

Chapter 3

Relational Database

Overview

- Database & Database Management System
- Relational Database
- Simple SQL Queries
- Database normalization
- RDBMS for an Inverted Text Index

What are DB & DBMS than?

- A database (DB) is a collection of data describing the activities of 1 or more related organization, eg. University database:

- Entities: students, faculty, courses, classrooms

- Relationship between entities:

- Students' enrollment in courses

- Faculty teaching courses

- The use of rooms for courses

- A Database Management System (DBMS) is a software designed to assist in maintaining & utilizing large collection of data eg.:

- Part of software industry: Oracle, Microsoft, Sybase

- Open source:

- Relational: MySQL, PostgreSQL, SQLite

- Text search: APACHE Lucene (SOLR, HADOOP), Ferret,

Storing Data: File System vs DBMS

- Data can be stored in RAM
 - That is what most programming language offers
 - RAM is fast, random access but volatile
- File System offered by every OS:
 - Stores data in files with diverse formats in disk
 - Implication \Rightarrow program using these files depend on the knowledge about that format
 - Allows data manipulation (open, read, write, etc.)
 - Allows protection to be set on a file
 - Drawbacks:
 - No standards of format
 - Data duplication & dependence
 - No provision for concurrency & security

Storing Data: File System vs DBMS

- Database Management system:
 - Simple, efficient, ad hoc queries
 - Concurrency controls
 - Recovery, Benefits of good data modelling
 - Stores information in disks
 - This has implication for database design:
 - READ : transfer data from disk to main memory (RAM)
 - WRITE : transfer data from RAM to disk
 - In relational DBMS:
 - Information is stored as *tuples* or *records* in *relations* or *tables*.
 - Making use of relational Algebra

Relational Database

- Relational Database Management System (RDBMS) consists of:
 - A set of tables
 - A schema
- A schema:
 - is a description of data in terms of data model
 - Defines tables and their attributes (field or column)
- The central data description construct is a relation:
 - Can be thought as records
 - eg. information on student is stored in a relation with the following schema:

Student(***sid***: string, ***sname***: string, ***login***: string, ***gpa***: numeric)

Relational Database

- Tables \equiv relation:
 - is a subset of the Cartesian product of the domains of the column data type.
 - Stores information about an entity or theme
 - Consist of columns (fields) and rows (records).
 - Rows \equiv tuple, describing information about a single item, eg. A specific student
 - columns \equiv attributes, describing a single characteristic (attributes) of its item, eg. Its ID number, GPA, etc
 - Every row is unique & identified by a key
 - Entity is
 - an object in the real world that is distinguishable from other objects.
- eg. Students, lecturers, courses, rooms.
- Described using a set of attributes whose domain values must be identified.
 - The attribute 'name of Student' \Rightarrow 20-character strings

Creating Relational Database

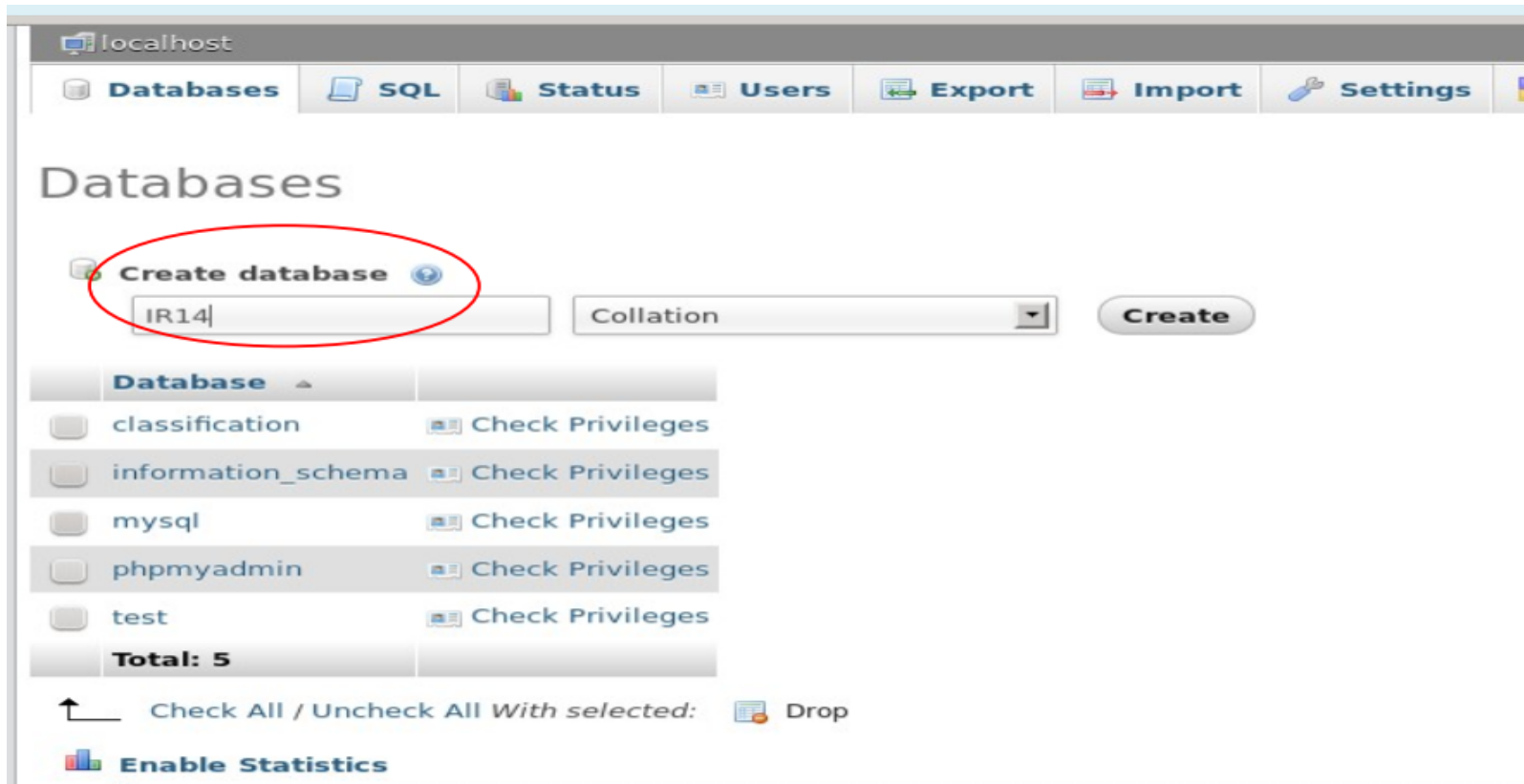
- How to create relational database?
 - Need RDBMS (MySQL, Oracle, etc)
 - Just take MySQL as an open source RDBMS
 - . With user Interface
 - . eg. phpMyAdmin → providing graphical user interface
 - Free to use any scripts or programming languages
 - Using SQL commands in terminal
 - Using SQL integrated in your code

Creating Relational Database

- How to create relational database in GUI?
 - Step 1: install XAMPP (just an example)
a cross-platform Apache HTTP Server, MySQL Server & interpreters for script
 - Step 2: start your XAMPP first:
/xampp_or_lampp_path start
eg.
/opt/lampp/lampp start
 - Open your browser, and type:
localhost/phpmyadmin

RDBMS Example


- Database Server: MySQL 5.5.27
- Web Server: Apache through XAMPP Package



RDBMS Example

Creating table, defining attributes & their domains

Table name: Add column(s)

Structure 














Name	Type 	Length/Values 	Default 	Collation	Attributes	Null	Index	A I <small>PK FK</small>	Comment
<input type="text"/>	INT 	<input type="text"/>	None 	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text"/>
<input type="text"/>	INT 	<input type="text"/>	None 	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text"/>
<input type="text"/>	INT 	<input type="text"/>	None 	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text"/>
<input type="text"/>	INT 	<input type="text"/>	None 	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text"/>	<input type="checkbox"/>	<input type="text"/>

Table comments:

Storage Engine: 

Collation:

PARTITION definition: 

RDBMS Example

- Creating table, defining attributes & their domains

The screenshot displays a database management interface. On the left, a sidebar shows a list of tables: 'Courses', 'Lecturer', and 'Students'. The 'Students' table is selected and highlighted with a red box. Below this list is a 'Create table' button. The main area shows the table structure for 'Students'. The table has four columns: 'Sid' (varchar(6)), 'SName' (varchar(35)), 'Login' (varchar(25)), and 'GPA' (float(2,1)). Each column is circled in red. The 'Sid' column is marked as 'Primary' and 'Unique'. The 'SName' column is marked as 'Primary' and 'Unique'. The 'Login' column is marked as 'Primary' and 'Unique'. The 'GPA' column is marked as 'Primary' and 'Unique'. The 'Action' column for each row contains icons for 'Change', 'Drop', 'Browse distinct values', 'Primary', 'Unique', and 'Index'. A red box highlights the 'Primary', 'Unique', and 'Index' icons for the 'Sid' column. Below the table structure, there are options to 'Check All / Uncheck All With selected:', 'Print view', 'Relation view', 'Propose table structure', and 'Track table'. The 'Relation view' option is circled in red. At the bottom, there is a '+ Indexes' button and an 'Information' tab.

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	<u>Sid</u>	varchar(6)	latin1_swedish_ci		No	None		Change Drop Browse distinct values Primary Unique Index More
2	<u>SName</u>	varchar(35)	latin1_swedish_ci		No	None		Change Drop Browse distinct values Primary Unique Index More
3	<u>Login</u>	varchar(25)	latin1_swedish_ci		No	None		Change Drop Browse distinct values Primary Unique Index More
4	<u>GPA</u>	float(2,1)			No	None		Change Drop Browse distinct values Primary Unique Index More

RDBMS Example

- Each relation is defined to be a set of unique tuples of rows

The diagram illustrates a database table structure. A light blue background on the left contains the text 'Tuples (Recods, row)' with five arrows pointing to the rows of the table. Above the table, the text 'Fields (Attributes, Columns)' has four arrows pointing to the columns. The table itself has four columns: 'Sid', 'SName', 'Login', and 'GPA'. It contains five rows of data.

Fields (Attributes, Columns)			
Sid	SName	Login	GPA
CL0001	David	david@cis	1.3
CL0002	Wenpeng	hansying@cis	1.5
CL0003	Yadoll	yalah@cs	1.7
CL0004	Bastian	basti@cis	1.3
CL0005	Dewika	krisna@cl	3.5

Tuples (Recods, row)

Key Constraints

- Key constraint is a statement that a certain minimal subset of the relation is a unique identifier for a tuple.
- Two Types of keys:
 - Primary key:
 - Foreign key
- Primary key:
 - a unique identifier for a tuple (row)
- Sid is a primary key for student,
- Cid is a primary key for Course
 - Primary key fields are indexed

Key Constraints

- Foreign key:
 - A kind of a logical pointer
 - a key to refer to relation with other tables & should match the primary key of the referenced relation.
 - Foreign key fields are also often indexed if they are important for retrieval.

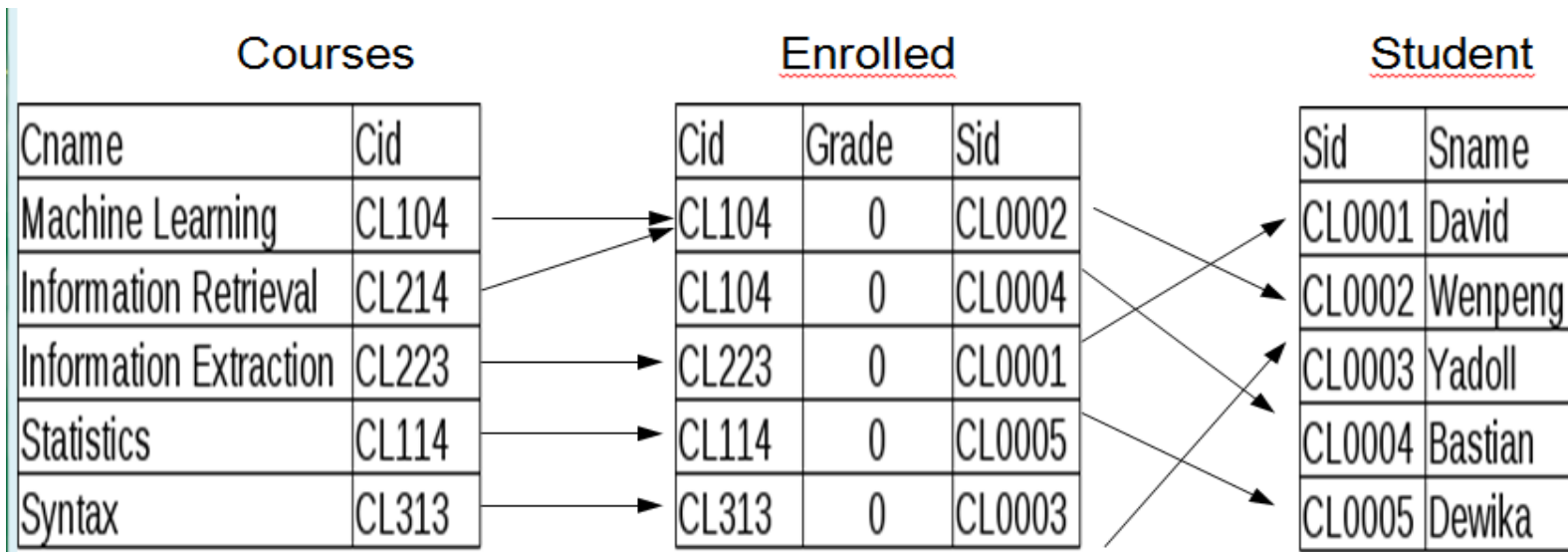
courses(Cid, Cname, Instructor, Semester)

Student(Sid, Sname, login, GPA)

How do you express which students take which course?

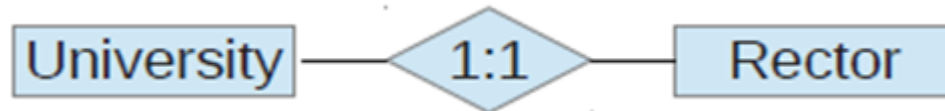
Key Constraints

- Need a new table :
 - enrolled(Cid, grade, Sid)
 - Sid/Cid in enrolled are foreign keys referring to Sid in Student table & Cid in Courses.

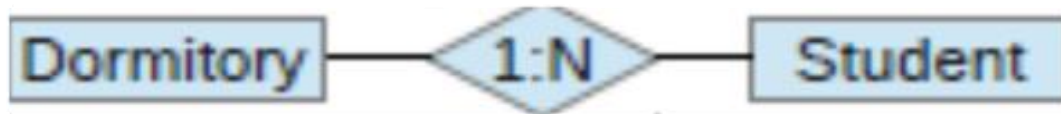


Relation

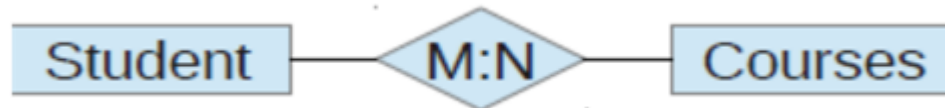
- One to one :
 - Each primary key relates only one record in related table



- One to many
 - The primary key relates to one or many records in related table



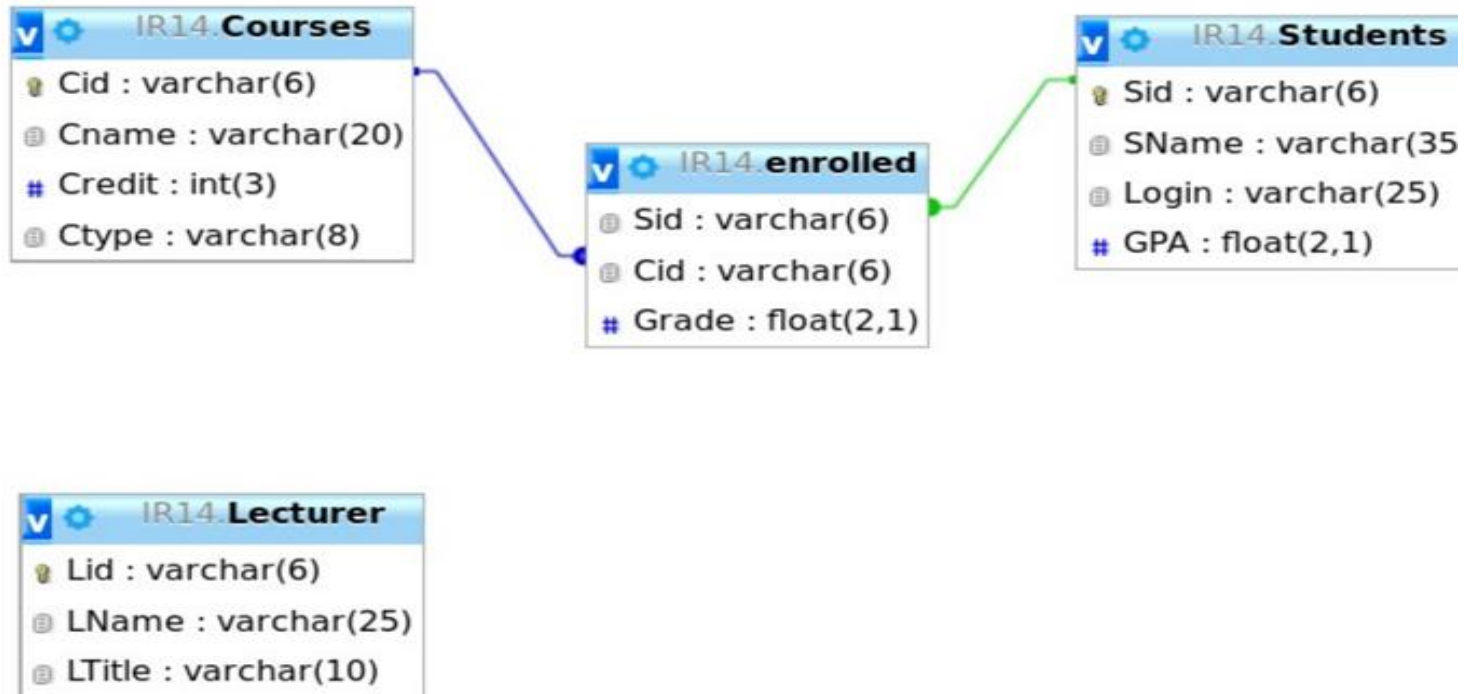
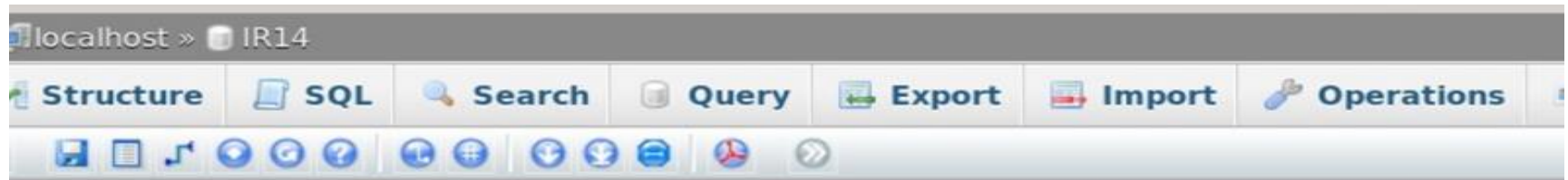
- Many to Many:
 - The primary key relates to many records in related table, and a record in related table can relate to many primary keys on another table



Storing Relationships using Keys

- Modeling data is one thing, storing it in a database is another one.
- In relational database, the 'rules' are:
 - If the relationship to be stored is 1:N, place the attribute identified as the primary key from the one table as a foreign key in another table.
 - If the relationship to be stored is M:N, a new table structure must be created to hold the association. This 'bridge' table will have as foreign key attributes, the primary key of each table that is part of relationship
- The key for the 'bridge' table then becomes either:
 - The combination of all the foreign keys OR
 - A new attribute will be added as a surrogate key

Storing Relationships using Keys



Indexes in MySQL

- A database index is
 - a data structure that improves the speed of operations in a table
 - Unseen table created by DB engine that keeps indexed fields and its pointers to each record into the actual table.
- Indexes in MySQL:
 - Primary key
 - Unique indexes:

All values in the indexed column must be distinct though it's unnecessarily indexed as a primary key
 - **Index:**

Refers to a non-unique index, used for speeding the retrieval

Indexes in MySQL

- Indexes in MySQL:
 - **Fulltext:**
 - . An index created for full text searches
 - . Supporting storage engines: InnoDB & MyISAM
 - . Data type: CHAR, VARCHAR, TEXT
 - **Spatial Index:**
 - . for spatial data types
 - . Uses R-tree indexes
- **Example of index usage:**
 - “Find all students with $GPA < 1.7$ ”
 - . May need to scan the entire table
 - . Index consists of a set of entries pointing to locations of each search key

Data Type in MySql

- String:
 - Char, varchar, text, (tiny, medium, long)
 - Binary, varbinary
 - Blob (tiny, medium, long), enum, set
- Date & time
- Numeric
 - Int (tiny, small, medium, big)
 - Decimal, float, double, real
 - BIT, boolean, serial
- Spatial:
 - Geometry, point, linestring, polygon, etc

SQL

- **Structured Query Language (SQL):**

- Is a standard language used to communicate with a relational database.

- Is used in conjunction with procedural or object-oriented languages/scripts such as Java, Perl, Ruby, Python, etc

- **Sql basic conventions:**

- Each statement begins with a command,

- eg. CREATE, SELECT

- Each statement ends with delimiter usually a semicolon (;)

- Statements are written in a free-form style,

- eg. SELECT...FROM... WHERE...

- SQL statement is not case-sensitive, except inside string constant, eg SELECT...FROM... WHERE SName = 'Yadoll'

Simple SQL Queries

- The basic form of SQL Queries is:

SELECT select-list (column_name)

FROM from-list (table_name)

WHERE condition

- Selecting all students with GPA above 1.7

SELECT Sid, Sname FROM student WHERE GPA <= 1.7

- Selecting all information from a table

SELECT * FROM enrolled

- Selecting course name with pattern matching

SELECT Cname FROM Courses WHERE Cname LIKE
"Machine %"

Simple SQL Querie

- **INSERT:**

INSERT *INTO* `Students` VALUES (CL0001, David,
david@cis, 1,3)

INSERT INTO `Students` VALUES (sid, sname, login, gpa)

- **ALTER:**

ALTER TABLE `Students` ADD `Intakeyear`

ALTER TABLE `Lecturer` ADD INDEX(`courses`)

Simple SQL Querie

- **Using logical connectives:**

- AND, OR, NOT may be used to construct a condition

```
SELECT `cname` FROM `courses` WHERE  
semester = 'summer' AND ctype = 'seminar'
```

- **Joining tables:**

- SELECT 'Sname'

- FROM 'Students' 'Courses'

- WHERE Student .sid=Courses.sid

Simple SQL Queries

- Creating Table:

```
CREATE TABLE `Students` (  
  `Sid` varchar(6) NOT NULL,  
  `SName` varchar(35) NOT NULL,  
  `Login` varchar(25) NOT NULL,  
  `GPA` float(2,1) NOT NULL,  
  PRIMARY KEY (`Sid`)  
  ENGINE=InnoDB CHARSET= Latin1
```

Creating Database Through Terminal

Creating Database Through Terminal

- Open your terminal console
- Go to the path where you save your MySql
- If you install XAMPP :
 - You need to start XAMPP as a SU/root
 - to get the action commands (in Linux), type:
`/opt/lampp/lampp`
 - Start only MySQL Server, type:
`/opt/lampp/lampp startmysql`
 - To stop MySQL, type:
`/opt/lampp/lampp stopmysql`
 - To start XAMPP (Apache, MySQL & others), type:
`/opt/lampp/lampp star`

Creating Database Through Terminal

- **If you install XAMPP :**

- go to the path where mysql is saved, in Linux it is usually saved in bin, so type:

- `/opt/lampp/bin/mysql -uusername -ppassword`

- If you are already in mysql path:

- **To see the databases. Type:**

- `SHOW DATABASES ;`

- **To create a database, use SQL command:**

- `CREATE DATABASE database_name ;`

- **Creating database does not select it for use, so type:**

- `USE database_name ;`

- **To delete database:**

- `DROP DATABASE database_name ;`

- **Use SQL commands to create tables, do table operation, etc**

Creating Database Through Terminal

```
+-----+
1 row in set (0.00 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| IR14 |
| cdcol |
| classification |
| mysql |
| performance_schema |
| phpmyadmin |
| test |
+-----+
8 rows in set (0.00 sec)

mysql> create database information_retrieval
-> ;
Query OK, 1 row affected (0.00 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| IR14 |
| cdcol |
| classification |
| information_retrieval |
| mysql |
| performance_schema |
| phpmyadmin |
| test |
+-----+
9 rows in set (0.00 sec)
```

Database Normalization

- Functional dependencies:
 - Require that the value for a certain set of attributes determines uniquely the value for another set of attributes
 - are akin to a generalization of the notion of a key
 - Let R be a relation and
 $\alpha \subseteq R$ and $\beta \subseteq R$

The functional dependency :

$$\alpha \rightarrow \beta$$

holds on R and only if for any tuples t_1 & t_2 that agree on the attributes α , they also agree on the attributes β .

- That is, $t_1[\alpha] = t_2[\alpha] \rightarrow t_1[\beta] = t_2[\beta]$

Database Normalization

- Functional dependencies

Example: consider student(Sid, Sname, DeptId)
instance of student

<u>Sid</u>	<u>Sname</u>	<u>DeptId</u>	Is this true?	Yes	No
CL12001	JOHN	13	$Sid \rightarrow Sname$		
CL13050	WENPENG	13	$Sid \rightarrow DeptId$		
DE10003	ALDI	15	$Sname \rightarrow DeptId$		
PS11123	ILJA	11	$Sname \rightarrow Sid$		
IT09256	LISANDRO	09	$DeptId \rightarrow Sname$		
CL13075	MATTHEW	13	$DeptId \rightarrow Sid$		

Database Normalization


- Functional dependencies

Example: consider student(Sid, Sname, DeptId)
instance of student.

<u>Sid</u>	<u>Sname</u>	<u>DeptId</u>	Is this true?	Yes	No
CL12001	JOHN	13	$Sid \rightarrow Sname$	✓	
CL13050	WENPENG	13	$Sid \rightarrow DeptId$	✓	
DE10003	ALDI	15	$Sname \rightarrow DeptId$		✓
PS11123	ILJA	11	$Sname \rightarrow Sid$		✓
IT09256	LISANDRO	09	$DeptId \rightarrow Sname$		✓
CL13075	MATTHEW	13	$DeptId \rightarrow Sid$		✓

Database Normalization

- examine the following poor database design:



	Sid	Cname	time	room	Lid
Edit Copy Delete	CL0001	Machine Learning	Wed 10.15	L155	PR145
Edit Copy Delete	CL0002	Information Retrieval	Tue 12.15	C131	PD220
Edit Copy Delete	CL0003	Machine Learning	Wed 10.15	L155	PR145
Edit Copy Delete	CL0004	Information Extraction	Thu 10.00	C149	PR111

- Problems:
 - No need to repeatedly store the class time & Professor ID
 - Which one is the key?

Database Normalization

- **First Normal Form (1NF):**

- A row of data cannot contain a repeating group of data.
- Each row of data must have a unique identifier, i.e primary key

- **This can be done by**

- Eliminating the repeated groups of data through creating separate tables of related data
- Identify each set of related data with a primary key
- All attributes are single valued (1 data type) & non-repeating

- **Student information:**

Sid Sname Major Minor IntakeYear

- **Course information**

Cid Cname Lid Time Room

- **Lecturer Information**

Lid Lname Ltitle

Database Normalization

- **Second Normal form (2NF):**

- A table should meet 1NF
- There must not be any partial dependency of any column on primary key (Records should not depend on anything other than a table's primary key)

- Recall our poor database design:

Sid → Cname or Cname → time ?



	Sid	Cname	time	room	Lid
<input type="checkbox"/> Edit Copy Delete	CL0001	Machine Learning	Wed 10.15	L155	PR145
<input type="checkbox"/> Edit Copy Delete	CL0002	Information Retrieval	Tue 12.15	C131	PD220
<input type="checkbox"/> Edit Copy Delete	CL0003	Machine Learning	Wed 10.15	L155	PR145
<input type="checkbox"/> Edit Copy Delete	CL0004	Information Extraction	Thu 10.00	C149	PR111

Database Normalization

- Second Normal Form (2NF) solution:
 - **Create** separate tables for sets of values that apply to multiple records
 - **Relates** the tables with a **foreign key**
 - **Remove** subsets of data that apply to multiple rows of a table and **place** them in separate tables enrolled

Sid Cid grade (?)

- What do we do with the attribute *time*, *room*, & *Lid*?

Database Normalization

- Third Normal Form (3NF):
 - Eliminate all attributes (columns) that do not directly dependent upon the primary key
 - Each non-primary key attribute must be dependent only on primary key (no transitive dependency)
- Example:

Student:

Sid Sname Major Minor IntakeYear

- which attribute is not directly dependent on sid?

Student:

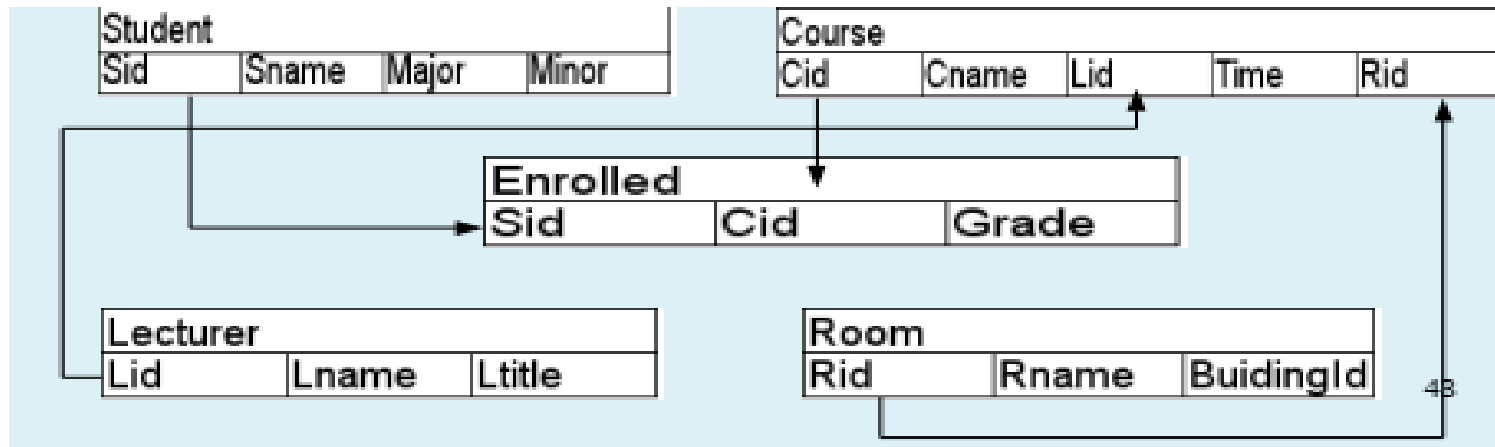
Sid Sname Minor Major

Database Normalization

Old design

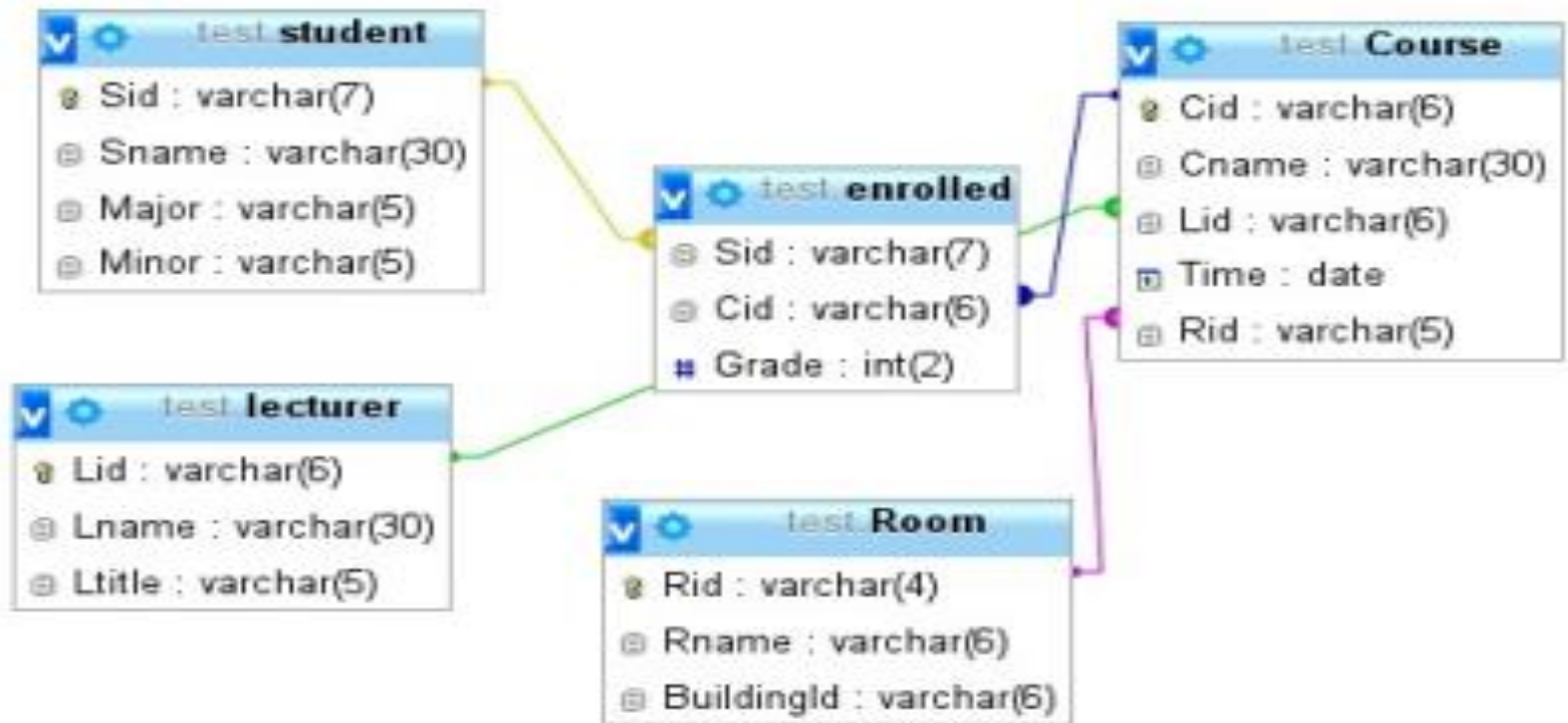
	Sid	Cname	time	room	Lid
Edit Copy Delete	CL0001	Machine Learning	Wed 10.15	L155	PR145
Edit Copy Delete	CL0002	Information Retrieval	Tue 12.15	C131	PD220
Edit Copy Delete	CL0003	Machine Learning	Wed 10.15	L155	PR145
Edit Copy Delete	CL0004	Information Extraction	Thu 10.00	C149	PR111

New design



Database Normalization

- Storing the relation among tables in database



Database Normalization

- **Exercise:**

- Which normal form does this table violate?
- And how do you normalize it?

Person	Title	Author	Pages	Year
Yakup	Database Management System	Ramakhrisnan, Raghu	903	2010
Wenpeng	Beyond Human-Computer Interaction	Preece, Jennifer	889	2009
Amy	Support Your Local Wizard	Duane, Diane	473	1990
Dwika	The Hobbit	Tolkien, JRR	389	1995
Yadoll	Beyond Human-Computer Interaction	Preece, Jennifer	889	2009
Irina	Support Your Local Wizard	Duane, Diane	473	1990