CS3200/CS5200: Databases

Dr. John Rachlin
Email: j.rachlin@northeastern.edu

MySQL Datatypes & CREATE TABLE basics

Objectives for today

- Review of most common datatypes
- Review basic table creation and a few variations
- More SQL elements

Most common datatypes

- CHAR(n), VARCHAR(n)
- INT   (SIGNED or UNSIGNED)
- FLOAT, DOUBLE, DECIMAL(m,d)
- DATE, TIME, DATETIME
- BLOB, TEXT (CLOB)
- ENUM

CHAR vs VARCHAR

- **CHAR(n)** is a fixed-width string with maximum length n.
  Typically used when the attribute always has the same size.
  For example: State abbreviations – VT, MA, CA uses CHAR(2)

- **VARCHAR(n)** is a variable-width string with maximum length n.
  Use for strings that vary in size: usernames, email, addresses…
  Adds a tiny bit of memory overhead (1 or 2 bytes) to store length.
  Most commonly used.

INT

- Integers are declared using INT
- INT(x) means x characters will be displayed in MySQL Workbench
- Can be declared as SIGNED INT or UNSIGNED INT
- Variations: TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT

Datatype	Bytes	Range
TINYINT (BOOLEAN)	1	Signed: -128 to 127 Unsigned: 0 to 255
SMALLINT	2	Signed: -32,768 to 32,767 Unsigned: 0 to 65,535
MEDIUMINT	3	Signed: -8,388,608 to 8,388,607 Unsigned: 0 to 16,777,215
INT	**4**	**Signed: -2,1447,483,648 to 2,147,483,647** **Unsigned: 0 to 4,294,967,295**
BIGINT	8	Signed: -9x10^{18} to 9x10^{18} Unsigned: 0 to 1.8x10^{19}

FLOAT, DOUBLE, DECIMAL(m,d)

- **FLOAT**: Single-precision real (4 bytes)
  $-3.4x10^{38}$ to $3.4x10^{38}$

- **DOUBLE**: Double-precision real (8 bytes)
  $-1.8x10^{308}$ to $1.8x10^{308}$

- **DECIMAL(m,d)** packs actual decimal characters into bytes.
  m = maximum number of digits total (precision) 1..65
  d = number of digits after the decimal point (scale) 0..30
  An “exact” numerical datatype
  Calculations on decimals may be slower due to conversions
  Used in financial / business-related databases

DATE, TIME, DATETIME

- **DATE** stores just a date

- **TIME** stores just a time (1 second precision)
  TIME(d) fractional seconds (d = number of digits after decimal point)
  e.g. TIME(4) = 22:05:16.5398

- **DATETIME** stores both a date and a time (1 second precision)
  DATETIME(d) fractional seconds
  e.g. DATETIME(6) = 2017-09-21 17:39.59.999999
  Used for timestamps, e.g., when was the user created?

- Fun fact: The function **now()** or **now(d)** returns the current date & time

BLOBs and TEXT (CLOBs)

- **BLOB:** Binary Large Object (64 kb)
  Example usages: audio, video, images, PDFs– anything not plain text

  LONGBLOB (4GB), **MEDIUMBLOB** (16MB), **TINYBLOB** (255 bytes)

- **TEXT:** Large text (64 kb)
  Example usages: large text files, e.g. XML files, log files
  Equivalent to the more standard datatype "CLOB"

  LARGETEXT (4GB), **MEDIUMTEXT** (16 MB), **TINYTEXT** (255 bytes)

ENUMs

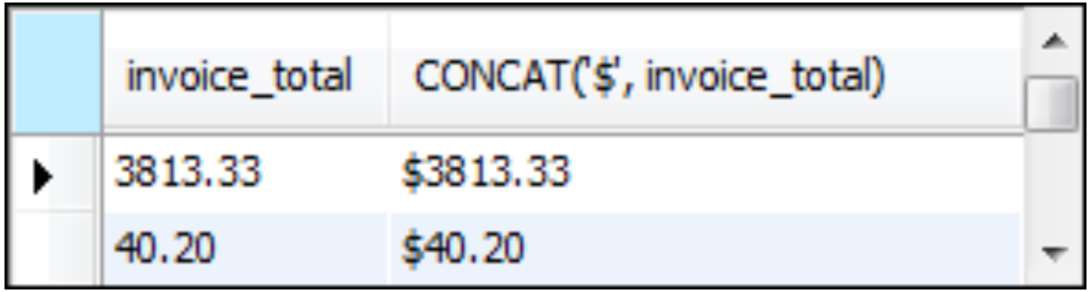
- Categorical values - usually with just a few valid choices

  CREATE TABLE order (
  .
  .
  size ENUM('x-small', 'small', 'medium', 'large', 'x-large'),
  status ENUM(‘pending’, ’in process’, ‘complete’),
  .
)

Datatype conversion

Implicit conversion: The datatype is automatically converted to satisfy the query

```
SELECT
  invoice_total, CONCAT('$', invoice_total)
FROM invoices
```



invoice_total	CONCAT('$', invoice_total)
3813.33	$3813.33
40.20	$40.20

Datatype conversion

Explicit conversion: The datatype of some expression is converted to the user-specified target datatype.

- **CAST** (expression **AS** *datatype*)
- **CONVERT** (expression, *datatype*)

Valid target datatypes:

CHAR, DATE, TIME, DATETIME, SIGNED, UNSIGNED, DECIMAL

CREATE TABLE

Keywords you should know when creating a database table.
(In addition to the datatypes!)

Getting Started	More Advanced (Later)
CREATE TABLE	**CONSTRAINT**
PRIMARY KEY	**FOREIGN KEY**
AUTO_INCREMENT	**REFERENCES**
NOT NULL	**ON DELETE / CASCADE / SET NULL**
UNIQUE	**COLLATE**
DEFAULT	**CHARACTER SET**
	ENGINE

Creating Table Examples

A very basic table with just four columns.

```
USE demo;

DROP TABLE IF EXISTS users;

CREATE TABLE users (
    user_id INT,
    username VARCHAR(20),
    dob DATE,
    class ENUM ('Basic', 'Premium')
);
```

What is a PRIMARY KEY?

- A **PRIMARY KEY** is a value or set of values that uniquely identifies a single row in your database table.
- Primary keys cannot be NULL (empty / unassigned)
- Primary keys must be unique
- Primary keys are usually INTs but they could be other datatypes

A Relational Data Model

Branch

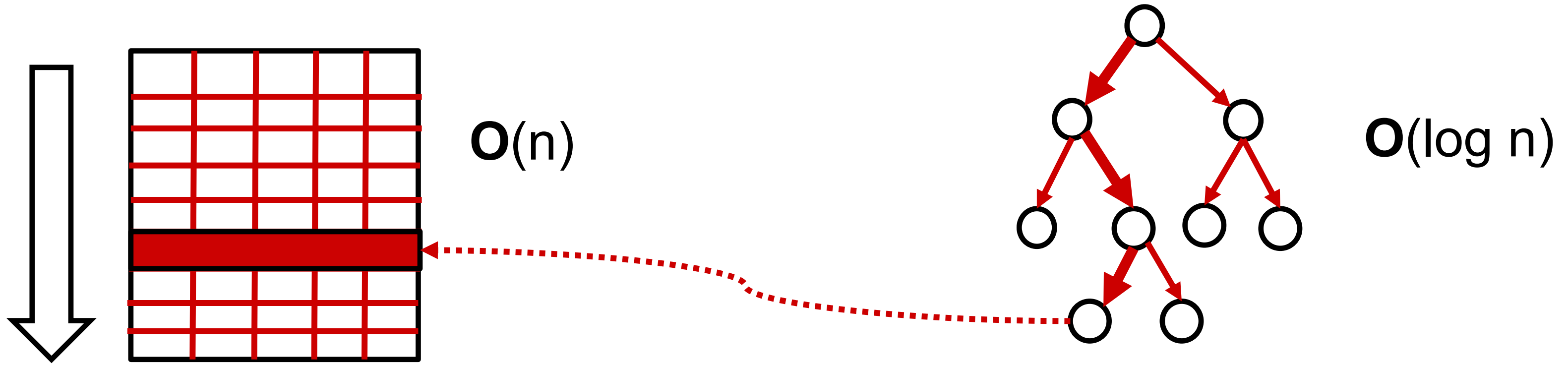
branchNo	street	city	postCode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

PRIMARY KEY: What's it good for?

The database management system builds an **index** for the table around the primary key so that, given the value of the primary key, it can look up the row containing the rest of the values very quickly.



Without a primary Key
we must scan row-by-row
looking for the right value

With Primary Key
we look up the the row location
by searching through a tree.

Why are Primary Keys usually Integers?

(Why does your bank treat you like a number?)

"No, Thursday's out. How about never—is never good for you?"

- They can be auto-incremented as new records are added
- INTs are more space efficient than VARCHARs
- It's faster to compare a lookup value to an INT
- Numbers are language neutral

Assigning a Primary Key

Make user_id the PRIMARY KEY

```
CREATE TABLE users (
    user_id INT PRIMARY KEY,
    username VARCHAR(20),
    dob DATE,
    class ENUM ('Basic', 'Premium')
);
```

or

```
CREATE TABLE users (
    user_id INT,
    username VARCHAR(20),
    dob DATE,
    class ENUM ('Basic', 'Premium'),
    PRIMARY KEY (user_id)
);
```

Auto-incrementing the primary key

Using AUTO_INCREMENT

- When a new record (row) is inserted into the table AND we don’t specify the value for the primary key, MySQL assigns the primary key value for you!   (New Value = Current Max Value + 1)

```
CREATE TABLE users (
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(20),
    dob DATE,
    class ENUM ('Basic', 'Premium')
);
```

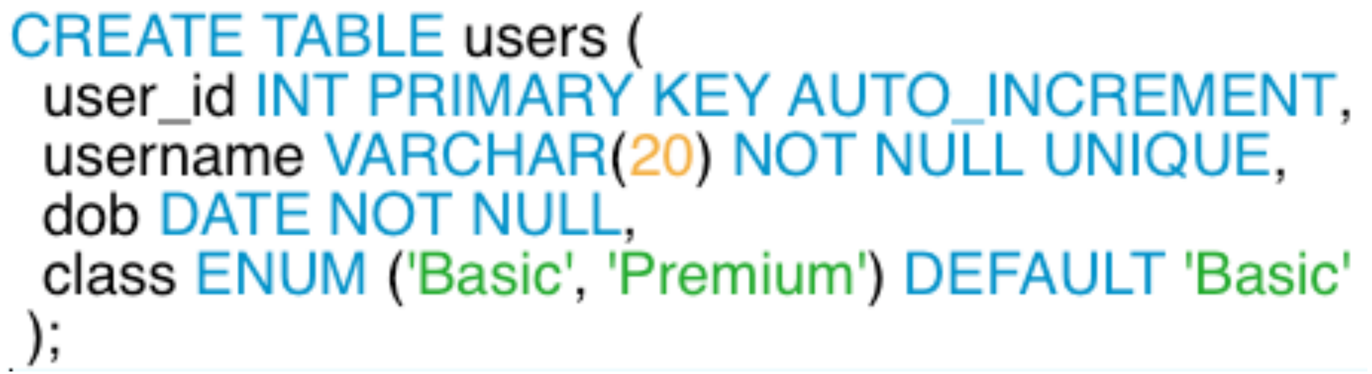
UNIQUE / NOT NULL

Everyone has to have a *unique* username.
Everyone has to specify a date of birth.   (Hey - this is a *dating* website!)

```
CREATE TABLE users (
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(20)  NOT NULL UNIQUE,
    dob DATE NOT NULL,
    class ENUM ('Basic', 'Premium')
);
```

DEFAULT

When users register, automatically assign them **Basic** status by default.



```
CREATE TABLE users (
  user_id INT PRIMARY KEY AUTO_INCREMENT,
  username VARCHAR(20) NOT NULL UNIQUE,
  dob DATE NOT NULL,
  class ENUM ('Basic', 'Premium') DEFAULT 'Basic'
 );
```