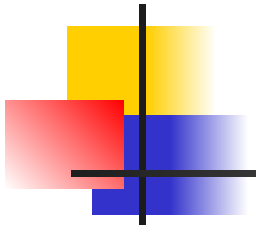


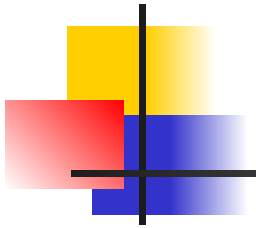
Лекция №2

План лекции:

1. Логическая архитектура MySQL.
2. Типы данных СУБД MySQL.
3. Выбор типов полей в MySQL.
4. Типы таблиц СУБД MySQL (MyISAM и InnoDB).
5. Создание таблиц MyISAM и InnoDB.
6. Ссылочная целостность связей.

Логическая архитектура MySQL

Типы данных СУБД MySQL



СУБД MySQL поддерживает следующие типы данных, допустимые для полей таблиц:

– числовые (точные и приближенные числовые типы);
– строковые;
– календарные;
– NULL (значение, записываемое в поле таблицы и обозначающее отсутствие информации).

Числовые типы данных СУБД MySQL (Точные числовые данные)

Объявление типа	Объем памяти	Диапазон
TINYINT[(M)] [UNSIGNED] [ZEROFILL]	M × 1 byte 1 ≤ M ≤ 3	От -128 до 127 (от -2^7 до 2^7-1)
TINYINT UNSIGNED TINYINT(1) UNSIGNED	1 byte M=1	от 0 до 255 (от 0 до 2^8-1)
SMALLINT[(M)] [UNSIGNED] [ZEROFILL]	M × 2 byte 1 ≤ M ≤ 5	От -32 768 до 32°767 (от -2^{15} до $2^{15}-1$)
SMALLINT UNSIGNED	2 byte 1 ≤ M ≤ 5	от 0 до 65535 (от 0 до $2^{16}-1$)
MEDIUMINT[(M)] [UNSIGNED] [ZEROFILL]	M × 3 byte 1 ≤ M ≤ 5	От -2^{23} до $2^{23}-1$
MEDIUMINT UNSIGNED MEDIUMINT(1) UNSIGNED	3 byte M=1	от 0 до 16 777°215 (от 0 до $2^{24}-1$)

Числовые типы данных СУБД MySQL (Точные числовые данные)

продолжение

Объявление типа	Объем памяти	Диапазон
INT[(M)] [UNSIGNED][ZEROFILL] INTEGER синоним INT	M × 4 byte 1 ≤ M ≤ 255	От -2^{31} до $2^{31}-1$
INT UNSIGNED INT(1) UNSIGNED	M × 4 byte M=1	от 0 до 2^{32} – 1
BIGINT[(M)] [UNSIGNED] [ZEROFILL]	M × 8 byte	От -2^{63} до 2^{63} – 1 (от 0 до 2^{M})
BIT[(M)]	(M+7)/8 1 ≤ M ≤ 64	От 1 до 64 битов
BOOL (синоним BOOLEAN)	1 byte	0 (false) или 1 (true)
TINYINT(1)	1 byte	0 (false) или 1 (true)
DECIMAL[(M[,D])] DEC[(M[,D])] NUMERUC[(M[,D])] с атрибутами [UNSIGNED] [ZEROFILL]	(M + 2) byte 1 ≤ M ≤ 64 0 ≤ D ≤ 30	Зависит от параметров точности (M) и шкалы (D)

Числовые типы данных СУБД MySQL (Приближенные числовые типы)

Объявление типа	Объем памяти	Диапазон
Число с плавающей запятой и одинарной точностью FLOAT[(M[,D])] [UNSIGNED] [ZEROFILL]	4 byte 1≤M≤24	От 1,175494351E-38 До 3,402823466E+38 Точность – примерно 7 знаков после запятой.
Число с плавающей запятой и двойной точностью REAL[(M[,D])] [UNSIGNED] [ZEROFILL] DOUBLE PRECISION[(M[,D])] [UNSIGNED] [ZEROFILL]	8 byte 25≤M≤53	от 1.797693134862315E+308 до -2.2250738585072E-308, 0, от 2.225073858507201E-308 до 1.79769313486231E+308.

Строковые типы данных СУБД MySQL

Объявление типа	Объем памяти Размер	Описание
[NATIONAL] VARCHAR(M) [BINARY]	L+1 byte 1≤M≤255 L≤M	Строка переменной длины. M – задано символов (byte), (L+1) – фактически выделяется для хранения (концевые пробелы удаляются). Нельзя смешивать типы CHAR и VARCHAR. Если есть поле VARCHAR, то все поля типа CHAR будут приведены к типу VARCHAR.
[NATIONAL] CHAR(M) [BINARY]	M byte 1≤M≤255	Строка фиксированной длины. Независимо от длины строки, использует для ее хранения все M символов (дополняется пробелами). CHAR синоним CHAR(1). CHAR(0) принимает только 2 значения: NULL или ""
BLOB	L+2 byte	При выполнении операций над полями типа BLOB кодировка не учитывается.
TEXT	L+2 byte L≤2^{16}-1= =65’535	При выполнении операций над полями типа TEXT учитывается кодировка. Только этот тип используется для полнотекстового поиска

Строковые типы данных СУБД MySQL

продолжение

Объявление типа	Объем памяти Размер	Описание
TINYBLOB TINYTEXT	L+1 byte L<2^8-1=255	Уменьшенный BLOB Уменьшенный TEXT
MEDIUMBLOB MEDIUMTEXT	L+3 byte L<2^24-1= =16'777'215	Средний BLOB Средний TEXT
LONGBLOB LONGTEXT	L+4 byte L<2^32-1= =4'294'967'296	Длинный BLOB Длинный TEXT
ENUM('element1', 'element2', ...)	1 byte или 2 byte	Перечисление. Список ENUM может содержать 256 элементов (1 байт) или от 257 до 65536 элементов (2 байта). Объект строки может иметь только одно значение, выбранное из заданного списка элементов.
SET('element1', 'element2', ...)	1, 2, 3, 4 или 8 byte	Набор. Список SET может содержать: от 1 до 8 элементов (1 байт), от 9 до 16 элементов (2 байта), от 17 до 24 элементов (3 байта), от 25 до 32 элементов (4 байта), от 33 до 64 элементов (8 байт). Таблица может содержать не более 255 уникальных определений ENUM и SET.

Календарные типы данных СУБД MySQL

Тип	Объем памяти	Диапазон допустимых значений
DATE	3 byte	от '1000-01-01' до '9999-12-31'
TIME	3 byte	от '-828:59:59' до '828:59:59'
DATATIME	8 byte	от '1000-01-01 00:00:00' до '9999-12-31 00:00:00'
TIMESTAMP(M)	4 byte	от '1970-01-01 00:00:00' до '2038-12-31 59:59:59'
YEAR(2)		формат YY, диапазон – от 1970 до 2069
YEAR(4)	1 byte	формат YYYY, диапазон – от 1901 до 2155

Выбор типов полей в MySQL

Общие рекомендации по использованию типов полей можно свести к следующим принципам:

– не используйте символьные поля для цифровых данных;
– используйте столбцы фиксированной длины;
– используйте как можно более короткие столбцы;
– по возможности используйте типы ENUM и SET;
– избегайте выборки полей с типом TEXT и по возможности храните их в отдельных таблицах;
– не забывайте объявлять столбцы как NOT NULL;
– используйте типы даты и времени;
– хранить IP-адреса как UNSIGNED INT (INET_ATON (), INET_NTOA ());
– хранить пароли как CHAR() или BIN() (UNHEX(), HEX())
– не забывайте оптимизировать таблицы, подверженные частым изменениям, и проверяйте характеристики столбцов уже наполненных таблиц (*PROCEDURE ANALYZE()*).

Различие MyISAM и InnoDB

Описание	*MyISAM*	*InnoDB*
Возможность использования транзакций	Нет	Да
Поддержка внешних ключей	Нет	Да
Уровень блокировки	на уровне таблиц	на уровне строк
Одновременные запросы к разным частям таблицы.	Медленнее	Быстрее
При смешанной нагрузке в таблице (select/update/delete/insert)	Медленнее	Быстрее
Операция Insert	Быстрее	Медленнее (из-за транзакции)
Если преобладают операции чтения (SELECT)	Быстрее	Медленнее
Deadlock	Не возникают	Возможны.

Различие MyISAM и InnoDB

продолжение

Описание	*MyISAM*	*InnoDB*
Поддержка полнотекстового поиска	Да	Нет (доступен начиная с версии MySQL 5.6.4)
Запрос Count(*)	Быстрее	Медленнее
Поддержка mysqlhotcopy	Да	Нет
Файловое хранение таблиц	Каждой таблице отдельный файл	Данные хранятся в больших совместно используемых файлах (по умолчанию)
Бинарное копирование таблиц	Да	Нет
Размер таблиц в БД	Меньше	Больше
Поведение в случае сбоя	«Ломается» вся таблица	По логам можно все восстановить
В случае хранения «логов» и подобного	Лучше	Хуже

Создание таблиц MyISAM

Синтаксис CREATE TABLE	Пример
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] *`db_name`.`tbl_name`* (*create_definition,...,...*) [*table_options*] *create definition:* (*`col_name`* DATA_TYPE | [[DEFAULT] CHARACTER SET [=] `charset_name` | [DEFAULT] COLLATE [=] `collation_name`] | {NOT NULL	NULL} | [AUTO_INCREMENT] | [DEFAULT *value*] *`col_name2`*,...,*`col_name3`*,... PRIMARY KEY (*`col_name1`*,...,...) INDEX [*`index_name1`*] (*`col_name2`*, [ASC | DESC]), INDEX ..., INDEX ...) *table option:* [ENGINE [=] {MyISAM	InnoDB}] | [[DEFAULT] CHARACTER SET [=] *charset_name* | COLLATE [=] *collation_name*]	CREATE TABLE IF NOT EXISTS `Shop_innodb`.`Order` (`id_order` INT UNSIGNED NOT NULL AUTO_INCREMENT, `fk_client` INT UNSIGNED NOT NULL DEFAULT 0, `fk_delivery` INT UNSIGNED NULL, `data` DATE NOT NULL, `st_initiate` BIT(1) NOT NULL DEFAULT 0, `st_executed` BIT(1) NOT NULL DEFAULT 0, `address` TEXT CHARACTER SET 'utf8' COLLATE 'utf8_unicode_ci' NULL, `price_delivery` FLOAT(6,2) NULL, `data_delivery` TIME NULL, `time_delivery` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8_unicode_ci' NULL, `phone_delivery` INT UNSIGNED NULL, *#объявление первичного ключа* PRIMARY KEY (`id_order`, `fk_client`), *#объявление индексов* INDEX `fk_Order_Client_id_name` (`fk_client` ASC), INDEX `fk_Order_Delivery_id_name` (`fk_delivery` ASC),) *#объявление опций таблицы* ENGINE = MyISAM DEFAULT CHARACTER SET = utf8 COLLATE = utf8_unicode_ci

Создание таблиц InnoDB

Синтаксис CREATE TABLE	Пример

CREATE [TEMPORARY] TABLE
[IF NOT EXISTS]
`db_name`.`tbl_name1`
(
create_definition,...
)
[table_options]
 	
create definition:
(
`col_name1` DATA_TYPE
| [[DEFAULT] CHARACTER
SET[=]`charset_name`
| [DEFAULT]
COLLATE[=]`collation_name`]
| {NOT NULL | NULL}
| [AUTO_INCREMENT]
| [DEFAULT value]
`col_name2`,...,`col_name3`,...
PRIMARY KEY (`col_name1`,...)
INDEX [`index_name1`]
(`col_name2`, [ASC | DESC]),
INDEX ..., INDEX ...
CONSTRAINT FOREIGN KEY
(`index_name1`)
REFERENCES `db_name`.`tbl_name2`
(`col_name`)
ON DELETE {RESTRICT | CASCADE |
SET°NULL | NO ACTION}
ON UPDATE {RESTRICT | CASCADE |
SET NULL | NO ACTION},
CONSTRAINT ..., CONSTRAINT ...
)
table option:
[ENGINE [=] {MyISAM | InnoDB}]
| [[DEFAULT] CHARACTER
SET[=]charset_name
| COLLATE[=]collation_name]
	CREATE TABLE IF NOT EXISTS
	`Shop_innodb`.`Order`
	(
	`id_order` INT UNSIGNED NOT NULL
	...AUTO_INCREMENT,
	`fk_client` INT UNSIGNED NOT NULL
	...DEFAULT 0,
	`fk_delivery` INT UNSIGNED NULL,
	`data` DATE NOT NULL,
	`st_initiate` BIT(1) NOT NULL DEFAULT
	0,
	`st_executed` BIT(1) NOT NULL DEFAULT
	0,
	`address` TEXT CHARACTER SET 'utf8'
	...COLLATE 'utf8_unicode_ci' NULL,
	`price_delivery` FLOAT(6,2) NULL,
	`data_delivery` TIME NULL,
	`time_delivery` VARCHAR(45)
	...CHARACTER SET 'utf8'
	...COLLATE 'utf8_unicode_ci' NULL,
	`phone_delivery` INT UNSIGNED NULL,
	#объявление первичного ключа
	PRIMARY KEY (`id_order`, `fk_client`),
	#объявление индексов
	INDEX `fk_Order_Client_id_name`
	...(`fk_client` ASC),
	INDEX `fk_Order_Delivery_id_name`
	...(`fk_delivery` ASC),
	#объявление ограничений
	CONSTRAINT `fk_Order_Client_name`
	...FOREIGN KEY (`fk_client`)
	...REFERENCES `Shop_innodb`.`Client`
	...(`id_client`)
	...ON DELETE RESTRICT
	...ON UPDATE RESTRICT,
	CONSTRAINT `fk_Order_Delivery_name`
	...FOREIGN KEY (`fk_delivery`)
	...REFERENCES `Shop_innodb`.`Delivery`
	...(`id_delivery`)
	...ON DELETE RESTRICT
	...ON UPDATE RESTRICT
	)
	#объявление опций таблицы
	ENGINE = InnoDB
	DEFAULT CHARACTER SET = utf8
	COLLATE = utf8_unicode_ci

Определение ссылочной целостности связей

СУБД MySQL поддерживает ссылочную целостность связей для двух SQL-операций манипулирования данными – UPDATE и DELETE.

Ссылочная целостность реализуется в виде четырех вариантов действий: RESTRICT, CASCADE, SET NULL, NO ACTION.

При задании ссылочной целостности требуется учитывать следующие условия:

– если инструкции ограничения внешних ключей ON DELETE и/или ON UPDATE при создании таблицы не определены, по умолчанию поддерживается ссылочное действие ON UPDATE RESTRICT и/или ON DELETE RESTRICT;

– если внешний ключ (FK) определен в двух таблицах и обе таблицы являются родительским и дочерними, для каждого внешнего ключа (FK) должно быть определено ссылочное ограничение (действие) ON UPDATE CASCADE и/или ON DELETE CASCADE;

– если при выполнении операции ссылочных ограничений ON UPDATE CASCADE или ON UPDATE SET NULL для обновления одной и той же таблицы возникает рекурсия, то их действие изменяется на RESTRICT;

– ссылочные ограничения ON DELETE SET NULL и ON UPDATE SET NULL поддерживаются только для внешних ключей (FK), которые могут принимать значения NULL;

– каскадные ссылочные ограничения внешнего ключа (ON UPDATE CASCADE, ON DELETE CASCADE) не активируют триггеры;

– ссылочные ограничения ON DELETE SET DEFAULT и ON UPDATE SET DEFAULT не поддерживаются для таблиц типа InnoDB (в СУБД MySQL это ключевое слово зарезервировано, но не обрабатывается).

Задание ссылочной целостности для таблиц типа InnoDB

Ссылочные ограничения	Дочерняя таблица (Foreign Key, FK)	Родительская таблица (Primary Key, PK)
ON DELETE RESTRICT	Удаление кортежа (FK) возможно без ограничений. Ситуации, когда в родительской таблице отсутствует кортеж с PK=FK, не может существовать (иначе — операция удаления приведет к ошибке).	Удаление кортежа (PK) невозможно, если в дочерней таблице есть кортежи отвечающие условию FK=PK.
ON UPDATE RESTRICT	Обновление FK возможно, если в родительской таблице имеется PK, со значением нового FK. Исключение — присвоение FK значения NULL, если при создании таблицы определено, что FK может принимать значение NULL.	Обновление PK возможно, если в дочерней таблице нет FK, совпадающего с новым значением PK (FK≠PK).
ON DELETE CASCADE	Эквивалентно ON DELETE RESTRICT	Удаление кортежа (PK) приводит к удалению всех кортежей в дочерней таблице, у которых FK совпадает с PK (FK=PK).
ON UPDATE CASCADE	Эквивалентно ON UPDATE RESTRICT	Изменение значения PK приводит к такому же изменению всех FK с совпадающими значениями (FK=PK) в дочерней таблице.

Задание ссылочной целостности для таблиц типа InnoDB

продолжение

Ссылочные ограничения	Дочерняя таблица (Foreign Key, FK)	Родительская таблица (Primary Key, PK)
ON DELETE SET NULL	Эквивалентно ON DELETE RESTRICT Работает только для FK, допускающих значение NULL.	Удаление кортежа (PK) приводит к присвоению всем FK с совпадающими значениями (FK=PK) в дочерней таблице значений NULL.
ON UPDATE SET NULL	Эквивалентно ON UPDATE RESTRICT Работает только для FK, допускающих значение NULL.	Изменение значения PK приводит к присвоению всем FK с совпадающими значениями (FK=PK) в дочерней таблице значений NULL.
ON DELETE NO ACTION	Эквивалентно ON DELETE RESTRICT	Эквивалентно ON DELETE RESTRICT
ON UPDATE NO ACTION	Эквивалентно ON UPDATE RESTRICT	Эквивалентно ON UPDATE RESTRICT