:
2. Column Name
   - Any word or phrase (explained earlier as identifier)
     - e.g. Name, Date_Of_Birth, …
3. Column Data-Type
   - Determines a domain for a column
     - e.g. Char, Varchar, Integer, …
- Constraints (Optional)
   - Determines additional features for a column
     - e.g. Primary Key, Null, Not Null, Default, …

SQL-DDL (Table Constraints)

- To specify Primary Keys:

```
CREATE TABLE PROJECT (
  PRIMARY KEY (ProjectID) );
CREATE TABLE EMPLOYEE (
  PRIMARY KEY (EmployeeNumber) );
CREATE TABLE ASSIGNMENT (
  PRIMARY KEY (ProjectID, EmployeeNum) );
```

SQL-DDL (Table Constraints)

- An alternative way (for existing tables)
  - Type →ALTER TABLE tablename ADD PRIMARY KEY (tablefieldnames)
    - e. g. →alter table project add primary key (projectid);
    - e. g. →alter table employee add primary key (employeenumber);
    - e. g. →alter table assignment add primary key (projectid, employeenum);

SQL-DDL (Table Constraints)

- You can delete a primary key from an existing table
- To do:

  Type →ALTER TABLE TableName DROP PRIMARY KEY

  e.g. →alter table project drop primary key;

SQL-DDL (Table Constraints)

- To set relationships, specify Foreign Keys:

```
CREATE TABLE ASSIGNMENT (
        “                “                “
        “                “                “
    FOREIGN KEY (ProjectID) REFERENCES
    PROJECT (ProjectID) ON DELETE CASCADE,
    FOREIGN KEY (EmployeeNum) REFERENCES
    EMPLOYEE (EmployeeNumber) ON DELETE
    NO ACTION
);
```

SQL-DDL (Table Constraints)

- An alternative way (for existing tables)
- To do:

  Type →ALTER TABLE tableone ADD FOREIGN KEY (tableone.fieldnames) REFERENCES tabletwo (tabletwo.fieldnames) ON DELETE CASCADE / ON DELETE NO ACTION

  Examples (Next Slide)

SQL-DDL (Table Constraints)

- ALTER TABLE Examples

→alter table assignment add foreign key (projectid) references project (projectid) on delete cascade;

→alter table assignment add foreign key (employeenum) references employee (employeenumber) on delete no action;

SQL-DDL (Foreign Key Constraints)

- You can use the following two keywords for references in a relationship
- ON DELETE and/or ON UPDATE
- Each of the keywords can use the following options:
  - CASCADE – SET NULL
  - NO ACTION – RESTRICT
  - SET DEFAULT

SQL-DDL (Foreign Key Constraints)

- CASCADE
  - Delete or update the row from the parent table and automatically delete or update the matching rows in the child table
  - ON DELETE CASCADE is supported starting from MySQL 3.23.50 and ON UPDATE CASCADE is supported from MySQL 4.0.8

SQL-DDL (Foreign Key Constraints)

- SET NULL
  - Delete or update the row from the parent table and set the foreign key column or columns in the child table to NULL
  - This is valid only if the FK columns do not have the NOT NULL option specified
  - ON DELETE SET NULL is available starting from MySQL 3.23.50 and ON UPDATE SET NULL is available starting from 4.0.8

SQL-DDL (Foreign Key Constraints)

- NO ACTION
  - In standard SQL, NO ACTION means no action in the sense that an attempt to delete or update a primary key value will not be allowed to proceed if there is a related foreign key value in the referenced table
  - Starting from MySQL 4.0.18 the InnoDB rejects the delete or update operation for the parent table

SQL-DDL (Foreign Key Constraints)

- RESTRICT
  - Rejects the delete or update operation for the parent table Specifying RESTRICT (or NO ACTION) is the same as omitting the ON DELETE or ON UPDATE clause
  - Some database systems have deferred checks, and NO ACTION is a deferred check
  - In MySQL, FK constraints are checked immediately, so NO ACTION is the same as RESTRICT

SQL-DDL (Foreign Key Constraints)

- SET DEFAULT
  - Delete or update the row from the parent table and set the FK column or columns in the child table to DEFAULT
  - The InnoDB rejects table definitions containing ON DELETE SET DEFAULT or ON UPDATE SET DEFAULT clauses