The Relational Model and the Structured Query Language (SQL)

Abdu Alawini

University of Illinois at Urbana-Champaign

CS411: Database Systems

June 12, 2020

Some slides are adopted from D. Maier, S. Davidson and A. Parameswaran

Leaning Objectives

After this lecture, you should be able to:

- Define a **data model**
- Distinguish among different **data models**
- Define the **relational data model**
- Articulate the **basic terminologies** of the **relational** data **model** (from a practical prospective)
- Define **Primary and Foreign keys** for a relational table
- Distinguish between **Schema** and **Instance**
- Write **single-relation SQL** queries

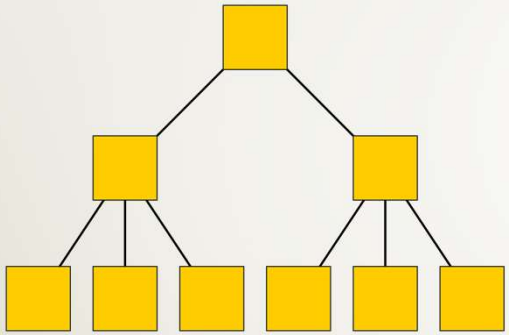
What is a Data Model?

A data model is a notation for **describing data or information**. The description generally consists of three parts:

1. *Structure of the data:*
   - data structures used to implement data in the computer (physical data model)
2. *Operations on the data:*
   - limited set of queries (operations that retrieve information) and modifications (operations that change the database).
3. *Constraints on the data:*
   - ways to describe limitations on what the data can be. These constraints often come from the real-world application requirements

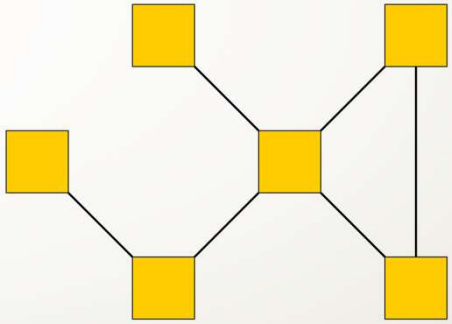
Types of Data Models

Hierarchal Model



Src: Wikipedia

Network Model



Src: Wikipedia

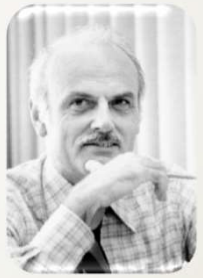
Semi-structured Model (XML)

```
<note>
  <date>2017-11-08</date>
  <hour>08:30</hour>
  <to>Raj</to>
  <from>Ravi</from>
  <body>Meeting at 8am.</body>
</note>
```

Outline

- ✓ Data Models
- Relational Database Model
  - Basic Concepts and Terminology
  - Keys and Foreign Keys
  - Schema Specifications
- SQL query language
  - Single-Relation Queries

The Relational Data Model



It all began with a breakthrough paper by E.F. Codd in 1970:
"A relational model of data for large shared data banks".
Communications of the ACM 13 (6): 377
Codd's insights:

- Separate physical implementation from logical
- Model the data independently from how it will be used (accessed, printed, etc.)
  - Describe the data minimally and mathematically
    - A relation describes an association between data items – tuples with attributes
    - We generally think of tables and rows, but that's somewhat imprecise
  - Use standard mathematical (logical) operations over the data – these are the relational algebra or relational calculus

Database from a user Perspective

- We'll assume that a DB has been already been implemented and loaded with data
- Our roles is to query/modify the data using SQL
- But before that, we need to learn the basics of relational model (from a practical point of view)

Introduction to Relational Databases from a Practical Point of View

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Imagine that this table (or relation) has been defined to help keep track of bank accounts.

Table Structure

The *name* of the table

Account

The name of the *columns* (*attributes*)

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Table Schema

The *schema* for the table

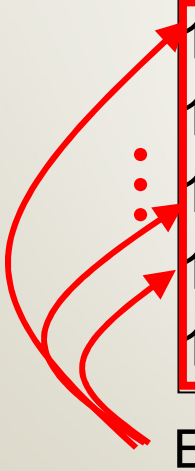
Account			
Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

The schema sets the structure of the table. You can think of the schema as the *definition* of the table. (Note, the schema specifies more information than what is shown.)

Table Rows

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking



Each entry in the table is called a *row* (*tuple*).

Sometimes an entry in the table is called a record.

Table Instance

An *instance* of the table...

the current contents or data in the table.

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Another Table Instance

Another *instance* of the table

(two rows added, one (103) deleted)

Account

Number	Owner	Balance	Type
101	J. Smith	1,000.00	checking
102	W. Wei	2,000.00	checking
104	M. Jones	1,000.00	checking
105	H. Martin	10,000.00	checking
107	W. Yu	7,500.00	savings
109	R. Jones	432.55	checking

new

Intension vs. Extension

The *intension* of the table

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

The *extension* of the table. Also called the *extent*.

"Size" of a Table

Degree or arity of a table is the number of columns

Degree of this relation (or table) is 4 because there are 4 attributes

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Cardinality of this instance is 5 (because there are 5 rows)

Cardinality of a table = the number of rows in the current instance

Outline

- ✓ Data Models
- Relational Database Model
  - ✓ Basic Concepts and Terminology
  - Keys and Foreign Keys
  - Schema Specifications
- SQL query language
  - Single-Relation Queries

Database (One or More Tables)

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

Deposit	AcctNo	Transaction-id	Date	Amount
	102	1	10/22/00	500.00
	102	2	10/29/00	200.00
	104	3	10/29/00	1000.00
	105	4	11/02/00	10,000.00

Check	AcctNo	Check-number	Date	Amount
	101	924	10/23/00	125.00
	101	925	10/24/00	23.98

Table Keys

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

Deposit	AcctNo	Transaction-id	Date	Amount
	102	1	10/22/00	500.00
	102	2	10/29/00	200.00
	104	3	10/29/00	1000.00
	105	4	11/02/00	10,000.00

Check	AcctNo	Check-number	Date	Amount
	101	924	10/23/00	125.00
	101	925	10/24/00	23.98

Each table has a key…. where the values must be unique.

Table Keys (cont.)

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

Deposit	AcctNo	Transaction-id	Date	Amount
	102	1	10/22/00	500.00
	102	2	10/29/00	200.00
	104	3	10/29/00	1000.00
	105	4	11/02/00	10,000.00

Check	AcctNo	Check-number	Date	Amount
	101	924	10/23/00	125.00
	101	925	10/24/00	23.98

Key may consist of one column or two (or more) columns.

Connections between Tables

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

Deposit	AcctNo	Transaction-id	Date	Amount
	102	1	10/22/00	500.00
	102	2	10/29/00	200.00
	104	3	10/29/00	1000.00
	105	4	11/02/00	10,000.00
	106	5	12/05/00	555.00

Is this legal?

If not, how do we prevent it from happening?

Foreign Key

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

Deposit	AcctNo	Transaction-id	Date	Amount
	102	1	10/22/00	500.00
	102	2	10/29/00	200.00
	104	3	10/29/00	1000.00
	105	4	11/02/00	10,000.00
	106	5	12/05/00	555.00

We say that Deposit.AcctNo is a *foreign key* that *references* Account.Number. If the DBMS enforces this constraint, we have ***referential integrity***.

Foreign Keys (cont.)

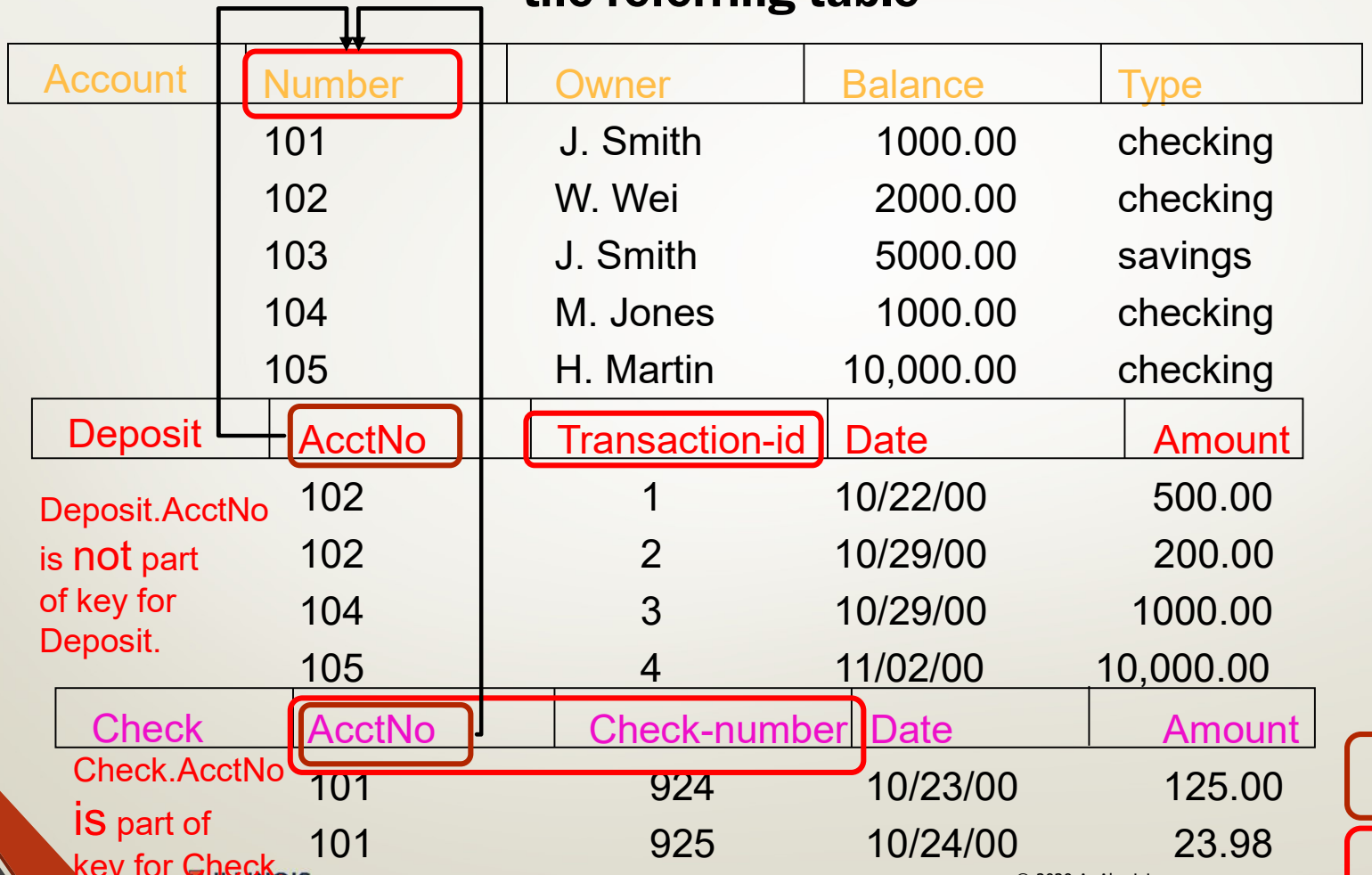
Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

Check	AcctNo	Check-number	Date	Amount
	101	924	10/23/98	125.00
	101	925	10/24/98	23.98

Are there any foreign keys in the Check table?

Yes, Check.AcctNo is a foreign key that references Account.Number.

Foreign keys might or might not be part of the key for the referring table



Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

Deposit	AcctNo	Transaction-id	Date	Amount
	102	1	10/22/00	500.00
	102	2	10/29/00	200.00
	104	3	10/29/00	1000.00
	105	4	11/02/00	10,000.00

Check	AcctNo	Check-number	Date	Amount
	101	924	10/23/00	125.00
	101	925	10/24/00	23.98

Deposit.AcctNo is not part of key for Deposit.

Check.AcctNo is part of key for Check

Foreign Key

Primary Key

Outline

- ✓ Data Models
- Relational Database Model
  - ✓ Basic Concepts and Terminology
  - ✓ Keys and Foreign Keys
  - Schema Specifications
- SQL query language
  - Single-Relation Queries

Specification of a Database Schema

- Select the tables, with a name for each table.
- Select columns for each table and give the domain for each column.
- Specify the key(s) for each table.
- Specify all appropriate foreign keys.

There can be more than one key for a table.

Database Domains for Columns

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	...			

For every column of every table, the schema specifies allowable values. For example,

Number must be a 3-digit number

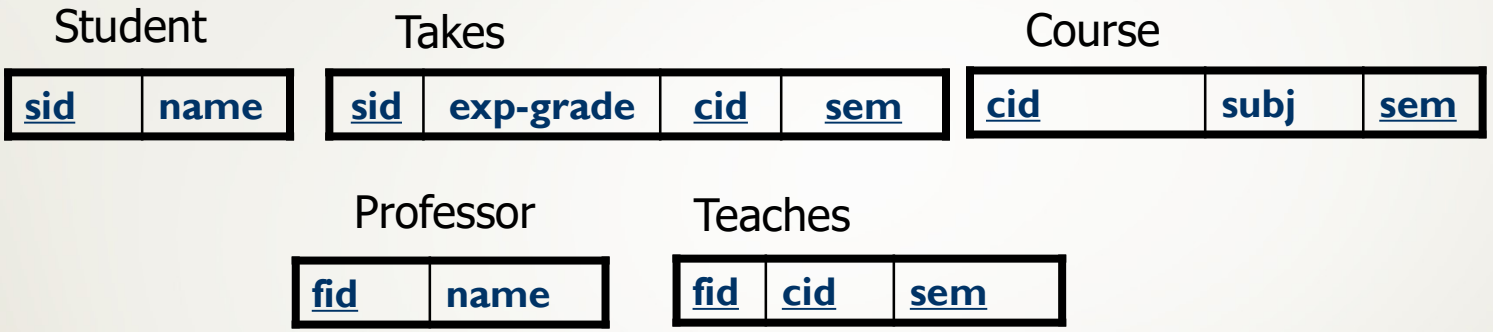
Owner must be a 30-character string

Type must be “checking” or “savings”

The set of allowable values for a column is called the domain of the column.

Example Database Schema

(Keys are underlined. Each table has one key.)



In relational DBs, we use *relation(attribute:domain)*

```
STUDENT(sid:int, name:string)
Takes(sid:int, exp-grade:char[2], cid:string, sem:char[3])
COURSE(cid:string, subj:string, sem:char[3])
Teaches(fid:int, cid:string, sem:char[3])
PROFESSOR(fid:int, name:string)
```

Outline

- ✓ Data Models
- Relational Database Model
  - ✓ Basic Concepts and Terminology
  - ✓ Keys and Foreign Keys
  - ✓ Schema Specifications
- SQL query language
  - Single-Relation Queries

SQL: Structured Query Language

The standard language for relational data

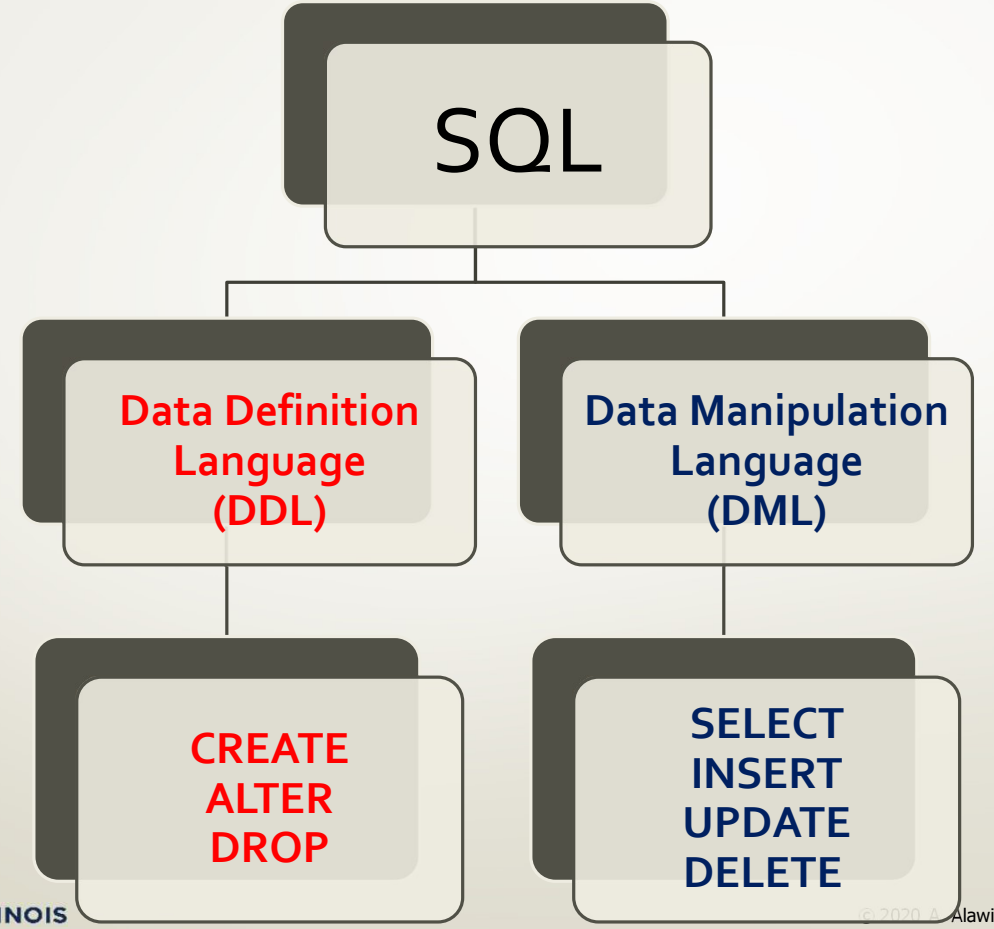
- Invented by folks at IBM, esp. Don Chamberlin
- Actually not a particularly elegant language…
- Beat a more elegant competing standard, QUEL, from Berkeley

Separated into

- DDL (data definition language)
- DML (data manipulation language)

DML based on relational calculus, which we discuss later

SQL – the language we use to talk to the Database Management System



SQL (cont.)

SQL is a standard…
and there have been a series of SQL standards: 1986, 1989, 1992 (SQL2), 1999 (SQL3), …, SQL:2011

But DBMS products differ in how much of the standard they support … and how many extra features they have.

Single-Relation Queries

Structured Query Language

Single-Relation Query Form

- The principal form of a single-relation query is:

SELECT desired attributes

FROM one table (relation)

WHERE condition about tuples of the table

Example Query

Account	Number	Owner	Balance	Type
	101	J. Smith	1000.00	checking
	102	W. Wei	2000.00	checking
	103	J. Smith	5000.00	savings
	104	M. Jones	1000.00	checking
	105	H. Martin	10,000.00	checking

```
SELECT    Number, Owner
FROM      Account
WHERE     Type = "savings";
```

Number	Owner
103	J. Smith

How a Single-Relation SQL query is evaluated

Third, the SELECT clause tells us which columns to keep in the query answer.

```
SELECT    Number, Owner
FROM      Account
WHERE     Type = "savings"
```

First, the FROM clause tells us the input tables.

Second, the WHERE clause is evaluated for all rows from the input table.

Renaming Attributes

- If you want the result to have different attribute names, use "AS <new name>" to rename an attribute.
- Example based on Account(number, owner, balance, type):

```
SELECT Number AS Acc_Num, Owner
FROM Accounts
WHERE Type = "savings";
```

Acc_Num	Owner
103	J. Smith

Another Database Schema

- We will be using the following database schema as a running example.
  - Underline indicates key attributes.

Drinks(<u>name</u>, manf)

Cafe(<u>name</u>, addr, license)

Customers(<u>name</u>, addr, phone)

Likes(<u>customer</u>, <u>drink</u>)

Sells(<u>cafe</u>, <u>drink</u>, price)

Frequents(<u>customer</u>, <u>cafe</u>)

Expressions in SELECT Clauses

- Any expression that makes sense can appear as an element of a SELECT clause.

- Example: from Sells (cafe, drink, price):

```
SELECT cafe, drink, price * 120 AS priceInYen
FROM Sells;
```

Complex Conditions in WHERE Clause

- From Sells (cafe, drink, price), find the price Caffe bene charges for Mocha:

```
SELECT price
FROM Sells
WHERE cafe = 'Caffe bene' AND
      drink = 'Mocha';
```

WHERE Clause Syntax

What you can use in WHERE:

- attribute names of the relation(s) used in the FROM.
- comparison operators: =, <>, <, >, <=, >=
- apply arithmetic operations: stockprice*2
- operations, comparisons on strings
- pattern matching: s LIKE p
- special stuff for comparing dates and times.

See the textbook for more…

Then, create bigger expressions using Boolean connectives

Syntax: Patterns and "LIKE"

- WHERE clauses can have conditions in which a string is compared with a pattern, to see if it matches.

- General form: <Attribute> LIKE <pattern>
  or <Attribute> NOT LIKE <pattern>

- Pattern is a quoted string with
  - % "any string"
  - _ "any character."

Example

- From Customers(name, addr, phone) find the customers that have phone numbers with middle three numbers 555:

```
SELECT name
FROM Customers
WHERE phone LIKE '%555-_ _ _ _';
```

Ordering the results

```
SELECT *
FROM  Account
WHERE Name LIKE 'J%'
ORDER BY Balance
```

- Ordering is ascending, unless you specify the DESC keyword.
- Ties are broken by the second attribute on the ORDER BY list, etc.

Summary

- A data model is an **abstract model** that **organizes elements of data** and standardizes how they **relate to one another** and to properties of the real world entities [wikipedia].
- There are several types of data models: Network, Hierarchal,…
  - We will study the relational, NoSQL and graph models.
- Relational database model
  - Data is structured as relations
  - Defines a limited set of operations (query and modification) to interact with the data
- Structured Query Language (SQL) is a declarative language (standard) that enable users/applications to query and modify the database.