Normalization

- It is a process that we can use to remove design flaws from a database
- A number of normal forms, which are sets of rules describing what we should and should not do in our table structure
- 3NF is sufficient to avoid the data redundancy problem of a designed relational database

Normal Form (NF)

- 1NF: each attribute or column value must be atomic
- 2NF: if a schema is 1NF, and if its all attributes that are not part of the primary key are fully functionally dependent on the primary key
- 3NF: if a schema is 2NF, and all transitive dependencies have been removed

Ex: employeeDept(<u>employeeID</u>, name, job, deptID, deptName) has to convert to

employee(employeeID, name, job, deptID)

Dept(deptID, deptName)

2NF

- It means that each non-key attribute must be functionally dependent on all parts of the primary key (i.e., the combination of the composite attributes of the key).
- Example: not 2NF

Employee(employeeID, name, job, departmentID, skill)

employeeID, skill → name, job, departmentID

employeeID → name, job, departmentID

(Note: → determine)

- Break the table into two tables to become 2NF

Employee(employeeID, name, job, departmentID)

employeeSkills(employeeID, skill)

3NF

- Example: 2NF but not 3NF

Employee(employeeID, name, job, departmentID, departmentName)

Here employeeID → departmentID

employeeID → departmentName

Also departmentID → departmentName, departmentID is not a key

Therefore, employeeID → departmentName is a transitive dependency

- Convert the schema to 3NF by breaking to two tables:

Employee(employeeID, name, job, departmentID)

Department(departmentID, departmentName)

Normal Forms Defined Informally

- 1st normal form
  - All attributes depend on **the key**
- 2nd normal form
  - All attributes depend on **the whole key**
- 3rd normal form
  - All attributes depend on **nothing but the key**

SUMMARY OF NORMAL FORMS based on Primary Keys

Table 10.1
Summary of Normal Forms Based on Primary Keys and Corresponding Normalization

Normal Form	Test	Remedy (Normalization)
First (1NF)	Relation should have no multivalued attributes or nested relations.	Form new relations for each multivalued attribute or nested relation.
Second (2NF)	For relations where primary key contains multiple attributes, no nonkey attribute should be functionally dependent on a part of the primary key.	Decompose and set up a new relation for each partial key with its dependent attribute(s). Make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it.
Third (3NF)	Relation should not have a nonkey attribute functionally determined by another nonkey attribute (or by a set of nonkey attributes). That is, there should be no transitive dependency of a nonkey attribute on the primary key.	Decompose and set up a relation that includes the nonkey attribute(s) that functionally determine(s) other nonkey attribute(s).

Nested Relation

- Each tuple can have a relation within it
- Multivalued attributes that are themselves composite
- It is not allowed in the First Normal Form

MySQL

1. Is an excellent database server product
2. Is fast and stable
3. Is available as free software and as commercial software
4. Supports the majority of features considered important by the database community
5. Is a great tool for learning about databases

Installing MySQL

- You can download MySQL from

http://www.mysql.com/downloads/mysql/

- You may refer to Classnote #9 for the information of installing MySQL on Windows
- You can run MySQL on Linux on the computer system of our department by executing:

  mysql –h stretch –u username -p

Introduction to MySQL Monitor

- After logged in you can see what databases exist on the system by:

  show database;

- You can select a database from the listed database names by:

  use *databasename*;

- After selecting a database, you can see what table are in it by:

  show tables;

Introduction to MySQL Monitor

- You can get information on a particular table by:
  describe *tablename*;
- Log out of the monitor by: \q
- Get a help by: \h
- You can execute a file of commands
  - When logged into the monitor by : Source *filename*
  - When not logged into the monitor by:

  mysql –u username –p < filename

Creating Databases and Tables

- Creating a database by

  create database *databasename*;
- Creating a table by

  create table *tablename (table definition) [type = table_type]*;

Ex: create table dept

(deptID int not null auto_increment primary key,

Name varchar(30)) type=InnoDB

Dropping Databases and Tables

- Drop a database by:

  drop database *databasename*;
- Drop a table by:

  drop table *tablename*;

Inserting, Deleting, and Updating Data

- Deleting by: delete from dept
- Inserting by:

```
insert into dept values
(42, 'Finance' ),
(128, 'Research  and Development' );
```

- Update by:

```
update employee
    set job = 'DBA'
    where employeeID = "6651';
```

Uploading data with Load Data Infile

- The Load Data Infile command allows you to bulk insert data from a text file into a single table without having to write Insert statements.
- Example:

  load data local infile 'dept_infile.txt'

  into table dept;

Querying MySQL

- Select columns from tables:

  select * from dept;

  select name, employeeID from employee;

- Using joins to run queries over multiple tables:

```
select employee.name as EmployeeName, dept.name as deptName
from employee, dept
where employee.deptID = dept.deptID;
```

Table Types in MySQL

- ISAM, having replaced by MyISAM
- MyISAM as the default type
  - Offer very fast but not transaction-safe storage
- InnoDB
  - Offer very fast and transaction-safe storage
  - Row-level locking
  - Sopports for foreign keys
  - Tables are portable from systems to systems
- BerkeleyDB (BDB)
  - Offer transaction-safe storage but slightly worse performance than InnoDB
- MERGE
  - Used to treat multiple MyISAM tables as a single table
- HEAP
  - Stored only in memory and need to be limited in size