CS430/630
Database Management Systems
Fall, 2019

Betty O’Neil

University of Massachusetts at Boston

People & Contact Information

- Instructor: Prof. Betty O'Neil
  - **Email:** eoneil AT cs DOT umb DOT edu (preferred contact)
  - **Web:** http://www.cs.umb.edu/~eoneil
  - **Office:** Science Building, 3rd Floor, Room 169 (S-3-169)

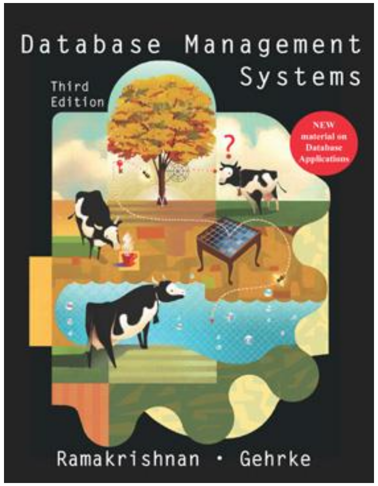
- TA: Brian Le
  - **Email**: Brian.Le002 AT umb DOT edu

Course Info

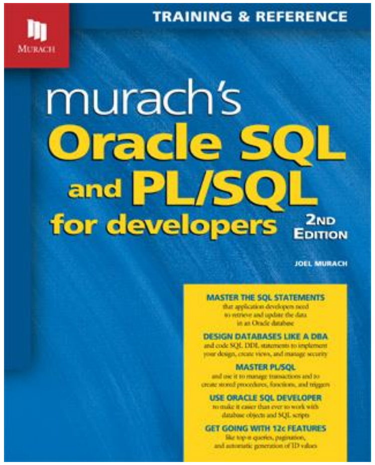
- Lecture Hours
  - Mon and Wed, 7:00-8:15pm
- Office Hours
  - Mon and Wed 3:30-3:45pm, 5:30-6:45 in S/3/169
  - By appointment (send email)
- Class URL
  - http://www.cs.umb.edu/cs630/

Textbooks

- Textbooks
  - *Database Management Systems*, 3rd Edition by Ramakrishnan and Gehrke


Database Management Systems
Third Edition
NEW material on Database Applications
Ramakrishnan • Gehrke

  - J. Murach (M), "*Oracle SQL and PL/SQL for Developers*", 2nd edition.
    - Please purchase a print copy (under $50).
    - Note: No electronic devices are allowed in open-book exams. No book: sit in front to share mine.


TRAINING & REFERENCE
MURACH
murach's
Oracle SQL and PL/SQL for developers
2ND EDITION
JOEL MURACH
MASTER THE SQL STATEMENTS
DESIGN DATABASES LIKE A DBA
MASTER PL/SQL
USE ORACLE SQL DEVELOPER
GET GOING WITH 12c FEATURES

Prerequisites

- Data Structures and Algorithms, Java
  - CS310 officially, but CS210 or equivalent is acceptable
  - If an undergraduate, take cs310 before or with cs430/cs630, not after (rule against out-of-order courses)

- Familiarity with UNIX/Linux OS
  - Exercises will be executed on Oracle 12c server running on a Linux machine in the CS dept (DBS3 Machine)
  - Access is via sqlplus on pe07.cs.umb.edu, another Linux machine, and later, JDBC from anywhere on the Internet.

Grading

- Final exam (150 points) – open books
- Midterm (100 points) – open books
  - Open books does NOT include electronic devices!

- Approximately 6 homework assignments
  - 10 points each, drop lowest score.
  - Assignments for CS630 will have additional questions
  - Assignments are individual – submit your own work only!
  - No plagiarism! See student code of conduct

Course Materials

- Class URL

  http://www.cs.umb.edu/cs630/

- Make sure you obtain a cs630 Unix course account
  - Apply process: See documentation here.
    - You need to apply even if you already have an account
    - You need a "UNIX account" for cs630 (one group for both cs430 and cs630)
    - Once you have your class account, you will be able to login on our server pe07.
    - Go to S/3/158 if you need assistance

University Policies

ACADEMIC CONDUCT:

It is the expressed policy of the University that every aspect of academic life—not only formal coursework situations, but all relationships and interactions connected to the educational process—shall be conducted in an absolutely and uncompromisingly honest manner.The University presupposes that any submission of work for academic credit indicates that the work is the student's own and is in compliance with University policies. In cases where academic dishonesty is discovered after completion of a course or degree program, sanctions may be imposed retroactively, up to and including revocation of the degree. Students are required to adhere to the Code of Student Conduct, including requirements for academic honesty, delineated in the University of Massachusetts Boston Bulletin, found at: http://www.umb.edu/life_on_campus/policies/community/code.

University Policies

Accommodations:

The University of Massachusetts Boston is committed to providing reasonable academic accommodations for all students with disabilities. This syllabus is available in alternate format upon request. Students with disabilities who need accommodations in this course must contact the instructor to discuss needed accommodations. Accommodations will be provided after the student has met with the instructor to request accommodations. Students must be registered with the Ross Center for Disability Services, UL 211, www.ross.center@umb.edu 617.287.7430 before requesting accommodations from the instructor.

Course Overview

- Relational Data Model
- Relational Algebra
- Structured Query Language
  - The most important part of the course
- Conceptual design – the ER model
- Database application development
  - Java/JDBC, PL/SQL
- Design Theory
- Database Security
- If time, some coverage of NoSQL Databases

What is a Relational DBMS (RDBMS)?

- Specialized software that provides:
  - Uniform and transparent access to data
    - Application-independence
    - Application/user is oblivious to internal data organization
    - Data organization may change, but applications need not change
  - Efficient access to data
    - Fast search capabilities, indexing
  - Data consistency
    - E.g., cannot delete student record if grade records still in DBMS
  - Concurrent access to data
  - Persistent storage and recovery from failure
  - Security

Why study databases?

- Databases are ubiquitous
  - Behind all web service providers there is a DBMS
    - In some cases a very large-scale one (RDBMSs at EBay, Facebook, etc.)
    - Amazon uses RDBMS for money handling
  - Corporations use RDBMS for business processes, HR, etc
  - Scientific computing relies on very large amounts of data
    - Humane genome data
    - Biochemistry data (protein sequences)
    - Astronomy data
    - High-energy physics
- DBAs are very well–paid!
  - And even in other IT areas, DBMS skills are a must
  - A "full stack developer" needs to know these topics.

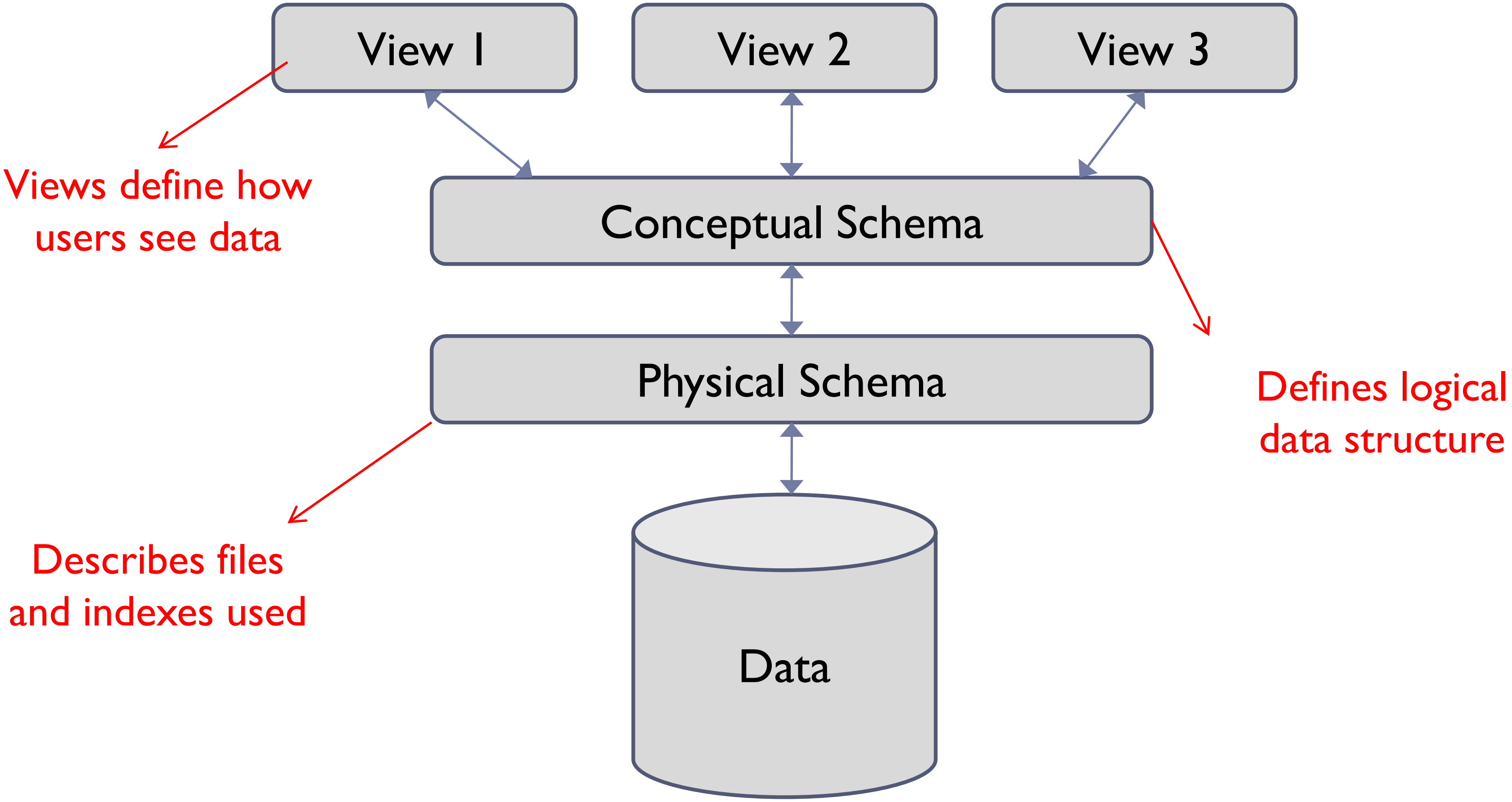
A bit of history …

- First data stores were file systems
  - Does not conform to transparency and uniformity desiderata
  - Search (within file) most often linear
  - Not portable
  - Doesn't handle concurrency properly
    - Sequential access only
- Early DBMS appeared in the 60s
  - Driven by banking and airline industry
  - Relatively small record size, and many concurrent accesses
  - Two prominent models: hierarchical model (tree) and network model (graph)
    - Lack of support for high-level query languages

A bit of history (contd.)

- Relational Databases
  - Major breakthrough, paper written by Codd (1970)
  - *Relations* (tables) with rows (records) and columns (fields)
    - Relationships and constraints among tables
  - Structured Query Language (SQL): high-level, declarative
    - Data definition/ manipulation language
  - Fast search – use of index structures
  - Data access language independent from internal organization
- Newer paradigms
  - Data Stream Management Systems
  - "No-SQL" Databases, Big Data, MapReduce

Levels of Abstraction



View 1

View 2

View 3

Conceptual Schema

Physical Schema

Data

Views define how users see data

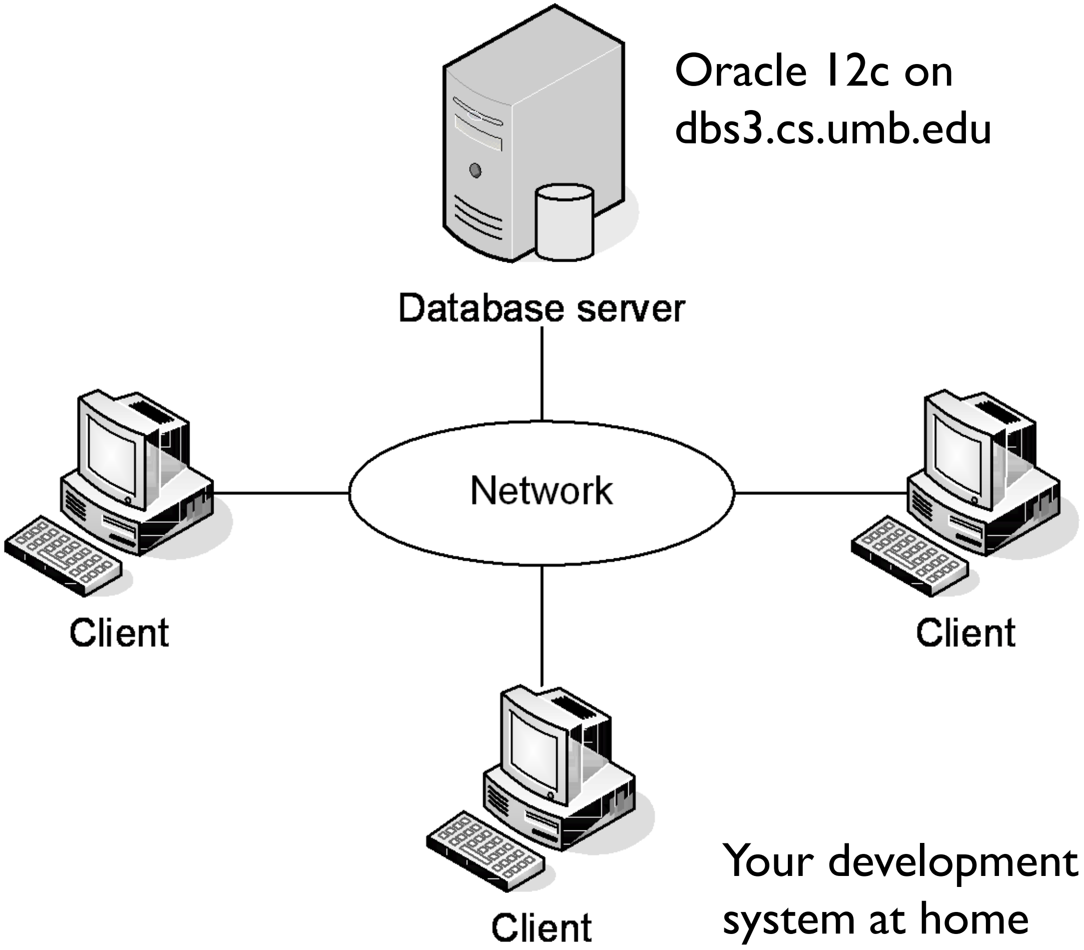
Defines logical data structure

Describes files and indexes used

Database Management Systems 3rd ed, Ramakrishnan and Gehrke

System architecture

Clients access the database server over the Internet

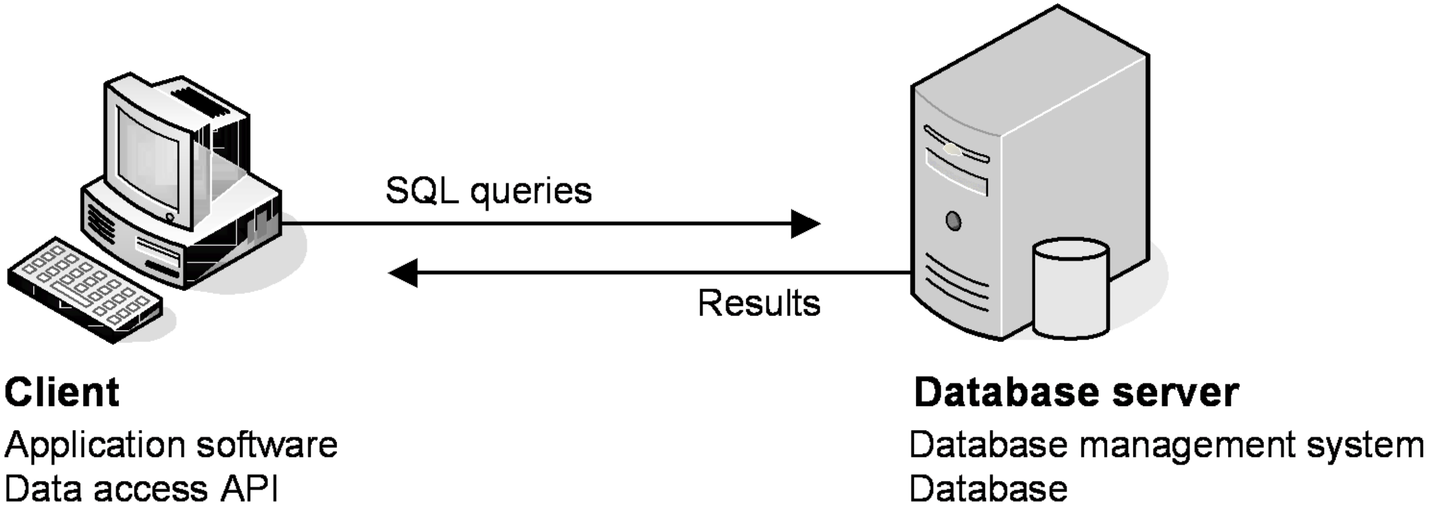


Oracle 12c on
dbs3.cs.umb.edu

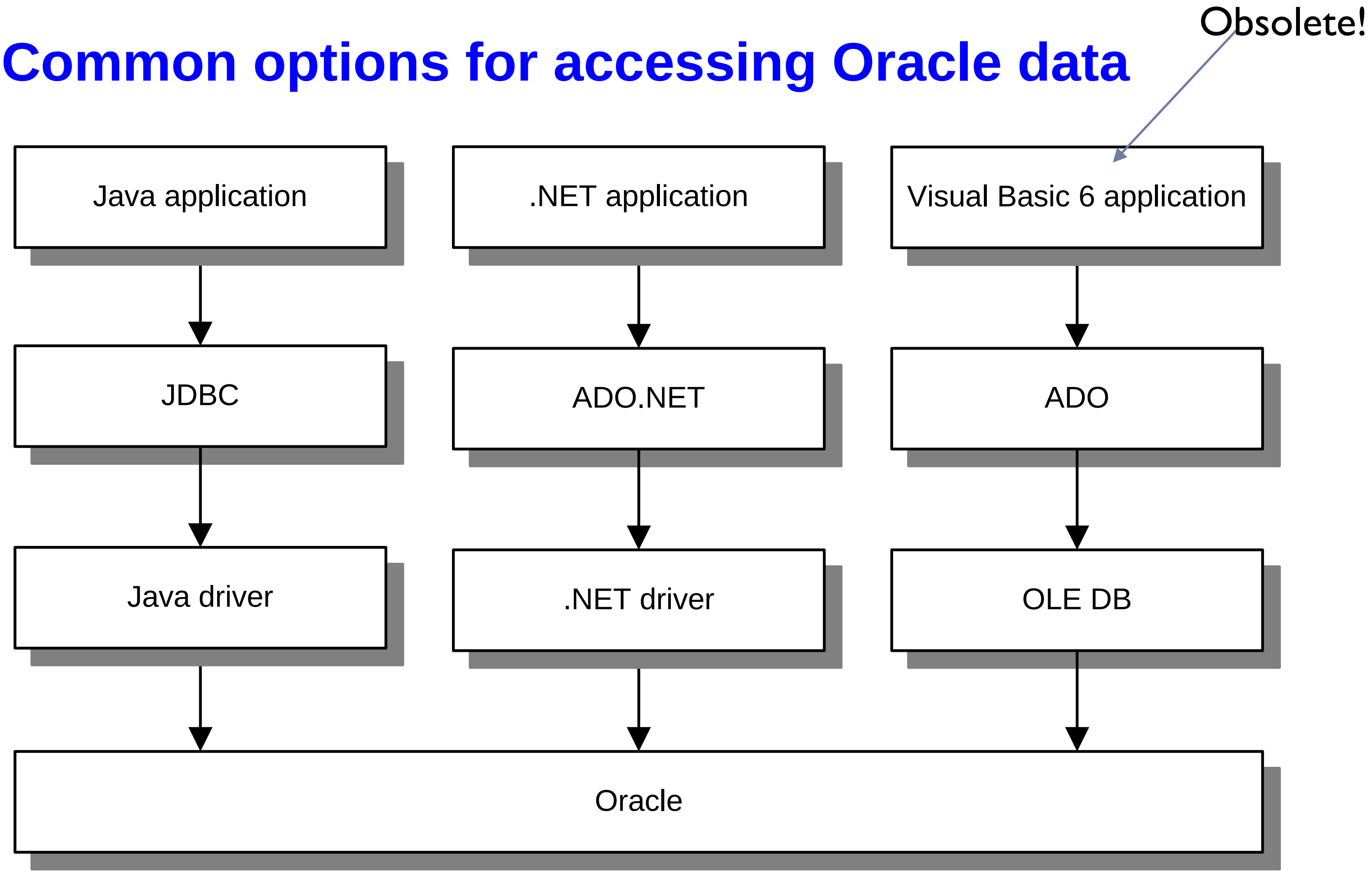
Your login on
topcat.cs.umb.edu
or pe07.cs.umb.edu

Your development
system at home

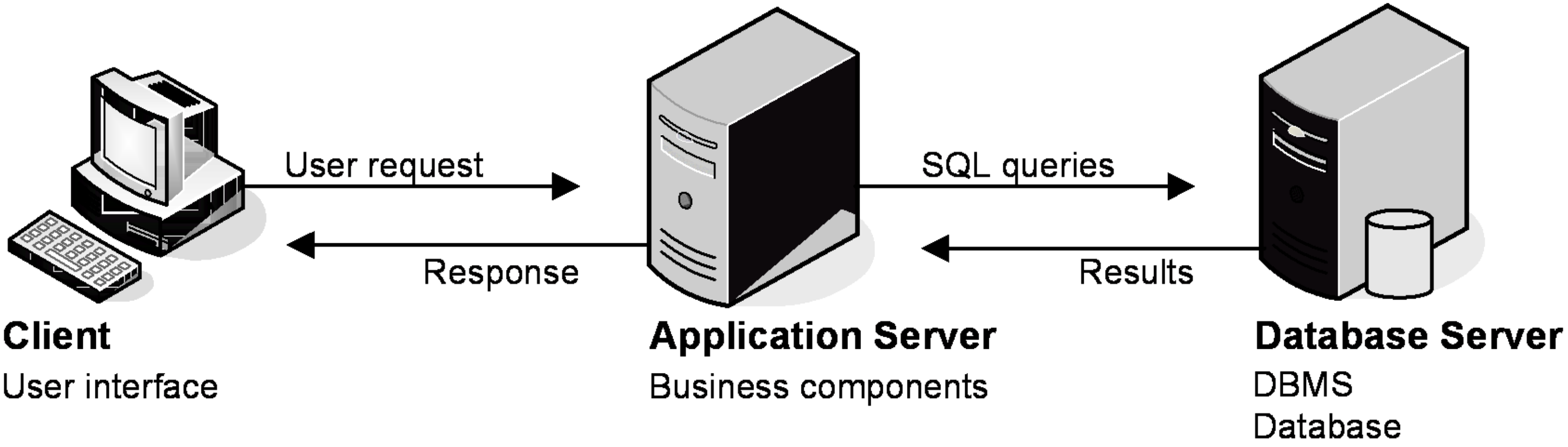
Client software, server software, and the SQL interface



Common options for accessing Oracle data



An application that uses an application server



User request

Response

SQL queries

Results

Client
User interface

Application Server
Business components

Database Server
DBMS
Database

←Over Internet→

←Over Local Area Network→

We won't be using an application server in this class, but we do in CS636, a follow-on course, with Java running in the application server (tomcat).

The Relational Model

The Relational Data Model

- Structure of data
  - Relational model uses tables: just rows, columns, no ‘pointers’
  - Vs. Programming languages deal with arrays, collections, etc. with explicit references from one item to another
- Operations on the data (maintaining data consistency)
  - Queries: operations that retrieve information
  - Modifications: operations that change data
- Constraints
  - Domain constraints (the simplest): e.g., age must be numeric
  - Other constraints: each student has unique student #

Prominent Data Models

- Relational model: tables with rows and columns, constraints, relationships, data consistency, but effectively inflexible schema (column definitions).
- "NoSQL" means not following relational model (may support SQL!), various types:
  - Key-value store (like hash table, key → value) Example: Redis
  - Document store (key → self-contained document) Ex.: Mongodb
  - Graph DB (nodes with links) Example: Neo4j
  - Wide Column Store. Example: Apache HBase, part of Hadoop project. Has tables, but with flexible schema
  - If time permits, we'll explore NoSQL at the end of the course

Relational Model

- *Relational database*: a set of *relations*
- *Relation:*
  - two-dimensional *table*, with rows and columns

    #Rows = *cardinality*
  - Each row represents an entity
    - A student, a course, a movie
  - Each column represents a property of the entity
    - Student age, student matriculation #, student gpa
    - Column values are atomic (e.g., integer or string) within given domain
  - Rows are also called tuples or records; columns are also called fields or attributes

“Students” Relation or Table

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

Relational Schema

- *Schema*: specifies name of relation, plus name and domain of each column

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

- Each relation must have a schema
  - Similar to a data type in programming languages
- Relational database schema = collection of relations' schemas
- Vs. NoSQL DBs: these have more flexible schemas: add new data items, etc.

Use of “string” as a domain in R&G

- If you already know some SQL, you know there is no data type “string” in SQL, but we see “string” in many schemas in the R&G book
- The first use is on pg. 59, for students, as on the last slide.
- In SQL, we have char(20) for example, for 20-character strings.
- So think of “string” in R&G as standing for strings with some length limit.

More about Relations

- Relations are *sets of tuples*
  - Sets are NOT ordered
  - Do NOT retrieve by row number, but by content!
- Relation Instance
  - Contents of a relation may change over time
    - Tuples are added/deleted/modified
    - E.g., Students join or leave the university
  - Instance represents set of tuples at a certain point in time
- Schemas may change too
  - Although this is not frequent in practice
  - Changing schema can be very expensive

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

Another Instance of "Students"

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
53660	Korth	korth@math	22	3.6

Cardinality = 4

Typical SQL Application life cycle

- Install application: create tables for application, once and for all
- Use application: add/delete rows, update rows, query rows

Of course, some applications create tables dynamically.

Keys

- A key of a relation is a set of fields *K* such that:
  1. No two distinct tuples in ANY relation instance have same values in all key fields, and
  2. No subset of *K* is a key (otherwise *K* is a *superkey*)
- Key may not be unique
  - Multiple candidate keys may exist
  - One of the keys is chosen (by DBA) to be the *primary key*
- Keys are shown underlined in schema
- In the relational model, duplicate tuples do not exist!
  - But most DBMS implementations do allow duplicates
  - Key constraints must be set by DBA to avoid duplicates
  - Rule of thumb: every table should have a primary key

Example of Keys

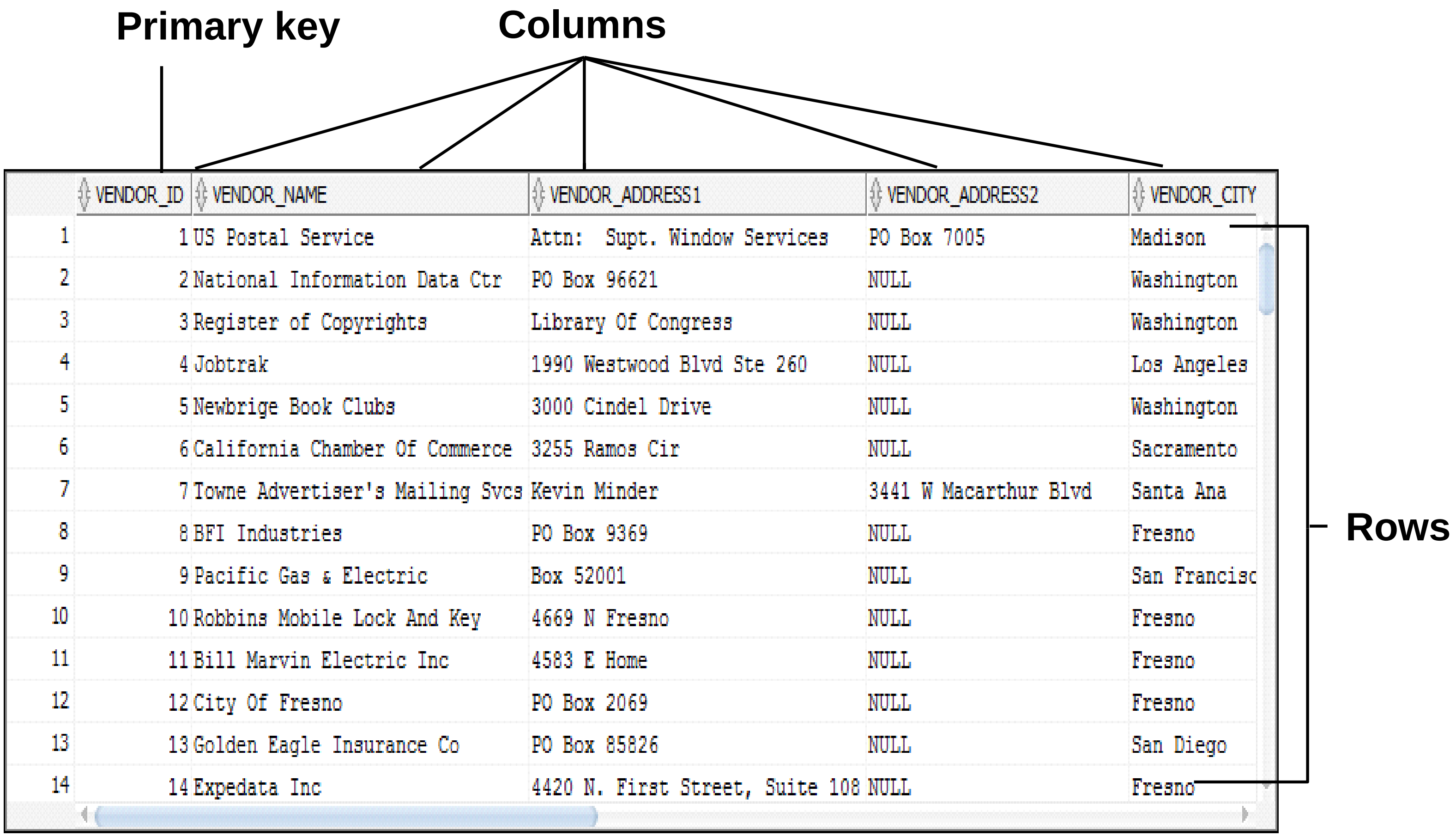
Students(*sid*: string, *name*: string, *login*: string, *age*: integer, *gpa*: real)

- *sid* is a key; {*sid, name*} is a superkey

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

- In practice, it is not easy to know when there exists a unique attribute combination in the data (e.g., names)
  - artificial keys are created: student ID, customer ID, etc.
  - SSN is also often used for keys (but what if multinational…)
  - Also, must keep SSNs hidden from public

Murach Example: The Vendors table viewed here in Oracle SQL Developer



Primary key

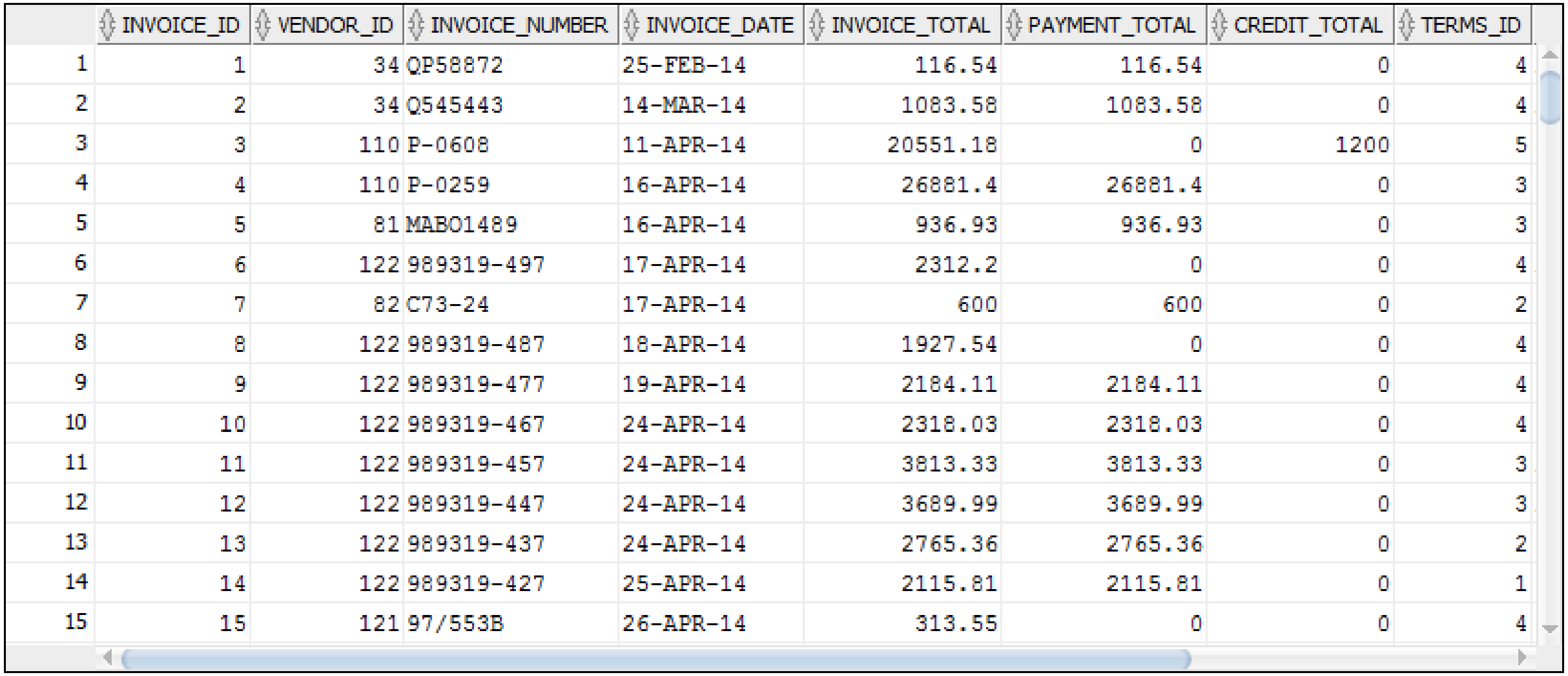
Columns

Rows

	VENDOR_ID	VENDOR_NAME	VENDOR_ADDRESS1	VENDOR_ADDRESS2	VENDOR_CITY
1	1	US Postal Service	Attn: Supt. Window Services	PO Box 7005	Madison
2	2	National Information Data Ctr	PO Box 96621	NULL	Washington
3	3	Register of Copyrights	Library Of Congress	NULL	Washington
4	4	Jobtrak	1990 Westwood Blvd Ste 260	NULL	Los Angeles
5	5	Newbrige Book Clubs	3000 Cindel Drive	NULL	Washington
6	6	California Chamber Of Commerce	3255 Ramos Cir	NULL	Sacramento
7	7	Towne Advertiser's Mailing Svcs	Kevin Minder	3441 W Macarthur Blvd	Santa Ana
8	8	BFI Industries	PO Box 9369	NULL	Fresno
9	9	Pacific Gas & Electric	Box 52001	NULL	San Francisc
10	10	Robbins Mobile Lock And Key	4669 N Fresno	NULL	Fresno
11	11	Bill Marvin Electric Inc	4583 E Home	NULL	Fresno
12	12	City Of Fresno	PO Box 2069	NULL	Fresno
13	13	Golden Eagle Insurance Co	PO Box 85826	NULL	San Diego
14	14	Expedata Inc	4420 N. First Street, Suite 108	NULL	Fresno

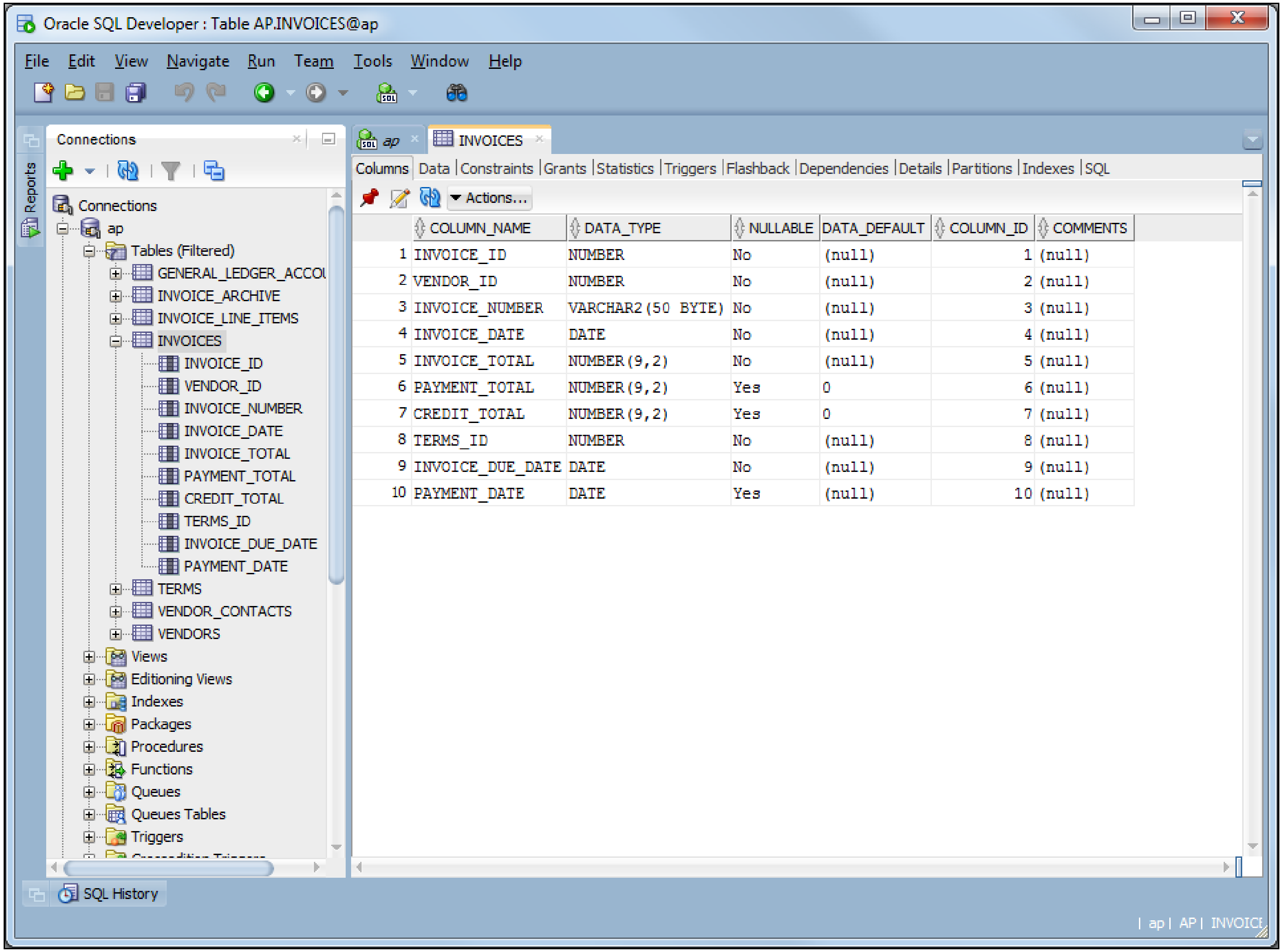
(Row numbers are not part of the database table)

The Murach Invoices table



	INVOICE_ID	VENDOR_ID	INVOICE_NUMBER	INVOICE_DATE	INVOICE_TOTAL	PAYMENT_TOTAL	CREDIT_TOTAL	TERMS_ID
1	1	34	QP58872	25-FEB-14	116.54	116.54	0	4
2	2	34	Q545443	14-MAR-14	1083.58	1083.58	0	4
3	3	110	P-0608	11-APR-14	20551.18	0	1200	5
4	4	110	P-0259	16-APR-14	26881.4	26881.4	0	3
5	5	81	MABO1489	16-APR-14	936.93	936.93	0	3
6	6	122	989319-497	17-APR-14	2312.2	0	0	4
7	7	82	C73-24	17-APR-14	600	600	0	2
8	8	122	989319-487	18-APR-14	1927.54	0	0	4
9	9	122	989319-477	19-APR-14	2184.11	2184.11	0	4
10	10	122	989319-467	24-APR-14	2318.03	2318.03	0	4
11	11	122	989319-457	24-APR-14	3813.33	3813.33	0	3
12	12	122	989319-447	24-APR-14	3689.99	3689.99	0	3
13	13	122	989319-437	24-APR-14	2765.36	2765.36	0	2
14	14	122	989319-427	25-APR-14	2115.81	2115.81	0	1
15	15	121	97/553B	26-APR-14	313.55	0	0	4

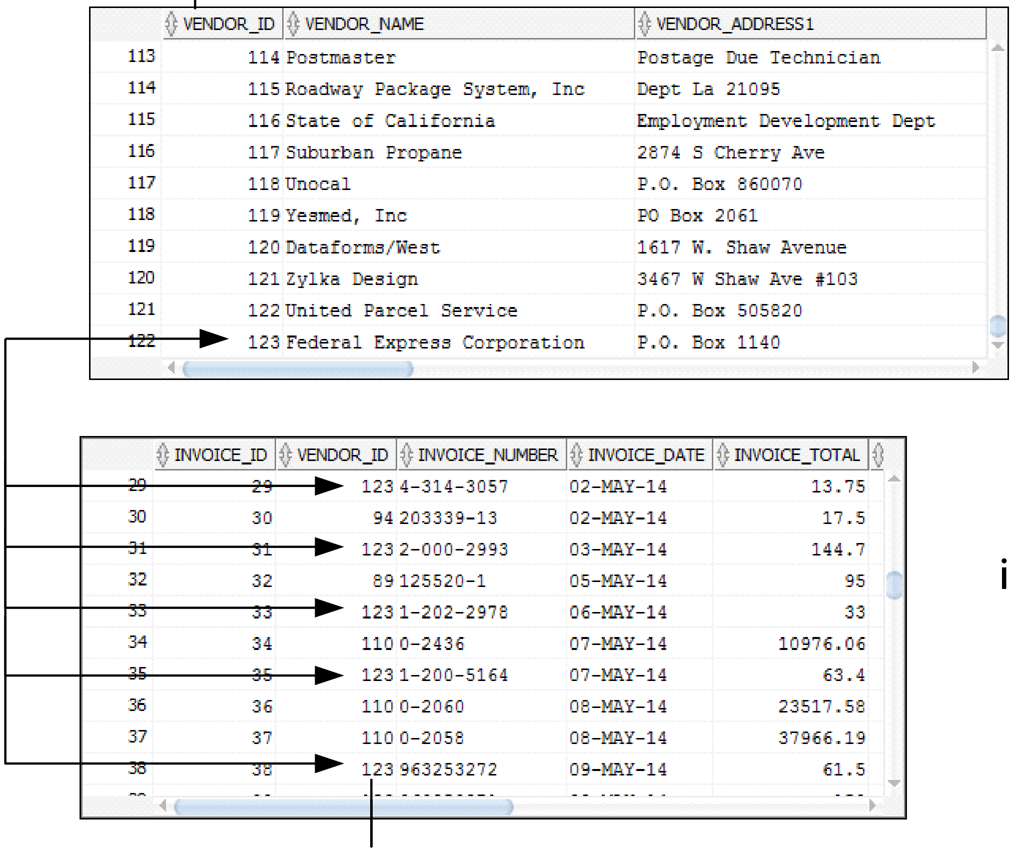
The columns of the Murach Invoices table



	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	INVOICE_ID	NUMBER	No	(null)	1	(null)
2	VENDOR_ID	NUMBER	No	(null)	2	(null)
3	INVOICE_NUMBER	VARCHAR2(50 BYTE)	No	(null)	3	(null)
4	INVOICE_DATE	DATE	No	(null)	4	(null)
5	INVOICE_TOTAL	NUMBER(9,2)	No	(null)	5	(null)
6	PAYMENT_TOTAL	NUMBER(9,2)	Yes	0	6	(null)
7	CREDIT_TOTAL	NUMBER(9,2)	Yes	0	7	(null)
8	TERMS_ID	NUMBER	No	(null)	8	(null)
9	INVOICE_DUE_DATE	DATE	No	(null)	9	(null)
10	PAYMENT_DATE	DATE	Yes	(null)	10	(null)

The relationship between two tables

Primary key



	VENDOR_ID	VENDOR_NAME	VENDOR_ADDRESS1
113	114	Postmaster	Postage Due Technician
114	115	Roadway Package System, Inc	Dept La 21095
115	116	State of California	Employment Development Dept
116	117	Suburban Propane	2874 S Cherry Ave
117	118	Unocal	P.O. Box 860070
118	119	Yesmed, Inc	PO Box 2061
119	120	Dataforms/West	1617 W. Shaw Avenue
120	121	Zylka Design	3467 W Shaw Ave #103
121	122	United Parcel Service	P.O. Box 505820
122	123	Federal Express Corporation	P.O. Box 1140

vendors table

	INVOICE_ID	VENDOR_ID	INVOICE_NUMBER	INVOICE_DATE	INVOICE_TOTAL
29	29	123	4-314-3057	02-MAY-14	13.75
30	30	94	203339-13	02-MAY-14	17.5
31	31	123	2-000-2993	03-MAY-14	144.7
32	32	89	125520-1	05-MAY-14	95
33	33	123	1-202-2978	06-MAY-14	33
34	34	110	0-2436	07-MAY-14	10976.06
35	35	123	1-200-5164	07-MAY-14	63.4
36	36	110	0-2060	08-MAY-14	23517.58
37	37	110	0-2058	08-MAY-14	37966.19
38	38	123	963253272	09-MAY-14	61.5

invoices table

Foreign key