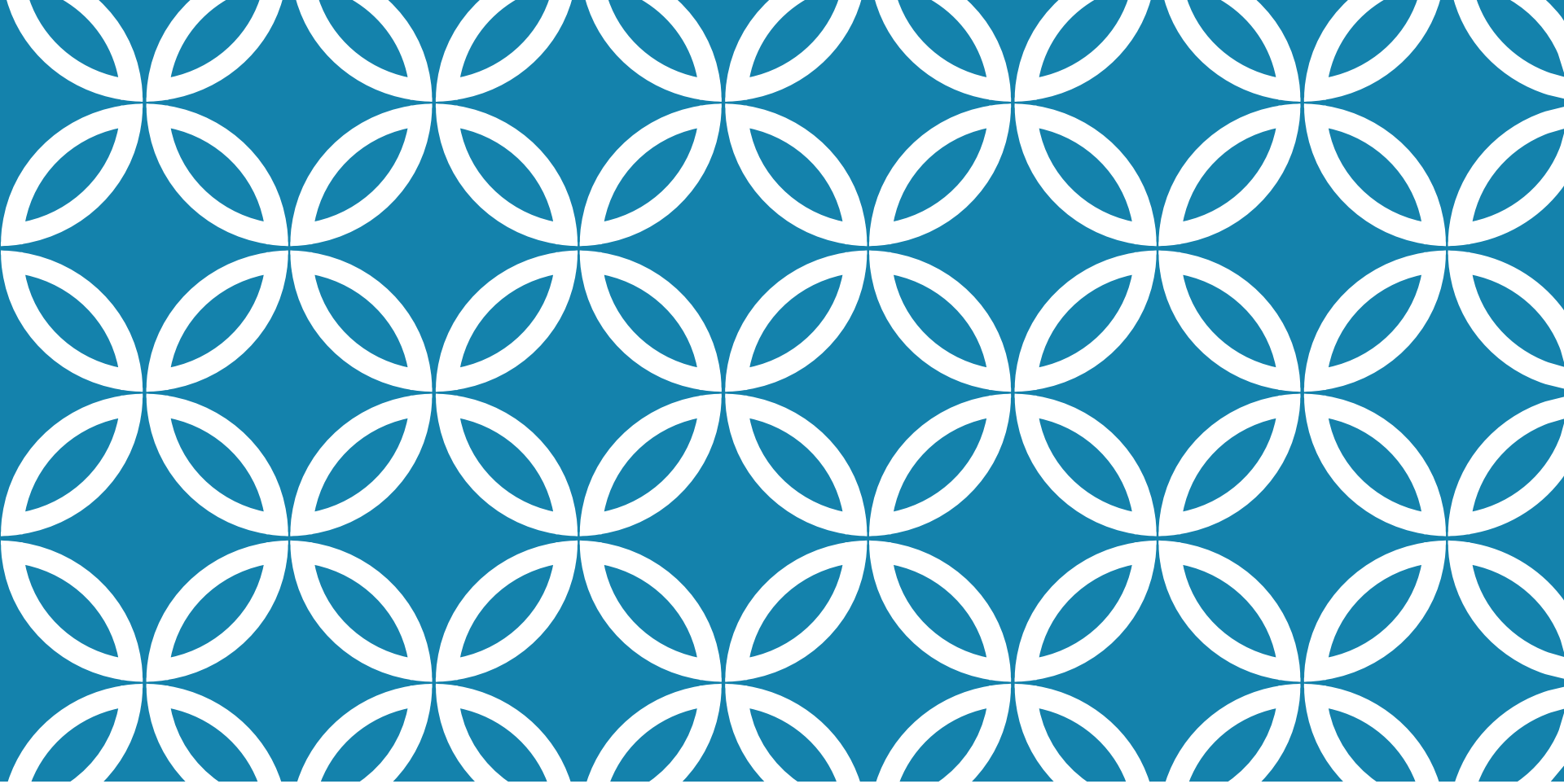


INTRODUCTION TO SQL AND DATA DEFINITION LANGUAGE (DDL)

CT230 Database Systems

LABS NEXT WEEK ….

Mon 19th 4-6 in IT106
Tue 20th 3-5 in IT101
Thur 23rd 10-2 in IT106 – will have assigned time before then

Please attend if you are able!

Nice working environment and you can get help if needed.

Main goals of the lab next week are:

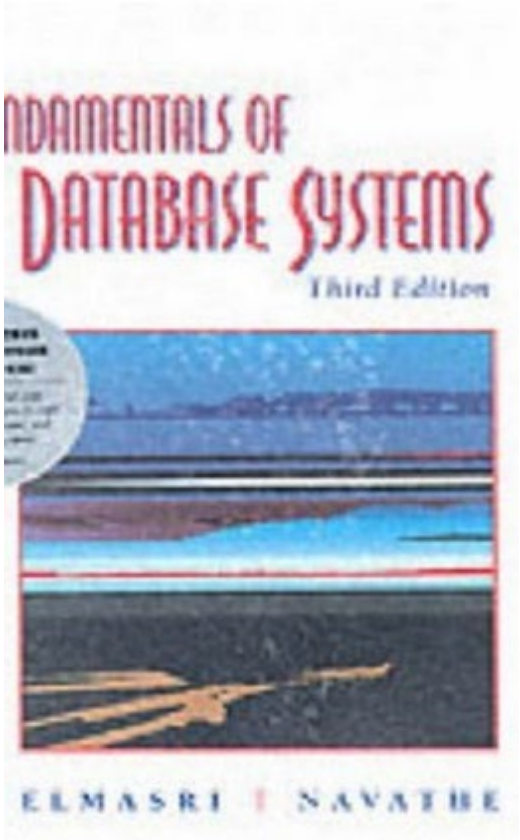
- Becoming familiar with phpMyAdmin and/or Adminer.
- Becoming familiar with the company database.
- Creating tables – GUI and DDL `CREATE TABLE`
- Adding data using `INSERT INTO`

SQL AND DDL

Relevant chapter in recommended book:

Elmasri and Navathe

Chapter 8 (3rd Edition)



QUESTIONS?

SQL:

- Structured
  - Query
    - Language

A special purpose programming language for relational database systems

FEATURES OF SQL:

- SQL is based on *relational algebra*:
  - All relational, set and hybrid operators are supported but SQL has additional operators to allow easier query development.
- SQL has been standardised since 1987 (SQL-86/SQL-87)
- The American National Standards Institute (ANSI) and the International Standards Organization (ISO) form SQL standards committees. Many vendors also take part.
- Recent standards include XML-related features in addition to many others (e.g., JSON data types etc.)

ANSI/ISO SQL

Despite standards there can be lack of portability between database systems due to:

- Complexity and size of standards (not all vendors will implement all of the standard).
- Vendor wants to keep syntax consistent with their other software products/OS or develop features to support user base.
- Want to maintain backward compatibility.
- Want to maintain "Vendor lock-in".

ANSI/ISO SQL

We will concentrate on the standardised SQL syntax that should work across vendors:

Comprises three components:

DDL – data definition language

DCL – data control language

DML – data manipulation language

DCL: DATA CONTROL LANGUAGE

Used to control access to the database and to database relations.

Role of database administrator.

Very important in multi-user systems.

Typical commands:

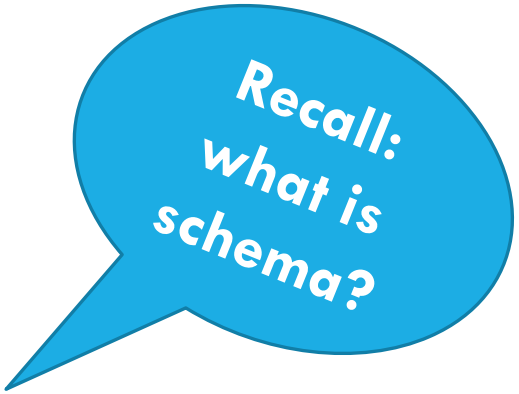
- `GRANT`
- `REVOKE`

Each of these can be used to:

Grant/revoke access to database.

Grant/revoke access to individual relations.

DDL:
DATA DEFINITION LANGUAGE



Standardised language to **define** the **schema** of a database.

Back-end of "Design" options on Interface (e.g. Create options).

Typical tasks: create, modify, and remove database objects such as tables and indexes.

Common DDL keywords are:

CREATE

ALTER

DROP

ADD

CONSTRAINT

DML:
DATA MANIPULATION LANGUAGE

4 DML statements:

`INSERT`	insert data
`SELECT`	query data
`UPDATE`	update data
`DELETE`	delete data

BACK TO DDL COMMANDS:

We use the DDL commands to mostly create tables and add constraints to our database:

Common DDL keywords are:

CREATE

ALTER

DROP

ADD

CONSTRAINT

Create a table and it's indexes and constraints

Steps:

1. Specify table (relation) name.

2. For each attribute in the table specify:
   - Attribute Name (e.g., ssn)
   - Data Type (e.g., bigint).
   - Any constraints (e.g. not null).

Recall: what is a primary key?

3. Specify Primary key of table: choose one or more attributes.

4. Specify Foreign keys *if they exist* and assuming the attributes and table you are referencing exists (may have to return to this step).

** Steps 1-3 MUST be completed for all tables.

Recall: what is a foreign key?

DATA TYPES
3 MAIN TYPES: strings, numeric and date/time

The main ones you will use:

- char(size)
- varchar(size)
- bool/boolean
- tinyint, smallint(size), mediumint(size), int(size)/integer(size), bigint(size)
- double(size, d)
- float()
- decimal(size, d)
- date, datetime, timestamp, time, year

Important to pick a suitable data type and a suitable size (based on the sample data)

Strings	**can contain letters, numbers, and special characters** *size* parameter specifies the maximum column length in characters
`char(size)`	FIXED length. *size can* be from 0 to 255. Default is 1
`varchar(size)`	VARIABLE length. *size* can be from 0 to 65535
`text`	string

Date/time	
`date`	Format: YYYY-MM-DD
`time`	Format: hh:mm:ss
`datetime`	Format: YYYY-MM-DD hh:mm:ss
`year`	A year in four-digit format

… Important to pick a suitable data type and a suitable size (based on the sample data) *ctd.*

Numeric	Max *size* value is 255 (mySQL supports UNSIGNED numeric types but not all DBMS do)
`Integers`	*See next slide*
`Bool/Boolean`	0 is False; non zero is True
`FLOAT`	Floating point number. 4 bytes, single precision
`DOUBLE`	Floating point number. 8 bytes, double precision
`DECIMAL(`*`size`*`,` *`d`*`)` `or dec(size,d)`	An exact fixed-point number. size = total number of digits (max 65, default 10) d = number of digits after the decimal point (max 30, default 0).

INTEGERS

Type	Bytes	Range
`tinyint`	1	-128 to 127
`smallint`	2	-32768 to 32767
`mediumint`	3	-8388608 to 8388607
`int`	4	-2147483648 to 2147483647
`bigint`	8	-9223372036854775808 to 9223372036854775807

Note:

Number in brackets (for integers) only refers to display not size

OTHERS

Unicode Char/String
Binary
Blob, Json etc.

AUTONUMBER

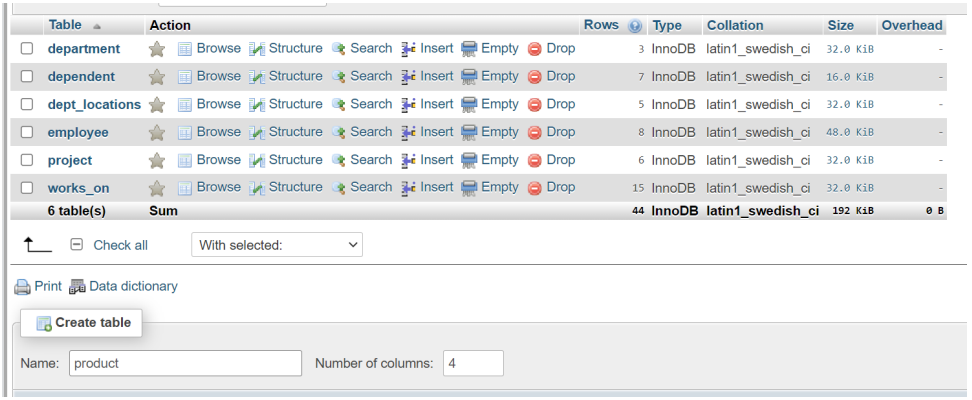
AUTO_INCREMENT in mySQL

Specifying an attribute to be "AUTO-INCREMENT" tells the DBMS to generate a number automatically when a new tuple is inserted into a table.

Often this is used for an "artificial" primary key value which is needed to ensure we have a primary key but has no meaning for the data being stored – using auto-increment means that the DBMS takes care of inserting a unique value automatically every time a new tuple is inserted.

By default, **AUTO_INCREMENT** is 1, and is incremented by 1 for each new tuple inserted.

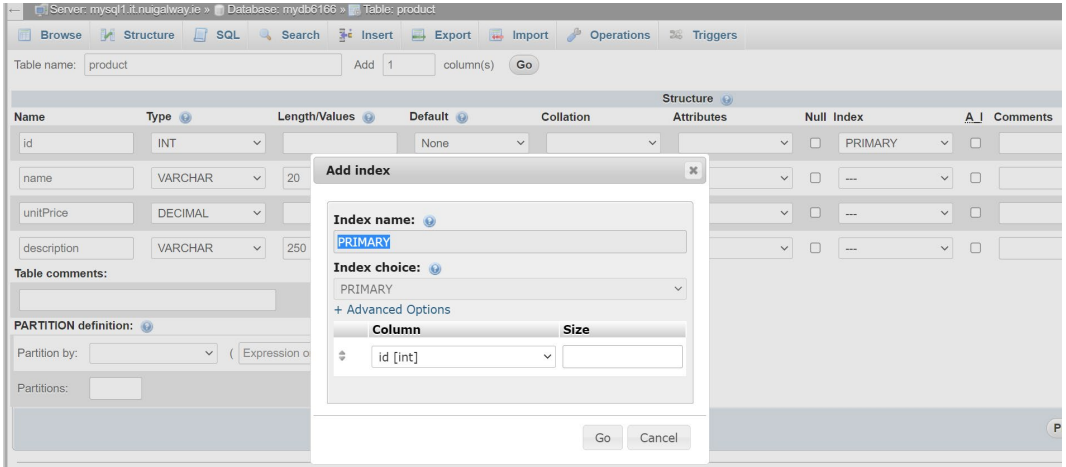
USING phpMyAdmin GUI to create a table and PK



Steps:

1: In the "Structure" view, in the "Create table" section, enter the new table name and number of columns and click the "Go" button.

2: In the new window, enter details of attributes (name and data types). Specify the keys in the Index option – "Primary" (for primary keys) and "Index" for Foreign keys (if they exist) and choose "Save". Note you may wish to view the SQL generated by choosing the "Preview SQL" option.

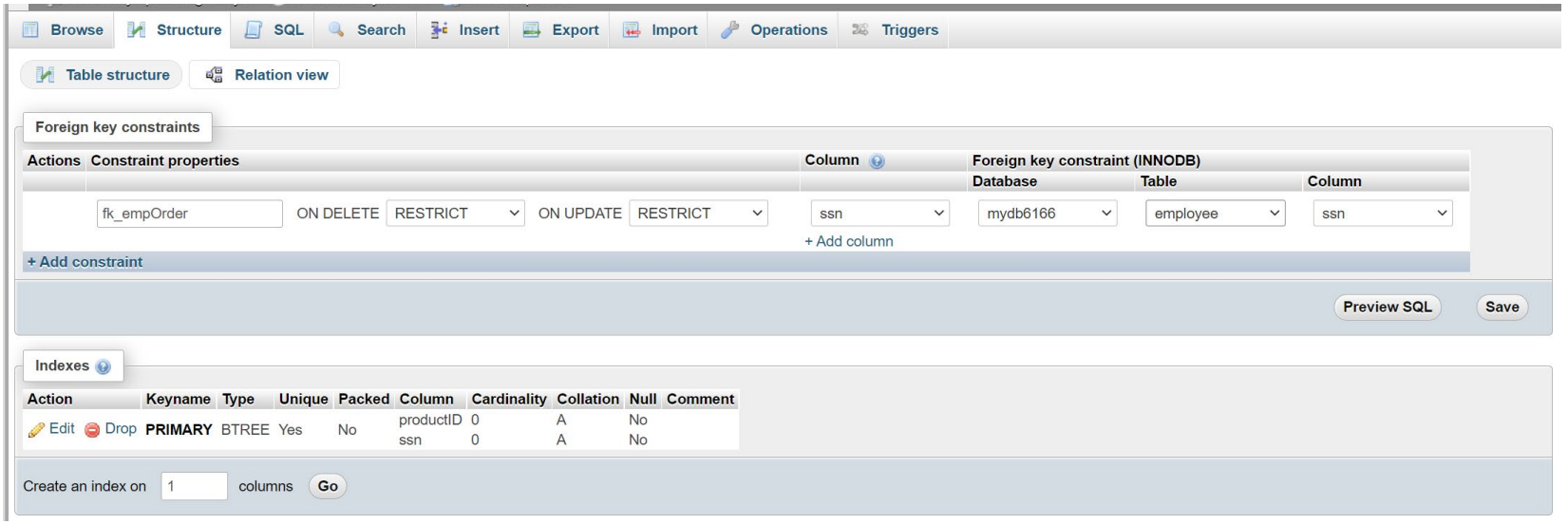


USING phpMyAdmin GUI to create Foreign keys

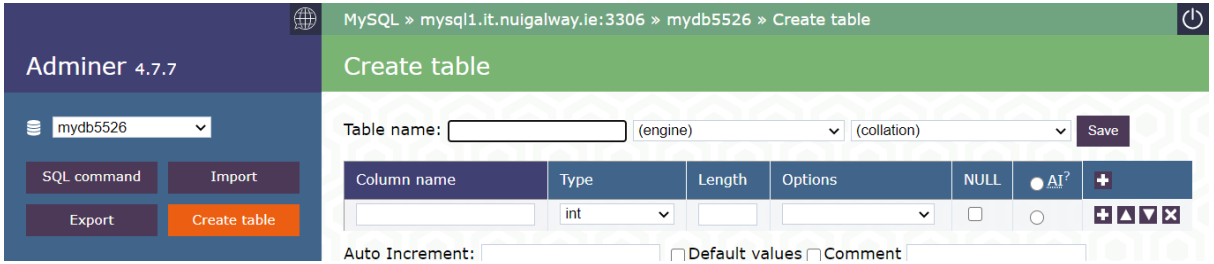
Steps:

3. Specify the FK by choosing the "Relation view" and choose the name, table and attribute that the FK references. Keep the ON DELETE and ON UPDATE as the default "RESTRICT" and choose save. (Note you might want to check "Preview SQL" again).

4. Look in Designer View to see the changes made.



USING GUI TO CREATE A TABLE
with Adminer

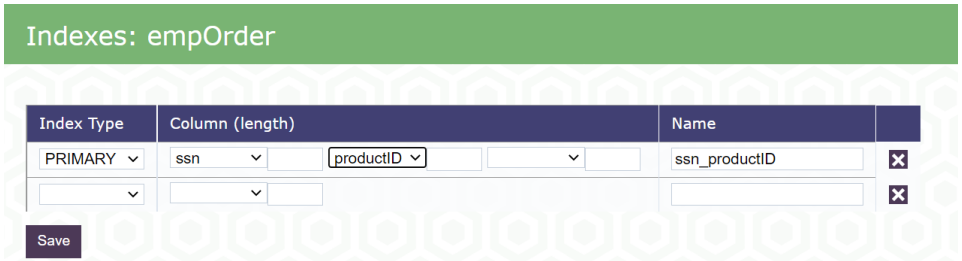


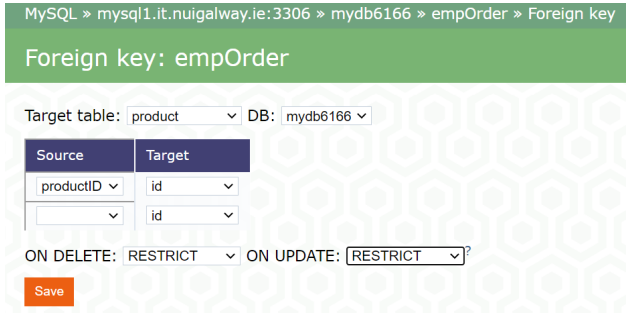
Steps:

1 and 2: Choose Create Table option and enter table name and details on attributes (name and data types). Choose the Save option.

3. Click on the table you created and choose the Alter Indexes option and specify Primary Key Index. Choose the Save option.

4. If there are foreign key(s) and the table being referenced exists, choose Add foreign key option and specify foreign keys. Choose the Save option. Else return to this step when other table(s) are created.





Using SQL DDL to create a table with index and constraints – when only one attribute is part of primary key

Syntax 1 (equivalent when only one Primary Key):

CREATE TABLE *tablename*

(attribute1 datatype [NOT NULL] [PRIMARY KEY],

attribute2 datatype [DEFAULT NULL],

attribute3 datatype,

….. ,

FOREIGN KEY (*attributename*) REFERENCES *tablename*(*attributename*)

);

Using SQL DDL to create a table with index and constraints – when more than one attribute is part of primary key

(See company2022.sql for examples!)

Syntax 2:

CREATE TABLE *tablename*

(attribute1 datatype [NOT NULL],

attribute2 datatype [DEFAULT NULL],

attribute3 datatype,

….. ,

PRIMARY KEY(*attributename(s)*),

FOREIGN KEY (*attributename*) REFERENCES *tablename*(*attributename*)

);

Naming the constraints …

Syntax 3 (name the constraints):

CREATE TABLE *tablename*

(attribute1 datatype [NOT NULL],

attribute2 datatype [DEFAULT NULL],

attribute3 datatype,

….. ,

CONSTRAINT *constraintname* PRIMARY KEY (*attributename*),

CONSTRAINT *constraintname* FOREIGN KEY (*attributename*)
REFERENCES *tablename*(*attributename*)

);

Looking at DDL code for department

```
CREATE TABLE `department` (
`dnumber` int(20) NOT NULL PRIMARY KEY,
`dname` varchar(50) DEFAULT NULL,
`mgrssn` bigint(20) DEFAULT NULL,
`mgrstartdate` date DEFAULT NULL)
ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

NOTE: CONSTRAINTS: FOREIGN KEYS:

FOREIGN KEY (*attributename*) REFERENCES *tablename*(*attributename*)

Need to specify:

* Keyword FOREIGN KEY to indicate it is a foreign key constraint and the attribute name or attribute names that will be the foreign key in current table. If there is more than one attribute they should be separated by commas. Attribute names should be enclosed in brackets.

* Keyword REFERENCES to specify attribute it references by specifying the table name and the attribute name. Again attribute name(s) should be in brackets. Table name is outside the bracket.

Constraint examples from COMPANY Schema for works_on table

```
CONSTRAINT pk_works_on PRIMARY KEY (essn, pno),

CONSTRAINT fk_works_on_employee FOREIGN KEY (essn)
REFERENCES employee(ssn),

CONSTRAINT fk_works_on_project FOREIGN KEY(pno)
REFERENCES project(pnumber)
```

Looking at DDL code in `company.sql` file

Note that:

- For this SQL dump the Foreign Keys were created after the tables, and after the data was entered (using INSERT INTO commands).
- Generally, it is better to create ALL the structure first and only then enter the data.
- Sometimes you can only add Foreign keys *after* all the tables have been created

USING ALTER TO MODIFY DESIGN

Remember: Cannot create a foreign key link *unless* the attribute it is referencing already exists

If you want to create everything but foreign keys initially you can add a foreign key later using the ALTER TABLE command

SYNTAX FOR ALTER COMMAND:

To add a constraint:

```
ALTER TABLE tablename

ADD CONSTRAINT constraintname FOREIGN KEY
(attributename) REFERENCES
tablename(attributename);
```

To add an attribute (column) constraint:

```
ALTER TABLE tablename

ADD attributename DATATYPE;
```

Looking at DDL code for Foreign Key constraint in department

```
ALTER TABLE `department`
ADD KEY `mgrssn` (`mgrssn`),
ADD CONSTRAINT `department_ibfk_2`
        FOREIGN KEY (`mgrssn`) REFERENCES `employee` (`ssn`);
```

HOW TO WORK WITH DDL IN ADMINER?

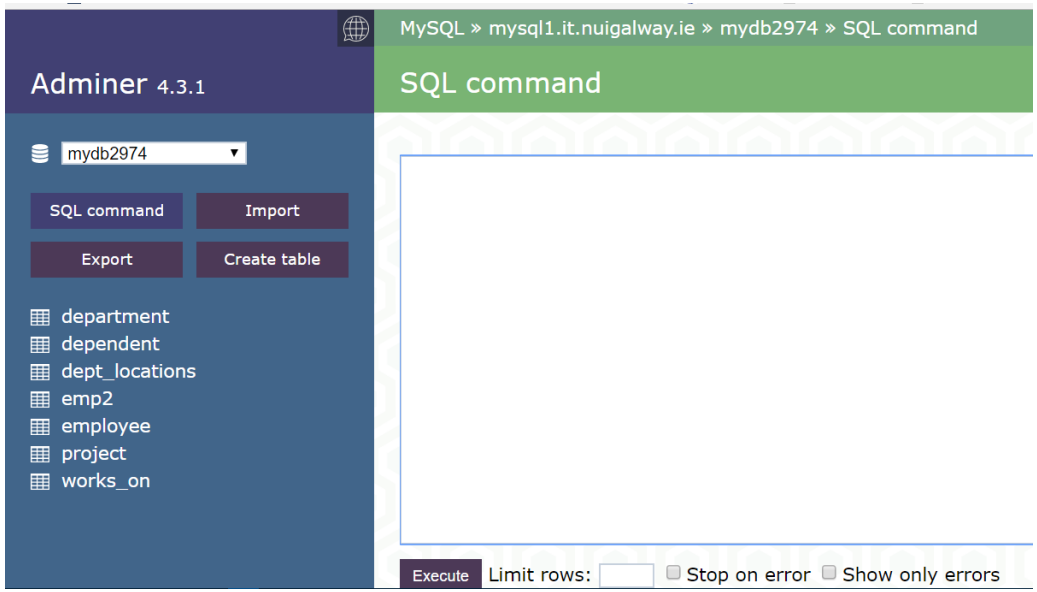
Choose:

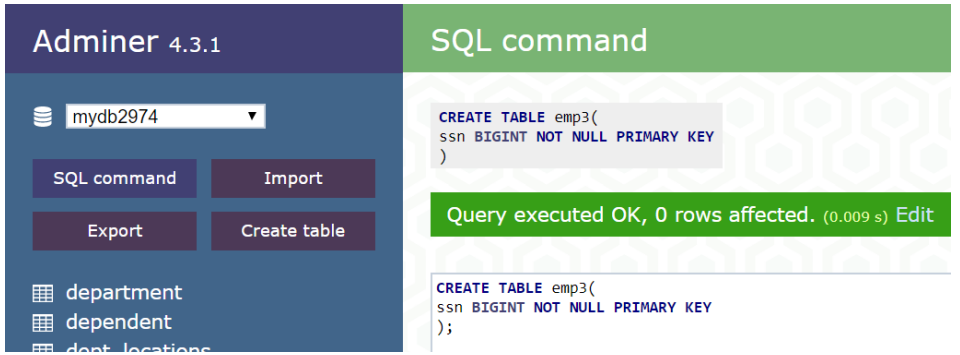
1. Choose SQL command option

2. Once you have typed in the SQL in the displayed editor choose the Execute option

(or CTRL+Enter)

* Note you may want to save your query





HOW TO WORK WITH DDL IN phpmyadmin

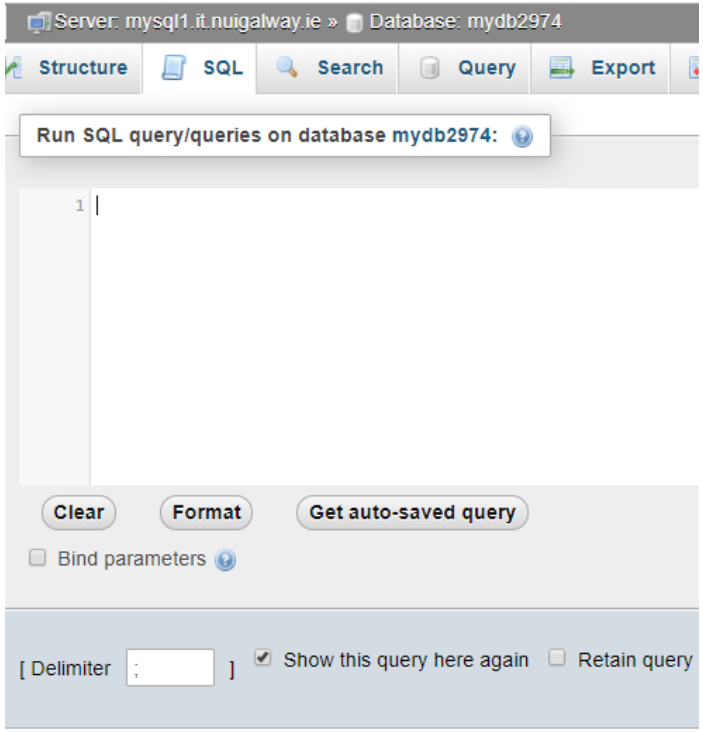
Choose:

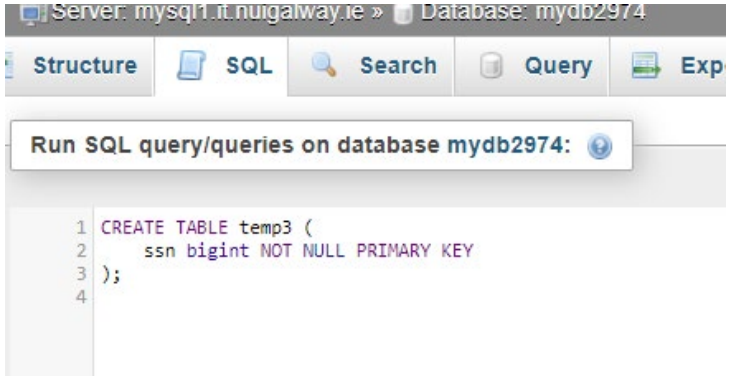
1. Choose SQL tab at the top

2. Type/Copy and Paste SQL in to the editor

3. Click “Go”

(or CTRL+Enter)





Looking at DML INSERT INTO code in `company2022.sql` file

Note that:

- Tuples are enclosed in brackets () and tuples are separated by commas
- Data type, format and order must correspond exactly to the data type, format and order specified when creating the tables.
- Strings, including dates, should be enclosed in single quotes
- Numbers are **not** enclosed in quotes

Looking at DML INSERT code for Foreign Key constraint in department

```
INSERT INTO `department`
(`dnumber`, `dname`, `mgrssn`, `mgrstartdate`) VALUES
(1, 'Headquarters', 888665555, '2019-06-19'),
(4, 'Administration', 987654321, '2015-01-01'),
(5, 'Research', 333445555, '2018-05-22');
```

IMPACT OF SETTING DATA TYPES, CONSTRAINTS (E.G., "NOT NULL), PRIMARY KEYS AND FOREIGN KEYS …

The DBMS has (Very!) strict checking of all constraints – and will not allow data to be entered if the data does not comply with the constraints set … this is one of the main advantages of a DBMS in terms of data correctness but it sometimes makes working with data entry difficult!

Consider the following examples ….

DOMAIN CONSTRAINTS

Definition: The value of each attribute A must be an atomic value from the domain dom(A).

- Can be tested easily by DBMS for data entry
- Queries can also be tested.
- Example attributes:
  - fname
  - minit
  - bdate

Column	Type
fname	varchar(50) *NULL*
minit	varchar(1) *NULL*
lname	varchar(50) *NULL*
ssn	bigint(20)
bdate	date *NULL*
address	varchar(100) *NULL*
gender	varchar(50) *NULL*
salary	double *NULL*
superssn	bigint(20) *NULL*
dno	int(11) *NULL*

Essentially: data types and formats must match to that specified

ENTITY INTEGRITY CONSTRAINTS (PRIMARY KEY CONSTRAINTS)

Definition: The primary key should uniquely identify each tuple in a relation. This means:

- No duplicate values for primary key allowed
- Null values not allowed for primary key
- Example:
- `ssn` in **employee** table
- `essn` and `pno` in **works_on** table

Essentially: "no null or missing values for primary key"

NOTE:

As we already discussed, Null values may not be permitted for other attributes also. e.g., name of student may be constrained to be NOT NULL

- We often see this constraint when filling out forms online (*required) and the constraint is often necessary for many non-key attributes
- However, we should be careful of only adding 'NOT NULL' constraints in the databases in our own assignments when they are **really necessary**

REFERENTIAL INTEGRITY CONSTRAINTS

Definition: Specified between two relations and require the concept of a foreign key. The constraint ensures that the database must **not** contain any unmatched foreign keys.

Therefore a relationship involving foreign keys MUST be between attributes of the **same type and size**

In addition, a value for a foreign key attribute MUST exist already as a candidate key value.

Essentially: "no unmatched foreign keys"

EXAMPLE (AGAIN):

employee

Modify	fname	minit	lname	ssn	bdate	address	gender	salary	superssn	dno
edit	John	B	Smith	123456789	1975-01-09	731 Fondren, Houston, Tx	Man	55250	333445555	5
edit	Franklin	T	Wong	333445555	1980-12-08	638 Voss, Houston, TX	Man	65000	888665555	5
edit	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	Woman	44183	333445555	5
edit	Ramesh	K	Narayan	666884444	1995-09-15	975 Fire Oak, Humble, TX	Man	60000	333445555	5
edit	James	E	Borg	888665555	1997-11-10	450 Stone, Houston, TX	Man	94199	*NULL*	1
edit	Jennifer	S	Wallace	987654321	1991-06-20	291 Berry, Bellaire, TX	Woman	69240	888665555	4
edit	Ahmad	V	Jabbar	987987987	2000-03-29	980 Dallas, Houston, TX	Man	44183	987654321	4
edit	Alicia	J	Zelaya	999887777	1998-07-19	3321 Castle, Spring, TX	Non-binary	44183	987654321	4

department

Modify	dnumber	dname	mgrssn	mgrstartdate
edit	1	Headquarters	888665555	2019-06-19
edit	4	Administration	987654321	2015-01-01
edit	5	Research	333445555	2018-05-22

Any referential integrity constraints problems with dno (a foreign key in relation employee) linking to dnumber in department?

SEMANTIC INTEGRITY CONSTRAINTS

Specified and enforced using a *constraint specification language*

Two types:

state constraints: e.g., "the maximum number of hours an employee can work on all projects per week is 39"

transition constraints: e.g., "the salary of an employee can only increase"; "the date entered for order delivery must not be in the past"

We will not use semantic integrity constraints

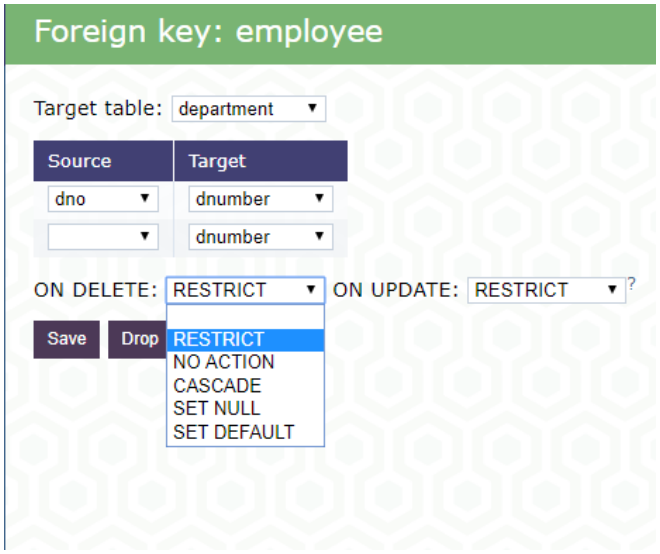
Consider the MySQL database and the associated data (company2022.sql):

Are there any unmatched foreign keys?

Are foreign and primary keys of same type and size?

SETTING CONSTRAINTS

- Domain constraints are set automatically once the data type is chosen
- Entity constrains are also set automatically once a primary key has been chosen
- Usually default constraints are set for foreign keys but these can be changed



UPDATE OPERATIONS AND CONSTRAINT VIOLATIONS

The DBMS must check that constraints are not violated whenever update operations are applied.

Three basic update operations on tables where constraints must be checked:

- insert
- delete
- modify

1. INSERT OPERATION

Provides a list of attribute values for a new tuple *t* that is to be inserted in to a relation *R*

This can happen directly via the interface or via a query

If a constraint is violated DBMS will reject insertion; usually with an explanation

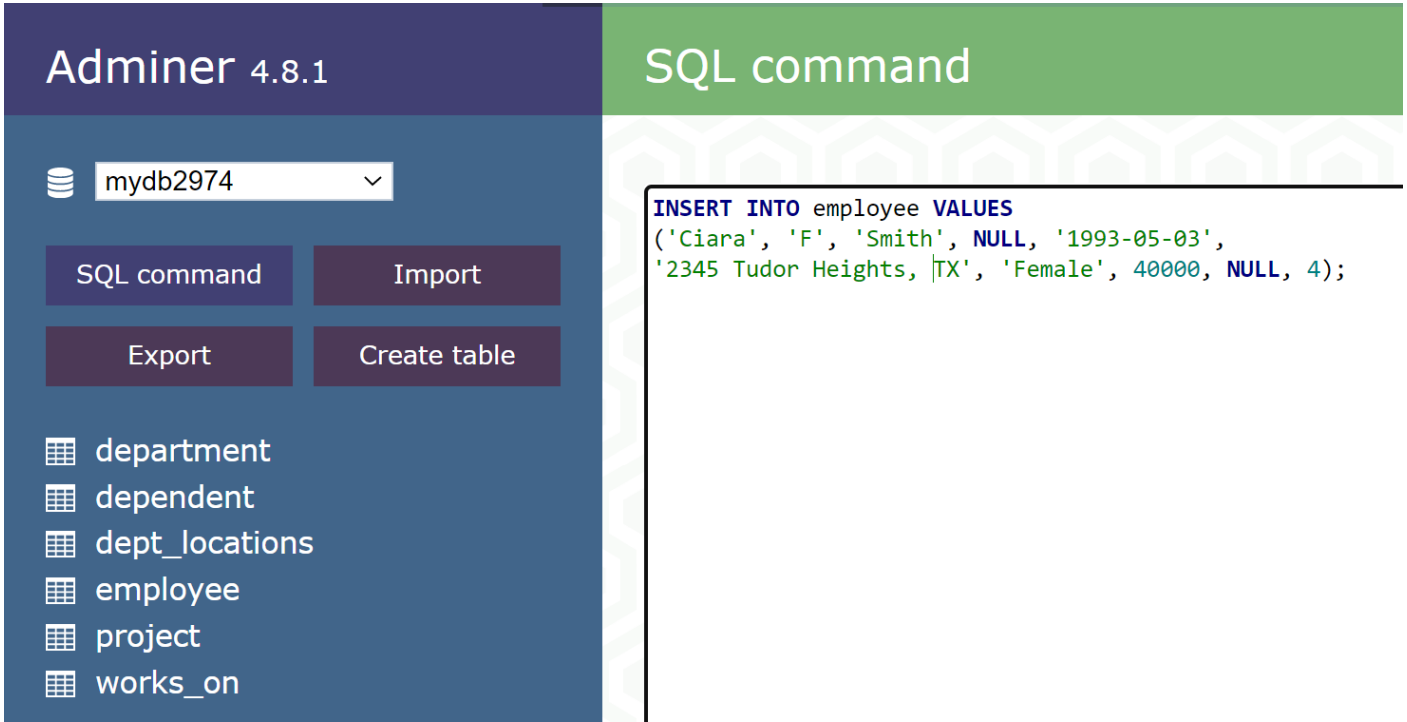
Examples:

Using the company database state the problems, if any, with the following insertions to the database:

```
INSERT INTO employee VALUES

('Ciara', 'F', 'Smith', NULL, '1993-05-03', '2345 Tudor Heights, TX', 'Female', 40000, NULL, 4);

INSERT INTO employee VALUES

('Tony', 'D', 'Burns', 523523523, '1983-05-03', '34 Sycamore Drive, TX', '2000', 50000, NULL, 4);

INSERT INTO employee VALUES

('Tony', 'D', 'Burns', 523523523, '1983-05-03', '34 Sycamore Drive, TX', 'Male', 50000, NULL, 14);

INSERT INTO employee VALUES

('Ciara', 'F', 'Smith', 4444555, '1993-05-03', '2345 Tudor Heights, TX', 'Female', 40000, NULL, 4);
```

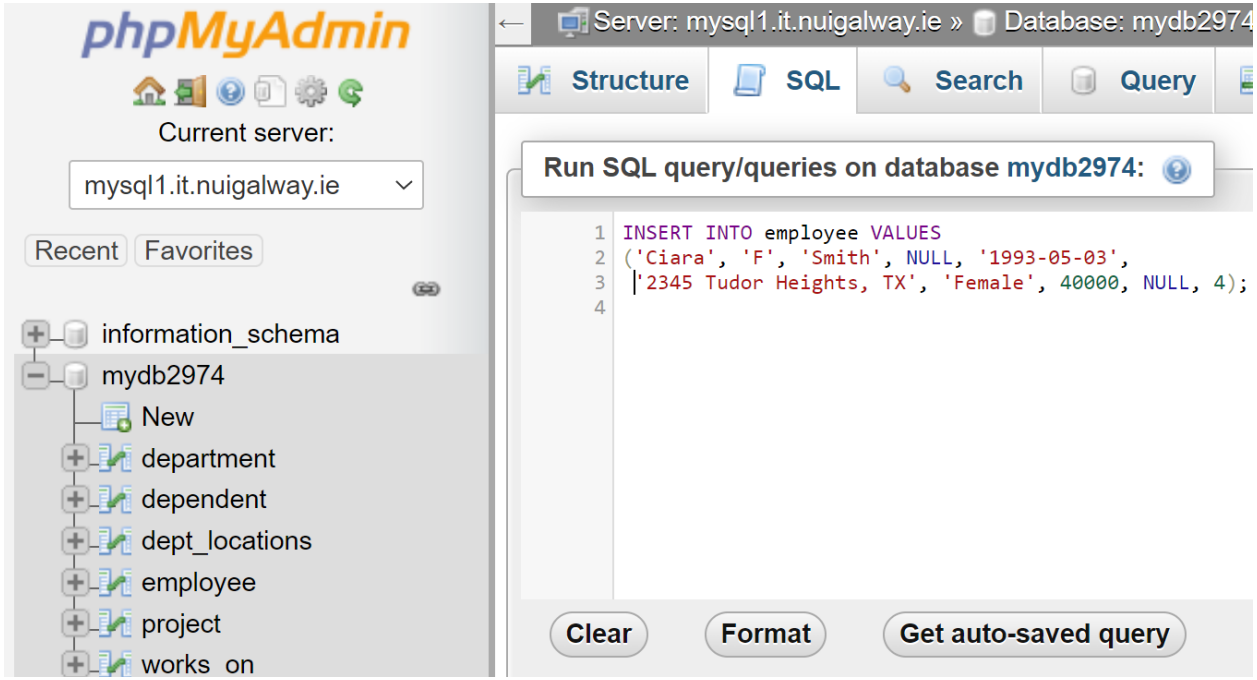
Trying this with Adminer:

1. Choose the “SQL command” button on LHS
2. A SQL editor is displayed on RHS
3. Type or copy and paste query in to editor
4. Choose “Execute” command



Trying this with phpMyAdmin

1. Choose the “SQL” tab on the top
2. A SQL editor is displayed in the middle of the screen
3. Type or copy and paste query in to editor
4. Choose “Go” button



2. DELETE OPERATION

A delete operation can only violate referential integrity constraints, i.e., if the tuple being deleted is referenced by the foreign keys from other tuples.

DBMS can:

reject deletion, with explanation

attempt to *cascade* deletion

modify referencing attribute

EXAMPLE: DELETE THE TUPLE JUST INSERTED (WITH SSN = 4444555)

```
DELETE FROM employee
WHERE ssn = 4444555;
```

3. UPDATE OPERATION

An update operation is used to change the values of one or more attributes in a tuple of a table

Issues already discussed with insert and delete could arise with this operation, specifically:

- if a primary key is modified … same as deleting one tuple and inserting another tuple in its place
- if a foreign key is modified … DBMS must ensure that new value refers to an existing tuple in the reference relation.

CASCADE UPDATE AND DELETE

Whenever tuples (rows) in the referenced (master) table are deleted (or updated), the respective tuples of the referencing (child) table with a matching foreign key column will be deleted (or updated) as well.

Note that if cascading DELETE is turned on there could be many deletions performed with the following query:

```
DELETE FROM employee
WHERE SSN = 123456789;
```

PROBLEM SHEETS/EXAM

- In problem sheet 1 you will practice DDL (and using the GUI (Create Table option) if you wish)
- In other assignments you will be asked to work with the DDL commands
- In exam, you will be asked for DDL commands but not any GUI questions

Therefore … it is important to know both approaches.

You try … Try adding these tables to the company database (choosing suitable data types):

These two tables keep track of products ordered by employees.

The product table contains a unique product id (the primary key of the table), name of the product, the unit price of the product and a description of the product).

The empOrder table contains the SSN of each employee who ordered a product, the ID of the product they ordered (productID) and the date they made the order. Note that ssn and productID are the primary keys, ssn is a foreign key to ssn in table employee and productID is a foreign key to id in table product:

```
product(id, name, unitPrice, description)
empOrder(ssn, productID, orderDate)
```