



10 – Database implementation - MySQL

ASE230 – Server-Side Programming
Nicholas Caporusso



NKU Objectives

- Create and query a database using MySQL

NKU Agenda

1. Introduction to PHPMyAdmin
2. Database creation and manipulation
3. Tools for manipulating data
4. Exporting and importing data
5. Working with users and privileges
6. Managing databases and tables

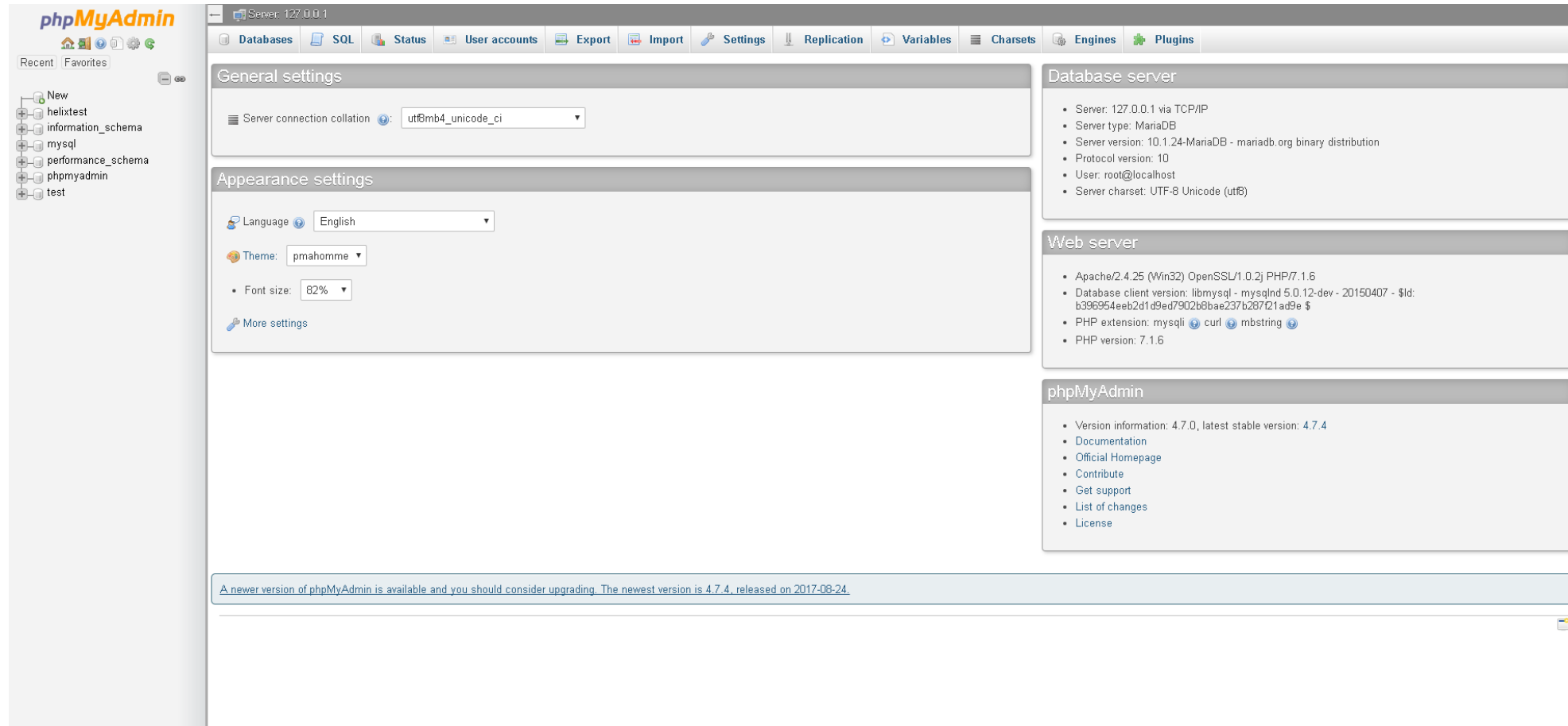


Introduction to PHPMyAdmin

NKU PhpMyAdmin

- myPhpAdmin: A popular web interface to MySQL that comes bundled with XAMPP and LAMP
- The default URL:
 - <http://localhost/phpmyadmin>
- Make sure you enabled MySQL from XAMPP panel!!
- Note: On new MySQL installations the single user account defined is “root” and the password is blank or “”.

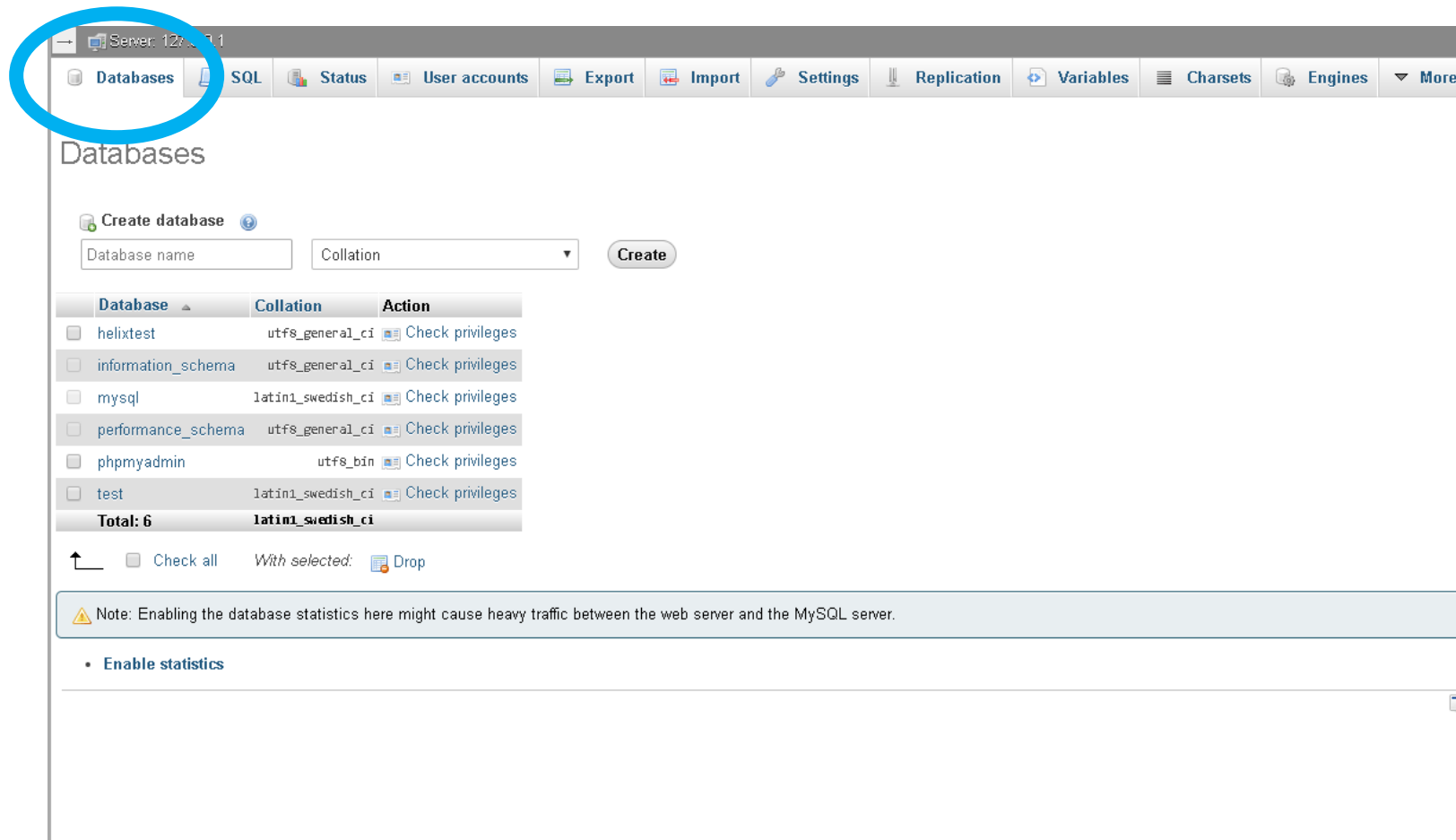
NKU The main interface





Database creation and manipulation

NKU Listing your databases



Server: 127.0.0.1

Databases | SQL | Status | User accounts | Export | Import | Settings | Replication | Variables | Charsets | Engines | More

Databases

Create database ⓘ

Database name: Collation:

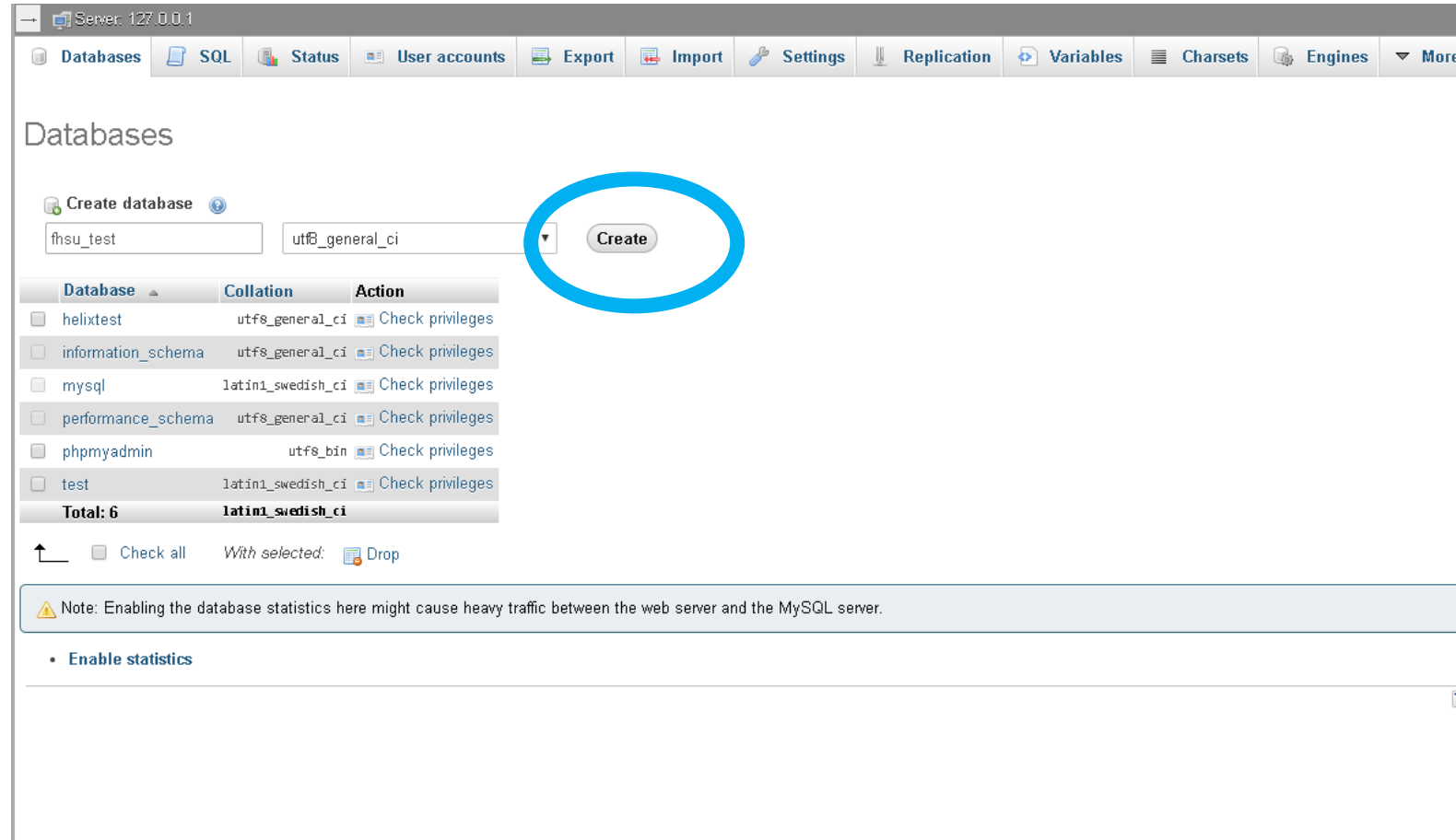
Database	Collation	Action
<input type="checkbox"/> helixtest	utf8_general_ci	Check privileges
<input type="checkbox"/> information_schema	utf8_general_ci	Check privileges
<input type="checkbox"/> mysql	latin1_swedish_ci	Check privileges
<input type="checkbox"/> performance_schema	utf8_general_ci	Check privileges
<input type="checkbox"/> phpmyadmin	utf8_bin	Check privileges
<input type="checkbox"/> test	latin1_swedish_ci	Check privileges
Total: 6	latin1_swedish_ci	

⬆ ☐ Check all With selected: [Drop](#)

⚠ Note: Enabling the database statistics here might cause heavy traffic between the web server and the MySQL server.

- [Enable statistics](#)

NKU Create a new database



NKU Listing all tables

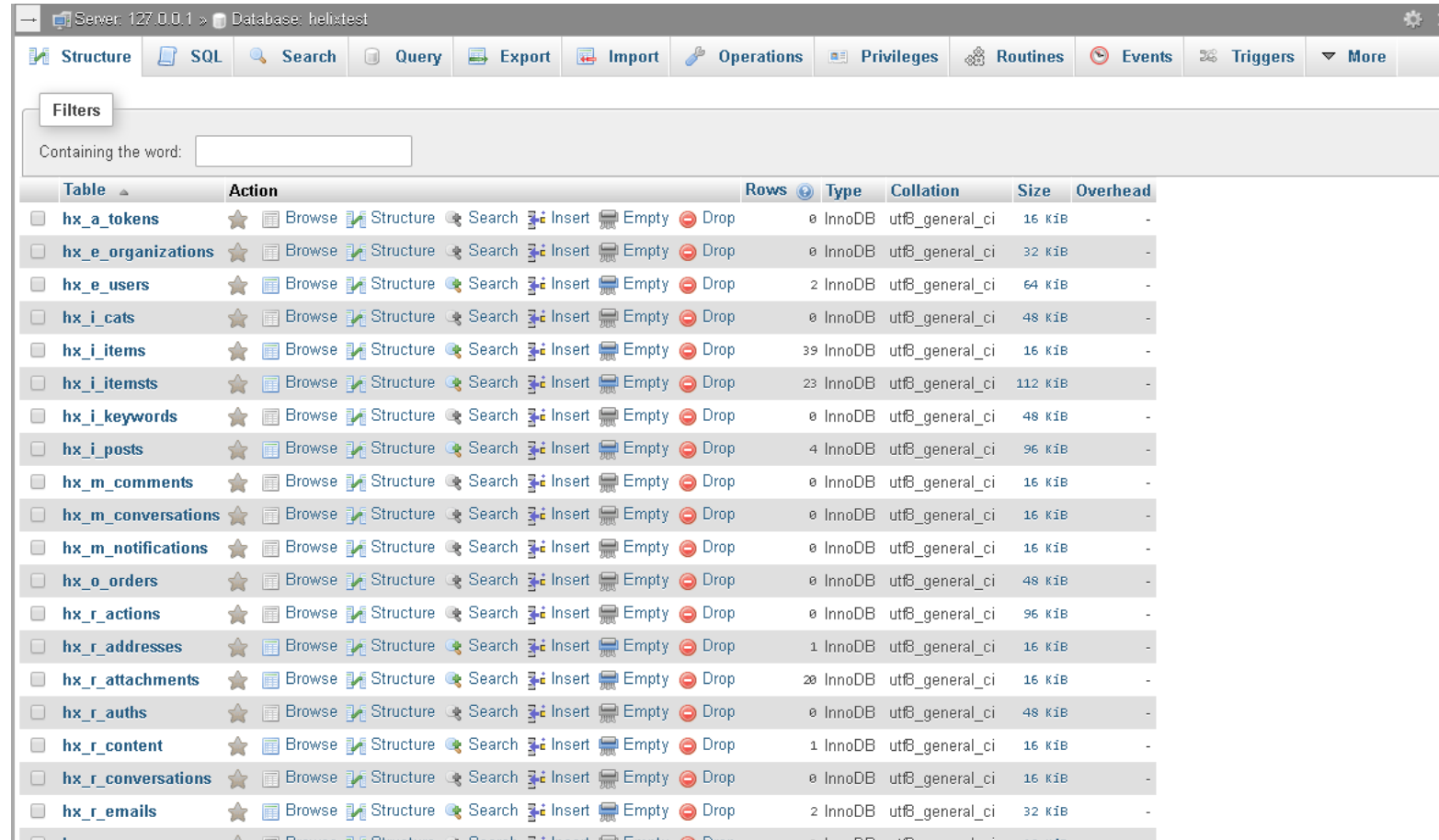


Table	Action	Rows	Type	Collation	Size	Overhead
hx_a_tokens	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8_general_ci	16 KiB	-
hx_e_organizations	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8_general_ci	32 KiB	-
hx_e_users	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8_general_ci	64 KiB	-
hx_i_cats	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8_general_ci	48 KiB	-
hx_i_items	Browse Structure Search Insert Empty Drop	39	InnoDB	utf8_general_ci	16 KiB	-
hx_i_itemsts	Browse Structure Search Insert Empty Drop	23	InnoDB	utf8_general_ci	112 KiB	-
hx_i_keywords	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8_general_ci	48 KiB	-
hx_i_posts	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8_general_ci	96 KiB	-
hx_m_comments	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8_general_ci	16 KiB	-
hx_m_conversations	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8_general_ci	16 KiB	-
hx_m_notifications	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8_general_ci	16 KiB	-
hx_o_orders	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8_general_ci	48 KiB	-
hx_r_actions	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8_general_ci	96 KiB	-
hx_r_addresses	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8_general_ci	16 KiB	-
hx_r_attachments	Browse Structure Search Insert Empty Drop	20	InnoDB	utf8_general_ci	16 KiB	-
hx_r_auths	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8_general_ci	48 KiB	-
hx_r_content	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8_general_ci	16 KiB	-
hx_r_conversations	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8_general_ci	16 KiB	-
hx_r_emails	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8_general_ci	32 KiB	-

NKU Creating a new table (1/2)

The screenshot shows a web-based MySQL management interface. At the top, a status bar indicates 'Server: 127.0.0.1' and 'Database: ihisu_test'. Below this is a horizontal menu with icons and labels for 'Structure', 'SQL', 'Search', 'Query', 'Export', 'Import', 'Operations', 'Privileges', 'Routines', 'Events', 'Triggers', and a 'More' dropdown. A light blue message box states 'No tables found in database.' Below this is a 'Create table' button. The main form area contains two input fields: 'Name: my_table' and 'Number of columns: 4'. A 'Go' button is located at the bottom right of the form.

NKU Creating a new table (2/2)

Server: 127.0.0.1 > Database: fhsu_test > Table: my_table

Browser Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

Table name: my_table Add 1 column(s) Go

Structure

Name	Type	Length/Values	Default	Collation	Attributes	Null Index	A I	Comments	Virtuality	MIME type
ID	INT		None			<input type="checkbox"/> PRIMARY	<input checked="" type="checkbox"/>			
Pick from Central Columns						PRIMARY				
name	VARCHAR	255	NULL			<input checked="" type="checkbox"/> ---	<input type="checkbox"/>			
Pick from Central Columns										
date	DATE		NULL			<input checked="" type="checkbox"/> ---	<input type="checkbox"/>			
Pick from Central Columns										
details	TEXT		NULL			<input checked="" type="checkbox"/> ---	<input type="checkbox"/>			
Pick from Central Columns										

Table comments: Collation: Storage Engine: InnoDB

PARTITION definition:

Partition by: (Expression or column list)

Partitions:

Console Preview SQL Save

NKU Table structure

Server: 127.0.0.1 > Database: fhsu_test > Table: my_table

Browser Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

Table structure Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	ID	int(11)			No	None		AUTO_INCREMENT	
2	name	varchar(255)	utf8_general_ci		Yes	NULL			
3	date	date			Yes	NULL			
4	details	text	utf8_general_ci		Yes	NULL			

☐ Check all With selected: Browse Change Drop Primary Unique Index Add to central columns Remove from central columns

Print Propose table structure Track table Move columns Improve table structure

Add 1 column(s) after details

Indexes

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
	PRIMARY	BTREE	Yes	No	ID	0	A	No	

Create an index on 1 columns

Partitions

No partitioning defined!

NKU Modifying a field

Server: 127.0.0.1 > Database: fhsu_test > Table: my_table

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

Structure

Name	Type	Length/Values	Default	Collation	Attributes	Adjust Null	privileges	A.I	Comments	Virtuality	Move column	MIME type
ID	INT	11	None			<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>				

[Pick from Central Columns](#)

Preview SQL Save

Console



Tools for manipulating data

NKU The SQL window

The screenshot shows a web-based SQL interface. At the top, a status bar indicates the connection: "Server: 127.0.0.1 > Database: fhsu_test > Table: my_table". Below this is a menu bar with icons and labels for "Browse", "Structure", "SQL", "Search", "Insert", "Export", "Import", "Privileges", "Operations", "Tracking", and "Triggers". The main area is titled "Run SQL query/queries on table fhsu_test.my_table:". It contains a text input field with the SQL query: `1 SELECT * FROM `my_table` WHERE 1`. Below the query field is a row of buttons: "SELECT *", "SELECT", "INSERT", "UPDATE", "DELETE", "Clear", "Format", and "Get auto-saved query". There is also a checkbox for "Bind parameters" with a help icon. Below these buttons is a "Bookmark this SQL query:" label followed by a text input field. At the bottom, there is a row of options: "[Delimiter :]", a checked checkbox for "Show this query here again", an unchecked checkbox for "Retain query box", an unchecked checkbox for "Rollback when finished", and a checked checkbox for "Enable foreign key checks". A "Go" button is located at the bottom right. On the right side of the main area, there is a "Columns" section with a list box containing "ID", "name", "date", and "details". Below the list box is a "<<" button.

NKU The search window

Server: 127.0.0.1 > Database: fhsu_test > Table: my_table

Browser Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

Table search Zoom search Find and replace

Do a "query by example" (wildcard: "%")

Column	Type	Collation	Operator	Value
ID	int(11)		=	
name	varchar(255)	utf8_general_ci	LIKE	
date	date		=	
details	text	utf8_general_ci	LIKE	

+ Options

Go

NKU Inserting a new record

Server: 127.0.0.1 > Database: fhsu_test > Table: my_table

Browser Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

Column	Type	Function	Null	Value
ID	int(11)			
name	varchar(255)		<input checked="" type="checkbox"/>	
date	date		<input checked="" type="checkbox"/>	
details	text		<input checked="" type="checkbox"/>	

Go

☒ Ignore

Column	Type	Function	Null	Value
ID	int(11)			
name	varchar(255)		<input checked="" type="checkbox"/>	
date	date		<input checked="" type="checkbox"/>	



Exporting and importing data

NKU Exporting data

Server: 127.0.0.1 > Database: fhsu_test > Table: my_table

Browser Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

Exporting rows from "my_table" table

Export templates:

New template: Template name:

Existing templates: Template: -- Select a template --

Export method:

☒ Quick - display only the minimal options

☐ Custom - display all possible options

Format:

SQL



Options for exporting data (1/3)

Server: 127.0.0.1 > Database: fhsu_test > Table: my_table

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

SQL

Rows:

☐ Dump some row(s)

Number of rows:

Row to begin at:

☒ Dump all rows

Output:

☐ Rename exported databases/tables/columns

☐ Use LOCK TABLES statement

☒ Save output to a file

File name template: ☒ use this for future exports

Character set of the file:

Compression:

☐ View output as text

Skip tables larger than MiB



Options for exporting data (2/3)

Server: 127.0.0.1 > Database: fhsu_test > Table: my_table

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

Skip tables larger than MiB

Format-specific options:

☒ Display comments (includes info such as export timestamp, PHP version, and server version)

Additional custom header comment (\n splits lines):

☐ Include a timestamp of when databases were created, last updated, and last checked

☐ Display foreign key relationships

☐ Display MIME types

☒ Enclose export in a transaction

☐ Disable foreign key checks

☐ Export views as tables

☐ Export metadata

Database system or older MySQL server to maximize output compatibility with:

☐ structure

☐ data

☒ structure and data

Object creation options

Add statements:



Options for exporting data (3/3)

Server: 127.0.0.1 > Database: test > Table: my_table

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

Object creation options

Add statements:

- ☐ Add DROP TABLE / TRIGGER statement
- ☒ Add CREATE TABLE statement
 - ☐ IF NOT EXISTS (less efficient as indexes will be generated during table creation)
 - ☒ AUTO_INCREMENT value
- ☒ Add CREATE VIEW statement
- ☐ Add CREATE PROCEDURE / FUNCTION / EVENT statement
- ☒ Add CREATE TRIGGER statement
- ☒ Enclose table and column names with backquotes (Protects column and table names formed with special characters or keywords)

Data creation options

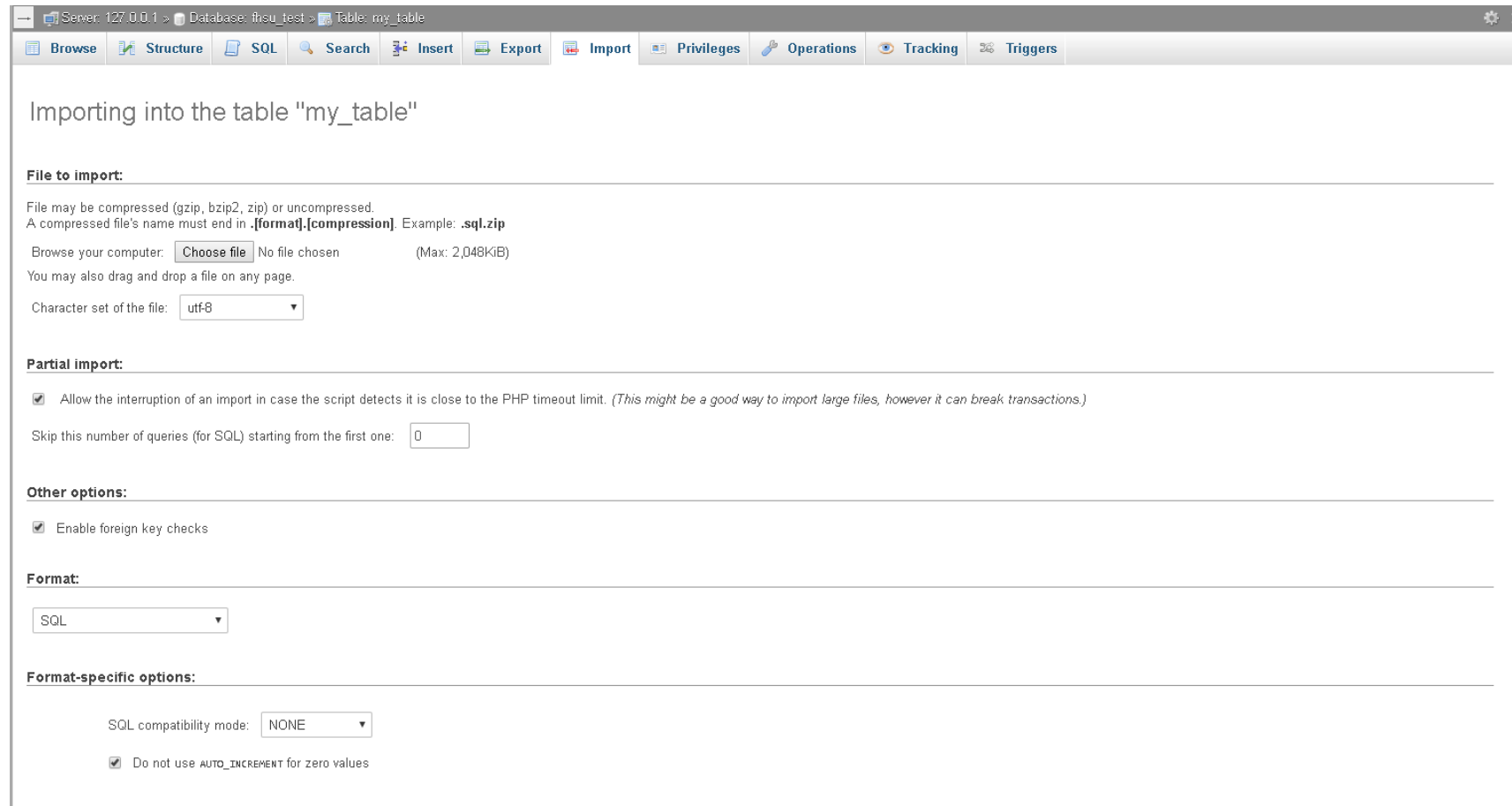
- ☐ Truncate table before insert
 - ☐ INSERT DELAYED statements
 - ☐ INSERT IGNORE statements

Function to use when dumping data:

Syntax to use when inserting data:

- ☐ include column names in every INSERT statement
Example: INSERT INTO tbl_name (col_A,col_B,col_C) VALUES (1,2,3)
- ☐ insert multiple rows in every INSERT statement
Example: INSERT INTO tbl_name VALUES (1,2,3), (4,5,6), (7,8,9)
- ☒ both of the above
Example: INSERT INTO tbl_name (col_A,col_B,col_C) VALUES (1,2,3), (4,5,6), (7,8,9)
- ☐ neither of the above
Example: INSERT INTO tbl_name VALUES (1,2,3)

NKU Importing data



Server: 127.0.0.1 > Database: fhsu_test > Table: my_table

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

Importing into the table "my_table"

File to import:

File may be compressed (gzip, bzip2, zip) or uncompressed.
A compressed file's name must end in **.[format].[compression]**. Example: **.sql.zip**

Browse your computer: No file chosen (Max: 2,048KiB)

You may also drag and drop a file on any page.

Character set of the file:

Partial import:

☒ Allow the interruption of an import in case the script detects it is close to the PHP timeout limit. *(This might be a good way to import large files, however it can break transactions.)*

Skip this number of queries (for SQL) starting from the first one:

Other options:

☒ Enable foreign key checks

Format:

Format-specific options:

SQL compatibility mode:

☒ Do not use AUTO_INCREMENT for zero values



Working with users and privileges

NKU Listing users

The screenshot shows the MySQL User accounts overview page. At the top, there's a navigation bar with tabs: Databases, SQL, Status, User accounts (selected), Export, Import, Settings, Replication, Variables, Charsets, Engines, and Plugins. Below the navigation bar, there are two sub-tabs: User accounts overview (selected) and User groups. The main heading is "User accounts overview". A warning message states: "A user account allowing any user from localhost to connect is present. This will prevent other users from connecting if the host part of their account allows a connection from any (%) host." Below this, there's a table listing user accounts.

	User name	Host name	Password	Global privileges	User group	Grant	Action
<input type="checkbox"/>	Any	%	No	USAGE	No	No	Edit privileges Export
<input type="checkbox"/>	Any	localhost	No	USAGE	No	No	Edit privileges Export
<input type="checkbox"/>	pma	localhost	No	USAGE	No	No	Edit privileges Export
<input type="checkbox"/>	root	127.0.0.1	No	ALL PRIVILEGES	Yes	Yes	Edit privileges Export
<input type="checkbox"/>	root	::1	No	ALL PRIVILEGES	Yes	Yes	Edit privileges Export
<input type="checkbox"/>	root	localhost	No	ALL PRIVILEGES	Yes	Yes	Edit privileges Export

Below the table, there are controls: a checkbox for "Check all" and a button for "With selected: Export". There are two main sections: "New" with an "Add user account" button, and "Remove selected user accounts" with a warning: "(Revoke all active privileges from the users and delete them afterwards.)" and a checkbox for "Drop the databases that have the same names as the users." A "Go" button is at the bottom right.

NKU Adding a new user (1/2)


Server: 127.0.0.1

Databases SQL Status User accounts Export Import Settings Replication Variables Charsets Engines Plugins

Add user account

Login Information

User name: Use text field:

Host name: Any host % 

Password: Use text field: Strength:

Re-type:

Authentication Plugin: Native MySQL authentication

Generate password:

Database for user account

☐ Create database with same name and grant all privileges.

☐ Grant all privileges on wildcard name (username_%).

NKU Adding a new user (2/2)

The screenshot shows the 'User accounts' configuration window for a MySQL server (127.0.0.1). The 'Global privileges' section is active, with a 'Check all' button. A note states: 'Note: MySQL privilege names are expressed in English.' Below this, there are four panels: 'Data', 'Structure', 'Administration', and 'Resource limits'. Each panel contains a list of privileges with checkboxes. The 'Data' panel includes SELECT, INSERT, UPDATE, DELETE, and FILE. The 'Structure' panel includes CREATE, ALTER, INDEX, DROP, CREATE TEMPORARY TABLES, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, EXECUTE, CREATE VIEW, EVENT, and TRIGGER. The 'Administration' panel includes GRANT, SUPER, PROCESS, RELOAD, SHUTDOWN, SHOW DATABASES, LOCK TABLES, REFERENCES, REPLICATION CLIENT, REPLICATION SLAVE, and CREATE USER. The 'Resource limits' panel includes a note: 'Note: Setting these options to 0 (zero) removes the limit.' and four input fields: MAX QUERIES PER HOUR, MAX UPDATES PER HOUR, MAX CONNECTIONS PER HOUR, and MAX USER_CONNECTIONS, all set to 0.

Server: 127.0.0.1

Databases SQL Status User accounts Export Import Settings Replication Variables Charsets Engines Plugins

Global privileges ☐ Check all

Note: MySQL privilege names are expressed in English.

☐ Data

- ☐ SELECT
- ☐ INSERT
- ☐ UPDATE
- ☐ DELETE
- ☐ FILE

☐ Structure

- ☐ CREATE
- ☐ ALTER
- ☐ INDEX
- ☐ DROP
- ☐ CREATE TEMPORARY TABLES
- ☐ SHOW VIEW
- ☐ CREATE ROUTINE
- ☐ ALTER ROUTINE
- ☐ EXECUTE
- ☐ CREATE VIEW
- ☐ EVENT
- ☐ TRIGGER

☐ Administration

- ☐ GRANT
- ☐ SUPER
- ☐ PROCESS
- ☐ RELOAD
- ☐ SHUTDOWN
- ☐ SHOW DATABASES
- ☐ LOCK TABLES
- ☐ REFERENCES
- ☐ REPLICATION CLIENT
- ☐ REPLICATION SLAVE
- ☐ CREATE USER

☐ Resource limits

Note: Setting these options to 0 (zero) removes the limit.

MAX QUERIES PER HOUR

MAX UPDATES PER HOUR

MAX CONNECTIONS PER HOUR

MAX USER_CONNECTIONS

NKU Assigning a user to a database (1/2)

Server: 127.0.0.1

Databases SQL Status User accounts Export Import Settings Replication Variables Charsets Engines Plugins

Global Database Change password Login Information

Edit privileges: User account 'test'@'%'

Database-specific privileges

Database	Privileges	Grant	Table-specific privileges	Action
None				

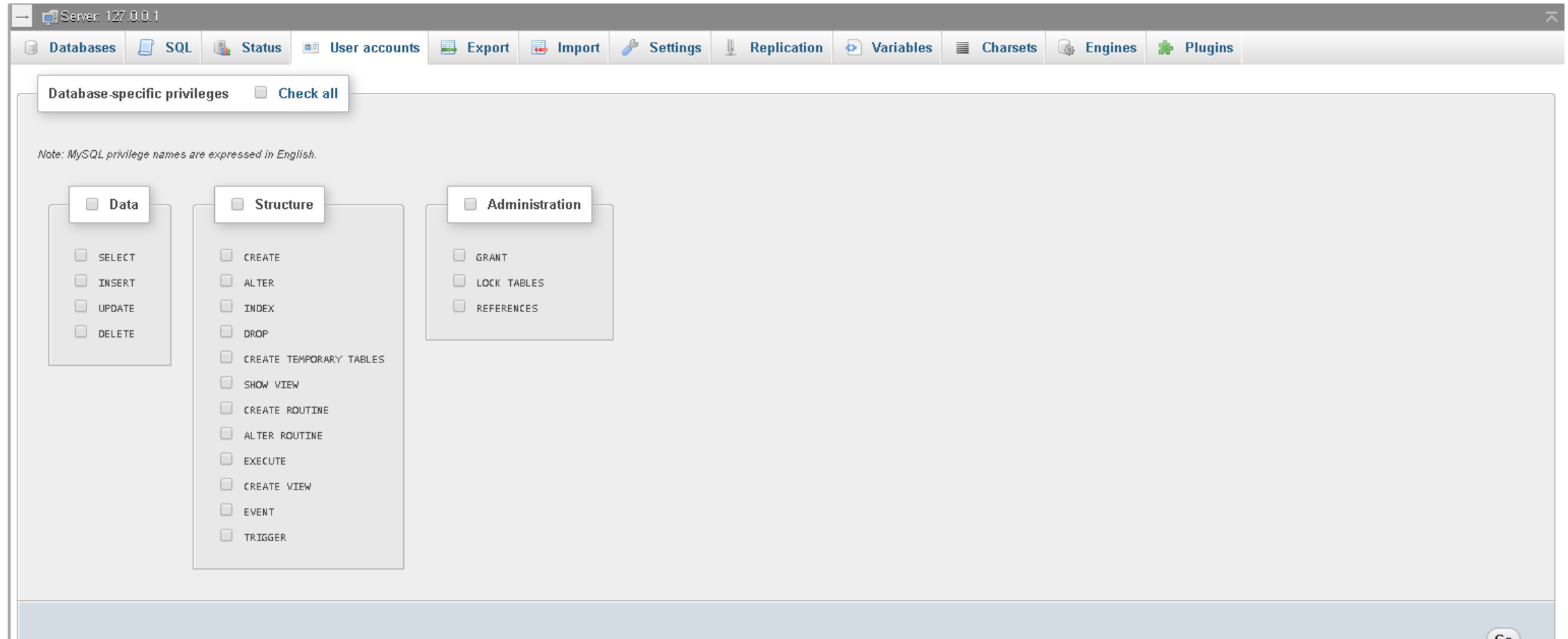
Database-specific privileges

Database: fhsu_test
helixtest
mysql
phpmyadmin

Add privileges on the following database(s):

Go

NKU Assigning a user to a database (2/2)





Managing databases and tables



Executing operations on a database

Server: 127.0.0.1 » Database: fhsu_test

Structure SQL Search Query Export Import Operations Privileges Routines Events Triggers Tracking Designer Central columns

Database comment

Go

Create table

Name:

Number of columns:

Go

Rename database to

☒ Adjust privileges ⓘ

Go

Remove database

Drop the database (DROP) ⓘ

Copy database to

☐ Structure only
☒ Structure and data
☐ Data only

☒ CREATE DATABASE before copying
☐ Add DROP TABLE / DROP VIEW
☒ Add AUTO_INCREMENT value
☒ Add constraints

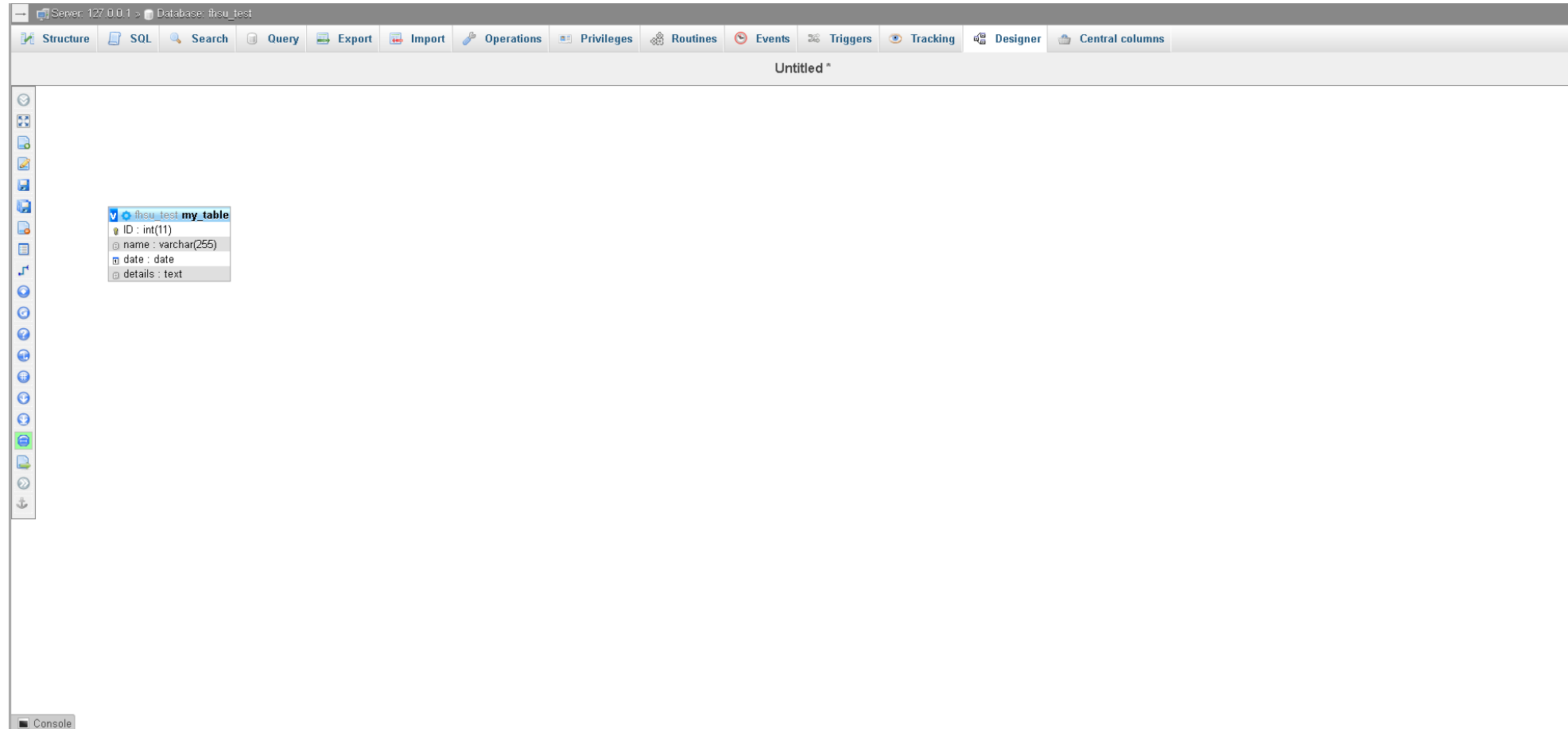
☒ Adjust privileges ⓘ
☐ Switch to copied database

Go

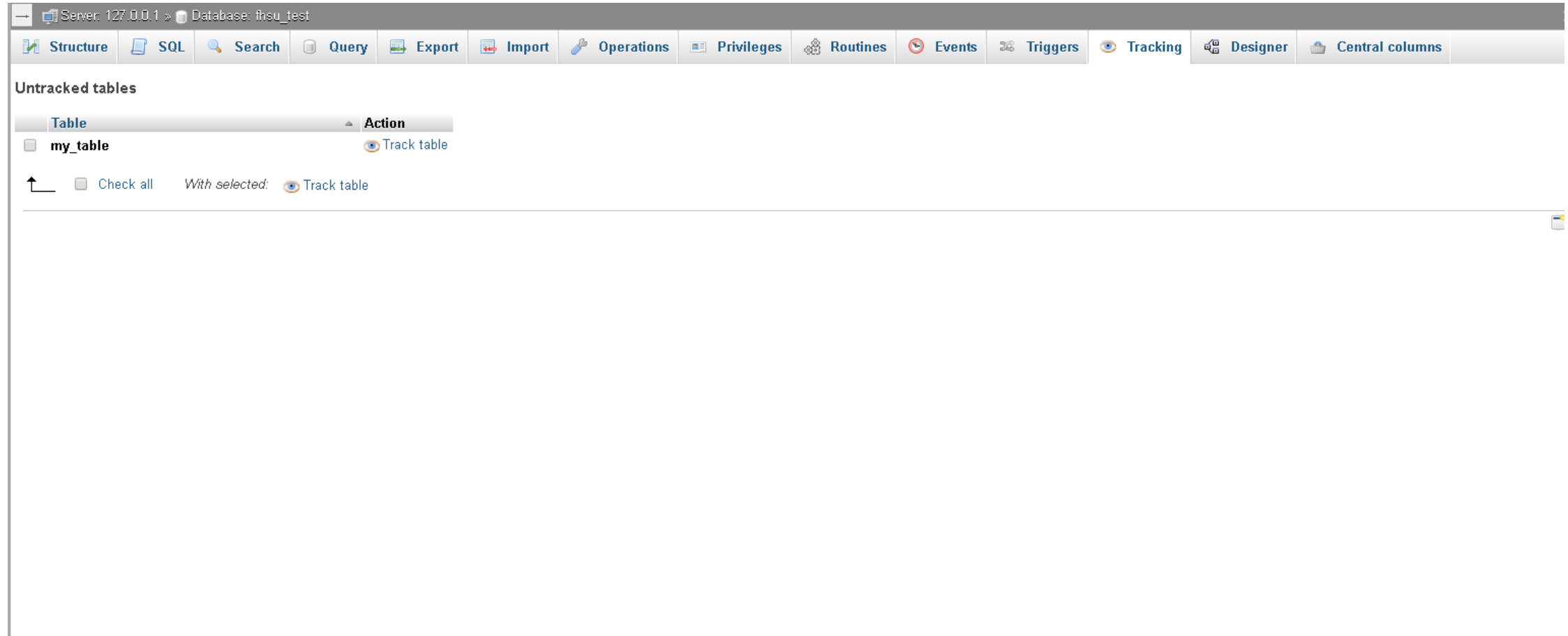
Collation

utf8_general_ci ▼

NKU Visual editor



NKU Tracking table versions (1/2)



NKU Tracking table versions (2/2)

Server: 127.0.0.1 » Database: fhsu_test » Table: my_table

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

Create version 1 of fhsu_test.my_table

Track these data definition statements:

- ☒ ALTER TABLE
- ☒ RENAME TABLE
- ☒ CREATE TABLE
- ☒ DROP TABLE
- ☒ CREATE INDEX
- ☒ DROP INDEX

Track these data manipulation statements:

- ☒ INSERT
- ☒ UPDATE
- ☒ DELETE
- ☒ TRUNCATE

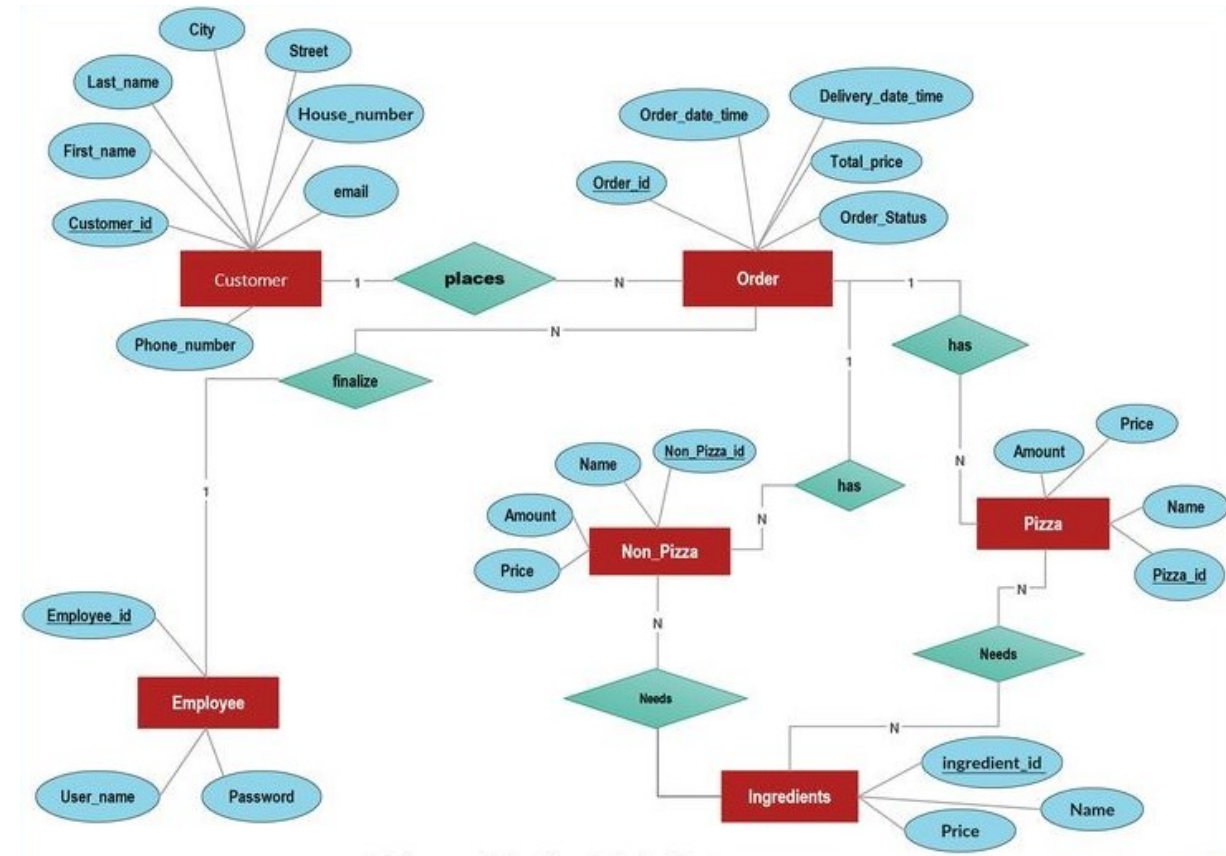
Create version



Wrap-up

NKU In-class exercise

- Implement the structure of the ER shown in the picture



NKU Agenda

1. Database and table creation
2. Modifying table structure
3. CRUD operations



Database and table creation

NKU The Structured Query Language

- Rise to dominance due in part to its powerful and flexible query language
- Structured Query Language (SQL) allows the user to specify what must be done without specifying how it must be done
- SQL-based relational database application involves:
 - User interface
 - A set of tables stored in the database
 - SQL engine

NKU SQL and MySQL

- MySQL uses Structured Query Language (SQL)
- SQL is language for retrieving, updating, deleting, information from a database
- Relational databases use a model that define data according to relationships
- Other databases: Oracle, Informix, DB2 (IBM) Access (Microsoft), SQL Server, PostgreSQL

NKU Why a database?

- powerful: can search it, filter data, combine data from multiple sources
- fast: can search/filter a database very quickly compared to a file
- big: scale well up to very large data sizes
- safe: built-in mechanisms for failure recovery (e.g. transactions)
- multi-user: concurrency features let many users view/edit data at same time
- abstract: provides layer of abstraction between stored data and app(s)
 - many database programs understand the same SQL commands

NKU Example tables (1/4): The Simpsons' school

id	name	email
123	Bart	bart@fox.com
456	Milhouse	milhouse@fox.com
888	Lisa	lisa@fox.com
404	Ralph	ralph@fox.com

students

id	name
1234	Krabappel
5678	Hoover
9012	Stepp

teachers

id	name	teacher_id
10001	Computer Science 142	1234
10002	Computer Science 143	5678
10003	Computer Science 190M	9012
10004	Informatics 100	1234

courses

student_id	course_id	grade
123	10001	B-
123	10002	C
456	10001	B+
888	10002	A+
888	10003	A+
404	10004	D+

grades

NKU Example tables (2/4): The world

code	name	continent	independence_year	population	gnp	head_of_state	...
AFG	Afghanistan	Asia	1919	22720000	5976.0	Mohammad Omar	...
NLD	Netherlands	Europe	1581	15864000	371362.0	Beatrix	...
...

countries (Other columns: region, surface_area, life_expectancy, gnp_old, local_name, government_form, capital, code2)

id	name	country_code	district	population
3793	New York	USA	New York	8008278
1	Los Angeles	USA	California	3694820
...

cities

country_code	language	official	percentage
AFG	Pashto	T	52.4
NLD	Dutch	T	95.6
...

languages

NKU Example tables (3/4): IMDB

id	first_name	last_name	gender
433259	William	Shatner	M
797926	Britney	Spears	F
831289	Sigourney	Weaver	F
...			

actors

id	name	year	rank
112290	Fight Club	1999	8.5
209658	Meet the Parents	2000	7
210511	Memento	2000	8.7
...			

movies

actor_id	movie_id	role
433259	313398	Capt. James T. Kirk
433259	407323	Sgt. T.J. Hooker
797926	342189	Herself
...		

roles

movie_id	genre
209658	Comedy
313398	Action
313398	Sci-Fi
...	

movies_genres

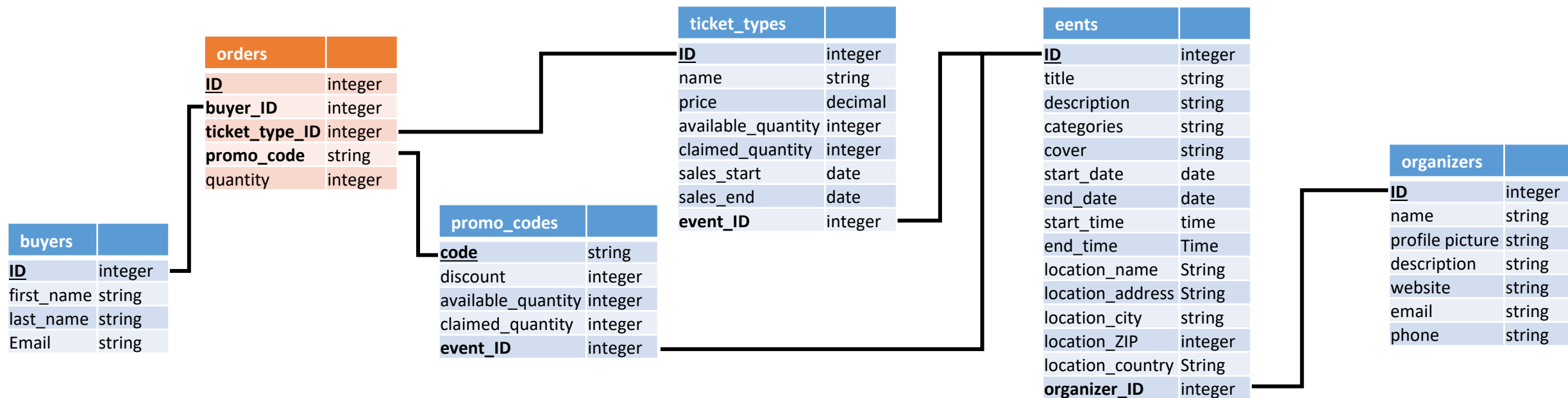
id	first_name	last_name
24758	David	Fincher
66965	Jay	Roach
72723	William	Shatner
...		

directors

director_id	movie_id
24758	112290
66965	209658
72723	313398
...	

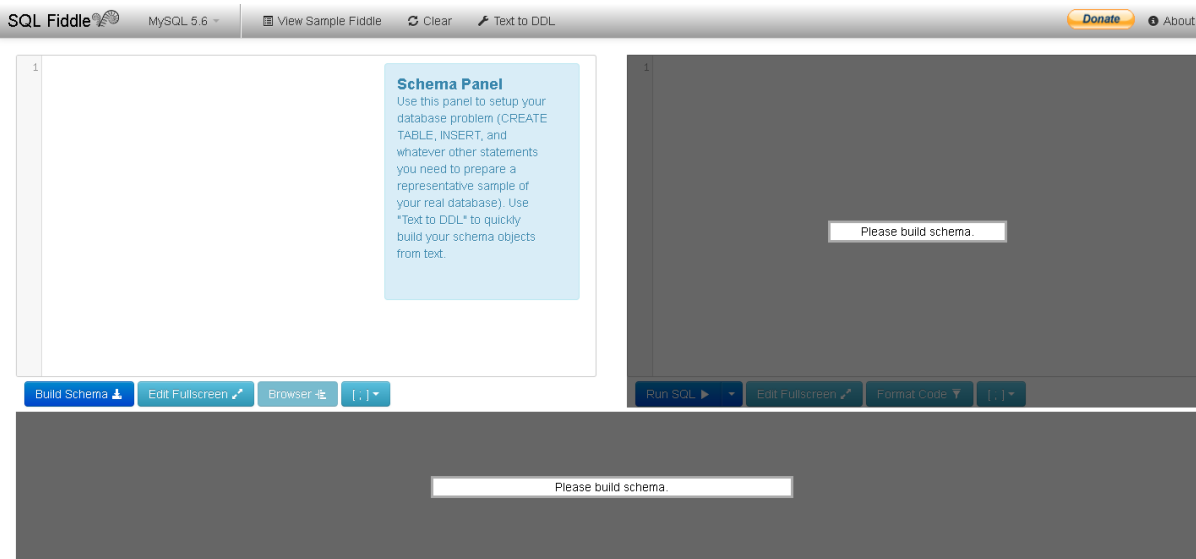
movies_directors

Example tables (4/4): Eventbrite

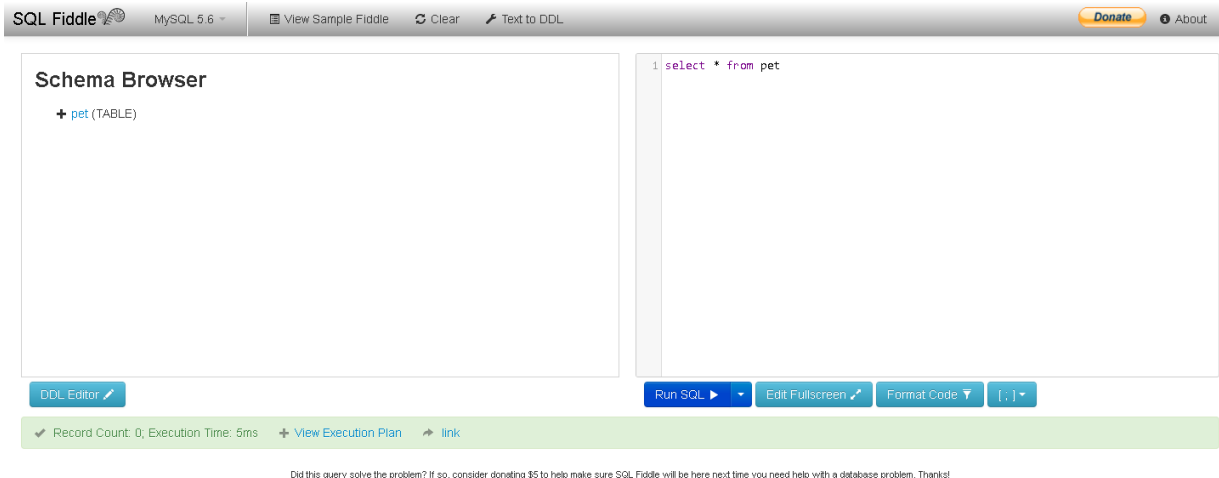


NKU Testing with SQLfiddle

Schema editor (left side)



SQL Browser (right side)





- MySQL permits syntax for three different comments.

```
/*  
 * Multi-line comment  
 * and /* nested comments  
 *         are legal */  
*/  
  
# Shell-like comment  
  
-- Standard SQL comment
```


NKU Creating a database



- The command `CREATE DATABASE` can be utilized to create a new database
- The command `USE` can be utilized to choose the database we want to use, after we created it

Syntax:

```
CREATE DATABASE dbName;
```

Example:

```
CREATE DATABASE carddb;  
USE carddb;
```

NKU Creating a table



- When we create a table, we want to define the columns of the table, so that we can use them later
- Columns are in parentheses, separated by commas

```
CREATE TABLE tblName (  
    colName1 dataType1  
    [colAttr1 ...]  
    [, colName2 dataType2]  
    [, colAttr2 ...]  
    [, tblAttr1, ...] );
```



- When we create a column, we specify
 - name of the column
 - data type
 - if it supports null values
 - if it supports automatic operations (e.g., auto increment)
- When we create a table, we also specify its keys, if any

```
CREATE TABLE notecard (  
    # id INT NOT NULL AUTO_INCREMENT,  
    id /* column name */  
    INT /* column data type */  
    NOT NULL /* data can't be null */  
    AUTO_INCREMENT, /* next integer */  
    name VARCHAR(50), /* 50 chr max */  
    content TEXT, /* unlimited char */  
    creation TIMESTAMP DEFAULT NOW(),  
    category_id INT, /* foreign key */  
    PRIMARY KEY(id) ); /* index */
```



- The table posts will contain blog posts
 - Id will be the unique identifier
 - Content will be the text of the post
 - Date will be the creation date
 - Topic is a foreign key which references a category id
 - Post_by is a foreign key which references an author

```
CREATE TABLE posts (  
    post_id INT(8) NOT NULL AUTO_INCREMENT,  
    post_content TEXT NOT NULL,  
    post_date DATETIME NOT NULL,  
    post_topic INT(8) NOT NULL,  
    post_by INT(8) NOT NULL,  
    PRIMARY KEY (post_id)  
) TYPE=INNODB;
```

NKU Viewing the table structure



- The describe command displays the column name, column type, and other attributes regarding a table

```
mysql> describe notecard;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
name	varchar(5)	YES		NULL	
content	text	YES		NULL	
creation	timestamp	NO		CURRENT_TIMESTAMP	
category_id	int(11)	YES		NULL	

```
5 rows in set (0.16 sec)  
mysql>
```

Syntax:

```
DESCRIBE tableName;
```



Modifying table structure

NKU Renaming a table



- We can modify the structure of the table, e.g., to rename it, using the command ALTER

Syntax:

```
ALTER TABLE oldTable  
    RENAME newTable;
```

Example:

```
ALTER TABLE notecard  
    RENAME recipe;
```

NKU Renaming a column



- The command ALTER also supports changing the data type of a field

Syntax:

```
ALTER TABLE tableName  
    CHANGE oldColNam  
        newColNam newColType;
```

Example:

```
ALTER TABLE author  
    CHANGE penname  
        pseudonym varchar(25);
```


NKU Modifying a column data type



- Warning: Changing the data type of a column in a table that contains values could cause a loss of data!

Syntax:

```
ALTER TABLE tableName  
    MODIFY colName colType;
```

Example:

```
ALTER TABLE book  
    MODIFY author  
        varchar(25);
```

NKU Adding a column



- We can add columns using the command ALTER and the attributes of the column

Syntax:

```
ALTER TABLE tblName  
    ADD colName1 colType1  
    FIRST|AFTER colName2;
```

Example:

```
ALTER TABLE book  
    ADD pseudonym  
        varchar(25)  
        AFTER id;
```



- The command ALTER is also used for changing the order of columns

Syntax:

```
ALTER TABLE tblName  
    MODIFY colNam1 colType  
    FIRST|AFTER colNam2;
```

Example:

```
ALTER TABLE book  
    MODIFY pseudonym  
        varchar(25)  
        AFTER author;
```

NKU Removing a column



- Pitfall: Dropping a column also removes the data stored in the column!
- Note: There is no “undo” on dropped columns.

Syntax:

```
ALTER TABLE tableName  
    DROP columnName;
```

Example:

```
ALTER TABLE book  
    DROP pseudonym;
```

NKU Removing a table



- Pitfall: Dropping a table also removes the data stored in the table!
- Note: There is no “undo” on dropped tables.

Syntax:

```
DROP TABLE tableName;
```

Example:

```
DROP TABLE book;
```



CRUD operations

NKU CRUD operations

- The acronym CRUD represents the most common SQL operations performed on a database.
- The following are the most often used SQL data commands:
 - SELECT: Querying data in a database
 - INSERT: Adding data to tables
 - UPDATE: Modifying existing table data
 - DELETE: Removing rows from tables

Letter	Operation	MySQL Statement
C	Create	INSERT
R	Retrieve	SELECT
U	Update	UPDATE
D	Destroy	DELETE

NKU Querying table data: SELECT



- SELECT: Used for extracting data contained in a table.
- The asterisk (*) can be used as a wildcard to specify all the columns in a query.

```
C:\WINDOWS\system32\cmd.exe - mysql -h localhost -u root -p
mysql> select * from notecard;
+----+-----+-----+-----+-----+
| id | name  | content                | creation           | category_id |
+----+-----+-----+-----+-----+
| 1  | Jello | Mix packet with water. | 2009-03-21 08:25:11 | 1           |
+----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> _
```

Syntax:

```
SELECT column1 [, ...]
    FROM table1 [, ...]
    [WHERE clause]
    # no [SANTA clause]
    [ ORDER BY clause [ASC|DESC] ]
    [LIMIT n];
```

Example:

```
SELECT * FROM recipe LIMIT 1;
```


NKU SELECT explained



- the SELECT statement searches a database and returns a set of results
 - the column name(s) written after SELECT filter which parts of the rows are returned
 - table and column names are case-sensitive
 - SELECT * FROM table; keeps all columns

```
SELECT name, code FROM countries;
```

name	code
China	CHN
United States	IND
Indonesia	USA
Brazil	BRA
Pakistan	PAK
...	...

NKU SELECT: examples

```
/* display all columns and all rows */  
SELECT * FROM recipe;
```

```
/* column names specified */  
SELECT name, content  
    FROM recipe;
```

```
/* specify matching row(s) */  
SELECT * FROM recipe  
    WHERE name = "Jello";
```

```
/* sort according to name */  
SELECT * FROM recipe ORDER BY name;
```

```
/* obtain most recent recipe */  
SELECT id FROM recipe ORDER BY creation  
    DESC LIMIT 1;
```

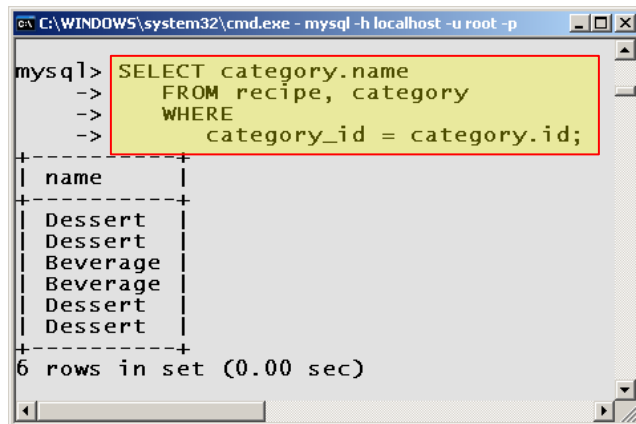
NKU Selecting unique values only



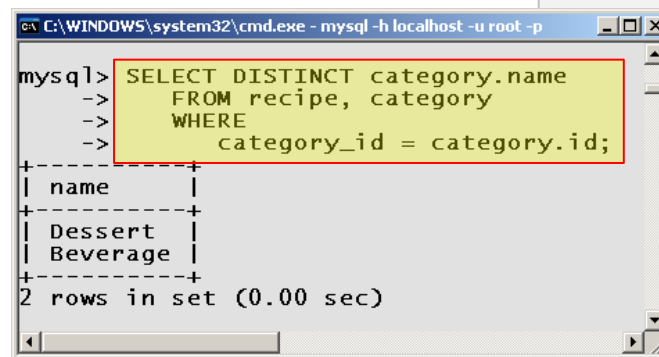
- A select statement may result in duplicate row values.
- Unique rows can be obtained by using the DISTINCT key word.

Syntax:

```
SELECT DISTINCT col1 [, ...]  
FROM tbl1 [, ...]  
[WHERE condition]  
[ORDER BY col1 [, ...]  
[LIMIT n];
```



```
C:\WINDOWS\system32\cmd.exe - mysql -h localhost -u root -p  
mysql> SELECT category.name  
-> FROM recipe, category  
-> WHERE  
-> category_id = category.id;  
+-----+  
| name |  
+-----+  
| Dessert |  
| Dessert |  
| Beverage |  
| Beverage |  
| Dessert |  
| Dessert |  
+-----+  
6 rows in set (0.00 sec)
```



```
C:\WINDOWS\system32\cmd.exe - mysql -h localhost -u root -p  
mysql> SELECT DISTINCT category.name  
-> FROM recipe, category  
-> WHERE  
-> category_id = category.id;  
+-----+  
| name |  
+-----+  
| Dessert |  
| Beverage |  
+-----+  
2 rows in set (0.00 sec)
```

SELECT DISTINCT explained

SELECT language FROM languages;

language
Dutch
English
English
Papiamentu
Spanish
Spanish
Spanish
...

SELECT DISTINCT language FROM languages;

language
Dutch
English
Papiamentu
Spanish
...

NKU Searching with the % wildcard



- Searching records in a database often involves matching part of string in a field
- wildcard % matches zero or more characters and is used with keyword “LIKE”.

The screenshot shows a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe - mysql -h localhost -u ...". It displays two MySQL queries and their results. The first query is "SELECT id, name, content FROM recipe;" which returns three rows: Jello, Pudding, and Koolaid, all with "Mix ingredients" as content. The second query is "SELECT id, name, content FROM recipe WHERE name LIKE '%o%';" which returns two rows: Jello and Koolaid, both with "Mix ingredients" as content. The results are displayed in a table format with columns for id, name, and content.

```
mysql> SELECT id, name, content
-> FROM recipe;
+----+-----+-----+
| id | name  | content |
+----+-----+-----+
| 1  | Jello | Mix ingredients |
| 2  | Pudding | Mix ingredients |
| 3  | Koolaid | Mix ingredients |
+----+-----+-----+
3 rows in set (0.00 sec)

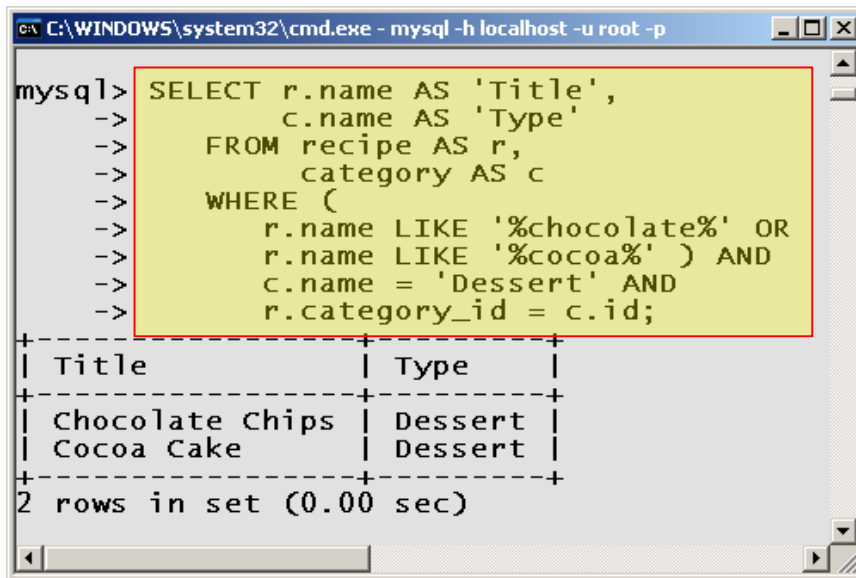
mysql> SELECT id, name, content
-> FROM recipe
-> WHERE name LIKE '%o%';
+----+-----+-----+
| id | name  | content |
+----+-----+-----+
| 1  | Jello | Mix ingredients |
| 3  | Koolaid | Mix ingredients |
+----+-----+-----+
2 rows in set (0.02 sec)
```

Example:

```
SELECT *
FROM recipe WHERE
(content LIKE "%rubarb%" AND
content LIKE "%strawberr%");
```



- The keywords AND, OR and NOT can be used in the WHERE clause to combine multiple statements



```
mysql> SELECT r.name AS 'Title',
->      c.name AS 'Type'
->      FROM recipe AS r,
->      category AS c
->      WHERE (
->          r.name LIKE '%chocolate%' OR
->          r.name LIKE '%cocoa%' ) AND
->          c.name = 'Dessert' AND
->          r.category_id = c.id;
```

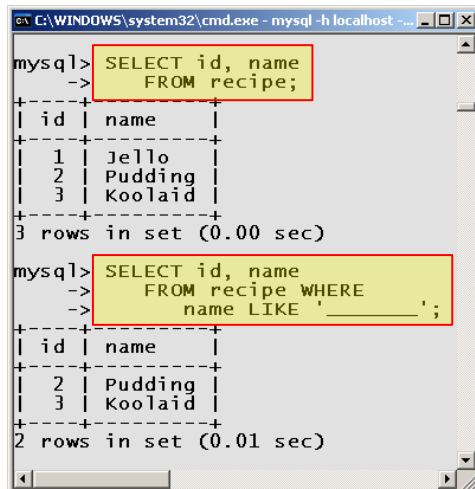
Title	Type
Chocolate Chips	Dessert
Cocoa Cake	Dessert

2 rows in set (0.00 sec)

Examples:

```
SELECT r.name AS 'Title',
       c.name AS 'Type'
FROM recipe AS r,
       category AS c
WHERE (
    r.name LIKE '%chocolate%' OR
    r.name LIKE '%cocoa%' ) AND
    c.name = "Dessert" AND
    r.category_id = c.id;
```

- Searching records in a database often involves matching part of string in a field
- Wildcard _ matches exactly one character and is used with keyword “LIKE”.



```
mysql> SELECT id, name
-> FROM recipe;
+----+-----+
| id | name  |
+----+-----+
| 1  | Jello |
| 2  | Pudding |
| 3  | Koolaid |
+----+-----+
3 rows in set (0.00 sec)

mysql> SELECT id, name
-> FROM recipe WHERE
-> name LIKE '____';
+----+-----+
| id | name  |
+----+-----+
| 2  | Pudding |
| 3  | Koolaid |
+----+-----+
2 rows in set (0.01 sec)
```

Example:

```
SELECT id, name
FROM recipe WHERE
name LIKE '____'; # 7
```



- A table alias is an alternative name (often abbreviated) which references a table.
 - Note: in the example “r” alias for table “recipe”, “c” alias for table “category”

Table Alias Example:

```
SELECT *  
    FROM recipe AS r,  
           category AS c  
WHERE r.category_id = c.id;
```




- A column alias is an alternative abbreviated name of which to reference a column
 - Note: in the example, “Title” alias for column “recipe.name”, “Type” alias for column “category.name”

```
mysql> SELECT r.name, c.name
-> FROM recipe AS r,
-> category AS c
-> WHERE r.category_id = c.id;
```

name	name
Jello	Dessert
Pudding	Dessert
Koolaid	Beverage

3 rows in set (0.00 sec)

```
mysql> SELECT r.name AS 'Title',
-> c.name AS 'Type',
-> FROM recipe AS r,
-> category AS c
-> WHERE r.category_id = c.id
-> ORDER BY title;
```

Title	Type
Jello	Dessert
Koolaid	Beverage
Pudding	Dessert

3 rows in set (0.00 sec)

Column Alias Example:

```
SELECT r.name AS 'Title',
       c.name AS 'Type'
FROM recipe AS r,
       category AS c
WHERE r.category_id = c.id
ORDER BY 'Title';
```

NKU Adding new data: INSERT



- Used for creating a new row of data in a table
- The order of the column names must match the order of the values

Syntax:

```
INSERT INTO table  
  [(column1, ... )]  
  VALUES (value1, ...);
```

Example:

```
INSERT INTO notecard  
  # COLUMNS  
  ( name,  
    content,  
    category_id )  
VALUES  
  ( "Pudding",      # name  
    "Add milk.",    # content  
    2 );            # category_id
```

NKU INSERT: omitting columns



- Column names can be omitted:
 - Values must be specified in the order which the columns were created.
 - No columns can be skipped.
 - Every column must have a value or NULL
- Don't do this

Example without columns:

```
INSERT INTO recipe
VALUES (
    NULL,          # id
    "Jello",       # name
    "Add water",   # content
    NULL,          # creation
    2 );           # category_id
```

NKU INSERT: rules

- If columns are not specified, values must be in the same order in which they were defined (via CREATE command)
- Numeric values should not be quoted
- String values must be quoted
- Date and time values should be quoted
- SQL functions should not be quoted.
- NULL should never be quoted.
- If a value is not specified, the value inserted is NULL, unless a default column value has been defined or column attribute is AUTO_INCREMENT.
- Quoting a column name is optional.

NKU How to get the last inserted ID



- There are many solutions.
- However, some of them are prone to some serious errors
- Always use `last_insert_ID()`

Worst Solution:

```
SELECT COUNT(id) FROM RECIPE;
```

Poor Solution:

```
SELECT MAX(id) FROM RECIPE;
```

Good Solution:

```
SELECT id FROM recipe ORDER BY  
        creation DESC LIMIT 1;
```

Best Solution:

```
SELECT LAST_INSERT_ID();
```

NKU Updating table data: UPDATE



- Used for modifying existing table data
- Pitfall
 - Failing to specify the WHERE or LIMIT clause modifies all the records in the table!
- Note
 - There is no “undo” on inadvertently modified data.

Syntax:

```
UPDATE tableName  
    SET colName = newValue  
    [WHERE colName = const]  
    [LIMIT n];
```

NKU UPDATE: examples

```
/* modify column 'content' on all rows */
UPDATE recipe
    SET content = "Mix ingredients";

/* replace "Drink" with "Beverage" (1st match) */
UPDATE category
    SET name = "Beverage"
    WHERE name = "Drink"
    LIMIT 1;

/* modify via unique id (most common) */
UPDATE recipe
    SET content = "Stir ingredients";
    WHERE id = 1;
```

NKU Deleting data: DELETE



- Used for removing row(s) in a table
- Pitfall:
 - Failing to specify the WHERE or LIMIT clause deletes all the records in the table!
- Note:
 - There is no “undo” on deleted data

Syntax:

```
DELETE FROM tableName  
    [WHERE colName = const]  
    [LIMIT n];
```


NKU DELETE: examples

```
/* delete all rows (you sure?) */
```

```
DELETE FROM book;
```

```
/* delete the first match */
```

```
DELETE FROM book
```

```
    WHERE lastName = "Wesley"
```

```
    LIMIT 1;
```

```
/* delete books having more than 30 pages*/
```

```
DELETE FROM book
```

```
    WHERE num_pages > 30;
```

```
/* delete via id (most common) */
```

```
DELETE FROM book
```

```
    WHERE id = 1;
```



Wrap-up

NKU In-class exercise

- Implement the database of a movie website similar to IMDB
 - movies
 - actors
 - directors
- Design queries to insert dummy data in the database and to modify the rating of a movie

NKU Agenda

1. Date and time functions
2. Extracting data using conditions
3. Extracting data from multiple tables
4. Indexes
5. Performance considerations

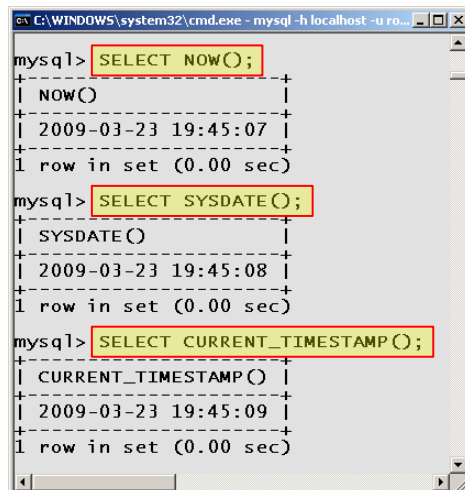


Date and time functions

NKU Date and time functions



- There are three functions that return the current time and date:
 - NOW()
 - SYSDATE()
 - CURRENT_TIMESTAMP()



```
mysql> SELECT NOW();
+-----+
| NOW() |
+-----+
| 2009-03-23 19:45:07 |
+-----+
1 row in set (0.00 sec)

mysql> SELECT SYSDATE();
+-----+
| SYSDATE() |
+-----+
| 2009-03-23 19:45:08 |
+-----+
1 row in set (0.00 sec)

mysql> SELECT CURRENT_TIMESTAMP();
+-----+
| CURRENT_TIMESTAMP() |
+-----+
| 2009-03-23 19:45:09 |
+-----+
1 row in set (0.00 sec)
```

Example:

```
SELECT NOW();
```

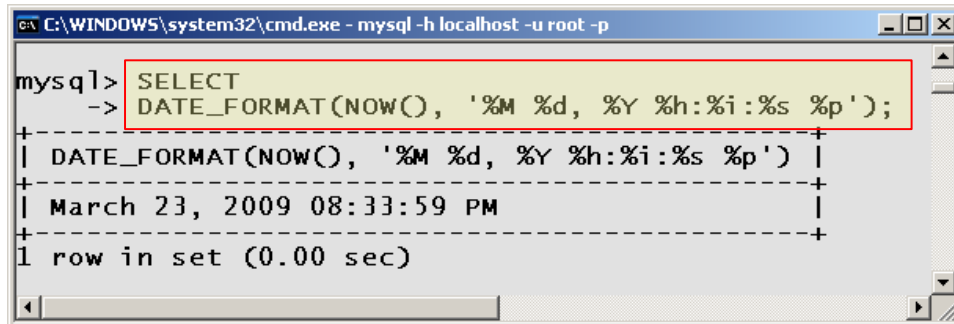
```
SELECT SYSDATE();
```

```
SELECT CURRENT_TIMESTAMP();
```

NKU Date format function



- SQL has one command to display the date in the format specified
 - DATE_FORMAT()



```
C:\WINDOWS\system32\cmd.exe - mysql -h localhost -u root -p

mysql> SELECT
-> DATE_FORMAT(NOW(), '%M %d, %Y %h:%i:%s %p');
+-----+
| DATE_FORMAT(NOW(), '%M %d, %Y %h:%i:%s %p') |
+-----+
| March 23, 2009 08:33:59 PM                    |
+-----+
1 row in set (0.00 sec)
```

Example:

```
SELECT
DATE_FORMAT (
NOW(), '%M %d, %Y %h:%i:%s %p');
```

Date format specifier characters (1/2)

Specifier	Description
%a	Abbreviated weekday name (Sun..Sat)
%b	Abbreviated month name (Jan..Dec)
%c	Month, numeric (0..12)
%D	Day of the month with English suffix (1st, 2nd, 3rd, ...)
%d	Day of the month, numeric (00..31)
%e	Day of the month, numeric (0..31)
%f	Microseconds (000000..999999)
%H	Hour (00..23)
%h	Hour (01..12)
%I	Hour (01..12)

Specifier	Description
%i	Minutes, numeric (00..59)
%j	Day of year (001..366)
%k	Hour (0..23)
%l	Hour (1..12)
%M	Month name (January..December)
%m	Month, numeric (00..12)
%p	AM or PM
%r	Time, 12-hour (hh:mm:ss with AM or PM)
%S	Seconds (00..59)
%s	Seconds (00..59)



Date format specifier characters (1/2)

Specifier	Description
%T	Time, 24-hour (hh:mm:ss)
%U	Week (00..53), where Sunday is the first day of the week
%V	Week (01..53), where Sunday is the first day of the week; used with %X
%W	Weekday name (Sunday..Saturday)
%w	Day of the week (0=Sunday..6=Saturday)
%X	Year for week where Sunday is first day of week, numeric, four digits; used with %V
%Y	Year, numeric, four digits
%y	Year, numeric (two digits)
%%	A literal “%” character

NKU Minimum and maximum value



- Finding the minimum and maximum value
 - MIN(): Finds the minimum value for a column
 - MAX(): Finds the maximum value for a column
- Note: in the example “Poor” and “Rich” are aliases for “MIN(salary)” and “MAX(salary)”

Example:

```
SELECT MIN(salary) AS Poor,  
       MAX(salary) AS Rich  
FROM paystub;
```



- COUNT() Finds the number of data rows
- Note: in the example “total” is an alias for “COUNT(id)”

Example:

```
SELECT COUNT(id) AS total  
FROM recipe;
```

NKU Getting the sum of values



- SUM(): Finds the total value of a column
- Note: in the example “Total is an alias for “SUM(salary)”

```
Example:  
SELECT SUM(salary)  
        AS Total  
FROM paystub;
```



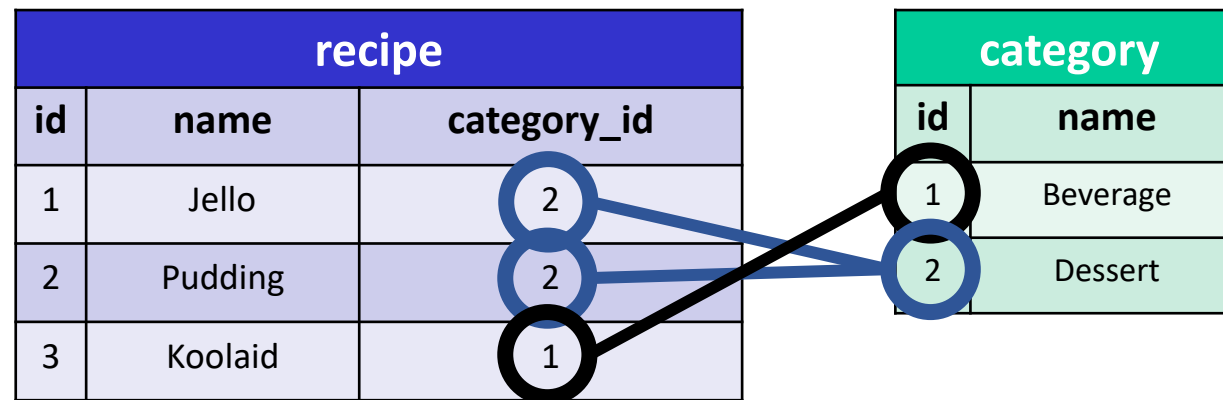
- **AVG():** Finds the average value for a column
- **Note:** in the example “Average” is an alias for “AVG(salary)”.

Example:

```
SELECT AVG(salary)
      AS Average
FROM paystub;
```

NKU Tables with foreign keys

- Often multiple tables are combined to obtain the information required.



- The tables are “joined” via the where clause.



Extracting data using conditions

NKU The WHERE clause



- The WHERE clause filters out rows based on their columns' data values
- in large databases, it's critical to use a WHERE clause to reduce the result set size
 - suggestion: when trying to write a query, think of the FROM part first, then the WHERE part, and lastly the SELECT part

```
SELECT name, population FROM cities WHERE  
country_code = "FSM";
```

name	population
Weno	22000
Palikir	8600



- The WHERE portion of a SELECT statement can use the following operators:
 - =, >, >=, <, <=
 - <> : not equal
 - BETWEEN min AND max
 - LIKE pattern
 - IN (value, value, ..., value)

```
SELECT name, gnp FROM countries WHERE gnp > 2000000;
```

code	name	gnp
JPN	Japan	3787042.00
DEU	Germany	2133367.00
USA	United States	8510700.00
...

NKU Using multiple clauses



- Multiple WHERE conditions can be combined using AND and OR
- When combining several clauses, you can conveniently use parentheses to group clauses

```
SELECT * FROM cities WHERE code = 'USA' AND  
population >= 2000000;
```

id	name	country_code	district	population
3793	New York	USA	New York	8008278
3794	Los Angeles	USA	California	3694820
3795	Chicago	USA	Illinois	2896016
...



- How to use LIKE
 - LIKE 'text%' searches for text that starts with a given prefix
 - LIKE '%text' searches for text that ends with a given suffix
 - LIKE '%text%' searches for text that contains a given substring

```
SELECT code, name, population FROM countries  
WHERE name LIKE 'United%';
```

code	name	population
ARE	United Arab Emirates	2441000
GBR	United Kingdom	59623400
USA	United States	278357000
UMI	United States Minor Outlying Islands	0



- You can write ASC or DESC to sort in ascending (default) or descending order:
 - `SELECT * FROM countries ORDER BY population DESC;`
- can specify multiple orderings in decreasing order of significance:
 - `SELECT * FROM countries ORDER BY population DESC, gnp;`

```
SELECT code, name, population FROM countries  
WHERE name LIKE 'United%' ORDER BY  
population;
```

code	name	population
UMI	United States Minor Outlying Islands	0
ARE	United Arab Emirates	2441000
GBR	United Kingdom	59623400
USA	United States	278357000



- can be used to get the top-N of a given category (ORDER BY and LIMIT)
- also useful as a sanity check to make sure your query doesn't return thousands of rows

```
SELECT name FROM cities WHERE name LIKE 'K%'  
LIMIT 5;
```

name
Kabul
Khulna
Kingston upon Hull
Koudougou
Kafr al-Dawwar



Extracting data from multiple tables

NKU Example of related tables

- **primary key:** a column guaranteed to be unique for each record (e.g. Lisa Simpson's ID 888)
- **foreign key:** a column in table A storing a primary key value from table B
 - (e.g. records in grades with student_id of 888 are Lisa's grades)
- **normalizing:** splitting tables to improve structure / redundancy (linked by unique IDs)

id	name	email
123	Bart	bart@fox.com
456	Milhouse	milhouse@fox.com
888	Lisa	lisa@fox.com
404	Ralph	ralph@fox.com

students

id	name	teacher_id
10001	Computer Science 142	1234
10002	Computer Science 143	5678
10003	Computer Science 190M	9012
10004	Informatics 100	1234

courses

student_id	course_id	grade
123	10001	B-
123	10002	C
456	10001	B+
888	10002	A+
888	10003	A+
404	10004	D+

grades

id	name
1234	Krabappel
5678	Hoover
9012	Stepp

teachers

NKU Querying multi-table databases

- When we have larger datasets spread across multiple tables, we need queries that can answer high-level questions such as:
 - What courses has Bart taken and gotten a B- or better?
 - What courses have been taken by both Bart and Lisa?
 - Who are all the teachers Bart has had?
 - How many total students has Ms. Krabappel taught, and what are their names?
- To do this, we'll have to join data from several tables in our SQL queries.

NKU Joining with ON clauses



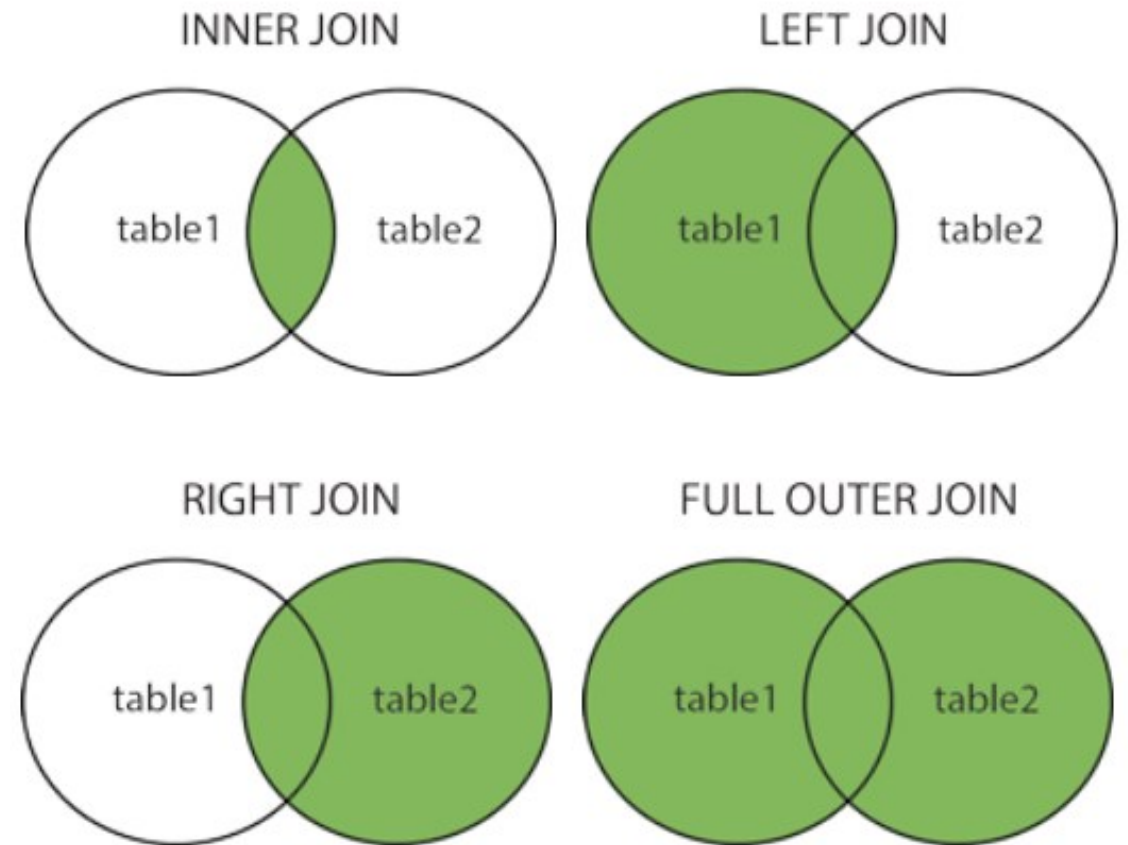
- join: combines records from two or more tables if they satisfy certain conditions
- the ON clause specifies which records from each table are matched
- the rows are often linked by their key columns (id)

```
SELECT *  
FROM students  
JOIN grades ON id = student_id;
```

id	name	email	student_id	course_id	grade
123	Bart	bart@fox.com	123	10001	B-
123	Bart	bart@fox.com	123	10002	C
404	Ralph	ralph@fox.com	404	10004	D+
456	Milhouse	milhouse@fox.com	456	10001	B+
888	Lisa	lisa@fox.com	888	10002	A+
888	Lisa	lisa@fox.com	888	10003	A+

NKU Different types of joins

- **(INNER) JOIN:** Returns records that have matching values in both tables
- **LEFT (OUTER) JOIN:** Return all records from the left table, and the matched records from the right table
- **RIGHT (OUTER) JOIN:** Return all records from the right table, and the matched records from the left table
- **FULL (OUTER) JOIN:** Return all records when there is a match in either left or right table



NKU Filtering columns in a join



- You can select individual columns
- The notation **table.column** can be used to disambiguate column names:
 - `SELECT * FROM students JOIN grades ON students.id = grades.student_id;`

```
SELECT name, course_id, grade  
FROM students  
JOIN grades ON id = student_id;
```

name	course_id	grade
Bart	10001	B-
Bart	10002	C
Ralph	10004	D+
Milhouse	10001	B+
Lisa	10002	A+
Lisa	10003	A+



- FROM / JOIN glue the proper tables together, and WHERE filters the results
- what goes in the ON clause, and what goes in WHERE?
 - ON directly links columns of the joined tables
 - WHERE sets additional constraints such as particular values (123, 'Bart')

```
SELECT name, course_id, grade
FROM students
JOIN grades ON id = student_id
WHERE name = 'Bart';
```

name	course_id	grade
Bart	10001	B-
Bart	10002	C

NKU Using table aliases



- You can give names to tables, like a variable name in Java
- to specify all columns from a table, write table.*
- (grade column sorts alphabetically, so grades C or better are ones <= it)

```
SELECT s.name, g.*  
FROM students s  
JOIN grades g ON s.id = g.student_id  
WHERE g.grade <= 'C';
```

name	student_id	course_id	grade
Bart	123	10001	B-
Bart	123	10002	C
Milhouse	456	10001	B+
Lisa	888	10002	A+
Lisa	888	10003	A+

NKU Multi-way joins



- More than 2 tables can be joined, as shown above
- What does the above query represent?
 - The names of all courses in which Bart has gotten a B- or better.

```
SELECT c.name
FROM courses c
JOIN grades g ON g.course_id = c.id
JOIN students bart ON g.student_id = bart.id
WHERE bart.name = 'Bart' AND g.grade <= 'B-';
```

name
Computer Science 142



Indexes

NKU Indexes

- Mechanism that enables a database to locate a record in row rapidly (e.g. table of contents or textbook index)
- Best Practice Don'ts:
 1. Don't index every column
 2. Indexing uses extra storage space
 3. Additional time is required to create indexed columns during data insert
 4. Too many indexes increase search time to locate record
- Don't index primary keys, they are already indexed



- As a best practice, columns which are the most utilized as conditions in queries should be indexed:
 1. Columns in a WHERE clause
 2. Columns in an ORDER BY clause
 3. Columns in MIN and MAX clauses

```
//e.g., WHERE
SELECT * FROM authors
    WHERE author = 'Max Lucado';
// author should be indexed

//e.g., in an ORDER BY clause
SELECT * FROM contacts
    ORDER BY author;
// author should be indexed

//e.g., in MIN and MAX clauses
SELECT MAX(elevation) FROM mountain;
// elevation should be indexed
```



- Indexes can be defined
 - While creating the table
 - On existing tables

```
// e.g., while creating the table
CREATE TABLE employee (
    id INT NOT NULL
        AUTO_INCREMENT, // id
    name VARCHAR(11),
    INDEX name_ix(name) );

// e.g., in an existing table
CREATE INDEX name_ix ON
    employee(name);
```



- Indexes can have the unique keyword
- Unique indexes can be defined
 - While creating the table
 - On existing tables

```
// e.g., Unique Index upon table creation:
```

```
CREATE TABLE employee (  
    id INT NOT NULL  
        AUTO_INCREMENT,  
    ss_number VARCHAR(11),  
    UNIQUE ss_uq(ss_number) );
```

```
// e.g., Unique Index for an existing table:
```

```
CREATE UNIQUE INDEX ss_uq ON  
    employee(ss_number);
```

NKU Multiple column indexes



- Unlike primary keys, an index can be applied to multiple columns.
- Multi-column unique key, for example, could assure that the combination of two columns is unique.

location	
city	state
Manhattan	KS
Manhattan	NY

Example:

```
CREATE UNIQUE INDEX state_uq ON  
location(city, state);
```



Performance considerations

NKU Performances

```
SELECT *  
FROM author  
WHERE name = 'John MacArthur';
```

Without index

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	author	ALL	NULL	NULL	NULL	NULL	7	Using where

With index

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	author	const	name_uq	name_uq	42	const	1	

Explanation:

- type: ALL means every record is scanned to determine match.
- possible_keys: NULL means no index defined.
- key: NULL means no key is used by query.
- rows: Number of rows searched for query (there are 7 records in the database).