Chapter 3, The Relational Model

Introduction to Relational Model

- Codd proposed the relational data model in 1970.
  - Prior to that, database systems were based on older data models (the and the network model); hierarchical model he relational model revolutionized the database field and largely supplanted these earlier models
  - Main idea was to organize data as groups of relations
  - Each relation describes a group of objects with similar attributes

Student ID	Name	Major
1161234	Ahmad	ENCS
1161455	Noor	COMP

Course ID	CODE	Name
56478	COMP333	Database management Systems
56479	COMP232	Data Structures

Relational data model example

Students(*sid:* `string`, *name:* `string`, *login:* `string`, *age:* `integer`, *gpa:* `real`)

The preceding schema says that each record in the Students relation has five fields, with field names and types as indicated.[2] An example instance of the Students relation

sid	*name*	*login*	*age*	*gpa*
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Keys

- A **super key** of an entity set is a set of one or more attributes whose values uniquely determine each entity.
- A candidate key of an entity set is a minimal super key
  - ID is **candidate key** of instructor
  - course_id is candidate key of course
- Although several candidate keys may exist, one of the candidate keys is selected to be the **primary key.**

Simplicity

- The relational model is very simple and elegant; a database is a collection of one or more relations, where each relation is a table with rows and columns.

- A DBMS permits the use of SQL to query, and manipulate data and relations in a database.

SQL

- DBMS Supports **Structured Query Language.**
  - Based on Relational Algebra

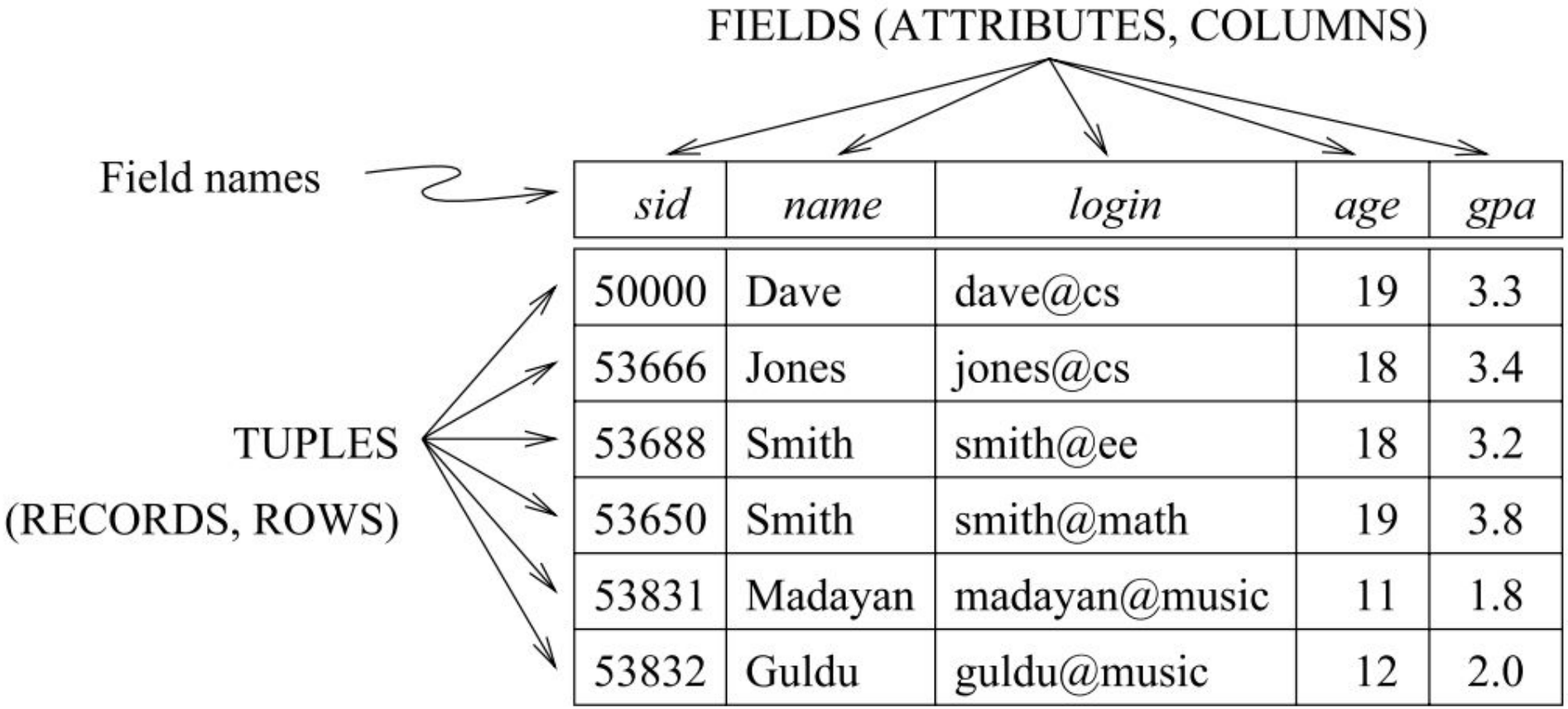
- Composed of
  - DDL
  - DML

Main Constructs

- The main construct in relational model is Relation
- A Relation consist of:
  - Schema
  - Instance
- There should be no redundant data (rows) inside a database
- Degree: number of fields (attributes)
- Cardinality: number of records (tuples)

Example:

Students(*sid:* `string`, *name:* `string`, *login:* `string`, *age:* `integer`, *gpa:* `real`)



sid	*name*	*login*	*age*	*gpa*
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Example Instance of Students Relation

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

- Cardinality = 3, degree = 5, all rows distinct
- Do all columns in a relation instance have to be distinct?

Example SQL

```
CREATE TABLE Students ( sid     CHAR(20),
                        name    CHAR(30),
                        login   CHAR(20),
                        age     INTEGER,
                        gpa     REAL )
```

```
INSERT
INTO    Students   (sid, name, login, age, gpa)
VALUES  (53688, 'Smith', 'smith@ee', 18, 3.2)
```

Example SQL..2

```
DELETE
FROM    Students S
WHERE   S.name = 'Smith'
```

```
UPDATE  Students S
SET     S.age = S.age + 1, S.gpa = S.gpa - 1
WHERE   S.sid = 53688
```

mySQL

- We will be using mySql server
  - Download from
  - https://dev.mysql.com/downloads/mysql/
- Must install a client to connect to server
  - Best: mySql WorkBench

MySQL :: Download MyS

Secure | https://dev.mysql.com/downloads/mysql/

Weather | M&N | edu | Misc | Apps | Ritaj | Plex | Selenium | CRM | Antalya | Shopping | QA | Estimates | Shopping Cart | Swar

Enterprise **Community** Yum Repository APT Repository SUSE Repository Windows Archives MyS

Thank you for your support!

Generally Available (GA) Releases

MySQL Community Server 8.0.12

Select Operating System:

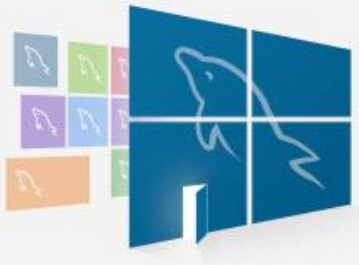
Microsoft Windows

Recommended Download:

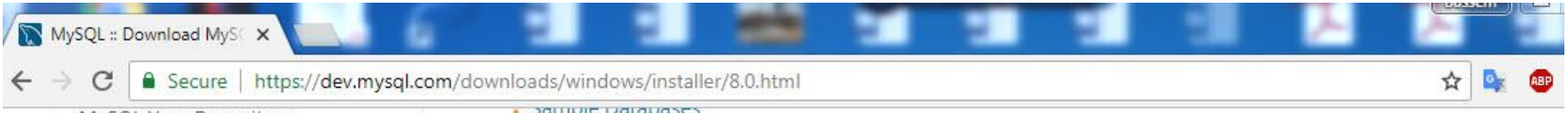
MySQL Installer for Windows

All MySQL Products. For All Windows Platforms. In One Package.

Starting with MySQL 5.6 the MySQL Installer package replaces the standalone MSI packages.



Windows (x86, 32 & 64-bit), MySQL Installer MSI **Go to Download Page >**



- MySQL Yum Repository
- MySQL APT Repository
- MySQL SUSE Repository
- MySQL Community Server
- MySQL Cluster
- MySQL Router
- MySQL Shell
- MySQL Workbench
- MySQL Connectors
- Other Downloads

- Sample Databases

Choosing the right file:

- If you have an online connection while running the MySQL Installer, choose the `mysql-installer-web-community` file.
- If you do NOT have an online connection while running the MySQL Installer, choose the `mysql-installer-community` file.

Note: MySQL Installer is 32 bit, but will install both 32 bit and 64 bit binaries.

Online Documentation

- MySQL Installer Documentation and Change History

Please report any bugs or inconsistencies you observe to our Bugs Database.
Thank you for your support!

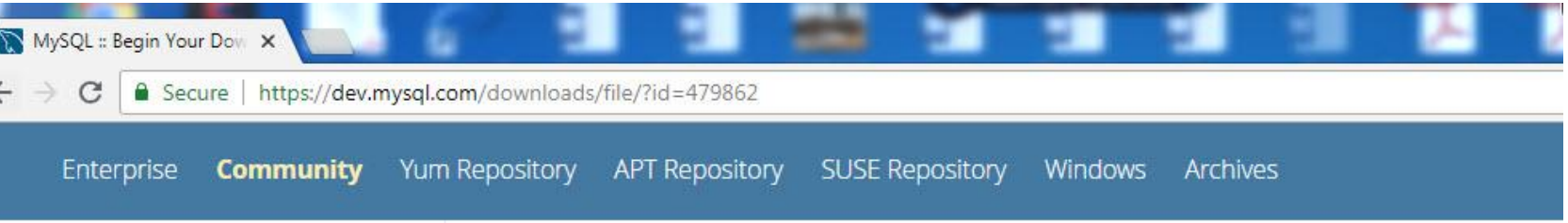
Generally Available (GA) Releases

MySQL Installer 8.0.12

Select Operating System: Microsoft Windows

Looking for previous GA versions?

Windows (x86, 32-bit), MSI Installer	8.0.12	15.9M	Download
(mysql-installer-web-community-8.0.12.0.msi)	MD5: 387bd57f0fb07e3880d10f0c21b81686 | Signature		
Windows (x86, 32-bit), MSI Installer	8.0.12	273.4M	Download
(mysql-installer-community-8.0.12.0.msi)	MD5: 53b3a9bb89db061862969b67c68b6667 | Signature		



- MySQL on Windows
- MySQL Yum Repository
- MySQL APT Repository
- MySQL SUSE Repository
- MySQL Community Server
- MySQL Cluster
- MySQL Router
- MySQL Shell
- MySQL Workbench
- MySQL Connectors
- Other Downloads

Begin Your Download

mysql-installer-community-8.0.12.0.msi

Login Now or Sign Up for a free account.

An Oracle Web Account provides you with the following advantages:

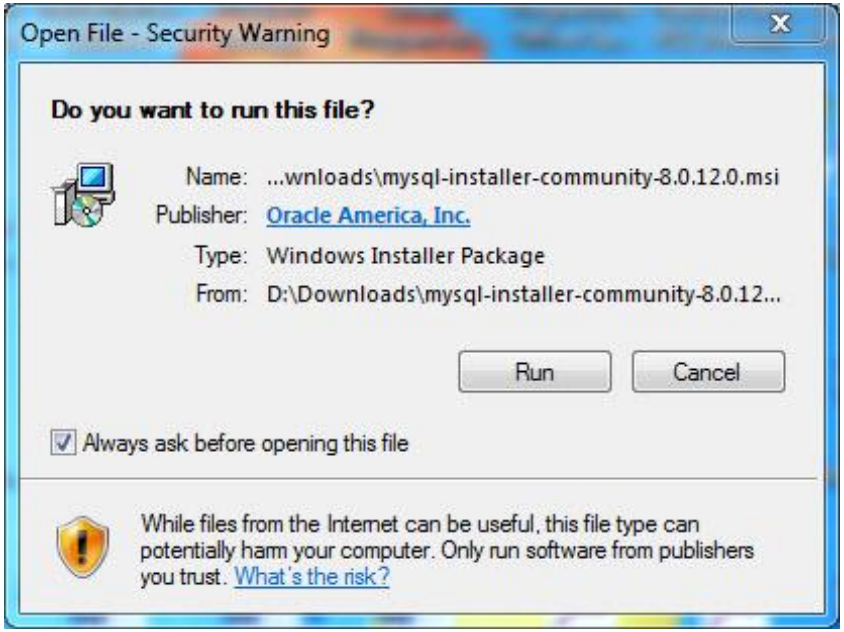
- Fast access to MySQL software downloads
- Download technical White Papers and Presentations
- Post messages in the MySQL Discussion Forums
- Report and track bugs in the MySQL bug system
- Comment in the MySQL Documentation

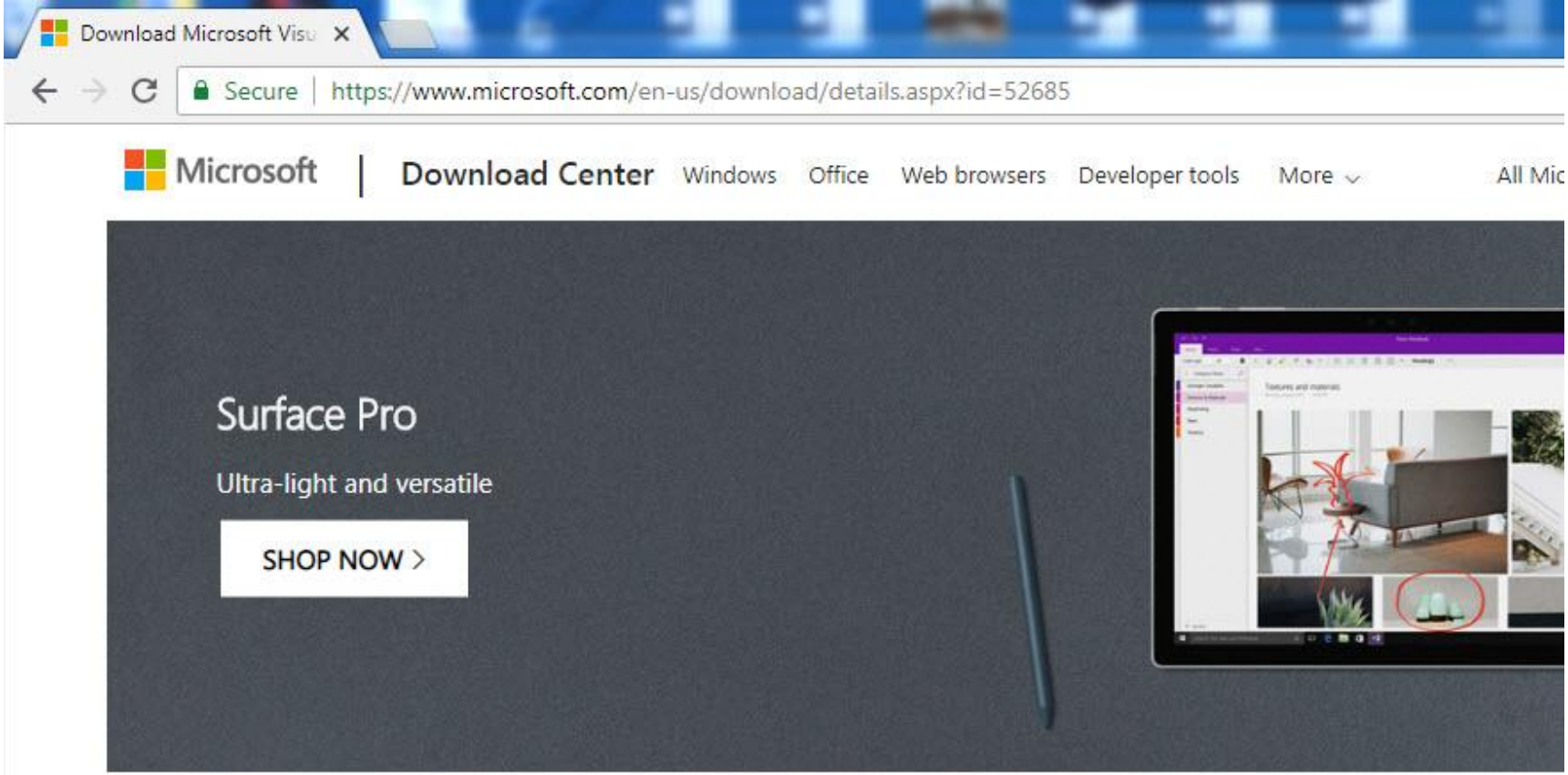
Login »
using my Oracle Web account

Sign Up »
for an Oracle Web account

MySQL.com is using Oracle SSO for authentication. If you already have an Oracle Web account, click the Login link. Otherwise, you can signup for a free account by clicking the Sign Up link and following the instructions.

No thanks, just start my download.



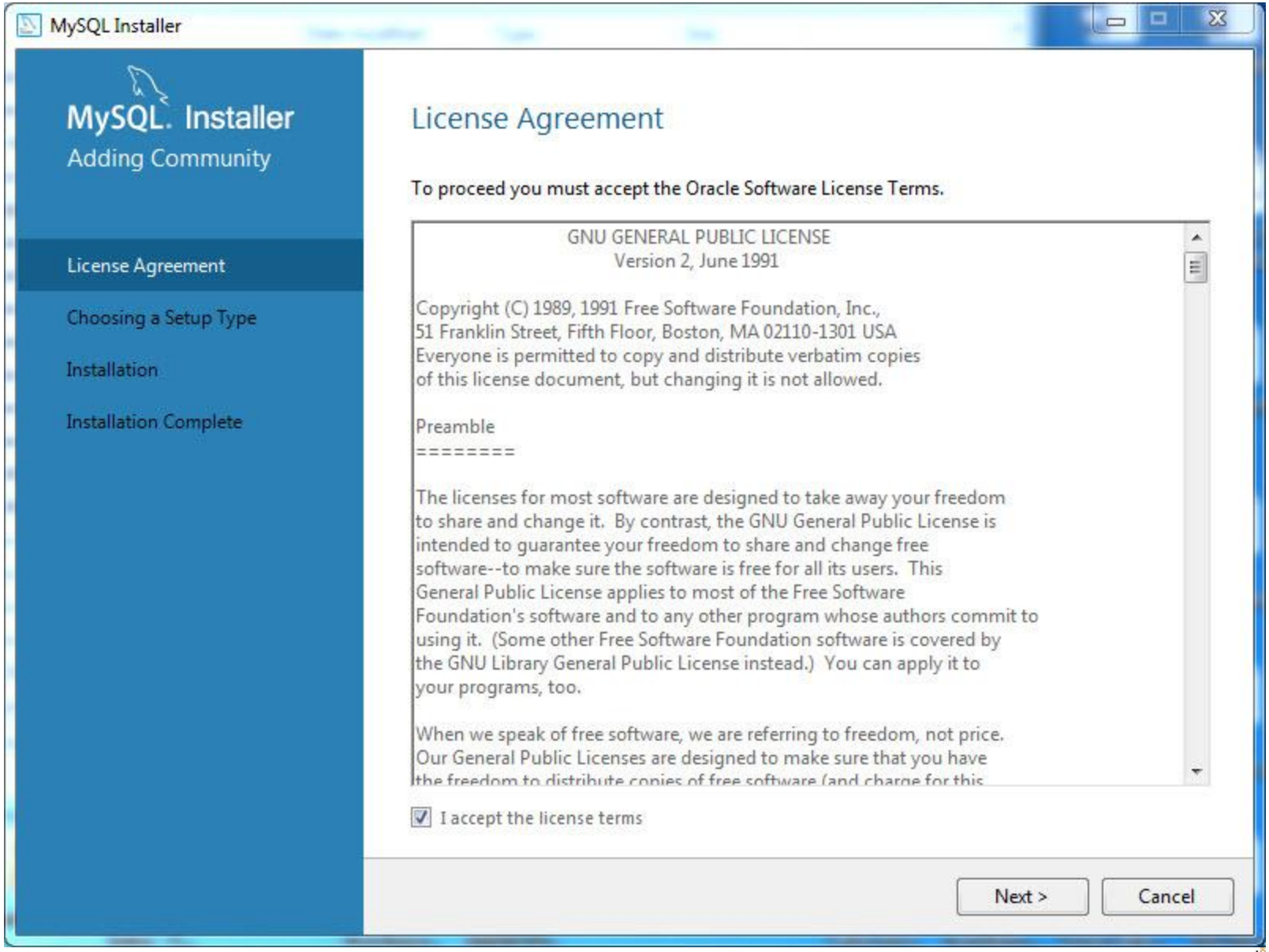


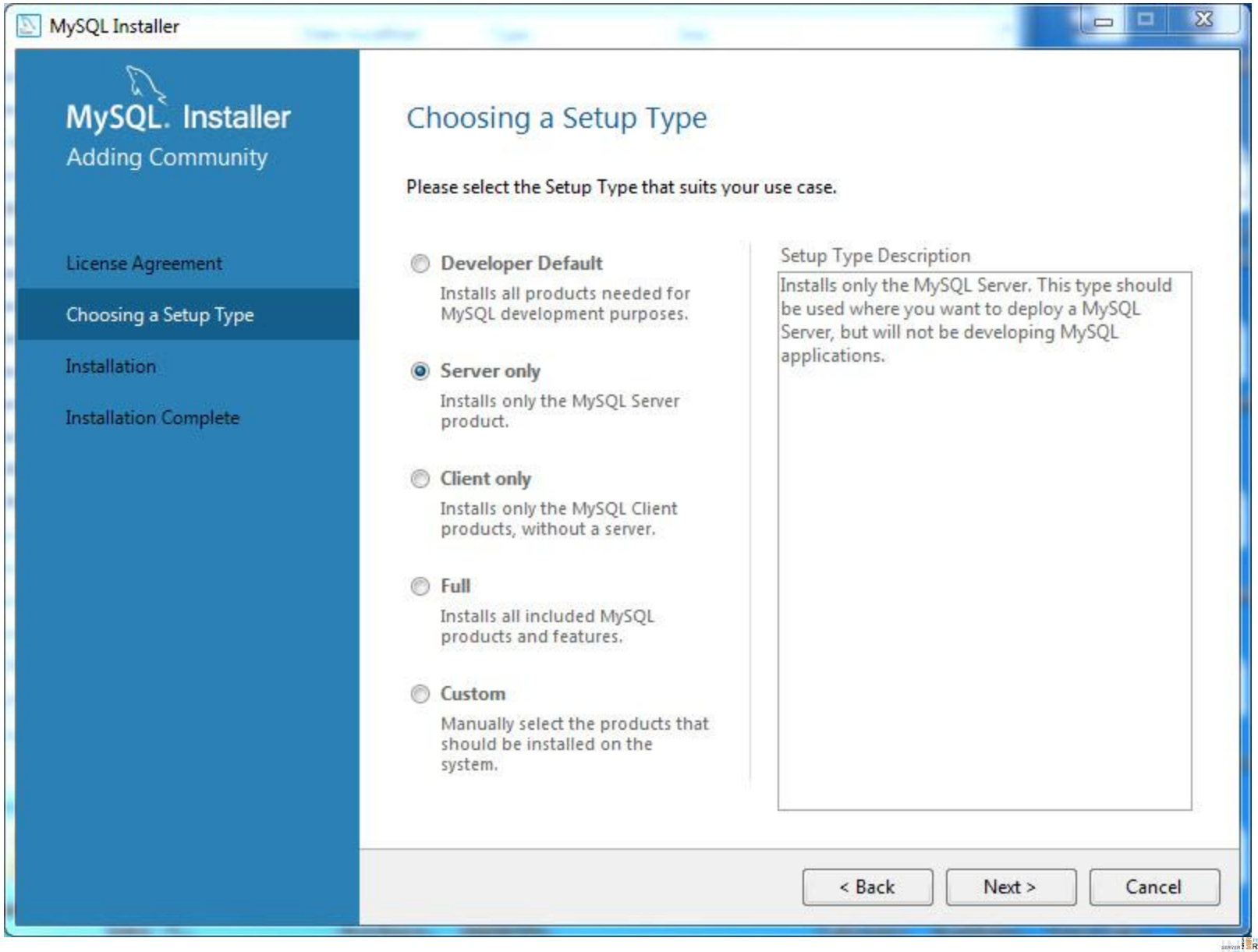
Microsoft Visual C++ 2015 Redistributable Update 3 RC

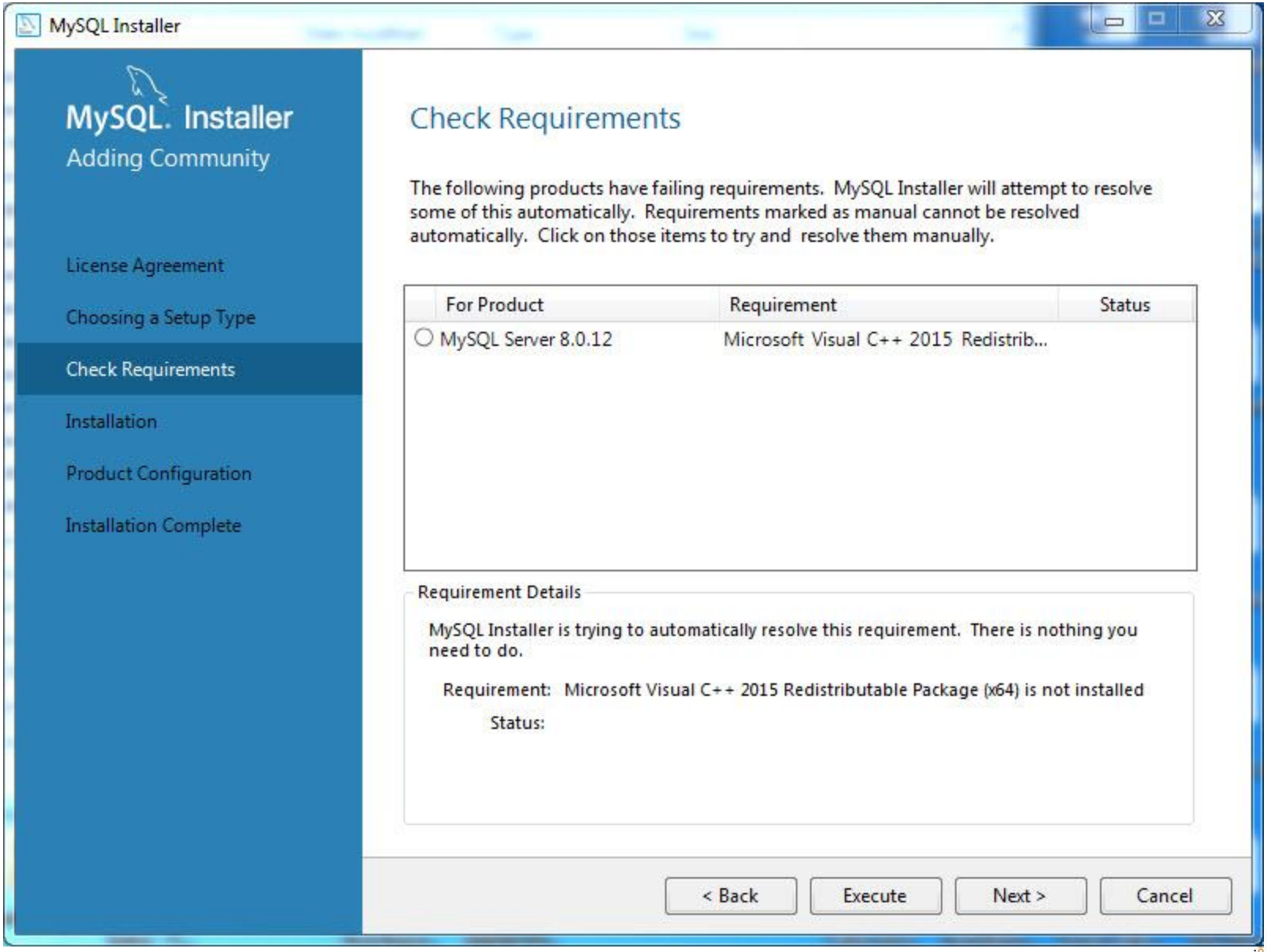
Important! Selecting a language below will dynamically change the complete page content to that language.

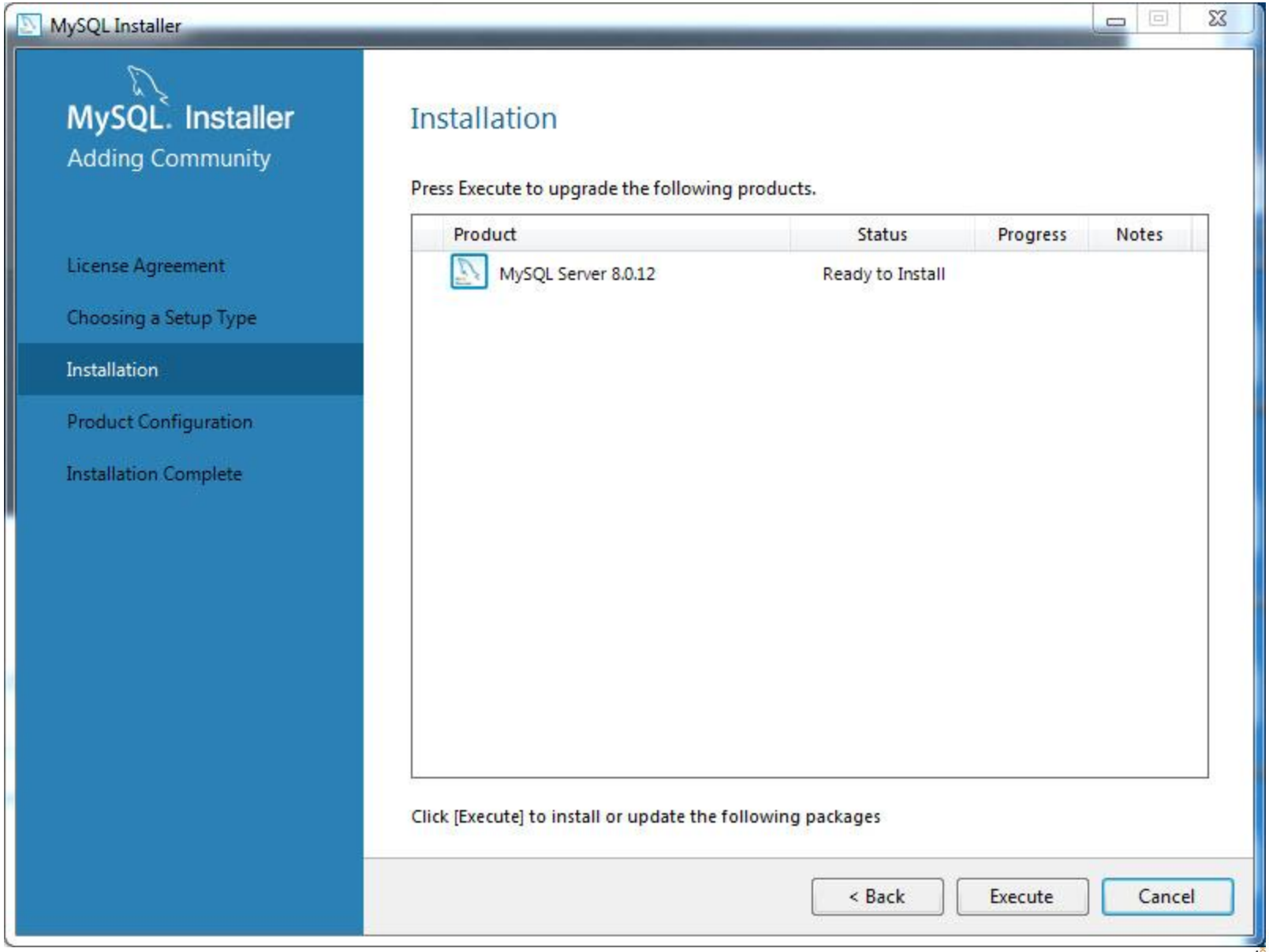
Select Language: English

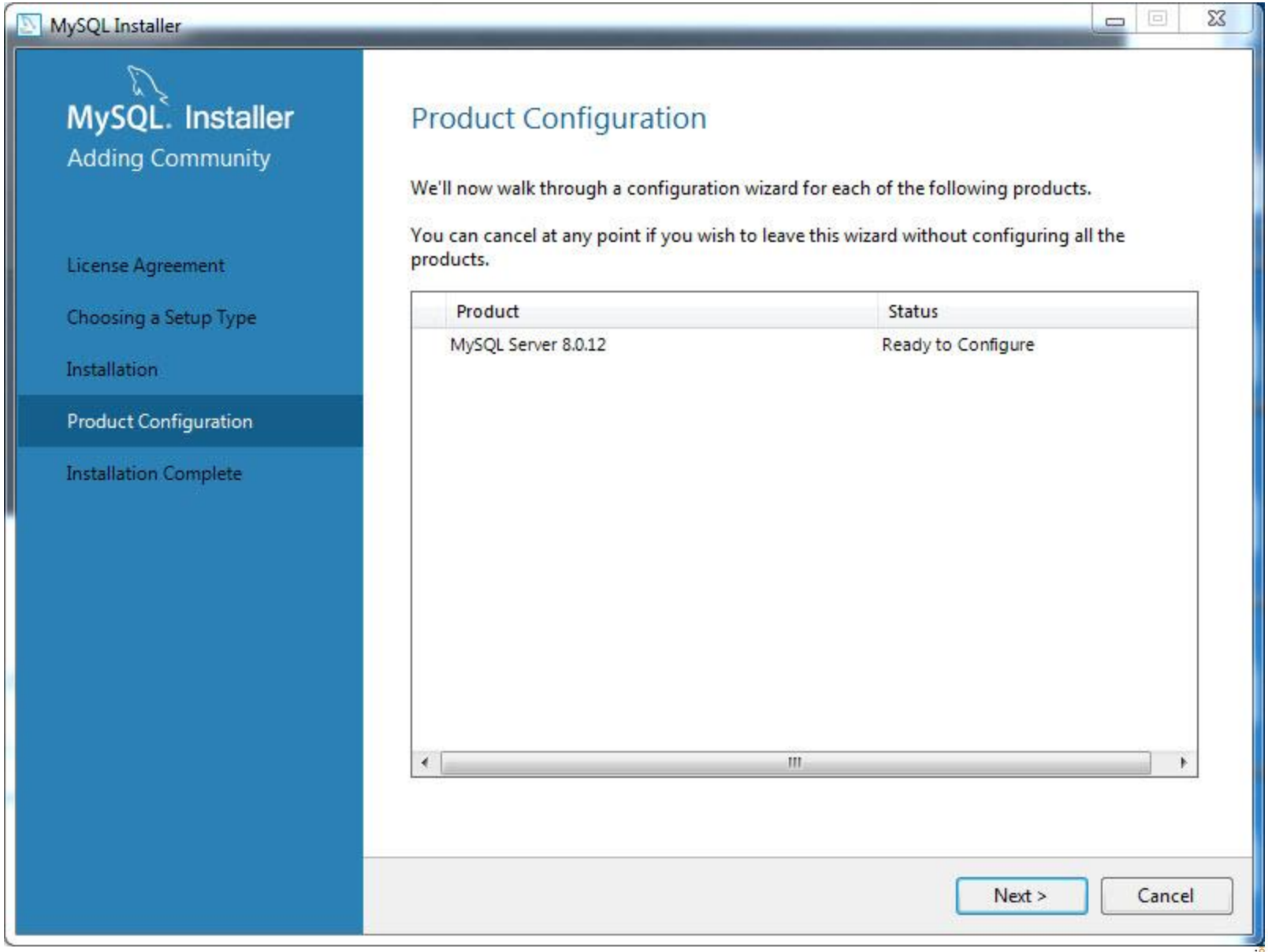
Download

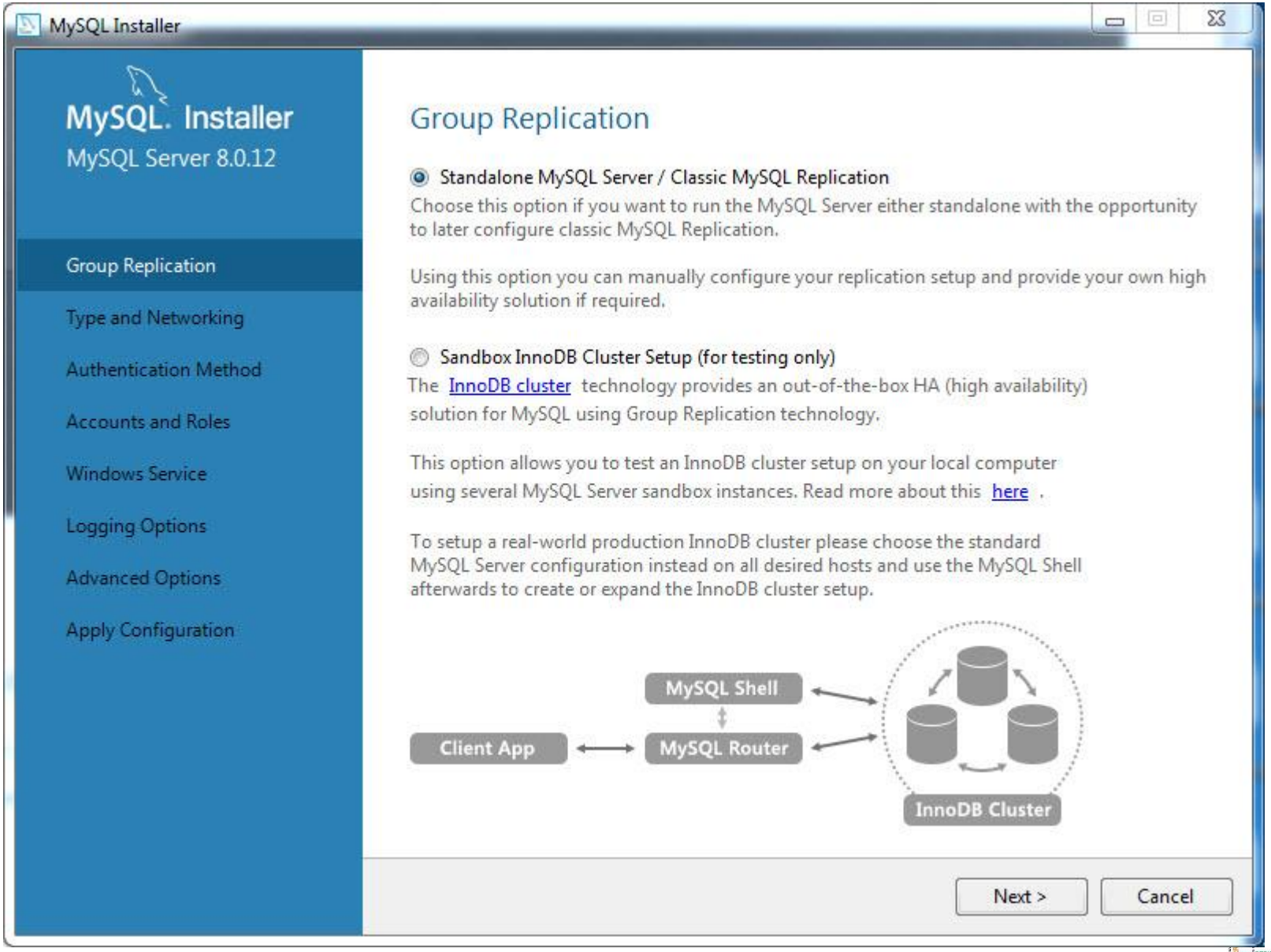


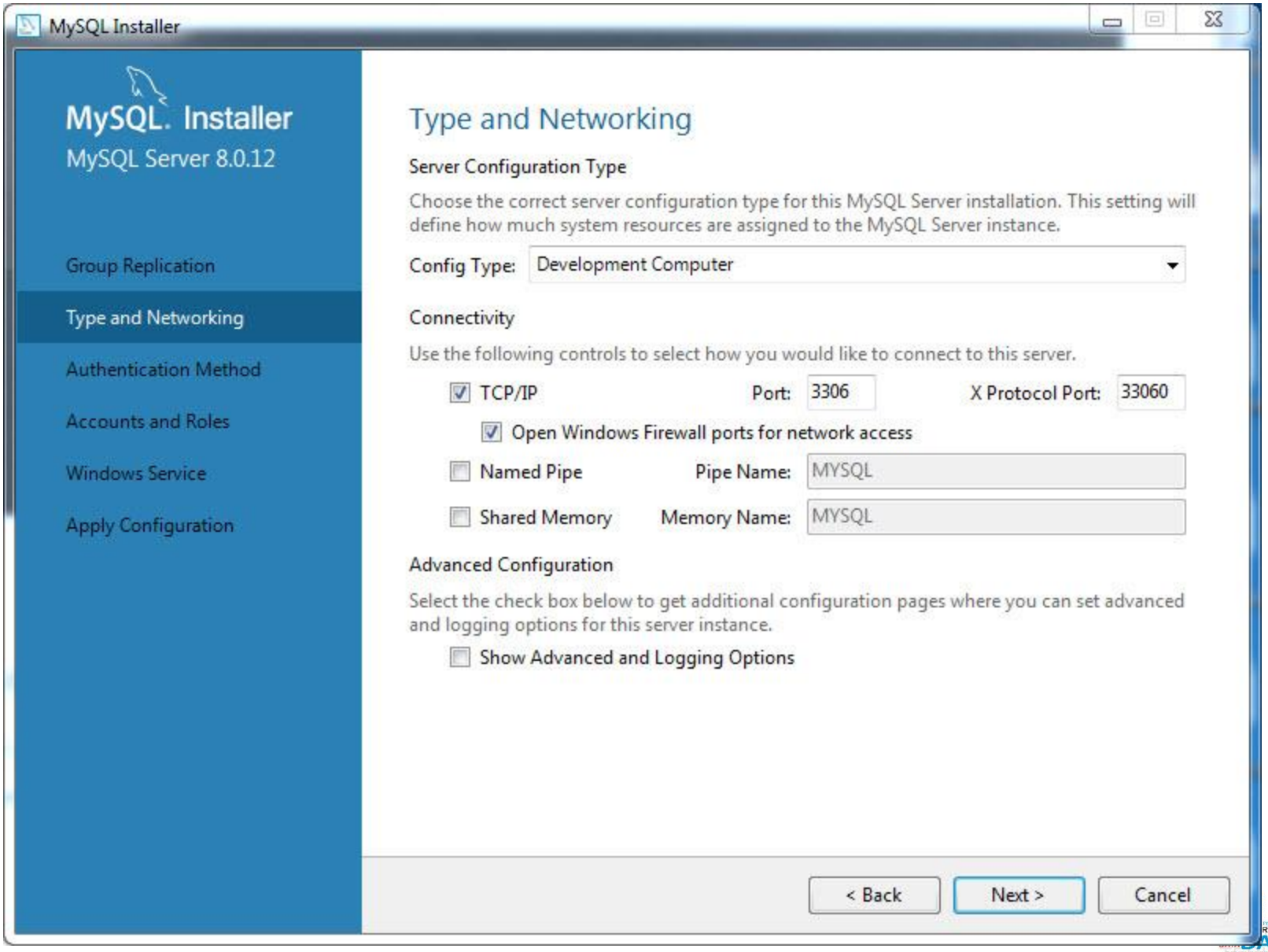


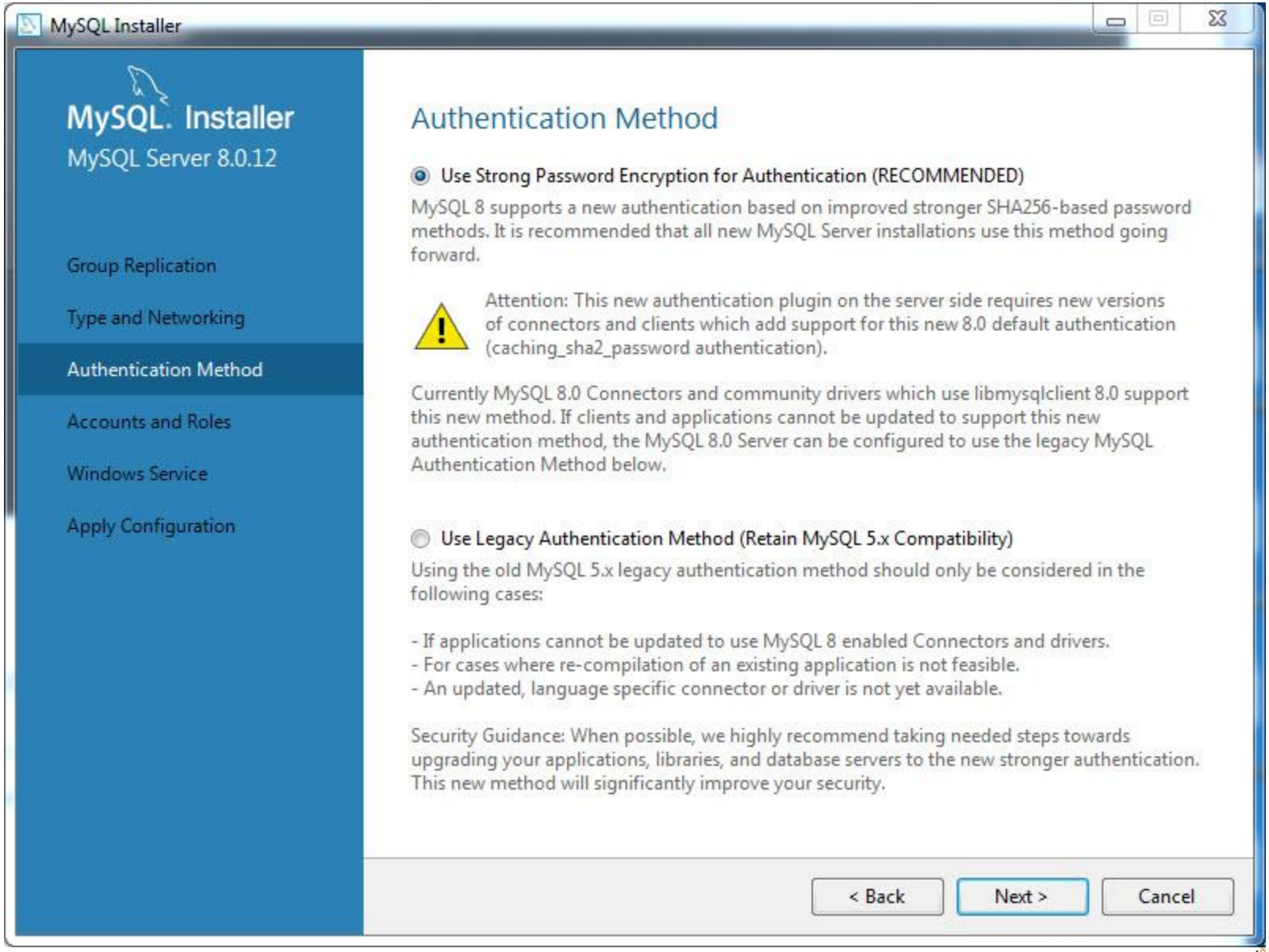


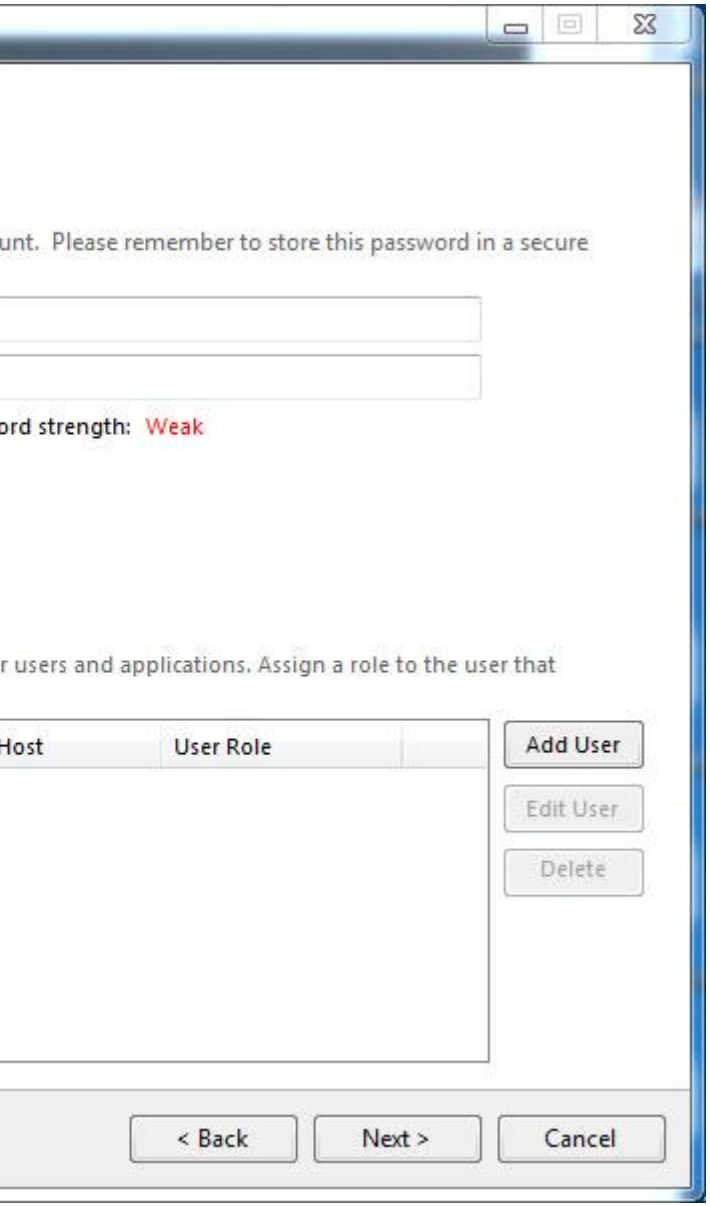


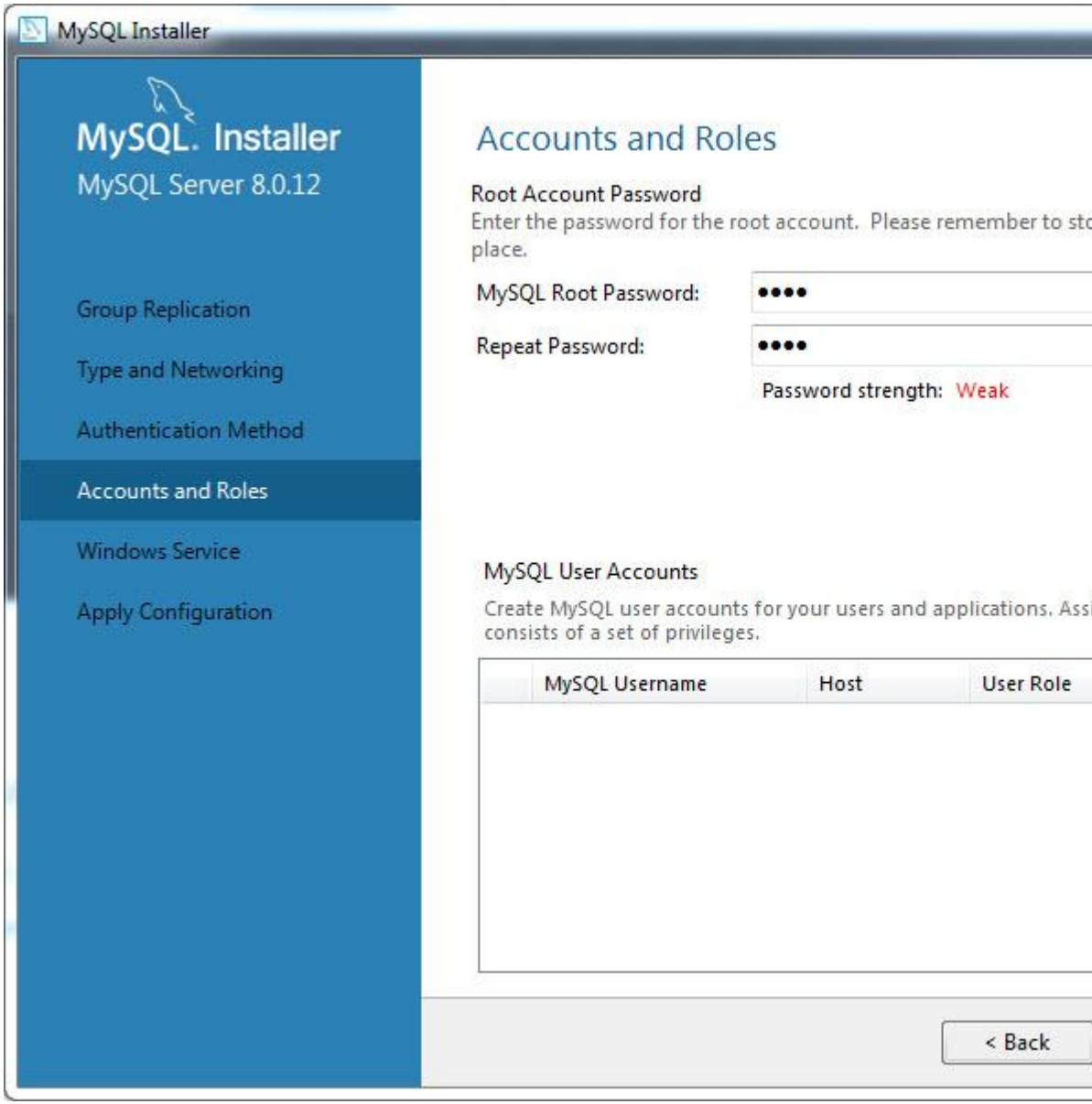


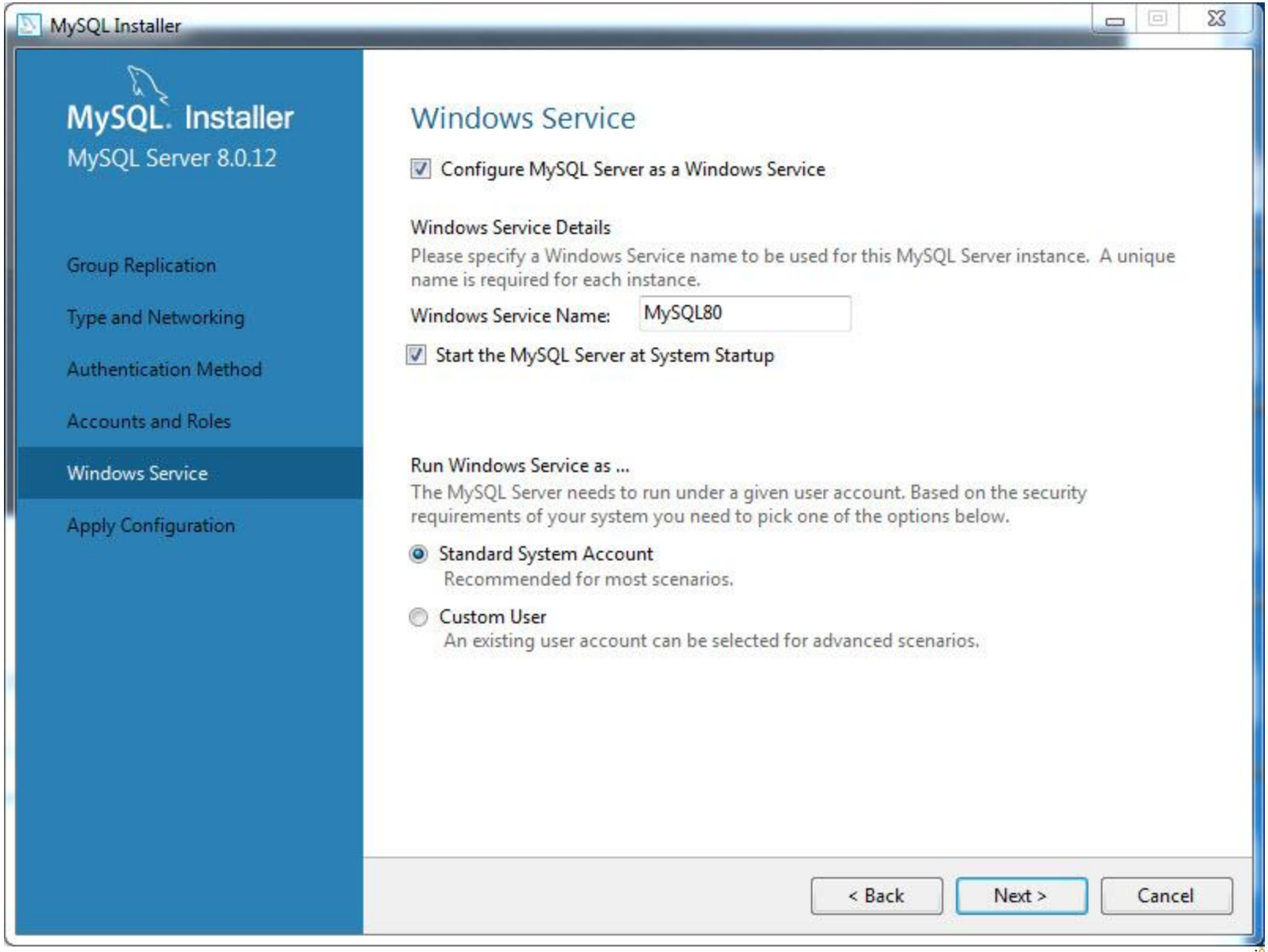


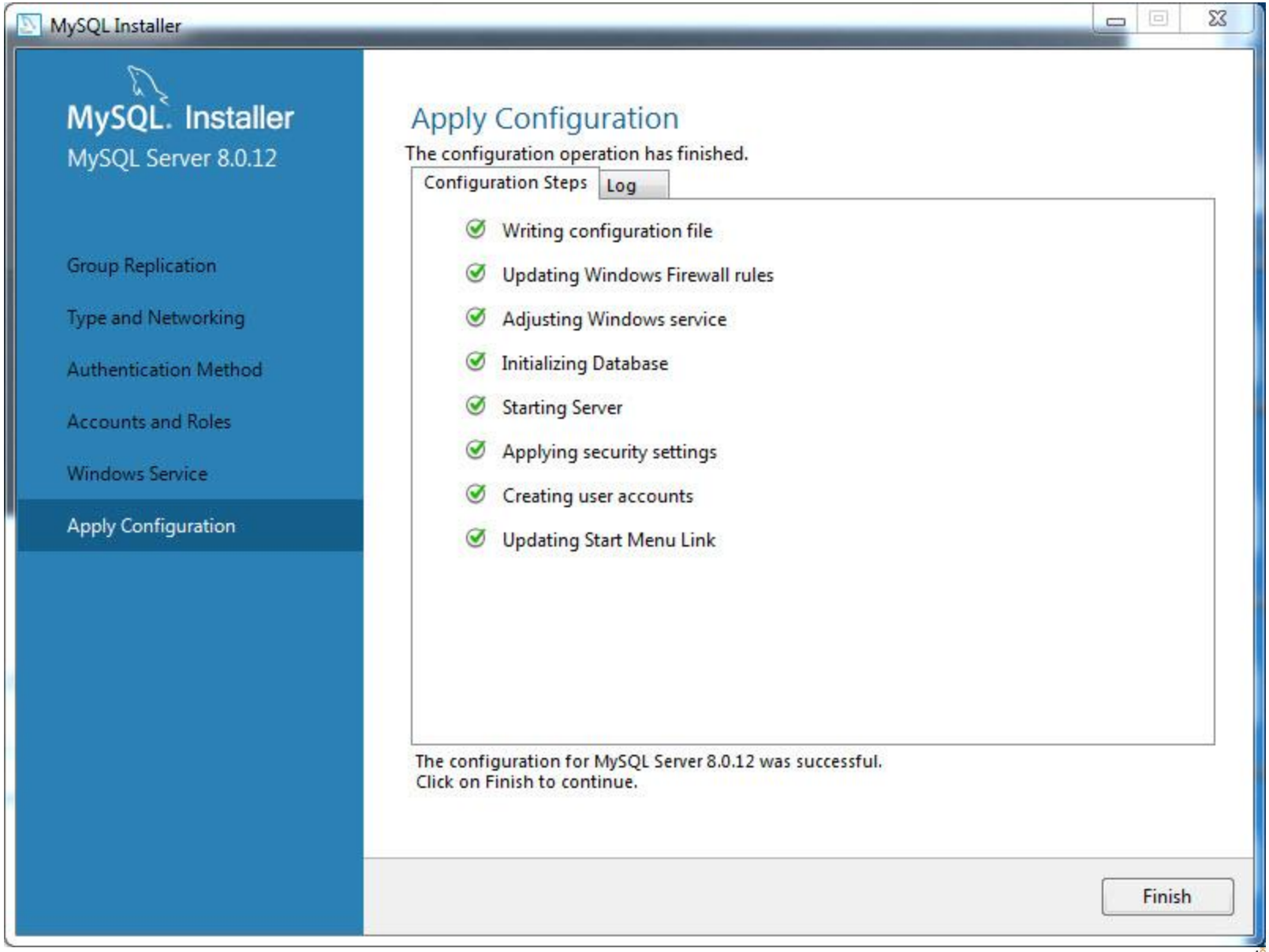


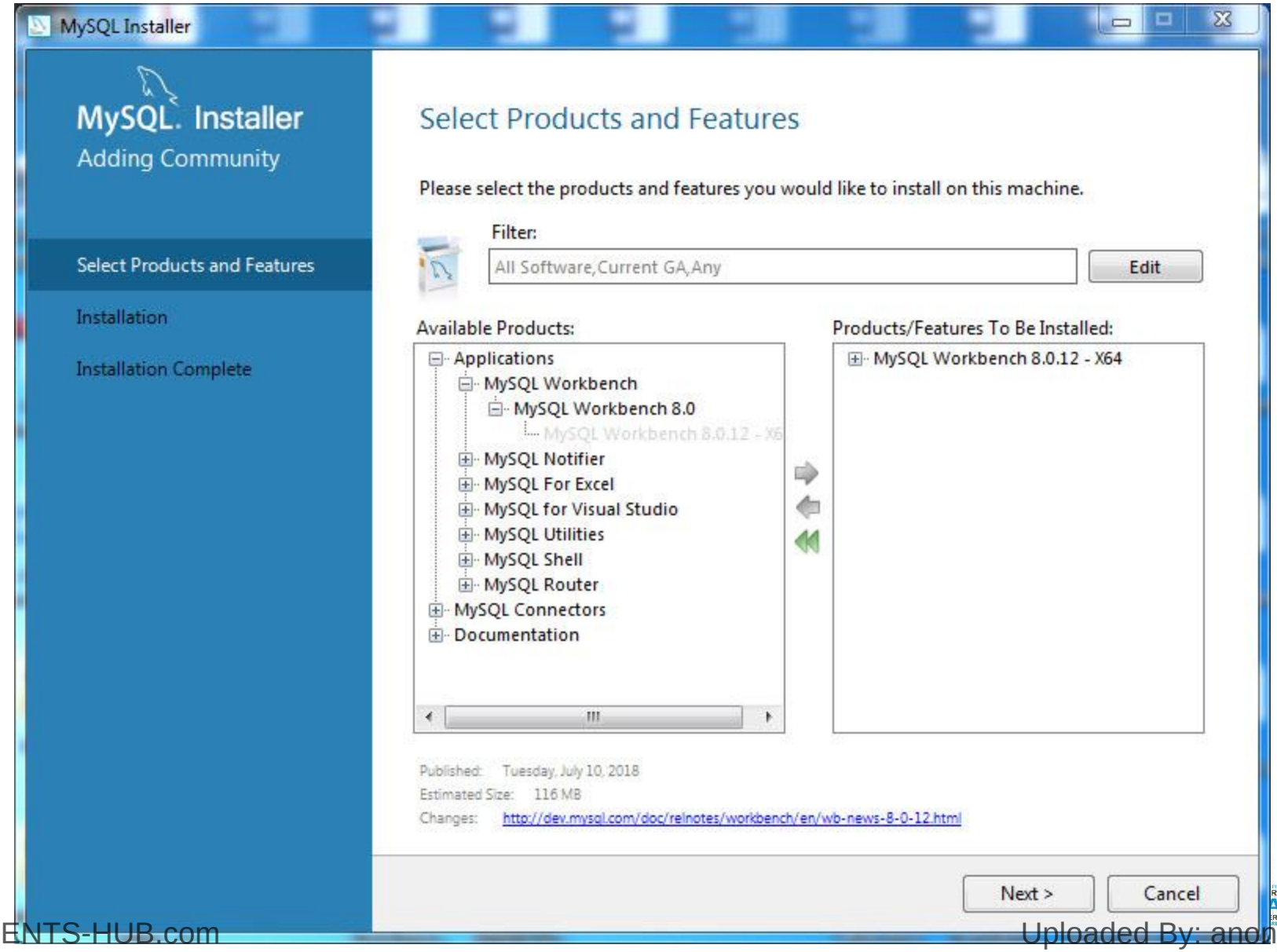


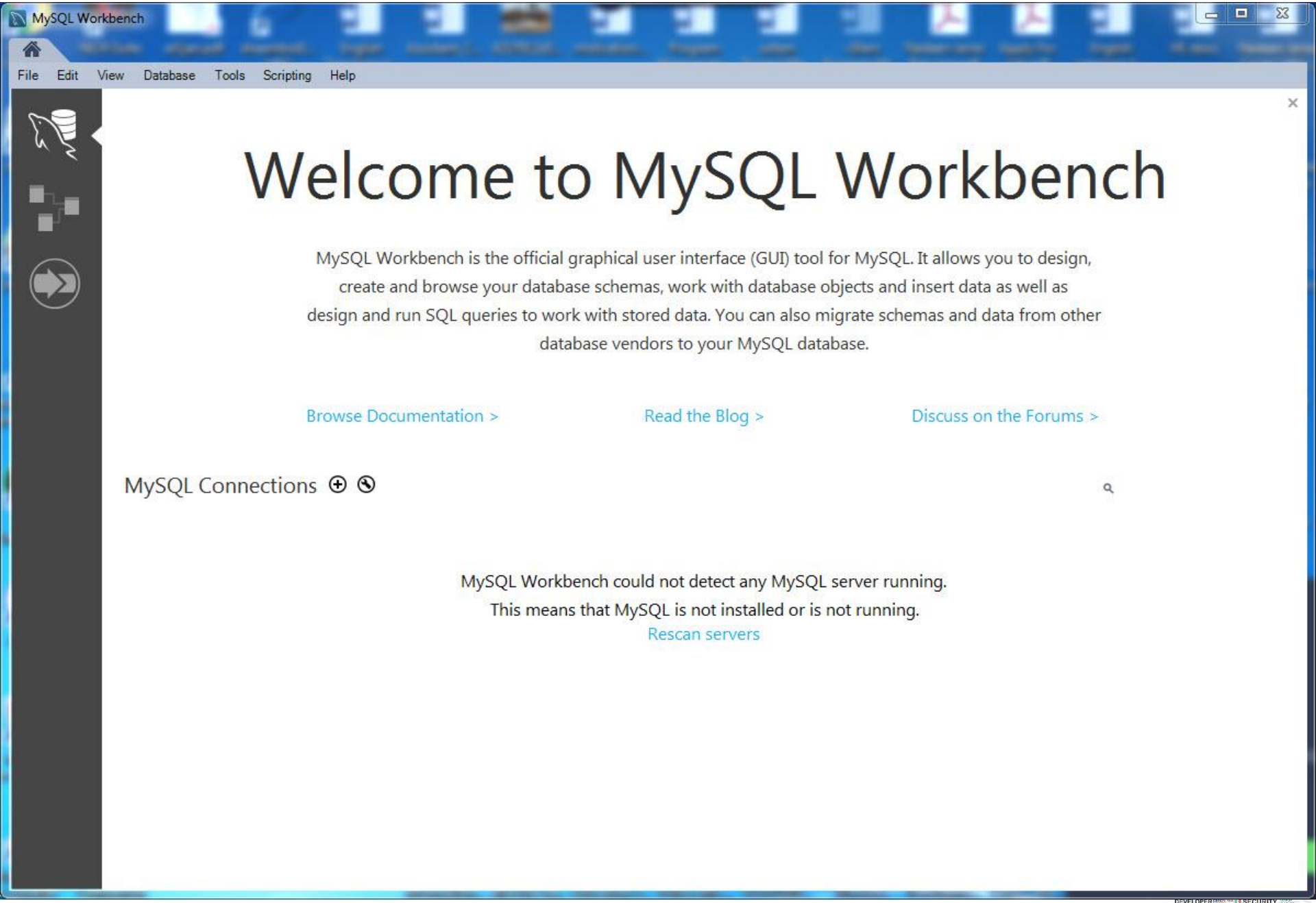


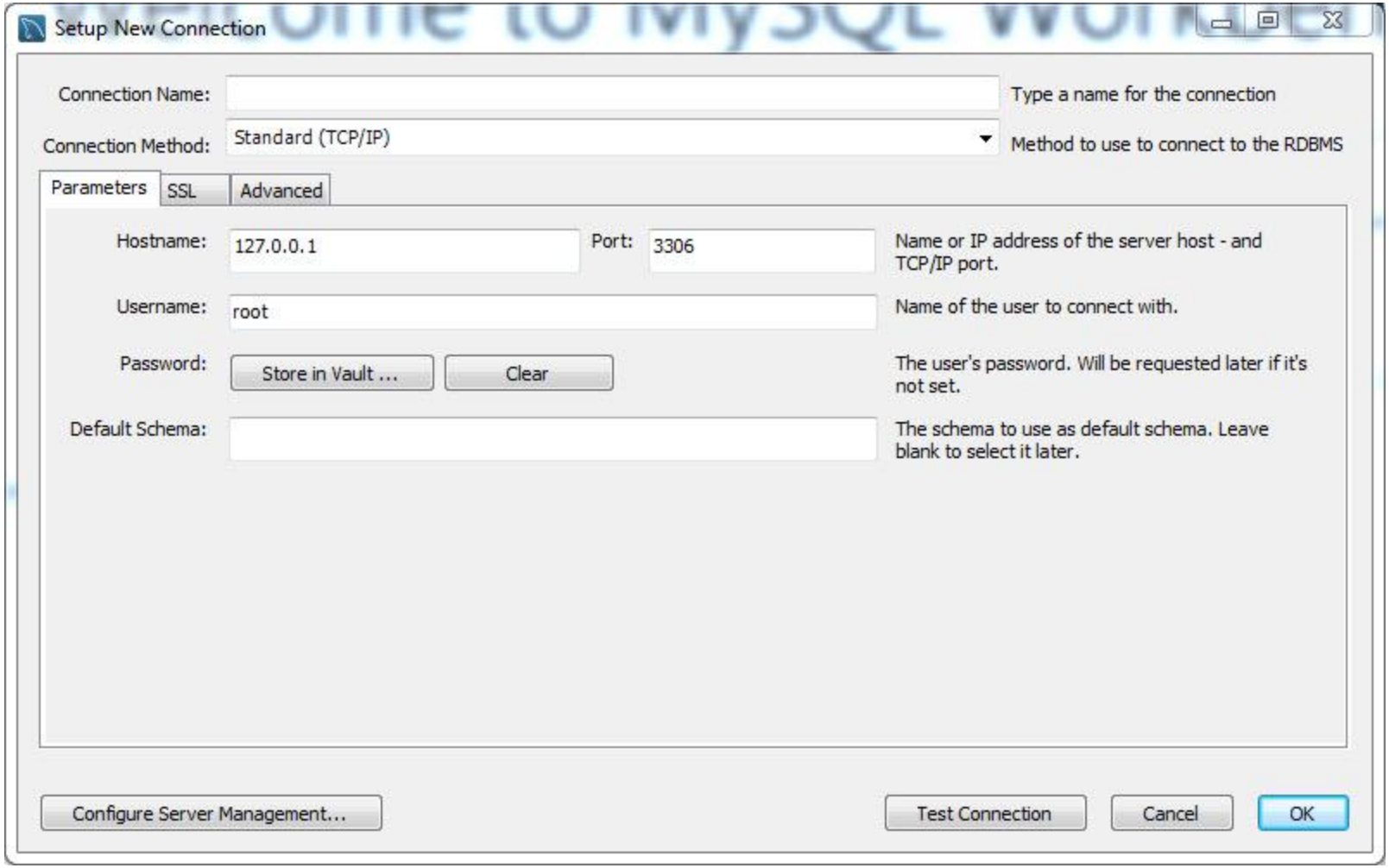


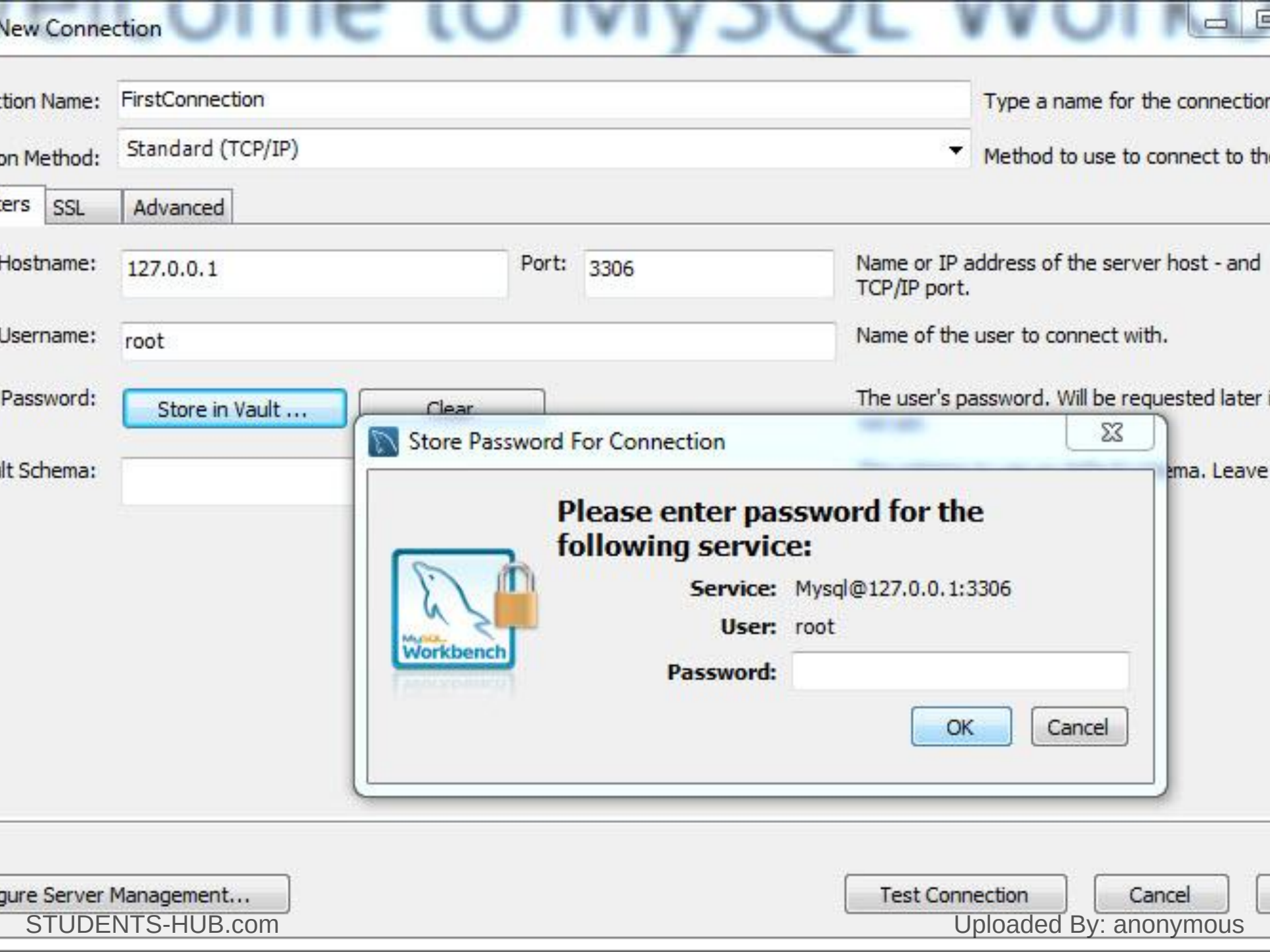


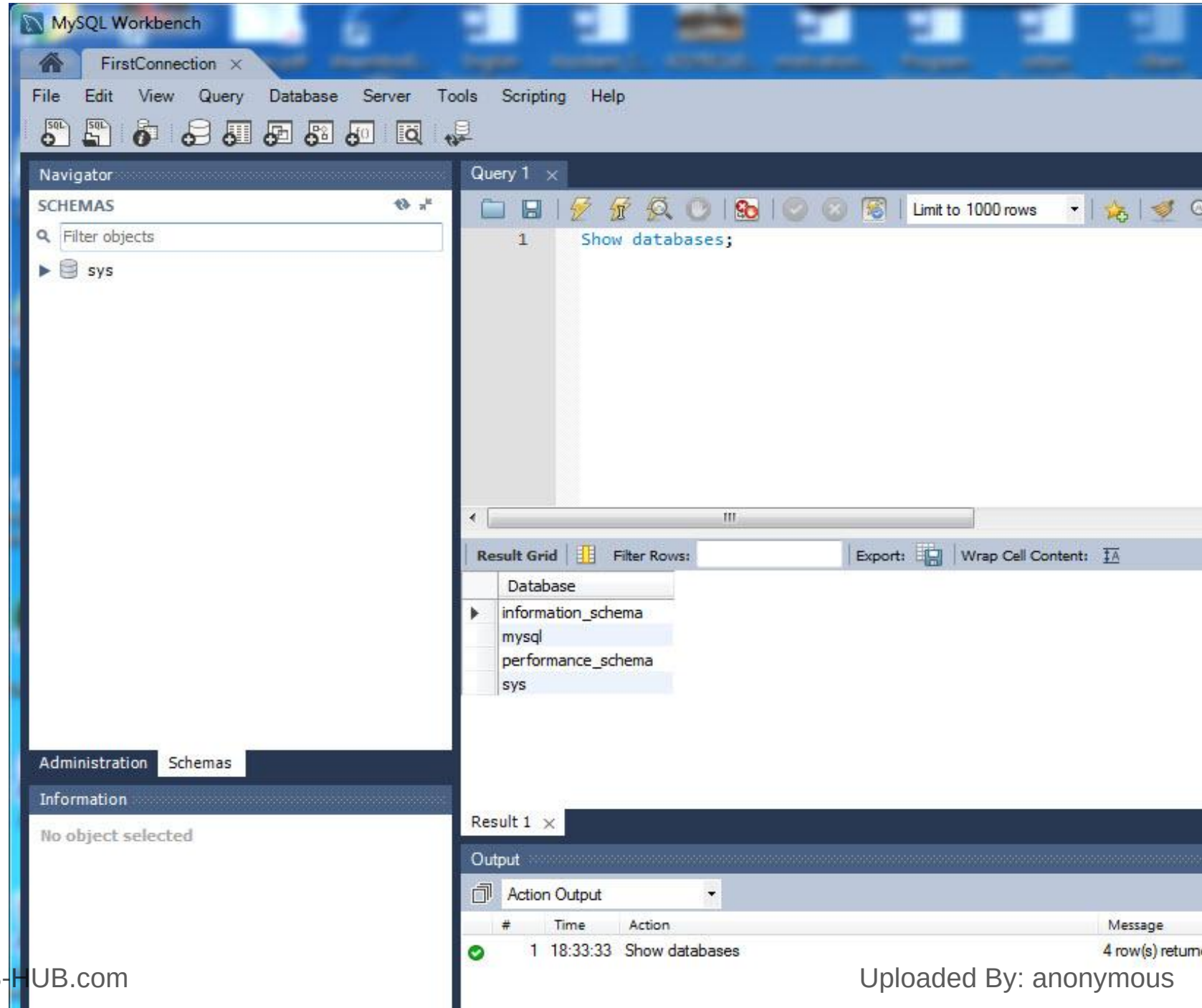












MySQL Basics – Data Definition

- SHOW DATABASES;
- CREATE DATABASE university;
- SHOW DATABASES;
- USE university;
- DROP DATABASE university;

MySQL Basics

- CREATE TABLE student (
  sid INT,
  sname VARCHAR(32),
  bdate DATE,
  gpa REAL,
  PRIMARY KEY (sid));
- SHOW TABLES;
- SHOW CREATE TABLE student;
- ALTER TABLE STUDENT ADD major VARCHAR(16);
- ALTER TABLE STUDENT ADD phone VARCHAR(16) AFTER bdate;
- DROP TABLE student;

MySQL Basics – Data Manipulation

- Query:

```
SELECT *
FROM student;
```

- INSERT INTO STUDENT VALUES (1051122, 'Ahmad', '1980-01-20', 99);
- SELECT * FROM student;
- INSERT INTO STUDENT (sid, sname) VALUES (1061122, 'Sireen’);
- DELETE FROM student WHERE sid>=1060000 AND sid<=1069999;
- Query:

```
SELECT sid, sname
FROM student
WHERE sname = ‘Ahmad’;
```

MySQL Basics – Auto increment

- ALTER TABLE student MODIFY sid int auto_increment;
- SELECT * FROM student;
- INSERT INTO student (sname) VALUES ('Iyad');
- ALTER TABLE student auto_increment=1070000;
- INSERT INTO student (sname) VALUES ('Gabi');
- ALTER TABLE student MODIFY gpa REAL DEFAULT 60;
- SELECT * FROM student;
- ALTER TABLE student MODIFY bdate DATE DEFAULT '1900-01-01';
- INSERT INTO student (sname) VALUES ('Gabi');

MySQL Basics – Data Control

- CREATE USER 'user1'@'localhost' IDENTIFIED BY ‘password';
- GRANT ALL PRIVILEGES ON university.* TO 'user1'@'localhost' WITH GRANT OPTION;
- CREATE USER 'user1'@'%' IDENTIFIED BY ‘password';
- GRANT ALL PRIVILEGES ON university.* TO 'user1'@'%' WITH GRANT OPTION;

- CREATE USER 'user2'@'localhost' IDENTIFIED BY ‘password2';
- GRANT SELECT ON university.* TO 'user2'@'localhost' WITH GRANT OPTION;
- CREATE USER 'user2'@'%' IDENTIFIED BY ‘password2';
- GRANT SELECT ON university.* TO 'user2'@'%' WITH GRANT OPTION;

Integrity Constraints Over Relations

- A database is only as good as the information stored in it, and a DBMS must therefore help prevent the entry of incorrect information.

- An integrity constraint (IC) is a condition that is specified on a database schema, and restricts the data that can be stored in an instance of the database.
- We already have seen the *Domain Constraints*

Key Constraints

- A key constraint is a statement that a certain minimal subset of the fields of a relation is a unique identifier for a tuple.
- Two Important Note:
  - Two distinct tuples in a legal instance cannot have identical values in all the fields of a key.
  - No subset of the set of fields in a key is a unique identifier for a tuple.
- Primary Key, Candidate Key, and Super key

Keys (continued)

- Composite key
  - Composed of more than one attribute
- Key attribute
  - Any attribute that is part of a key
- Superkey
  - Any key that uniquely identifies each row
- Candidate key
  - A superkey without redundancies

Keys (continued)

- Nulls:
  - No data entry
  - Not permitted in primary key
  - Should be avoided in other attributes
  - Can represent
    - An unknown attribute value
    - A known, but missing, attribute value
    - A “not applicable” condition
  - Can create problems when functions such as COUNT, AVERAGE, and SUM are used
  - Can create logical problems when relational tables are linked

SQL for Data Definition: CREATE with CONSTRAINT

- Creating database tables with PRIMARY KEY constraints
  - The SQL CREATE TABLE statement
  - The SQL CONSTRAINT keyword

```
CREATE TABLE Employee(
    EmpID Integer Not Null,
    EmpName Char(25) Not Null,
  PRIMARY KEY (EmpID)
);
```

SQL for Data Definition: CREATE with CONSTRAINT

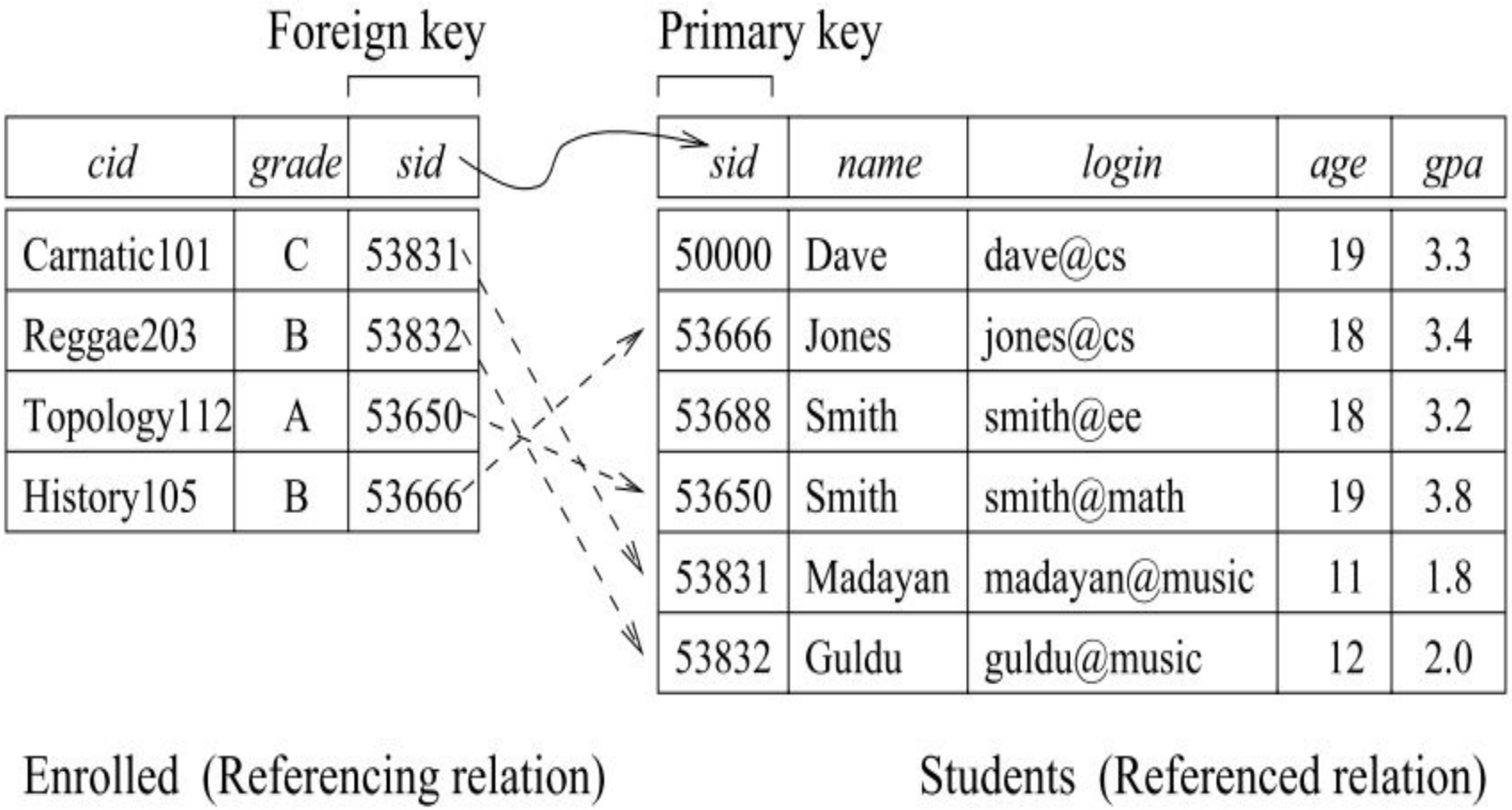
- Creating database tables with composite primary keys using PRIMARY KEY constraints
  - The SQL CREATE TABLE statement
  - The SQL CONSTRAINT keyword

```
CREATE    TABLE    Emp_Skill (
   EmpID          Integer        Not Null,
   SkillID        Integer        Not Null,
   SkillLevel     Integer,
   PRIMARY KEY (EmpID, SkillID)
);
```

Keys for Relationship Sets

- The combination of primary keys of the participating entity sets forms a super key of a relationship set.
  - (s_id, i_id) is the super key of advisor
  - NOTE: this means a pair of entity sets can have at most one relationship in a particular relationship set.
    - Example: if we wish to track multiple meeting dates between a student and her advisor, we cannot assume a relationship for each meeting. We can use a multivalued attribute though
- Must consider the mapping cardinality of the relationship set when deciding what are the candidate keys
- Need to consider semantics of relationship set in selecting the primary key in case of more than one candidate key

Foreign Key Constraints

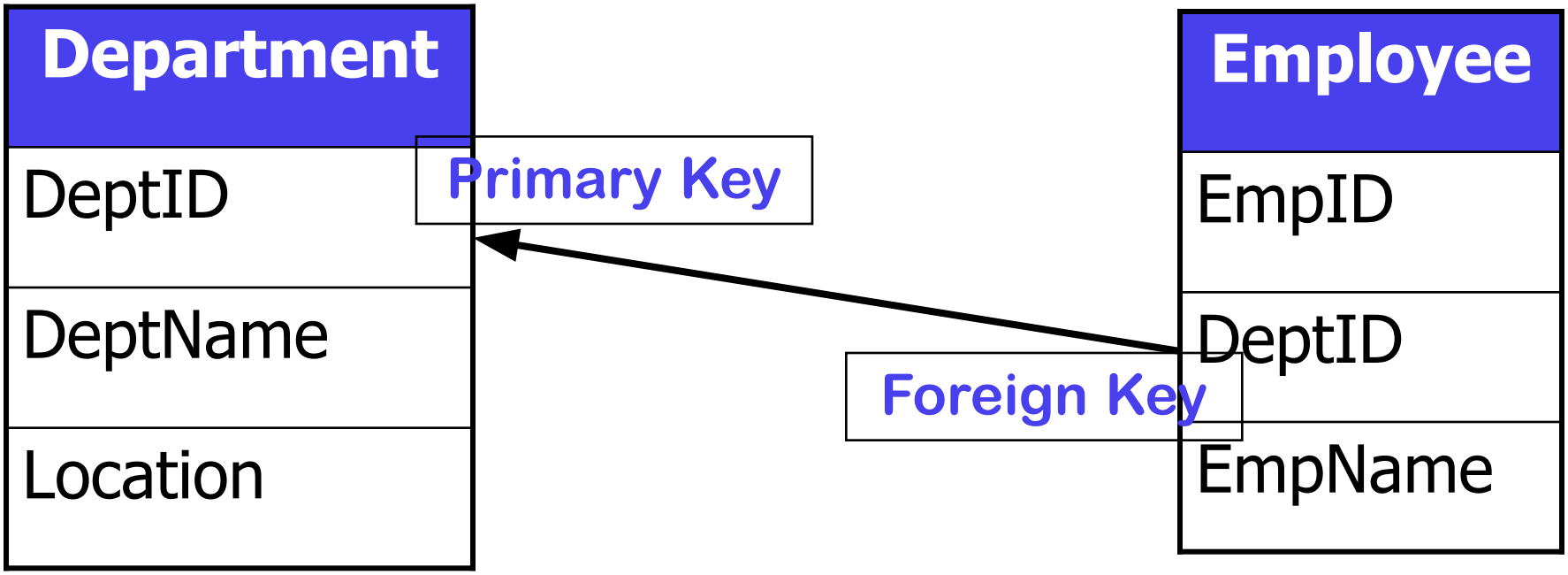


Specifying Foreign Keys

```
CREATE TABLE Enrolled ( sid     CHAR(20),
                        cid     CHAR(20),
                        grade   CHAR(10),
                        PRIMARY KEY (sid, cid),
```

Foreign Key (sid) References Students(sid)
Foreign Key (cid) References Course(cid));

Foreign Key Example



Referential Integrity

- Referential integrity states that every value of a foreign key must match a value of an existing primary key
- For example (see previous slide)
  - If EmpID = 4 in EMPLOYEE has a DeptID = 7 (a foreign key), a Department with DeptID = 7 **must exist** in DEPARTMENT

SQL for Data Definition: CREATE with CONSTRAINT

- Creating database tables using PRIMARY KEY and FOREIGN KEY constraints
  - The SQL CREATE TABLE statement
  - The SQL CONSTRAINT keyword

```
CREATE TABLE Emp_Skill (
   EmpID          Integer          Not Null,
   SkillID        Integer          Not Null,
   SkillLevel     Integer,
   PRIMARY KEY (EmpID, SkillID),
FOREIGN KEY (EmpID)          REFERENCES Employee (EmpID),
FOREIGN KEY (SkillID)        REFERENCES Skill (SkillID)
);
```

SQL for Data Definition: CREATE with CONSTRAINT

- Creating database tables using PRIMARY KEY and FOREIGN KEY constraints
  - The SQL CREATE TABLE statement
  - The SQL CONSTRAINT keyword
  - ON UPDATE CASCADE and ON DELETE CASCADE

```
CREATE      TABLE      Emp_Skill (
   EmpID            Integer          Not Null,
   SkillID          Integer          Not Null,
   SkillLevel       Integer,
         PRIMARY KEY (EmpID, SkillID),
               FOREIGN KEY (EmpID)
                     REFERENCES      Employee (EmpID)
                     ON DELETE CASCADE,
               FOREIGN KEY (SkillID)
                     REFERENCES      Skill (SkillID)
                     ON UPDATE CASCADE
);
```

When the row of EmpID (primary key) in Employee TABLE is deleted, the EmpFK (foreign key) is deleted also.

Deleting Database Objects: DROP

- To remove unwanted database objects from the database, use the SQL DROP statement
- Warning… The DROP statement will permanently remove the object and all data

```
DROP TABLE Employee;
```

Enforcing Integrity Constraints

- Deletion of ***Enrolled*** tuples do not violate referential integrity, but insertions could.
  - Inserting a tuple with an un-exist sid in ***Students***.

- Insertion of Students tuples do not violate referential integrity, but deletions could.

INSERT INTO *Enrolled* (sid, cid, grade) **VALUES** (51111, 'Hindi101', 'B');

Ways to handle foreign key violations

- If an ***Enrolled*** row with un-existing sid is inserted, it is **rejected**.
- If a ***Students*** row is deleted/updated,
  - Option 1: **Delete/Update** all ***Enrolled*** rows that refer to the deleted sid in ***Students*** (CASCADE). **Both are affected**
  - Option 2: **Reject** the deletion/updating of the ***Students*** row if an ***Enrolled*** row refers to it (NO ACTION). [The default action for SQL]. **None is affected**.
  - Option 3: **Set** the sid of ***Enrolled*** to some existing (**default**) sid value in ***Students*** for every involved ***Enrolled*** row (SET NULL / SET DEFAULT). **Both are affected**.

Referential Integrity in SQL

- When a ***Students*** row is deleted, all ***Enrolled*** rows that refer to it are to be **deleted** as well.
- When a ***Students*** sid is modified, the update is to be **rejected** if an ***Enrolled*** row refers to the modified ***Students*** row.

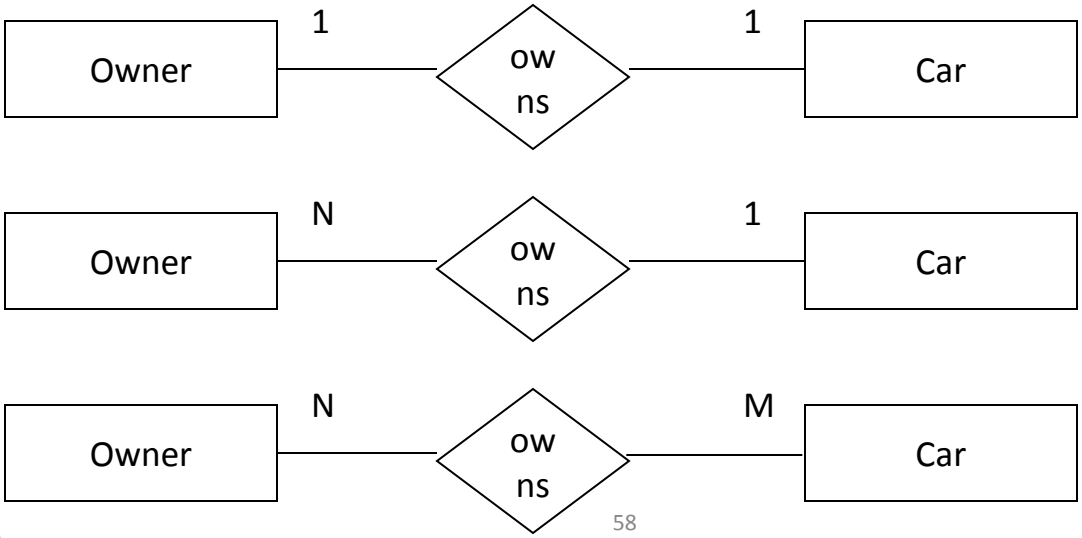
```
CREATE TABLE Enrolled
   (sid CHAR(20),
    cid CHAR(20),
    grade CHAR(2),
    PRIMARY KEY  (sid,cid),
    FOREIGN KEY (sid)
      REFERENCES Students (sid)
       ON DELETE CASCADE
       ON UPDATE No Action);
```

SQL Constraints

- NOT NULL constraint
  - Ensures that column does not accept nulls
- UNIQUE constraint
  - Ensures that all values in column are unique
- DEFAULT constraint
  - Assigns value to attribute when a new row is added to table
    - CUS_AREACODE CHAR(3) DEFAULT ‘615’ NOT NULL CHECK (CUS_AREACODE IN (‘615’, ‘713’, 931’))

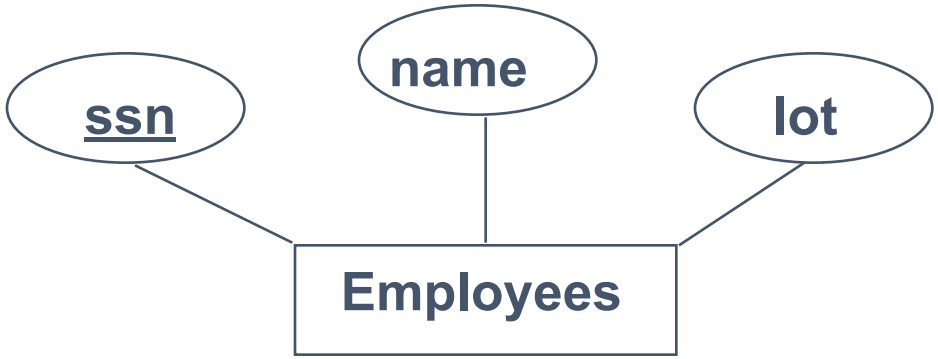
Relationship Types to Relational Model

- Possible cardinality ratio: 1:1, 1: N, N:1, and N:M
- Easiest is N:M
  - Every Entity is a relation
  - Every Relationship is a relation



ER to Relational Model - Entities

- Entity sets to tables:
  - Attributes to columns

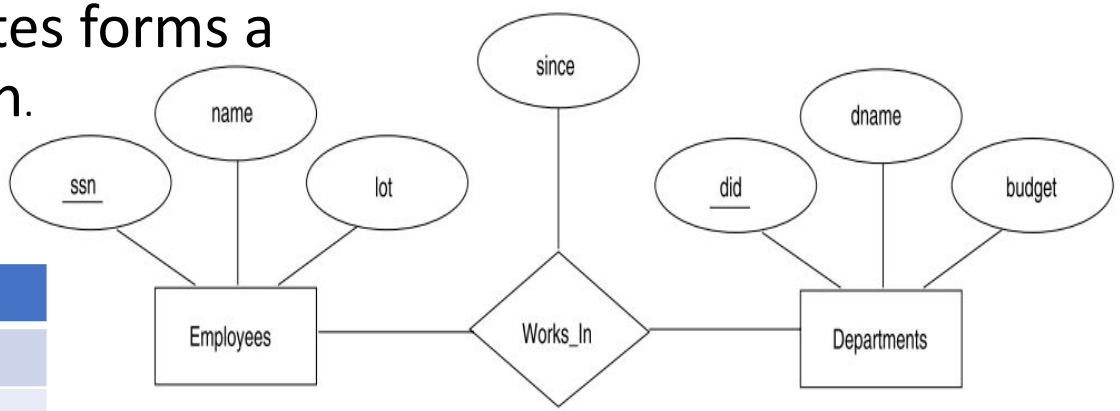


- CREATE TABLE Employees
  (ssn int,
  name CHAR(20),
  lot INTEGER,
  PRIMARY KEY (ssn))

ER to Relational Model - Relationships

- Relationship Sets to Tables
  - Attributes to columns
- In translating a relationship set to a relation, attributes of the relation must include:
  - Keys for each participating entity set (as foreign keys).
    - This set of attributes forms a *key* for the relation.

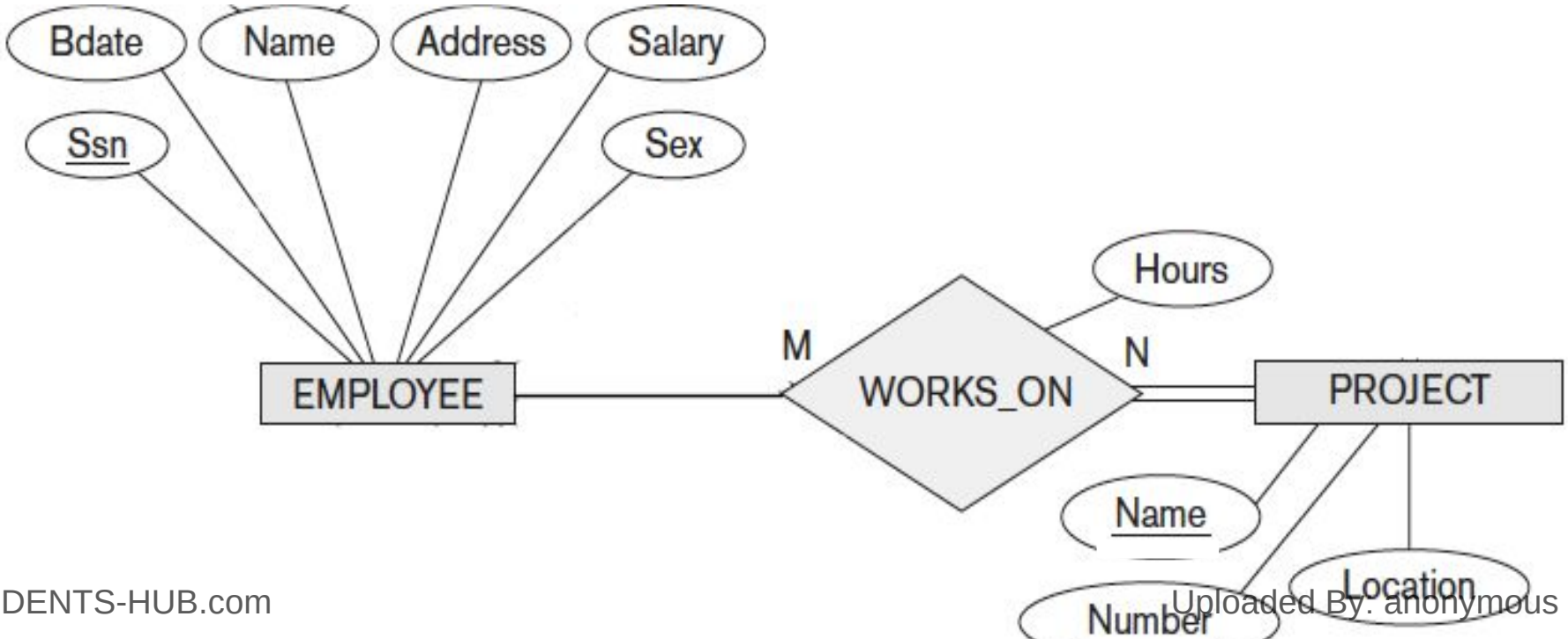
```
CREATE TABLE EMP ….
CREATE TABLE DEPT …

CREATE TABLE works_in (
        since date,
        ssn_emp int,
        did_dept int
primary key (ssn_emp, did_dept),
Foreign key (ssn_emp) references em
Foreign key (did_dept) references de
```



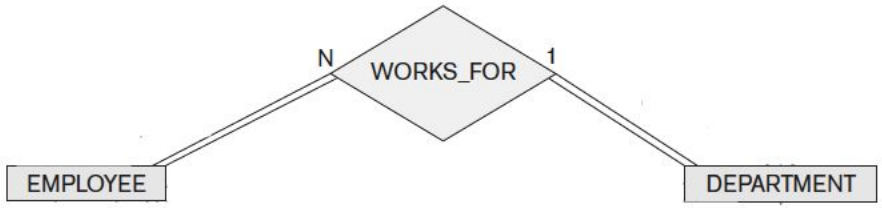
Since	Ssn	did
1/1/2019	1	2
1/1/2018	1	1
1/5/2020	2	3

ER to Relational Model - Relationships

- EMP (SSN: int primary key, name: varchar(32), etc…)
- PROJ (Number: int primary key, Name: varchar(32), etc..)
- CREATE TABLE EMP2PROJ (SSN int, Proj_num int not null, Hours int,
  PRIMARY KEY (SSN, Proj_num),
  Foreign Key (SSN) References EMP(SSN),
  Foreign Key (Proj_num) References PROJ(Number));

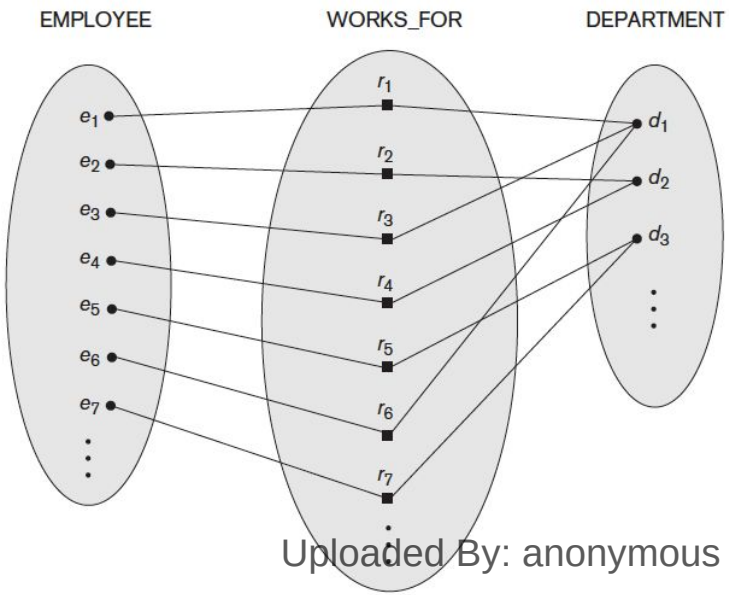


One-to-Many

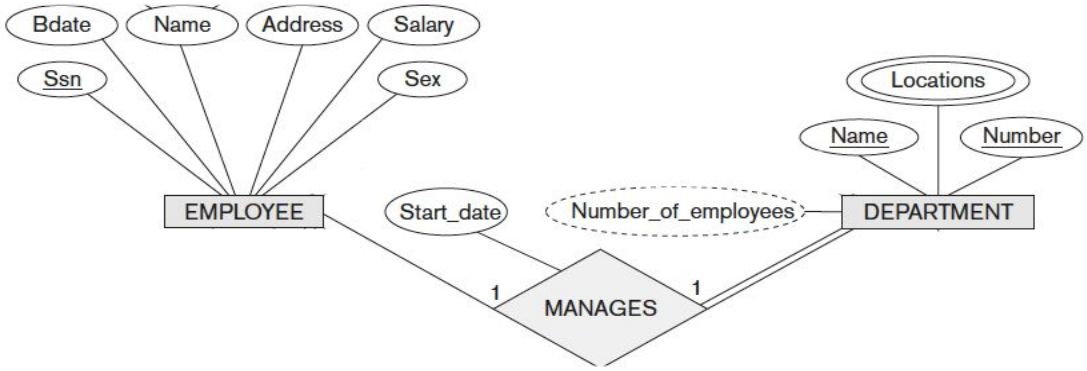


- Start with Each Entity as a relation
  - EMP(eid: int, name: varchar(32), etc..)
  - DEPT(did: int, dname: varchar(32), etc..)
- Relationship needs special care on the 1-1 side
  - Especially if total participation
- Relationship must be merged with Emp
- Result:
  - EMP_works(eid: int,
    name: varchar(32)
    rank int,
    salary real,
    did: int not null,
    primary key(eid),
    foreign key (did) references DEPT(did))

Replaces employee table
Replaces works_for table

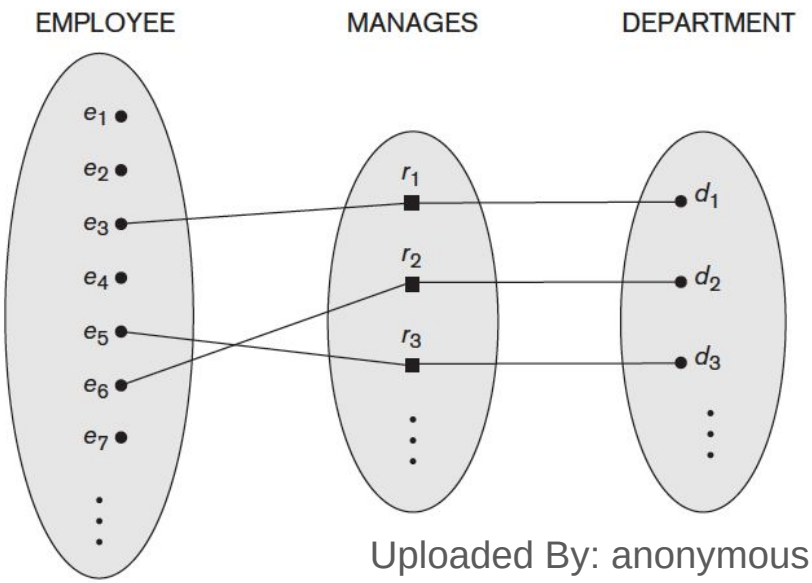


One-to-One



- Start with Each Entity as a relation
  - EMP(eid: int, name: varchar(32), etc..)
  - DEPT(did: int, dname: varchar(32), etc..)
- Relationship needs special care on the 1-1 side
  - Especially if total participation
- Relationship must be merged with DEPT
- Result:
  - DEPT(did int,
    name varchar(32),
    stdate date,
    mgr_ssn: int not null,
    primary key(did),
    foreign key (mgr_ssn)
    references EMP(eid));

Dept_locations
(did int,
Location int,
Primary key(did, location),
Foreign key (did) references dept(did));



Musicians Example

Exercise 2.5 Notown Records has decided to store information about musicians who perform on its albums (as well as other company data) in a database. The company has wisely chosen to hire you as a database designer (at your usual consulting fee of $2500/day).

- Each musician that records at Notown has an SSN, a name, an address, and a phone number. Poorly paid musicians often share the same address, and no address has more than one phone.
- Each instrument used in songs recorded at Notown has a name (e.g., guitar, synthesizer, flute) and a musical key (e.g., C, B-flat, E-flat).
- Each album recorded on the Notown label has a title, a copyright date, a format (e.g., CD or MC), and an album identifier.
- Each song recorded at Notown has a title and an author.
- Each musician may play several instruments, and a given instrument may be played by several musicians.
- Each album has a number of songs on it, but no song may appear on more than one album.
- Each song is performed by one or more musicians, and a musician may perform a number of songs.
- Each album has exactly one musician who acts as its producer. A musician may produce several albums, of course.

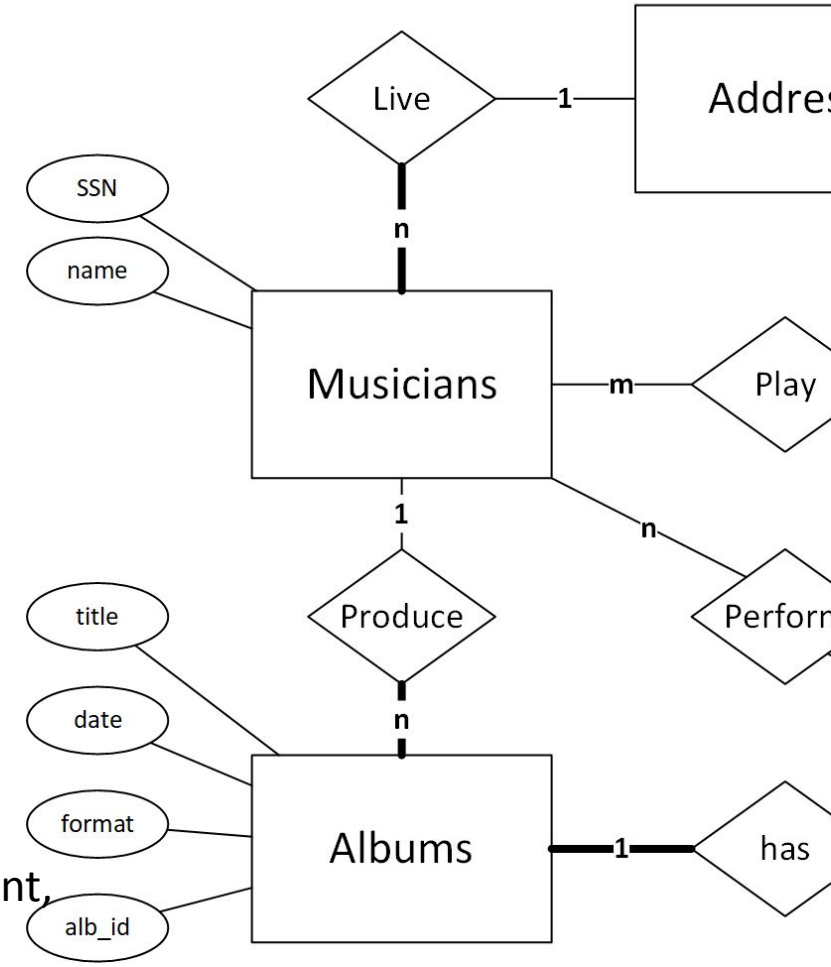
Create Table Address (Phone int primary key, City varchar(16), Street varchar(32));

CREATE TABLE Musicians (Ssn int primary key,
Name char(32), Phone int,
Foreign key(phone) references Address(phone));

Create table instruments(Inst_id int primary key,
Name varchar(32), Musical_key varchar(32));

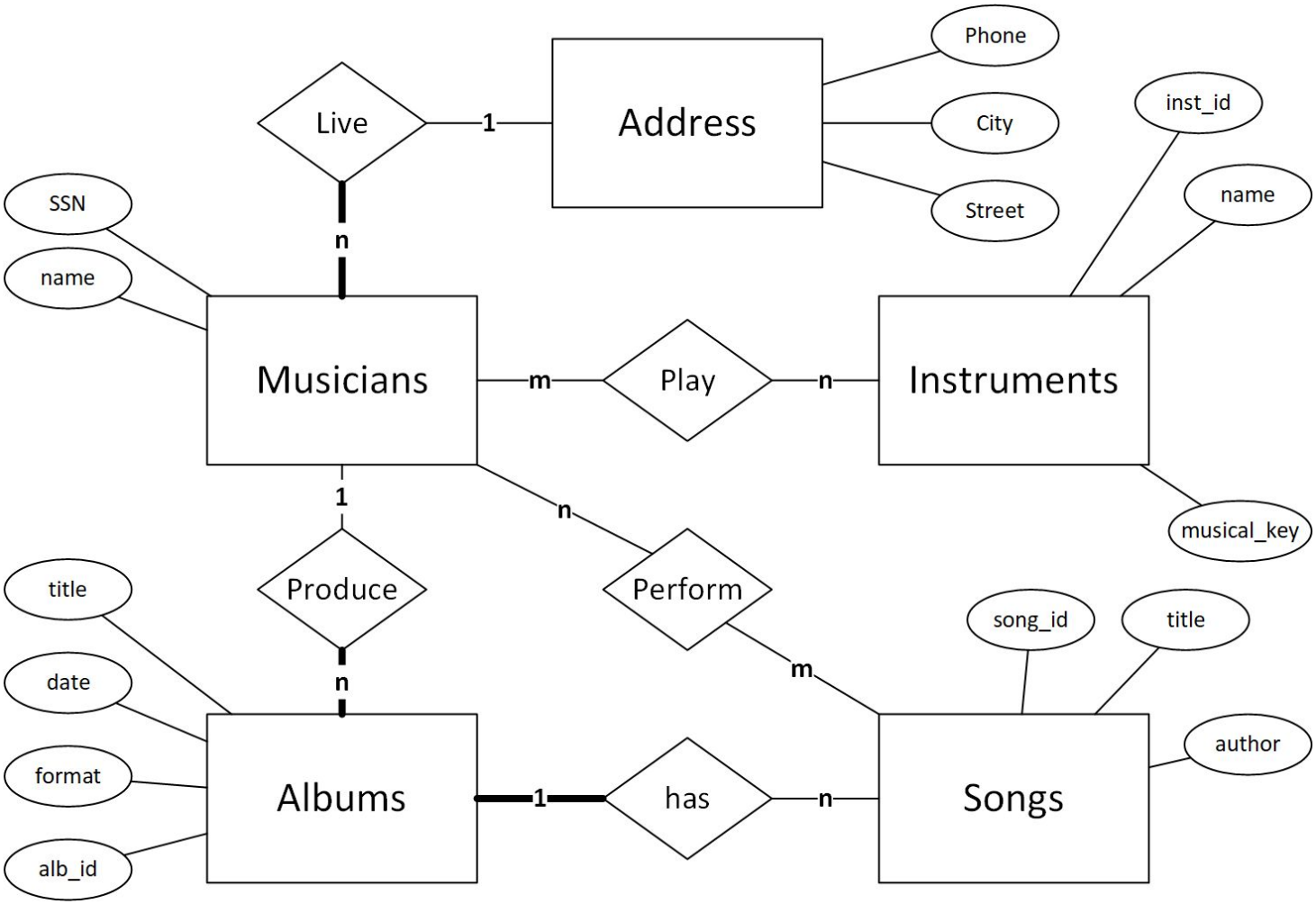
Create table Mus2Inst(ssn int, inst_id int,
Primary key(ssn, inst_id),
Foreign key (ssn) references musicians(ssn),
Foreign key (inst_id) references instruments(inst_id));

Create table albums (alb_id int primary key, Prod_ssn int,
A_date date, format varchar(32), Title varchar(32),
Foreign key (prod_ssn) references musicians(ssn));



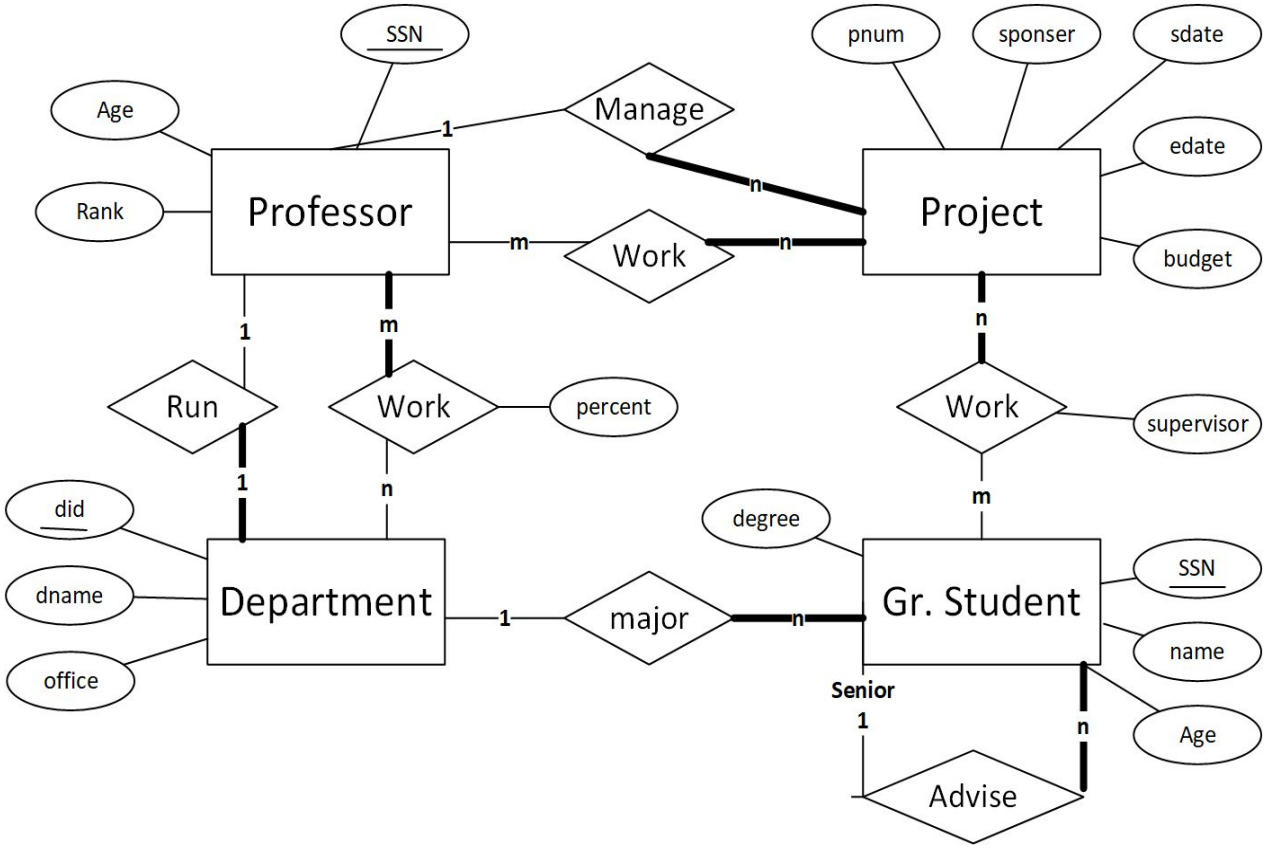
Create table Songs (song_id int primary key, title varchar(32), author varchar(32),
Alb_id int, foreign key (alb_id) references albums(alb_id));

Create table Mus2Songs(ssn int, song_id int, primary key (ssn, song_id),
Foreign key (ssn) References Musicians(ssn),
Foreign key (song_id) references Songs(song_id));



University Example

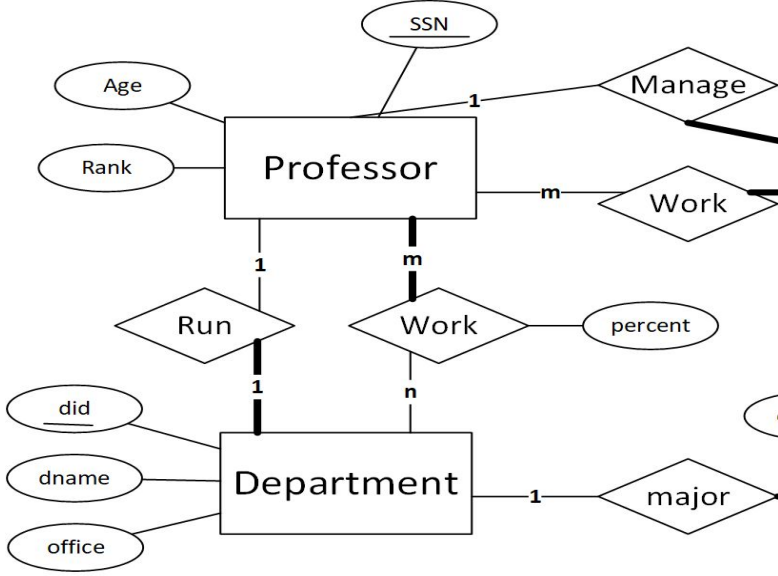
- Professors have an SSN, a name, an age, a rank, and a research specialty.
- Projects have a project number, a sponsor name (e.g., NSF), a starting date, an ending date, and a budget.
- Graduate students have an SSN, a name, an age, and a degree program (e.g., M.S. or Ph.D.).
- Each project is managed by one professor (known as the project's principal investigator).
- Each project is worked on by one or more professors (known as the project's co-investigators).
- Professors can manage and/or work on multiple projects.
- Each project is worked on by one or more graduate students (known as the project's research assistants).
- Graduate students can work on multiple projects, in which case they will have a (potentially different) supervisor for each one.
- Departments have a department number, a department name, and a main office.
- Departments have a professor (known as the chairman) who runs the department.
- Professors work in one or more departments, and for each department that they work in, a time percentage is associated with their job.
- Graduate students have one major department in which they are working on their degree.



Create Table Prof(ssn int primary key, age int,
rank int);

CREATE TABLE run_Dept (did int primary key,
office varchar(32),
dname varchar(32), mgr_ssn int,
Foreign key(mgr_ssn) references Prof(ssn));

Create table Prof2DeptW(did int, ssn int, percent int,
Primary key(did, ssn),
Foreign key (ssn) references Prof(ssn),
Foreign key (did) references run_Dept(did));



Create table Proj(pnum int primary key, sponser varchar(32), sdate date,
Edate date, budget real, mgr_ssn int,
Foreign key (mgr_ssn) references Prof(ssn));

Create table Prof2ProjW(pnum int, ssn int, Primary key(pnum, ssn),
Foreign key (ssn) references Prof(ssn),
Foreign key (pnum) references Proj(pnum));

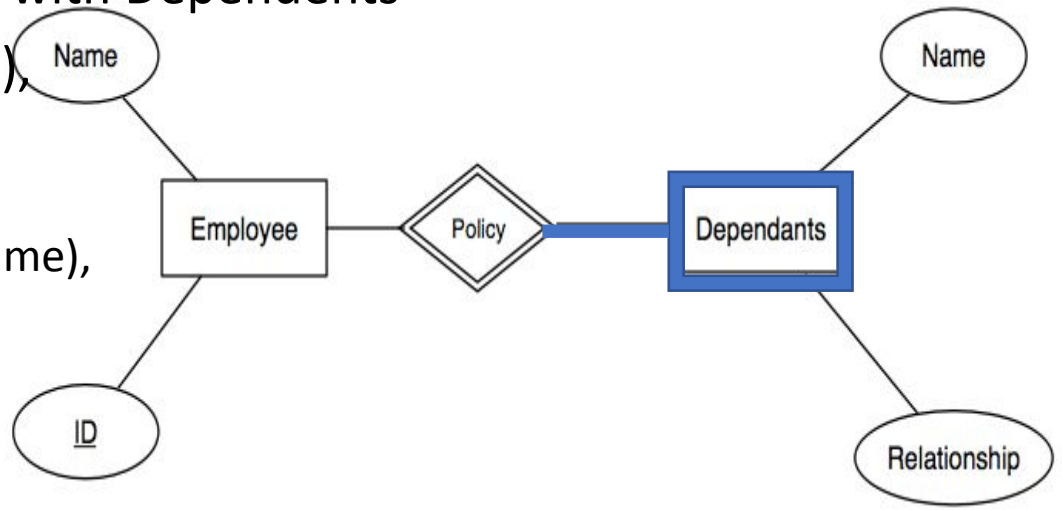
Create table gStudent(ssn int primary key, name varchar(32), age int,
degree varchar(32), senior_ssn int, major_did int,
Foreign key (major_did) references run_Dept(did));

Create table Student2ProjW(pnum int, ssn int, Primary key(pnum, ssn), supervisor varchar(32),
Foreign key (ssn) references gStudent(ssn), Foreign key (pnum) references Proj(pnum));

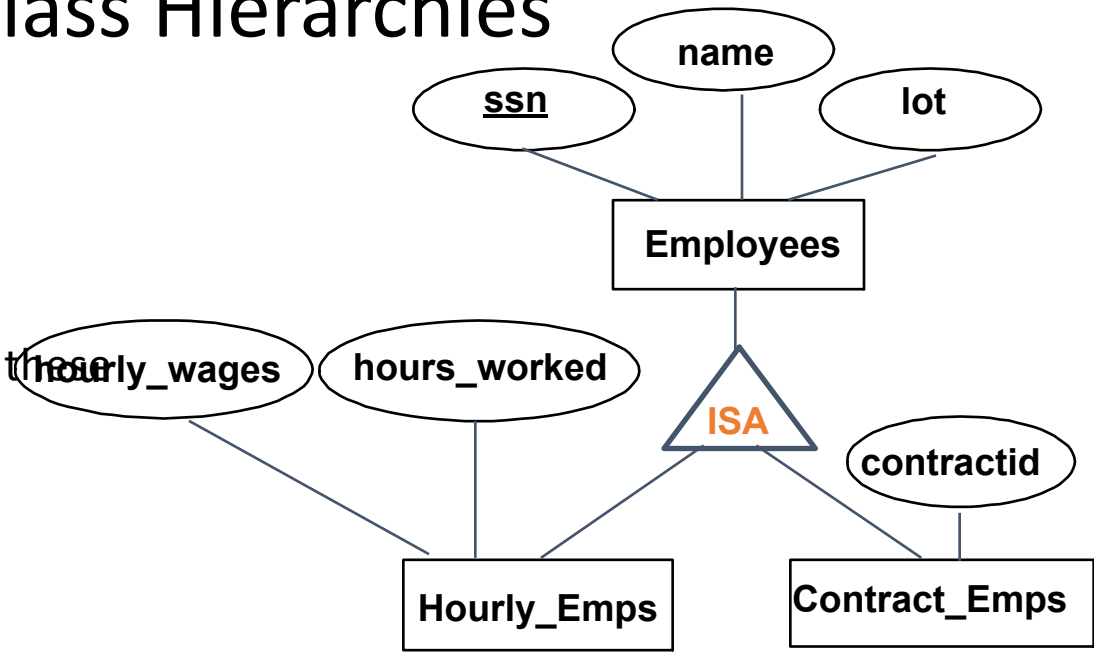
Relational Model for Weak Entity Sets

- Start with Each Entity as a relation
  - EMP(eid: int, name: varchar(32), etc..)
  - Dependents(Name: varchar(32), relationship: varchar(32), etc..)
- Weak Relationships needs special care
- Relationship must be merged with Dependents

Dependents(Name: Varchar(32),
Relationship: Varchar(32),
emp_id: int not null,
Primary key (emp_id*, Name),
Foreign Key (emp_id)
References EMP(eid)
on delete cascade)



Relational Model for Class Hierarchies



Just Hourly_Emps and Contract_Emps.

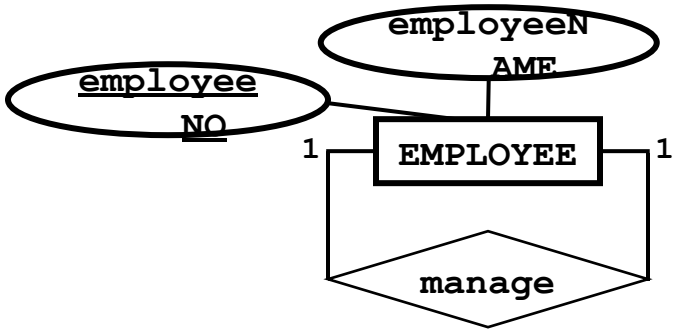
Hourly_Emps: *ssn*, *name, lot, hourly_wages, hours_worked.*

Each employee must be in one of these two subclasses.

- ***General approach:***
  - 3 relations: Employees, Hourly_Emps and Contract_Emps.
    - *Hourly_Emps*: Every employee is recorded in Employees. For hourly emps, extra info recorded in Hourly_Emps (*hourly_wages*, *hours_worked*, *ssn)*; must delete Hourly_Emps tuple if referenced Employees tuple is deleted).
    - Queries involving all employees easy, those involving just Hourly_Emps require a join to get some attributes.

Relational Model for Recursive Relationships

- EMP (employeeNo int primary key,
  employeeName varchar(32),
  ManagerSSN int))



Empno	Emp name	Mgr_ssn
1	Ahmad	-1
2	Dania	1