Web Programming

Lecture 8 – Database Access Through the Web

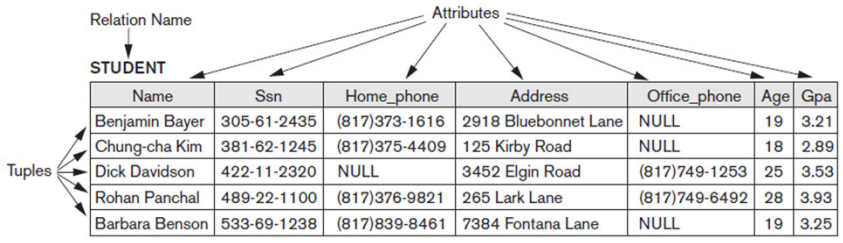
What is a database?

- A database is a collection of data organized to allow relatively easy access for retrievals, additions, modifications, and deletions.
- The most widely used type of databases is the ***relational database***, which was originally proposed by E. F Codd in the late 1960s.

What is a relational database?

- A relational database is a collection of tables of data.
  - Each row represents an ***entity*** (person, place or thing)
  - Each column has a name and represents an attribute (a property that an entity has).
  - There is usually one column (or set of columns) that uniquely identifies a particular row in the table. This is called the ***primary key***.
  - The intersection of a row and column is usually called a ***field***.

Relational Model Concepts

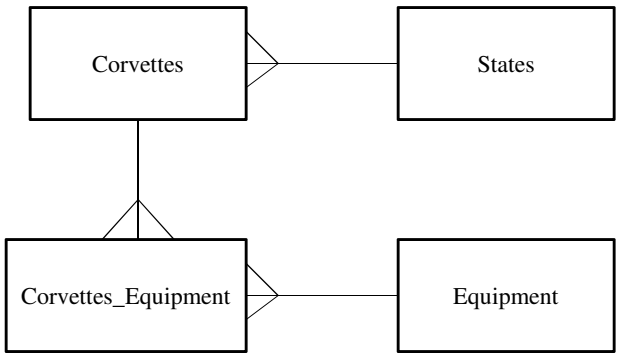


Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

Example – A Database Of Sports Cars

- We can set up a small database of Corvettes.
- Since each car can come with different equipment and from different states, we may choose to store this information in separate tables.

Data Model of the Corvettes Database



`Corvettes` Table

Vette_id	Body_Style	Miles	Year	State
1	coupe	18.0	1997	4
2	hatchback	58.0	1996	7
3	convertible	13.5	2001	1
4	hatchback	19.0	1995	2
5	hatchback	25.0	1991	5
6	hardtop	15.0	2000	2
7	coupe	55.0	1979	10
8	convertible	17.0	1999	5
9	hardtop	17.0	2000	5
10	hatchback	50.0	1995	7

`Equipment` Table

Equip_id	Equip
1	Automatic
2	4-speed
3	5-speed
4	6-speed
5	CD
6	Leather

`States` Table

State_id	State
1	Alabama
2	Alaska
3	Arizona
4	Arkansas
5	California
6	Colorado
7	Connecticut
8	Delaware
9	Florida
10	Georgia

`Corvettes_Equipment` Table

Vette_id	Equip	Vette_id	Equip
1	1	5	6
1	5	6	2
1	6	7	4
2	1	7	6
2	5	8	4
2	6	8	5
3	1	8	6
3	6	9	4
4	2	9	5
4	6	9	6
5	1	10	1
		10	5

Normalization

- We want our database to be a clear representation of the data, its relationships and constraints
- We can identify relationship using a technique called ***normalization***.
- Normalization is a bottom-up technique where we examine the relationship between attributes and reconfigure the tables accordingly.

Purpose of Normalization

- Characteristics of a suitable set of relations include:
  - the minimal number of attributes necessary to support the data requirements of the enterprise;
  - attributes with a close logical relationship are found in the same relation;
  - minimal redundancy with each attribute represented only once with the important exception of attributes that form all or part of foreign keys.

Our Example

- The DreamHome Customer Rental Details form holds details about property rented by a given customer.
  - To simplify things, we will assume that a renter rents a given property once and only one property at a time.

Our Original Table

CustNo	Cname	PropNo	PAddr	RntSt	RntFnsh	Rent	OwnerNo	OName
CR76	John Kay	PG4	6 Lawrence St, Elmont	7/1/10	8/31/06	700	CO40	Tina Murphy
		PG16	5 Nova Dr, East Meadow	9/1/06	9/1/08	900	CO93	Tony Shaw
CR56	Aline Stewart	PG4	6 Lawrence St, Elmont	9/1/02	6/10/04	700	CO40	Tina Murphy
		PG36	2 Manor Rd Scarsdale	8/1/04	12/1/05	750	CO93	Tony Shaw
		PG16	5 Nova Dr, East Meadow	8/1/06	9/1/10	900	CO93	Tony Shaw

First Normal Form (1NF)

- Unnormalized – A table with one or more repeating groups.
- First Normal Form (1NF) – A relation in which the intersection of each row and column contains one and only one value

Repeating Groups

- Any collection of attributes that repeat provides a complication for a database, both in terms of storing it (how many repeating groups would you allow for) as well as querying them.
- It is necessary to recognize them so we can eliminate them.
- E.g.,

  Repeating Group = (Property_no, Paddress, RentStart, RentFinish, Rent, Owner_No, OName)

Our Table in 1NF

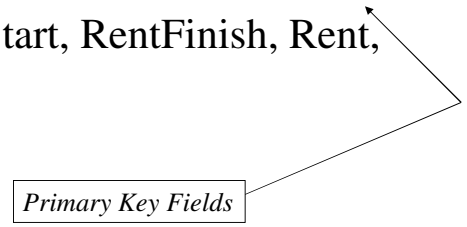
CustNo	CName	PropNo	PAddr	RntSt	RntFnsh	Rent	OwnerNo	OName
CR76	John Kay	PG4	6 Lawrence St, Elmont	7/1/10	8/31/06	700	CO40	Tina Murphy
CR76	John Kay	PG16	5 Nova Dr, East Meadow	9/1/06	9/1/08	900	CO93	Tony Shaw
CR56	Aline Stewart	PG4	6 Lawrence St, Elmont	9/1/02	6/10/04	700	CO40	Tina Murphy
CR56	Aline Stewart	PG36	2 Manor Rd Scarsdale	1/1/04	12/1/05	750	CO93	Tony Shaw
CR56	Aline Stewart	PG16	5 Nova Dr, East Meadow	8/1/06	9/1/08	900	CO93	Tony Shaw

Candidate Keys

- A candidate key for a given table is
  - unique (only one row has that value or combination of values)
  - irreducible (there is no subset of the candidate that is unique).
- Our candidate keys are:
  - (Customer_No, Property_No)
  - (Customer_No, RentStart)
  - (Property_No, RentStart)

The Customer_Rental Relation

Customer_Rental(Customer_No, Property_No, Cname, Paddress, RentStart, RentFinish, Rent, Owner_No, Oname)



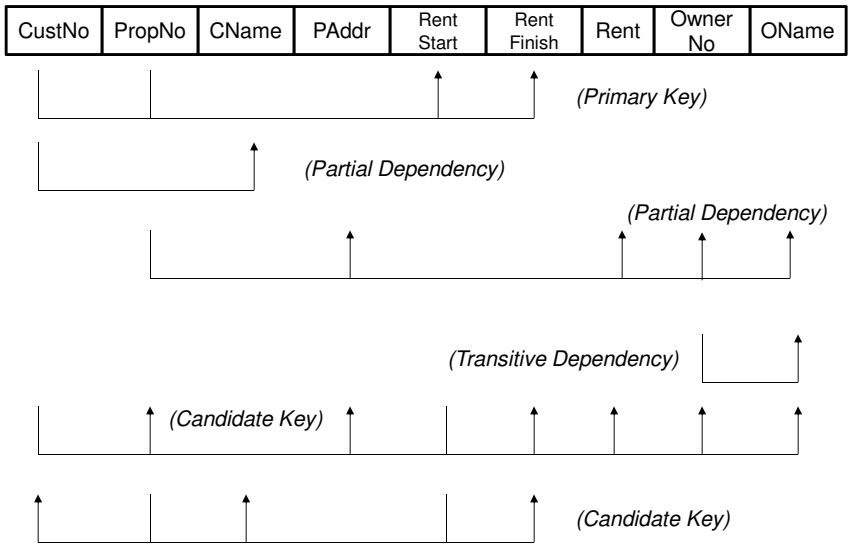
Functional Dependency

- If A and B are attributes of Relation R, B is functionally dependent on A (A→B) if each value of A is associated with one and only one value of B.
- B is fully functionally dependent on A if B is functional dependent on A and not on a proper subset of A.
- B is partially functionally dependent on A if there is some attribute that can be removed from A and the dependence still holds.

Listing All The Functional Dependencies

1. Cust_No, Prop_no → RentStart, RentFinish *(Primary Key)*
2. CustNo → Cname (Partial Dependency)
3. Prop_no → Paddress, Rent, Owner_No, Oname *(Partial Dependency)*
4. Owner_No → Oname *(Transitive Dependency)*
5. CustNo, RentStart → PropNo, Paddress, RentFinish, Rent, Owner_No, Oname *(Candidate Key)*
6. Prop_No, RentStart → CustNo, Cname, RentFinish *(Candidate Key)*

Functional Dependencies in Graphical Form



Functional Dependency in Our Table

- We have three relations with the following functional dependencies:
  - CustNo, PropNo → RentStart, RentFinish
  - CustNo → CustName
  - PropNo → Paddress, Rent, OwnerName, Oname
- Therefore, we have:
  - Customer(CustNo, Cname)
  - Rental(CustNo, PropNo, RentStart, RentFinish)
  - Property_Owner(PropNo, Paddress, Rent, OwnerNo, OName)

Second Normal Form (2NF)

- A relation is in 2NF if it is in 1NF and every non-primary key attribute is fully functionally dependent on the primary key

Our Database in 2NF

Customer Relation

CustNo	CName
CR76	John Kay
CR56	Aline Stewart

Our Database in 2NF

Rentals Relation

CustNo	PropNo	RentStart	RentFinish
CR76	PG4	7/1/10	8/31/06
CR76	PG16	9/1/06	9/1/08
CR56	PG4	9/1/02	6/10/04
CR56	PG36	1/1/04	12/1/05
CR56	PG16	8/1/06	9/1/10

Our Database in 2NF

Property-Owner Relation

PropNo	PAddr	Rent	OwnerNo	OName
PG4	6 Lawrence St, Elmont	700	CO40	Tina Murphy
PG16	5 Nova Dr, East Meadow	900	CO93	Tony Shaw
PG4	6 Lawrence St, Elmont	700	CO40	Tina Murphy
PG36	2 Manor Rd Scarsdale	750	CO93	Tony Shaw
PG16	5 Nova Dr, East Meadow	900	CO93	Tony Shaw

Transitive Dependency

- If A, B, and C are attributes of a relation R such that if A→B and B→C, then C is transitively dependent on A via B.

Third Normal Form

- A relation is in 3NF is if it is 2NF and there are no non-primary-key attributes that are transitively dependent on the primary key.

Functional Dependencies in 2NF

- Customer
  - CustNo → Cname
- Rental
  - CustNo, PropNo → RentStart, RentFinish
  - PropNo, RentStart → CustNo, RentFinish
- PropertyOwner
  - PropNo → Paddr, Rent, OwnerNo, OName
  - OwnerNo → Oname *(Oname is not f.d. on PropNo)*

Our 3NF Relations

We have 4 relations:

- Customer(CustNo, Cname)
- Rental(CustNo, PropNo, RentStart, RentFinish)
- Property_For_Rent(PropNo, Paddress, Rent, OwnerNo)
- Owner(OwnerNo, OName)

Our Database in 3NF

Customer Relation

CustNo	CName
CR76	John Kay
CR56	Aline Stewart

Our Database in 3NF

Rentals Relation

CustNo	PropNo	RentStart	RentFinish
CR76	PG4	7/1/10	8/31/06
CR76	PG16	9/1/06	9/1/08
CR56	PG4	9/1/02	6/10/04
CR56	PG36	1/1/04	12/1/05
CR56	PG16	8/1/06	9/1/10

Our Database in 3NF

Property-for-Rent Relation

PropNo	PAddr	Rent	OwnerNo
PG4	6 Lawrence St, Elmont	700	CO40
PG16	5 Nova Dr, East Meadow	900	CO93
PG36	2 Manor Rd, Scarsdale	750	CO93

Our Database in 3NF

Owner Relation

OwnerNo	OName
CO40	Tina Murphy
CO93	Tony Shaw

Basic SQL

- SQL language
  - Considered one of the major reasons for the commercial success of relational databases
- SQL
  - Structured Query Language
  - Statements for data definitions, queries, and updates (both Data Definition Language and Data Manipulation Language)

Our Example

- The DreamHome Customer Rental Details form holds details about property rented by a given customer.
  - To simplify things, we will assume that a renter rents a given property once and only one property at a time.

Our Original Table

CustNo	Cname	PropNo	PAddr	RntSt	RntFnsh	Rent	OwnerNo	OName
CR76	John Kay	PG4	6 Lawrence St, Elmont	7/1/10	8/31/06	700	CO40	Tina Murphy
		PG16	5 Nova Dr, East Meadow	9/1/06	9/1/08	900	CO93	Tony Shaw
CR56	Aline Stewart	PG4	6 Lawrence St, Elmont	9/1/02	6/10/04	700	CO40	Tina Murphy
		PG36	2 Manor Rd Scarsdale	8/1/04	12/1/05	750	CO93	Tony Shaw
		PG16	5 Nova Dr, East Meadow	8/1/06	9/1/10	900	CO93	Tony Shaw

First Normal Form (1NF)

- Unnormalized – A table with one or more repeating groups.
- First Normal Form (1NF) – A relation in which the intersection of each row and column contains one and only one value

Repeating Groups

- Any collection of attributes that repeat provides a complication for a database, both in terms of storing it (how many repeating groups would you allow for) as well as querying them.
- It is necessary to recognize them so we can eliminate them.
- E.g.,

  Repeating Group = (Property_no, Paddress, RentStart, RentFinish, Rent, Owner_No, OName)

Our Table in 1NF

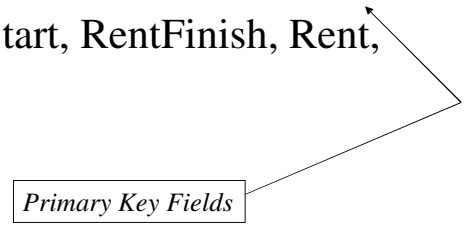
CustNo	CName	PropNo	PAddr	RntSt	RntFnsh	Rent	OwnerNo	OName
CR76	John Kay	PG4	6 Lawrence St, Elmont	7/1/10	8/31/06	700	CO40	Tina Murphy
CR76	John Kay	PG16	5 Nova Dr, East Meadow	9/1/06	9/1/08	900	CO93	Tony Shaw
CR56	Aline Stewart	PG4	6 Lawrence St, Elmont	9/1/02	6/10/04	700	CO40	Tina Murphy
CR56	Aline Stewart	PG36	2 Manor Rd Scarsdale	1/1/04	12/1/05	750	CO93	Tony Shaw
CR56	Aline Stewart	PG16	5 Nova Dr, East Meadow	8/1/06	9/1/08	900	CO93	Tony Shaw

Candidate Keys

- A candidate key for a given table is
  - unique (only one row has that value or combination of values)
  - irreducible (there is no subset of the candidate that is unique).
- Our candidate keys are:
  - (Customer_No, Property_No)
  - (Customer_No, RentStart)
  - (Property_No, RentStart)

The Customer_Rental Relation

Customer_Rental(Customer_No, Property_No, Cname, Paddress, RentStart, RentFinish, Rent, Owner_No, Oname)



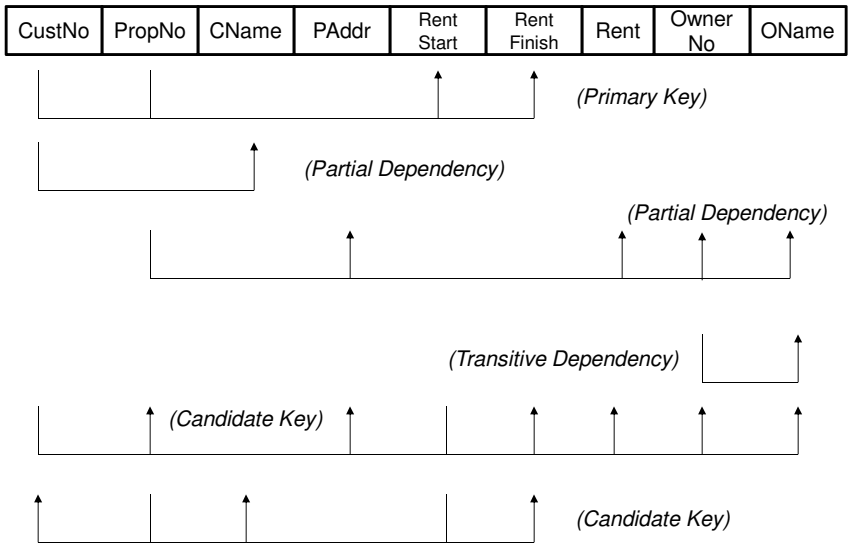
Functional Dependency

- If A and B are attributes of Relation R, B is functionally dependent on A (A→B) if each value of A is associated with one and only one value of B.
- B is fully functionally dependent on A if B is functional dependent on A and not on a proper subset of A.
- B is partially functionally dependent on A if there is some attribute that can be removed from A and the dependence still holds.

Listing All The Functional Dependencies

1. Cust_No, Prop_no → RentStart, RentFinish *(Primary Key)*
2. CustNo → Cname (Partial Dependency)
3. Prop_no → Paddress, Rent, Owner_No, Oname *(Partial Dependency)*
4. Owner_No → Oname *(Transitive Dependency)*
5. CustNo, RentStart → PropNo, Paddress, RentFinish, Rent, Owner_No, Oname *(Candidate Key)*
6. Prop_No, RentStart → CustNo, Cname, RentFinish *(Candidate Key)*

Functional Dependencies in Graphical Form



Functional Dependency in Our Table

- We have three relations with the following functional dependencies:
  - CustNo, PropNo → RentStart, RentFinish
  - CustNo → CustName
  - PropNo → Paddress, Rent, OwnerName, Oname
- Therefore, we have:
  - Customer(CustNo, Cname)
  - Rental(CustNo, PropNo, RentStart, RentFinish)
  - Property_Owner(PropNo, Paddress, Rent, OwnerNo, OName)

Second Normal Form (2NF)

- A relation is in 2NF if it is in 1NF and every non-primary key attribute is fully functionally dependent on the primary key

Our Database in 2NF

Customer Relation

CustNo	CName
CR76	John Kay
CR56	Aline Stewart

Our Database in 2NF

Rentals Relation

CustNo	PropNo	RentStart	RentFinish
CR76	PG4	7/1/10	8/31/06
CR76	PG16	9/1/06	9/1/08
CR56	PG4	9/1/02	6/10/04
CR56	PG36	1/1/04	12/1/05
CR56	PG16	8/1/06	9/1/10

Our Database in 2NF

Property-Owner Relation

PropNo	PAddr	Rent	OwnerNo	OName
PG4	6 Lawrence St, Elmont	700	CO40	Tina Murphy
PG16	5 Nova Dr, East Meadow	900	CO93	Tony Shaw
PG4	6 Lawrence St, Elmont	700	CO40	Tina Murphy
PG36	2 Manor Rd Scarsdale	750	CO93	Tony Shaw
PG16	5 Nova Dr, East Meadow	900	CO93	Tony Shaw

Transitive Dependency

- If A, B, and C are attributes of a relation R such that if A→B and B→C, then C is transitively dependent on A via B.

Third Normal Form

- A relation is in 3NF is if it is 2NF and there are no non-primary-key attributes that are transitively dependent on the primary key.

Functional Dependencies in 2NF

- Customer
  - CustNo → Cname
- Rental
  - CustNo, PropNo → RentStart, RentFinish
  - PropNo, RentStart → CustNo, RentFinish
- PropertyOwner
  - PropNo → Paddr, Rent, OwnerNo, OName
  - OwnerNo → Oname *(Oname is not f.d. on PropNo)*

Our 3NF Relations

We have 4 relations:

- Customer(CustNo, Cname)
- Rental(CustNo, PropNo, RentStart, RentFinish)
- Property_For_Rent(PropNo, Paddress, Rent, OwnerNo)
- Owner(OwnerNo, OName)

Our Database in 3NF

Customer Relation

CustNo	CName
CR76	John Kay
CR56	Aline Stewart

Our Database in 3NF

Rentals Relation

CustNo	PropNo	RentStart	RentFinish
CR76	PG4	7/1/10	8/31/06
CR76	PG16	9/1/06	9/1/08
CR56	PG4	9/1/02	6/10/04
CR56	PG36	1/1/04	12/1/05
CR56	PG16	8/1/06	9/1/10

Our Database in 3NF

Property-for-Rent Relation

PropNo	PAddr	Rent	OwnerNo
PG4	6 Lawrence St, Elmont	700	CO40
PG16	5 Nova Dr, East Meadow	900	CO93
PG36	2 Manor Rd, Scarsdale	750	CO93

Our Database in 3NF

Owner Relation

OwnerNo	OName
CO40	Tina Murphy
CO93	Tony Shaw

SQL Data Definition and Data Types

- Terminology:
  - **Table**, **row**, and **column** used for relational model terms relation, tuple, and attribute
- **`CREATE`** statement
  - Main SQL command for data definition

Schema and Catalog Concepts in SQL

- **SQL schema -** identified by a **schema name**
- Schema **elements** include tables and other constructs
- Each statement in SQL ends with a semicolon
- **`CREATE SCHEMA`** statement
  - **`CREATE SCHEMA Corvette;`**

The **`CREATE TABLE`** Command in SQL

- Specify a new relation
  - Provide name
  - Specify attributes and initial constraints
- Can optionally specify schema:
  - **`CREATE TABLE CORVETTE.CORVETTES`**
    `. . .`
    or
  - **`CREATE TABLE CORVETTES . . .`**

Creating the **Corvettes** Table

```
mysql> create table Corvettes
    -> (Vette_id        int     not null,
    -> Body_style       varchar(12) not null,
    -> miles            decimal(3, 1)   not null,
    -> year             int     not null,
    -> state            int     not null);
Query OK, 0 rows affected (0.23 sec)
mysql> alter table Corvettes add primary
    ->  key(Vette_id);
Query OK, 0 rows affected (0.36 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Creating the **Equipment** and **States** Tables

```
mysql> create table Equipment
    -> (Equip_id        int     not null,
    -> Equip            varchar(12)     not null,
    -> primary key(Equip_id));
Query OK, 0 rows affected (0.08 sec)

mysql> create table States
    -> (State_id        int     not null,
    -> State            varchar(12)     not null,
    -> primary key(State_id));
Query OK, 0 rows affected (0.13 sec)
```

Creating the **Corvettes_Equipment** Table

```
mysql> create table Corvettes_Equipment
    -> (Vette_id        int     not null,
    -> Equip            int     not null,
    -> primary key (Vette_id, Equip));
Query OK, 0 rows affected (0.16 sec)

mysql>
```

Load Data

```
mysql> load data local infile 'Equipment.txt' into
table Equipment;
Query OK, 6 rows affected (0.06 sec)
Records: 6  Deleted: 0  Skipped: 0  Warnings: 0
```

- The file must be located in the home directory of mysql (which is in

C:\Program Files\MySQL\MySQL Server 5.5\bin

Populating the **Corvettes** Table

```
mysql> load data local infile 'Corvettes.txt' into
table Corvettes;
Query OK, 9 rows affected, 7 warnings (0.03 sec)
Records: 9  Deleted: 0  Skipped: 0  Warnings: 7

mysql> insert into Corvettes values (10,
'hatchback', 50.0, 1995, 7);
Query OK, 1 row affected (0.09 sec)
```

Populating the **States** and **Corvettes_Equipment** Tables

```
mysql> load data local infile 'States.txt' into
table States;
Query OK, 10 rows affected (0.09 sec)
Records: 10  Deleted: 0  Skipped: 0  Warnings: 0

mysql> load data local infile
'Corvettes_Equipment.txt' into table Corvettes_Equ
ipment;
Query OK, 23 rows affected (0.09 sec)
Records: 23  Deleted: 0  Skipped: 0  Warnings: 0
```

The **SELECT** SQL Command

- Syntax:

  SELECT *ColumnNames* **FROM** *TableNames* **[WHERE** *condition***];**

SELECT – Some Examples

```
mysql> select * from corvettes;
+----------+-------------+-------+------+-------+
| Vette_id | Body_style  | miles | year | state |
+----------+-------------+-------+------+-------+
|        1 | coupe       |  18.0 | 1997 |     4 |
|        2 | hatchback   |  58.0 | 1996 |     7 |
|        3 | convertible |  13.5 | 2001 |     1 |
|  … …     |             |       |      |       |
|        7 | coupe       |  55.0 | 1979 |    10 |
|        8 | convertible |  17.0 | 1999 |     5 |
|        9 | hardtop     |  17.0 | 2000 |     5 |
|       10 | hatchback   |  50.0 | 1995 |     7 |
+----------+-------------+-------+------+-------+
10 rows in set (0.00 sec)
```

```
mysql> select body_style from Corvettes;
+-------------+
| body_style  |
+-------------+
| coupe       |
| hatchback   |
| convertible |
| hatchback   |
| hatchback   |
| hardtop     |
| coupe       |
| convertible |
| hardtop     |
| hatchback   |
+-------------+
10 rows in set (0.00 sec)
```

- mysql> select body_style from Corvettes where year > 1994;
- +-------------+
- | body_style |
- +-------------+
- | coupe |
- | hatchback |
- | convertible |
- | hatchback |
- | hardtop |
- | convertible |
- | hardtop |
- | hatchback |
- +-------------+
- 8 rows in set (0.00 sec)

Joining – An Example

```
mysql> select Corvettes.Vette_id,
    -> Corvettes.Body_Style, Corvettes.Miles,
    -> Corvettes.Year, Corvettes.State,
    -> Equipment.Equip
    -> from Corvettes, Equipment,
    -> Corvettes_Equipment
    -> where Corvettes.Vette_id =
    -> Corvettes_Equipment.Vette_id
    -> and corvettes_equipment.Equip =
    -> Equipment.Equip_id
    -> and Equipment.Equip = 'CD';
```

```
+----------+-------------+-------+------+-------+-------+
| Vette_id | Body_Style  | Miles | Year | State | Equip |
+----------+-------------+-------+------+-------+-------+
|        1 | coupe       |  18.0 | 1997 |     4 | CD    |
|        2 | hatchback   |  58.0 | 1996 |     7 | CD    |
|        8 | convertible |  17.0 | 1999 |     5 | CD    |
|        9 | hardtop     |   0.0 |   17 |  2000 | CD    |
|       10 | hatchback   |  50.0 | 1995 |     7 | CD    |
+----------+-------------+-------+------+-------+-------+
5 rows in set (0.00 sec)
```

DELETE SQL Command

- Syntax:

```
DELETE FROM TableName
WHERE PrimaryKey = Value;
```

- Example

```
mysql> delete from corvettes where vette_id = 27;
Query OK, 0 rows affected (0.00 sec)
```

DROP SQL Command

- Syntax:

```
Drop (TABLE | DATABASE) [IF EXISTS] Name;
```

- Example

```
mysql> drop table if exists transmission;
Query OK, 0 rows affected, 1 warning (0.00
sec)
```

Normalization

- We want our database to be a clear representation of the data, its relationships and constraints
- We can identify relationship using a technique called ***normalization***.
- Normalization is a bottom-up technique where we examine the relationship between attributes and reconfigure the tables accordingly.

Our Example

- The DreamHome Customer Rental Details form holds details about property rented by a given customer.
  - To simplify things, we will assume that a renter rents a given property once and only one property at a time.

Our Original Table

CustNo	Cname	PropNo	PAddr	RntSt	RntFnsh	Rent	OwnerNo	OName
CR76	John Kay	PG4	6 Lawrence St, Elmont	7/1/10	8/31/06	700	CO40	Tina Murphy
		PG16	5 Nova Dr, East Meadow	9/1/06	9/1/08	900	CO93	Tony Shaw
CR56	Aline Stewart	PG4	6 Lawrence St, Elmont	9/1/02	6/10/04	700	CO40	Tina Murphy
		PG36	2 Manor Rd Scarsdale	8/1/04	12/1/05	750	CO93	Tony Shaw
		PG16	5 Nova Dr, East Meadow	8/1/06	9/1/10	900	CO93	Tony Shaw

First Normal Form (1NF)

- Unnormalized – A table with one or more repeating groups.
- First Normal Form (1NF) – A relation in which the intersection of each row and column contains one and only one value

Repeating Groups

- Any collection of attributes that repeat provides a complication for a database, both in terms of storing it (how many repeating groups would you allow for) as well as querying them.
- It is necessary to recognize them so we can eliminate them.
- E.g.,

  Repeating Group = (Property_no, Paddress, RentStart, RentFinish, Rent, Owner_No, OName)

Our Table in 1NF

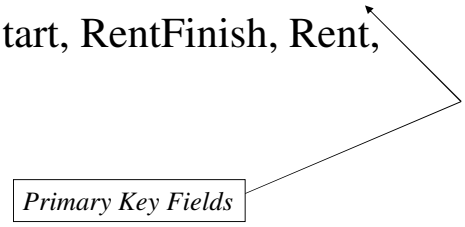
CustNo	CName	PropNo	PAddr	RntSt	RntFnsh	Rent	OwnerNo	OName
CR76	John Kay	PG4	6 Lawrence St, Elmont	7/1/10	8/31/06	700	CO40	Tina Murphy
CR76	John Kay	PG16	5 Nova Dr, East Meadow	9/1/06	9/1/08	900	CO93	Tony Shaw
CR56	Aline Stewart	PG4	6 Lawrence St, Elmont	9/1/02	6/10/04	700	CO40	Tina Murphy
CR56	Aline Stewart	PG36	2 Manor Rd Scarsdale	1/1/04	12/1/05	750	CO93	Tony Shaw
CR56	Aline Stewart	PG16	5 Nova Dr, East Meadow	8/1/06	9/1/08	900	CO93	Tony Shaw

Candidate Keys

- A candidate key for a given table is
  - unique (only one row has that value or combination of values)
  - irreducible (there is no subset of the candidate that is unique).
- Our candidate keys are:
  - (Customer_No, Property_No)
  - (Customer_No, RentStart)
  - (Property_No, RentStart)

The Customer_Rental Relation

Customer_Rental(Customer_No, Property_No, Cname, Paddress, RentStart, RentFinish, Rent, Owner_No, Oname)



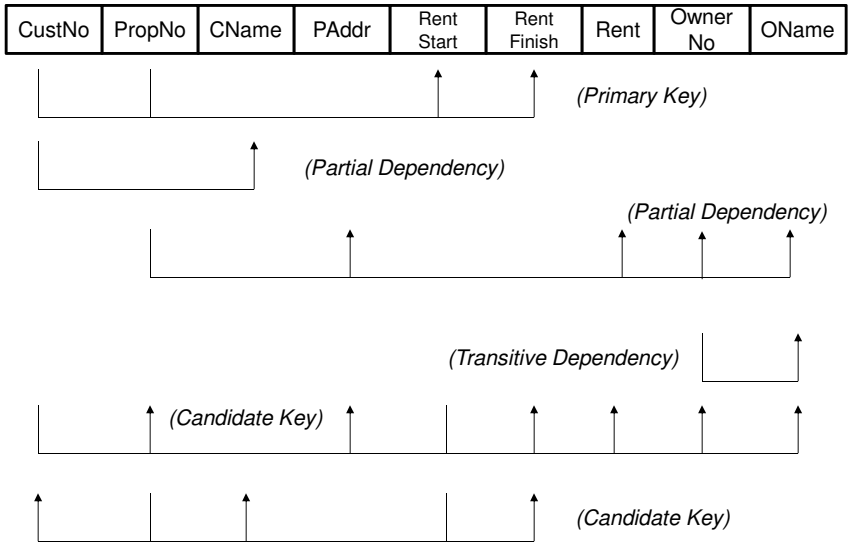
Functional Dependency

- If A and B are attributes of Relation R, B is functionally dependent on A (A→B) if each value of A is associated with one and only one value of B.
- B is fully functionally dependent on A if B is functional dependent on A and not on a proper subset of A.
- B is partially functionally dependent on A if there is some attribute that can be removed from A and the dependence still holds.

Listing All The Functional Dependencies

1. Cust_No, Prop_no → RentStart, RentFinish *(Primary Key)*
2. CustNo → Cname (Partial Dependency)
3. Prop_no → Paddress, Rent, Owner_No, Oname *(Partial Dependency)*
4. Owner_No → Oname *(Transitive Dependency)*
5. CustNo, RentStart → PropNo, Paddress, RentFinish, Rent, Owner_No, Oname *(Candidate Key)*
6. Prop_No, RentStart → CustNo, Cname, RentFinish *(Candidate Key)*

Functional Dependencies in Graphical Form



Functional Dependency in Our Table

- We have three relations with the following functional dependencies:
  - CustNo, PropNo → RentStart, RentFinish
  - CustNo → CustName
  - PropNo → Paddress, Rent, OwnerName, Oname
- Therefore, we have:
  - Customer(CustNo, Cname)
  - Rental(CustNo, PropNo, RentStart, RentFinish)
  - Property_Owner(PropNo, Paddress, Rent, OwnerNo, OName)

Second Normal Form (2NF)

- A relation is in 2NF if it is in 1NF and every non-primary key attribute is fully functionally dependent on the primary key

Our Database in 2NF

Customer Relation

CustNo	CName
CR76	John Kay
CR56	Aline Stewart

Our Database in 2NF

Rentals Relation

CustNo	PropNo	RentStart	RentFinish
CR76	PG4	7/1/10	8/31/06
CR76	PG16	9/1/06	9/1/08
CR56	PG4	9/1/02	6/10/04
CR56	PG36	1/1/04	12/1/05
CR56	PG16	8/1/06	9/1/10

Our Database in 2NF

Property-Owner Relation

PropNo	PAddr	Rent	OwnerNo	OName
PG4	6 Lawrence St, Elmont	700	CO40	Tina Murphy
PG16	5 Nova Dr, East Meadow	900	CO93	Tony Shaw
PG4	6 Lawrence St, Elmont	700	CO40	Tina Murphy
PG36	2 Manor Rd Scarsdale	750	CO93	Tony Shaw
PG16	5 Nova Dr, East Meadow	900	CO93	Tony Shaw

Transitive Dependency

- If A, B, and C are attributes of a relation R such that if A→B and B→C, then C is transitively dependent on A via B.

Third Normal Form

- A relation is in 3NF is if it is 2NF and there are no non-primary-key attributes that are transitively dependent on the primary key.

Functional Dependencies in 2NF

- Customer
  - CustNo → Cname
- Rental
  - CustNo, PropNo → RentStart, RentFinish
  - PropNo, RentStart → CustNo, RentFinish
- PropertyOwner
  - PropNo → Paddr, Rent, OwnerNo, OName
  - OwnerNo → Oname *(Oname is not f.d. on PropNo)*

Our 3NF Relations

We have 4 relations:

- Customer(CustNo, Cname)
- Rental(CustNo, PropNo, RentStart, RentFinish)
- Property_For_Rent(PropNo, Paddress, Rent, OwnerNo)
- Owner(OwnerNo, OName)

Our Database in 3NF

Customer Relation

CustNo	CName
CR76	John Kay
CR56	Aline Stewart

Our Database in 3NF

Rentals Relation

CustNo	PropNo	RentStart	RentFinish
CR76	PG4	7/1/10	8/31/06
CR76	PG16	9/1/06	9/1/08
CR56	PG4	9/1/02	6/10/04
CR56	PG36	1/1/04	12/1/05
CR56	PG16	8/1/06	9/1/10

Our Database in 3NF

Property-for-Rent Relation

PropNo	PAddr	Rent	OwnerNo
PG4	6 Lawrence St, Elmont	700	CO40
PG16	5 Nova Dr, East Meadow	900	CO93
PG36	2 Manor Rd, Scarsdale	750	CO93

Our Database in 3NF

Owner Relation

OwnerNo	OName
CO40	Tina Murphy
CO93	Tony Shaw

carsdata.html

```
<!-- carsdata.html
  Uses a form to collect a query against the cars
  database.
  Calls the PHP script access_cars.php to perform
  the given query and display the results
  -->

<!DOCTYPE html>
<html lang="en">
  <head>
    <title> Access to the cars database </title>
    <meta charset = "utf-8" />
  </head>
</html>
```

```
<body>
    <p>
      Please enter your query:
      <br />
      <form action = "access_cars.php"
            method = "post">
        <textarea rows = "2" cols = "80"
                  name = "query">
        </textarea>
        <br /> <br />
        <input type = "reset" value = "Reset" />
        <input type = "submit"
               value = "Submit request" />
      </form>
    </p>
  </body>
</html>
```

access_cars.php

```
<!-- access_cars.php
     A PHP script to access the cars database
     through MySQL
     -->

<!DOCTYPE html>
<html lang = "en">
  <head>
    <title>
      Access the cars database with MySQL
    </title>
    <meta charset = "utf-8" />
  </head>
```

```
  <body>
    <?php

  // Connect to MySQL
    $db  = mysqli_connect("localhost",
                          "root", "", "cars");
    if (mysqli_connect_errno()) {
      print "Connect failed:: "
                    . mysqli_connect_error();
      exit();
    }

  // Get the query and clean it up (delete leading
  // and trailing whitespace and remove backslashes
  // from magic_quotes_gpc)
```

```
    $query = $_POST['query'];
    trim($query);
    $query = stripslashes($query);

  //  Display the query, after fixing html
  //  characters
    $query_html = htmlspecialchars($query);
    print "<p> The query is: " . $query_html
          . "</p>";

  // Execute the query
    $result - mysqli_query($db, $query);
```

```
    if (!$result)       {
      print "Error - the query could not be
executed" .
        mysqli_error();
      exit;
    }

  // Display the reulst in a table
    print "<table> <caption> <h2> Query results
</h2> </caption>";
    print "<tr align = 'center'>";

  // Get the number of rows in the result
    $num_rows = mysqli_num_rows($result);
```

```
// If there are rows in the result, put them in an
// HTML table
  if ($num_rows > 0)  {
    $row = mysqli_fetch_assoc($result);
    $num_fields = mysqli_num_fields($result);
  }

// Produce the column labels
  $keys = array_keys($rows);
  for ($index = 0;  $index , $num_fields;
                                        $index++)
    print ("<th>" . $keys[$index] . "</th>");
    print ("</tr>");
```

```
// Output the values of the fields in the rows
  for ($row_num = 0;  $row_num < $num_rows;
                                $row_num++)      {
    print "<tr>";
    $row = mysqli_fetch_assoc($result);
  }
}
else {
  print "There were no such rows in the table <br />";
}
print "</table>";
 ?>
</body>
</html>
```