

Introduction to Relational Databases

Randy Julian

Lilly Research Laboratories

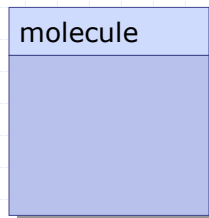
Database Basics

- ◆ Database:
 - A container (usually a file or set of files) to stored organized data.
 - Not the database software: (DBMS)
 - In practice: A set of Tables
- ◆ Tables
 - A structured list of data of a specific type
 - A database "Entity"
- ◆ Schema
 - Information about database and table layout and properties

Database Basics

- ◆ Column
 - A single field in a table - all tables have one or more columns (Entity Attribute)
- ◆ Datatype
 - A type of allowed data for a column
- ◆ Row
 - A record in a table
 - An "Entity Instance"

Database Entities



The "molecule" entity as it appears in the data model.

This will be represented in the DBMS as a table.

Entity Attributes

molecule
name
num_atoms
num_bonds
SMILES
CAS

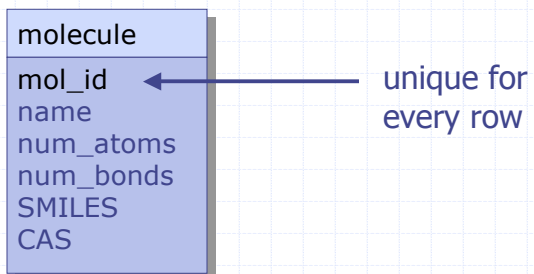
Attributes are columns in the table.

Each one of these will be represented in the DBMS as a row in a table.

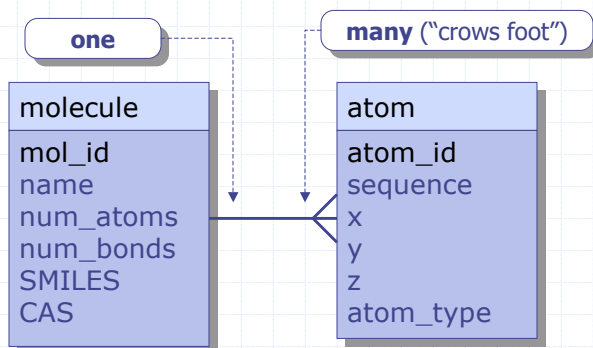
Primary Keys

- ◆ A column (or set of columns) whose values uniquely identify every row in a table.
 - No two rows can have the same primary key value
 - The primary key value can not be empty (NULL)
 - The primary key column cannot be modified or updated
 - Primary keys cannot be reused (if the record is deleted)

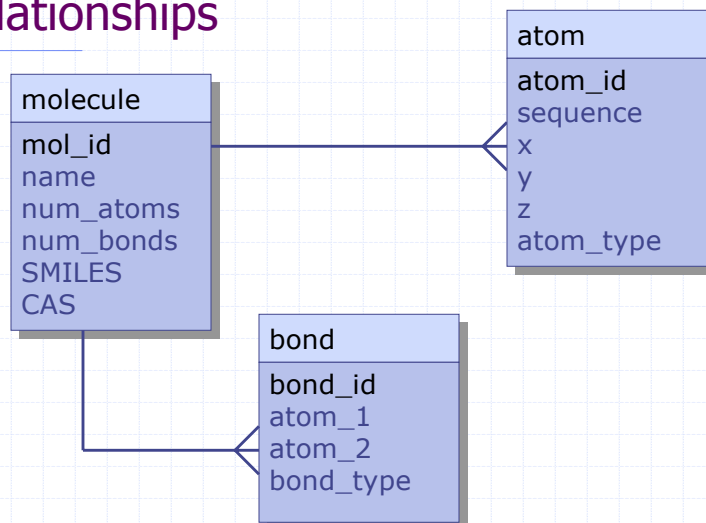
Primary Keys in the Data Model



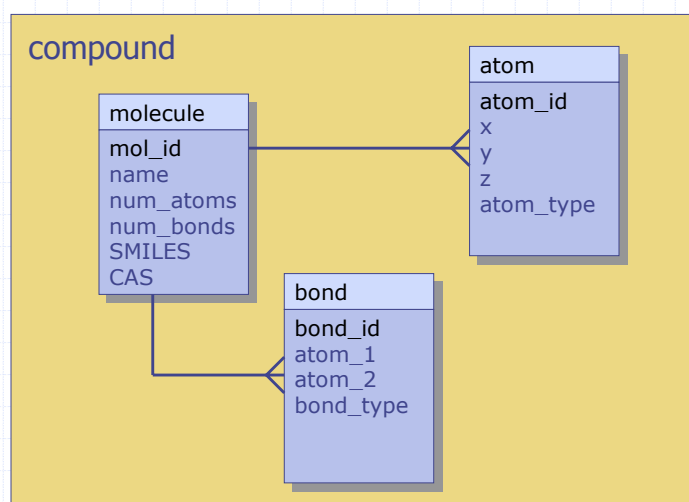
Relationships



Relationships



Logical Model



Normalization

- ◆ Goal: Increase clarity of representation
- ◆ First Normal Form (1NF)
 - When all attributes are single valued
 - If any attribute has repeating values it is not in 1NF

Example of un-normalized data

<u>name</u>	<u>num atoms</u>	<u>x</u>	<u>y</u>	<u>atom type</u>
Acetone	4	-0.3458	-2.9667	C
Acetone	4	0.3667	-2.55	C
Acetone	4	0.3621	-1.725	O
Acetone	4	1.0834	-2.9585	C

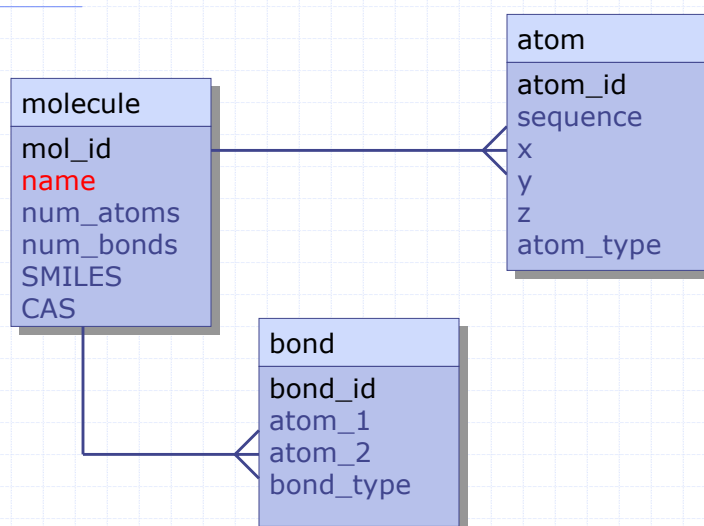
molecule

name
num_atoms
x
y
atom_type

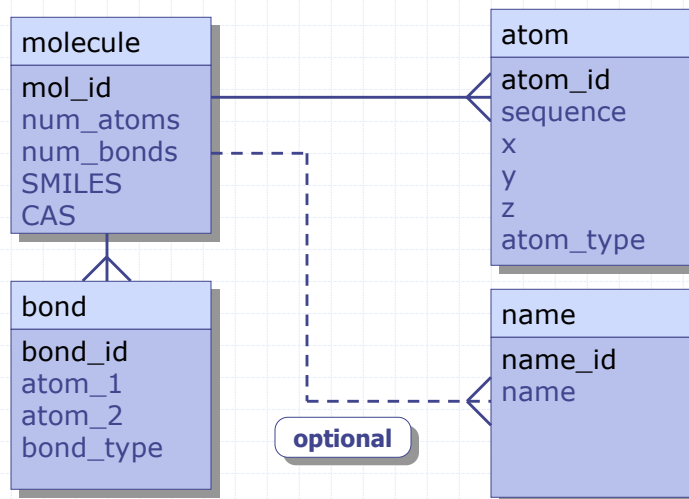
Second Normal Form (2NF)

- ◆ If the entity is already in 1NF and...
 - All non-identifying attributes are dependent on the entity's unique identifier.
 - Normalize along these attributes by:
 - ◆ Finding the entity where it belongs
 - ◆ Creating new entities where the attribute belongs
- ◆ Example:
 - Compound names - name not dependent on unique identifier of molecule

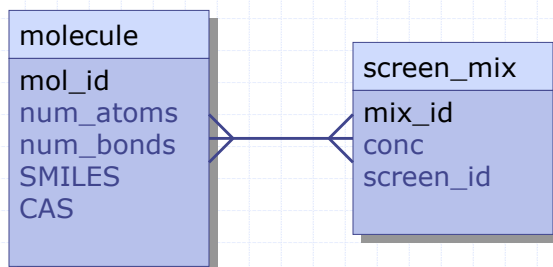
What about synonyms?



Adding another entity to further normalize



Many-to-many relationships

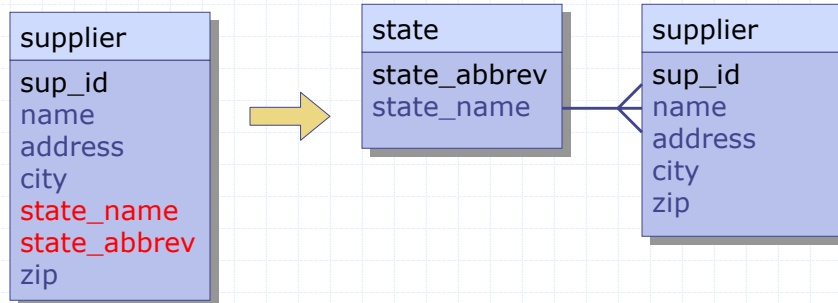


a molecule could be in many different screen mixtures

a screen mixture could contain many molecules

Third Normal Form (3NF)

- ◆ When no non-identifying attributes are dependent on any non-identifying attributes
 - The value of one attribute is controlled by another, non-identifying attribute.



Logical Modeling Method

- ◆ Identify and model the entities
- ◆ Identify and model the relationships between the entities
- ◆ Identify and model the attributes
- ◆ Identify unique identifiers for each entity
- ◆ Normalize

Physical Database Design

- ◆ Entities become tables in the database
 - You have already named the tables
- ◆ Attributes become columns
 - Choose an appropriate datatype for each column
- ◆ Unique identifiers become primary keys
 - These can never be NULL
- ◆ Relationships are modeled as 'foreign keys'
 - Attributes added to tables to make links

Table Definitions

Table	Column	Datatype	Notes
molecule	mol_id	INT	primary key
	num_atoms	INT	
	num_bonds	INT	
	SMILES	TEXT	
	CAS	TEXT	
atom	atom_id	INT	primary key
	sequence	INT	
	x	FLOAT	
	y	FLOAT	
	z	FLOAT	
	atom_type	TEXT	

Table Definitions (cont)

Table	Column	Datatype	Notes
bond	bond_id	INT	primary key
	atom_1	INT	
	atom_2	INT	
	bond_type	INT	
name	name_id	INT	primary key
	name	TEXT	

Physical Table Model

Table	Column	Datatype	Notes
molecule	mol_id	INT	primary key
	num_atoms	INT	
	num_bonds	INT	
	SMILES	TEXT	
	CAS	TEXT	
name	name_id	INT	primary key
	name	TEXT	
	mol_id	INT	foreign key

Physical Table Model

Table	Column	Datatype	Notes
atom	atom_id	INT	primary key
	sequence	INT	
	x	FLOAT	
	y	FLOAT	
	z	FLOAT	
	atom_type	TEXT	
	mol_id	INT	foreign key

Physical Table Model

Table	Column	Datatype	Notes
bond	bond_id	INT	primary key
	atom_1	INT	
	atom_2	INT	
	bond_type	INT	
	mol_id	INT	foreign key

Physical Databases

- ◆ A database management system (DBMS) accepts commands to create, change, fill, delete, etc. databases, tables, columns, records...
- ◆ The standard command language to perform these operations: Structured Query Language: (SQL)
 - Pronounced 'sequel' or S-Q-L

SQL

- ◆ Non proprietary language
- ◆ Easy to learn
- ◆ Very powerful
- ◆ Can be embedded in programs

Create the 'Compound' database

```
create database if not exists `compound`;
```

```
use `compound`;
```

```
CREATE TABLE `molecule` (  
  `mol_id` int(11) NOT NULL auto_increment,  
  `num_atoms` int(11) NOT NULL default '0',  
  `num_bonds` int(11) NOT NULL default '0',  
  `SMILES` varchar(128) default NULL,  
  `CAS` varchar(128) default NULL,  
  PRIMARY KEY (`mol_id`)  
);
```

CREATE TABLE

```
CREATE TABLE `atom` (  
  `atom_id` int(11) NOT NULL default '0',  
  `mol_id` int(11) NOT NULL default '0',  
  `sequence` int(11) NOT NULL default '0',  
  `x` float default NULL,  
  `y` float default NULL,  
  `z` float default NULL,  
  `atom_type` char(2) default NULL,  
  PRIMARY KEY (`atom_id`)  
);
```

DROP TABLE

```
drop table if exists `atom` ;
```

INSERT

```
INSERT INTO `molecule` VALUES  
(1,4,3,'67-64-1',''),  
(2,3,2,'75-05-8','');
```

```
INSERT INTO `atom` VALUES  
(1,1,1,-0.3458,-2.9667,0,'C') ,  
(2,1,2,0.3667,-2.55,0,'C') ,  
(3,1,3,0.3621,-1.725,0,'O') ,  
(4,1,4,1.0834,-2.9585,0,'C') ,  
(5,2,1,-20,-15,0,'C') ,  
(6,2,2,-1,-15,0,'C') ,  
(7,2,3,20,-15,0,'N') ;
```

```
INSERT INTO `name` VALUES  
(1,'67-64-1','Acetone') ,  
(2,'67-64-1','2-Propanone') ,  
(3,'67-64-1','Dimethyl ketone') ,  
(4,'75-05-8','Acetonitrile') ,  
(5,'75-05-8','Methyl cyanide') ;
```

Basic Query: SELECT

```
SELECT * FROM atom;
```

atom_id	mol_id	sequence	x	y	z	atom_type
1	1	1	-0.3458	-2.9667	0	C
2	1	2	0.3667	-2.55	0	C
3	1	3	0.3621	-1.725	0	O
4	1	4	1.0834	-2.9585	0	C
5	2	1	-20	-15	0	C
6	2	2	-1	-15	0	C
7	2	3	20	-15	0	N

SELECT

```
SELECT name.name, molecule.CAS  
FROM molecule,name  
WHERE molecule.CAS=name.cas;
```

name	CAS
Methyl cyanide	75-05-8
Dimethyl ketone	67-64-1
Acetonitrile	75-05-8
Acetone	67-64-1
2-Propanone	67-64-1

UPDATE

```
UPDATE molecule
SET SMILES ='CC(=O)C'
WHERE CAS='67-64-1';
```

> Query OK, 0 rows affected (0.29) sec

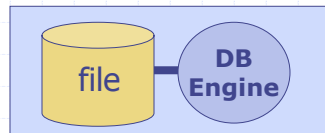
mol_id	num_atoms	num_bonds	CAS	SMILES
1	4	3	67-64-1	CC(=O)C
2	3	2	75-05-8	

DELETE

```
DELETE FROM molecule
WHERE CAS="99-99-99";
```

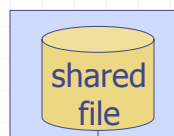
> Query OK, 0 rows affected (0.29) sec

Local Database Systems

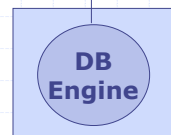
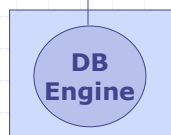


MS Access
dBase
Paradox

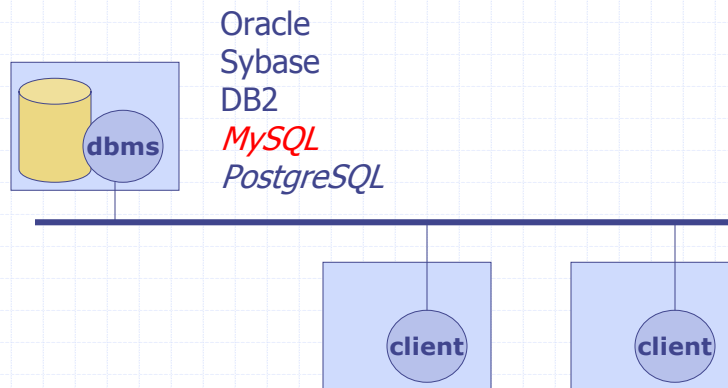
Engine-File Server Database Systems



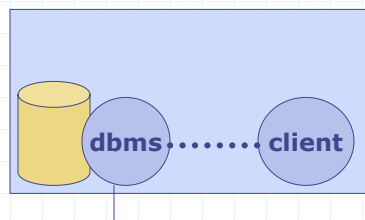
MS Access



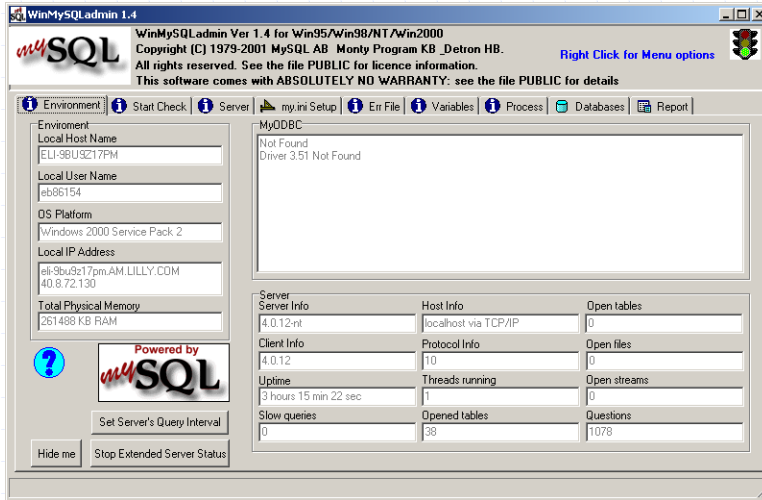
Client-Server Database Systems



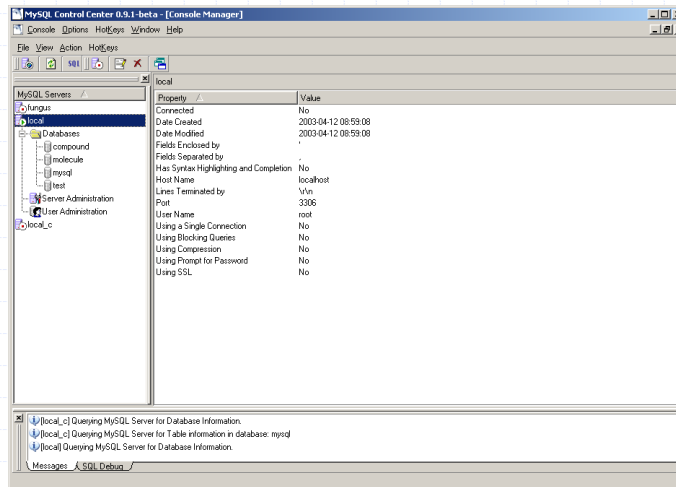
Local Client-Server



MySQL Server on Windows



MySQL Control Center GUI



Program Clients

