CS634
Architecture of Database Systems
Spring 2018

Elizabeth (Betty) O'Neil
University of Massachusetts at Boston

People & Contact Information

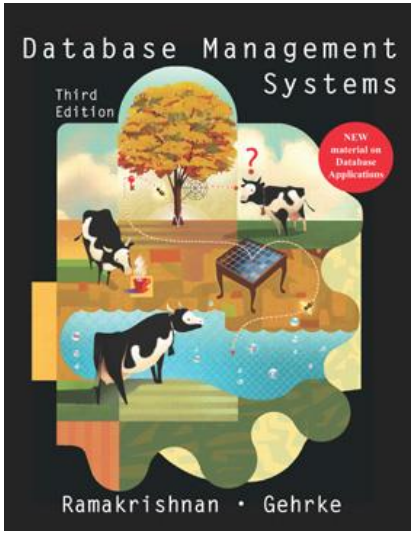
- Instructor: Prof. Betty O'Neil
  - **Email:** eoneil AT cs.umb.edu (preferred contact)
  - **Web:** http://www.cs.umb.edu/~eoneil
  - **Office:** Science Building, 3rd Floor, Room 169 (S-3-169)
- Grader (TA): TBD

Course Info

- Lecture Hours
  - MW 5:30-6:45 pm
  - McCormack M03-0204
- Office Hours
  - MW 3:45-5:15 pm in S/3/169
  - By appointment (send email or see me after class)
- Class URL
  - http://www.cs.umb.edu/cs634

Textbook & Recommended Readings

- Textbook
  - *Database Management Systems*, 3rd Edition by Ramakrishnan and Gehrke
  - Chapters 8-18, 20,25



- Other resources will be posted on the class home page

Prerequisites

- Database Management Systems
  - CS430/630
- Data Structures and Algorithms, Programming in Java
  - CS310, CS210
- Programming in C
  - CS240
- Familiarity with UNIX/Linux shell commands
  - Students have accounts on Linux system pe07.cs.umb.edu, as well as users.cs.umb.edu, with access to an Oracle 12c server running on a Linux machine (dbs3.cs.umb.edu), and a mysql server running on Linux (pe07.cs.umb.edu)

Grading: simple point system

- Midterm (100 points) – open book
- Final exam (150 points) – open book
- Open book does NOT include electronic devices!
- Need a print book or printouts of parts of .pdf.

- 5-6 homework assignments
  - 10-25 points each
  - Assignments are individual – submit your own work only! (unless specifically marked as group assignment)
  - No plagiarism please – see student code of conduct

Website

- Class URL

  http://www.cs.umb.edu/cs634/ Find slides, handouts, useful links, homework assignments, etc.

- Class email list: make sure you have a .forward in your cs.umb.edu home directory with your preferred email address—I'll check this on Friday.

- Make sure you create a Unix course account for cs634. It will give you access to our Linux system pe07. Also membership in the class email list.

University Policies

- **Student Conduct**: Students are required to adhere to the University Policy on Academic Standards and Cheating, to the University Statement on Plagiarism and the Documentation of Written Work, and to the Code of Student Conduct as delineated in the University Catalog and Student Handbook. The Code is available online at:

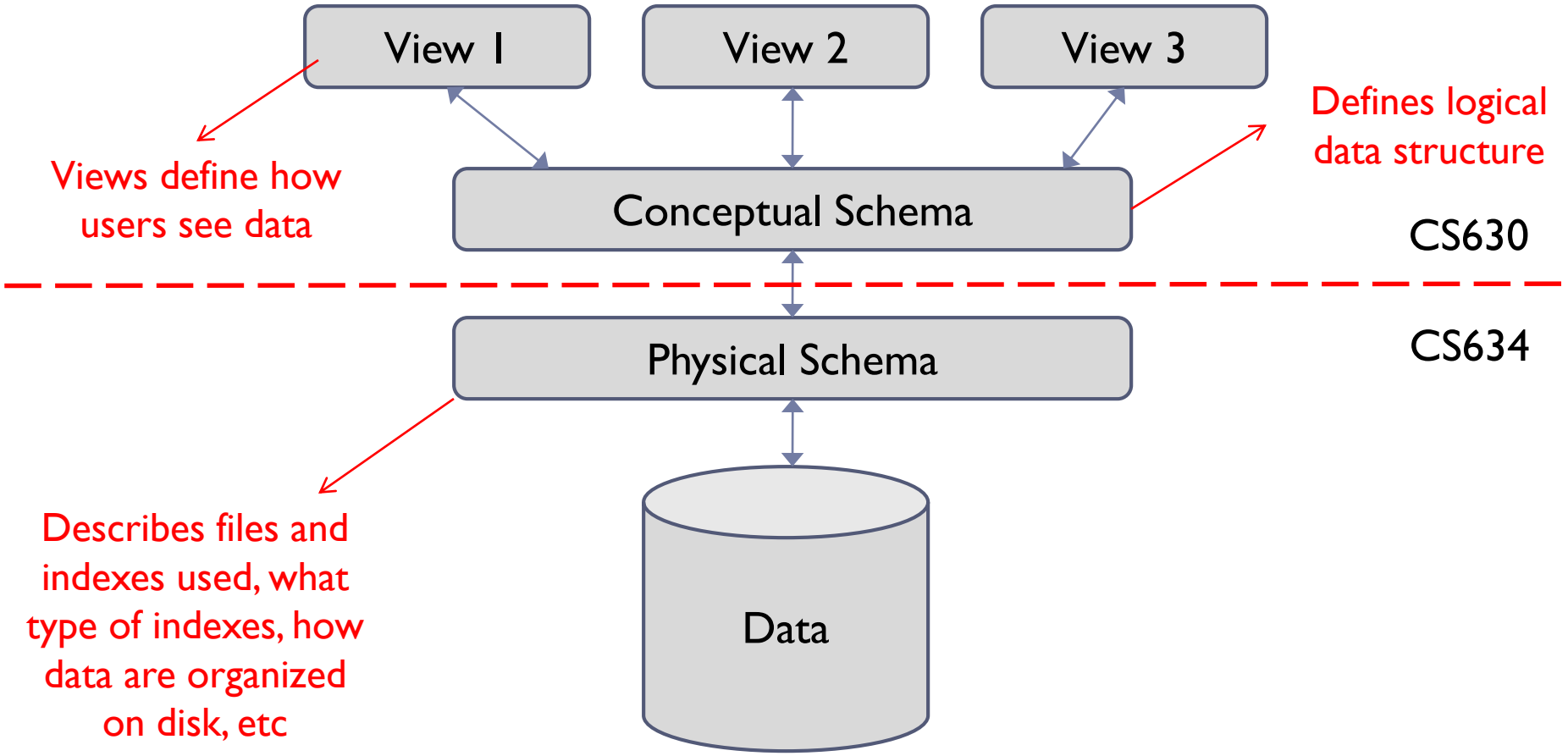
  http://www.umb.edu/life_on_campus/policies/code/

- **Accommodations:** Section 504 of the Americans with Disabilities Act of 1990 offers guidelines for curriculum modifications and adaptations for students with documented disabilities. If applicable, students may obtain adaptation recommendations from the Ross Center for Disability Services, CC-UL Room 211, (617-287-7430). The student must present these recommendations and discuss them with each professor within a reasonable period, preferably by the end of Drop/Add period.

What did we learn in 430/630?

- Relational Data Model
  - Data represented as table with row and columns, called a *relation;*
  - Each relation has a *schema*, which describes the table structure
- Querying relational DBMS
  - SQL language: single table queries, join queries, grouping and aggregates, nested queries, division
- Accessing relational DBMS from applications: JDBC, PLSQL
- Design theory
- Basics of Database Security (GRANT command, etc.)

Levels of Abstraction

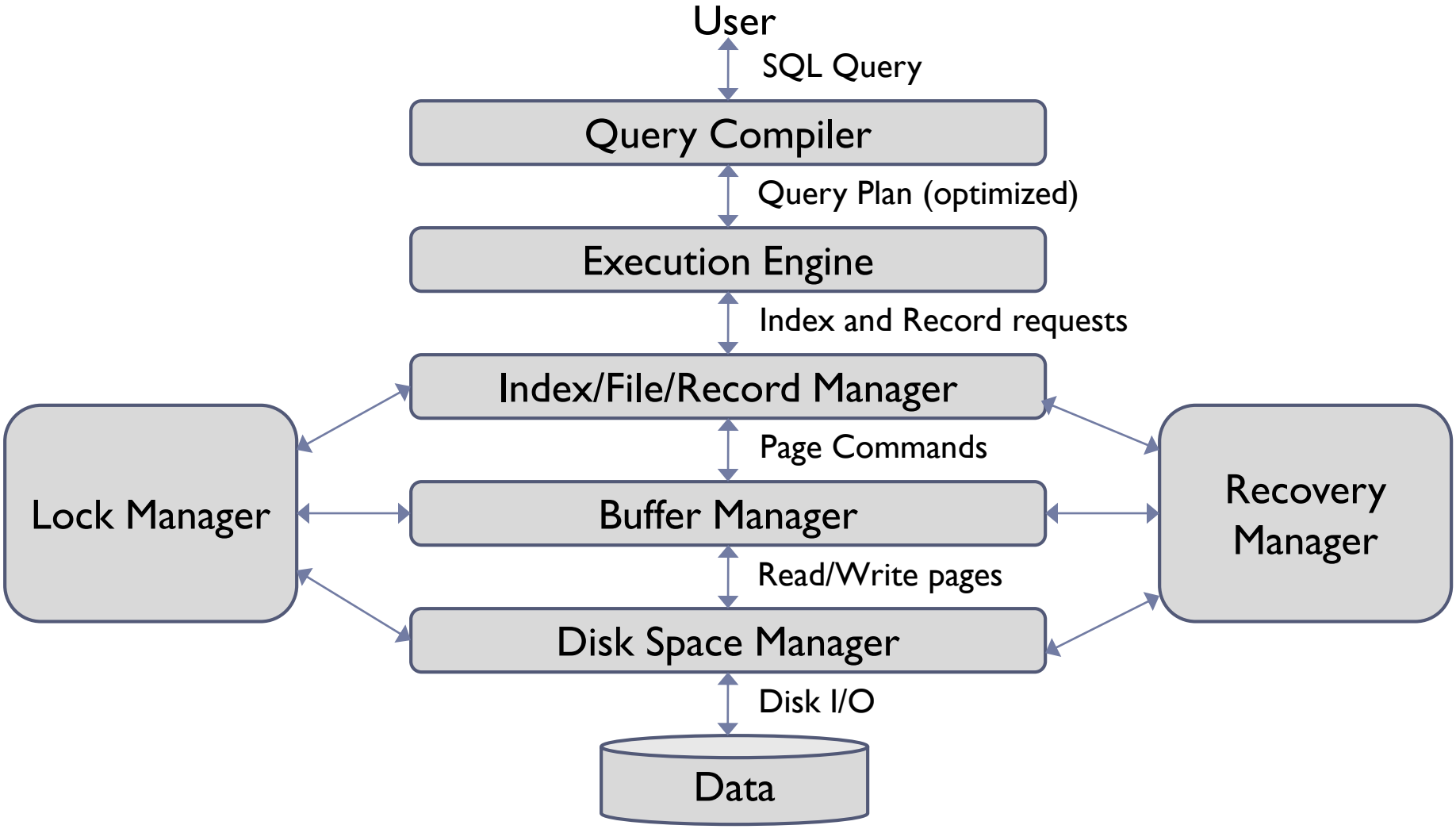


Database Management Systems 3rd ed, Ramakrishnan and Gehrke

What will we learn in 634?

- How data are stored inside DBMS
  - Internal data structure types and their trade-offs
- How to provide access to data **efficiently**
  - Indexing
- How to execute queries **efficiently**
  - Query execution plans and optimization
  - Tuning the database as DBA
- Transaction Management
  - Supporting concurrent access to data
  - Persistent storage and recovery from failure
- Data Warehousing
- Intro to Big Data

Architecture of a DBMS



A first course in database systems, 3rd ed, Ullman and Widom

Data Storage and Indexing

- Storage
  - Disk Space Management
  - RAID, SSD
  - Buffer Management and its tuning
  - Page and record formats
- Indexing
  - General Index Structure
  - Hierarchical (tree-based) indexing
  - Hash-based indexing
  - Index operations
  - Cost analysis and trade-offs

Query Evaluation and Optimization

- Operator Evaluation
  - Algorithms for relational operations
  - Selection, projection, join, etc
  - Query evaluation plans

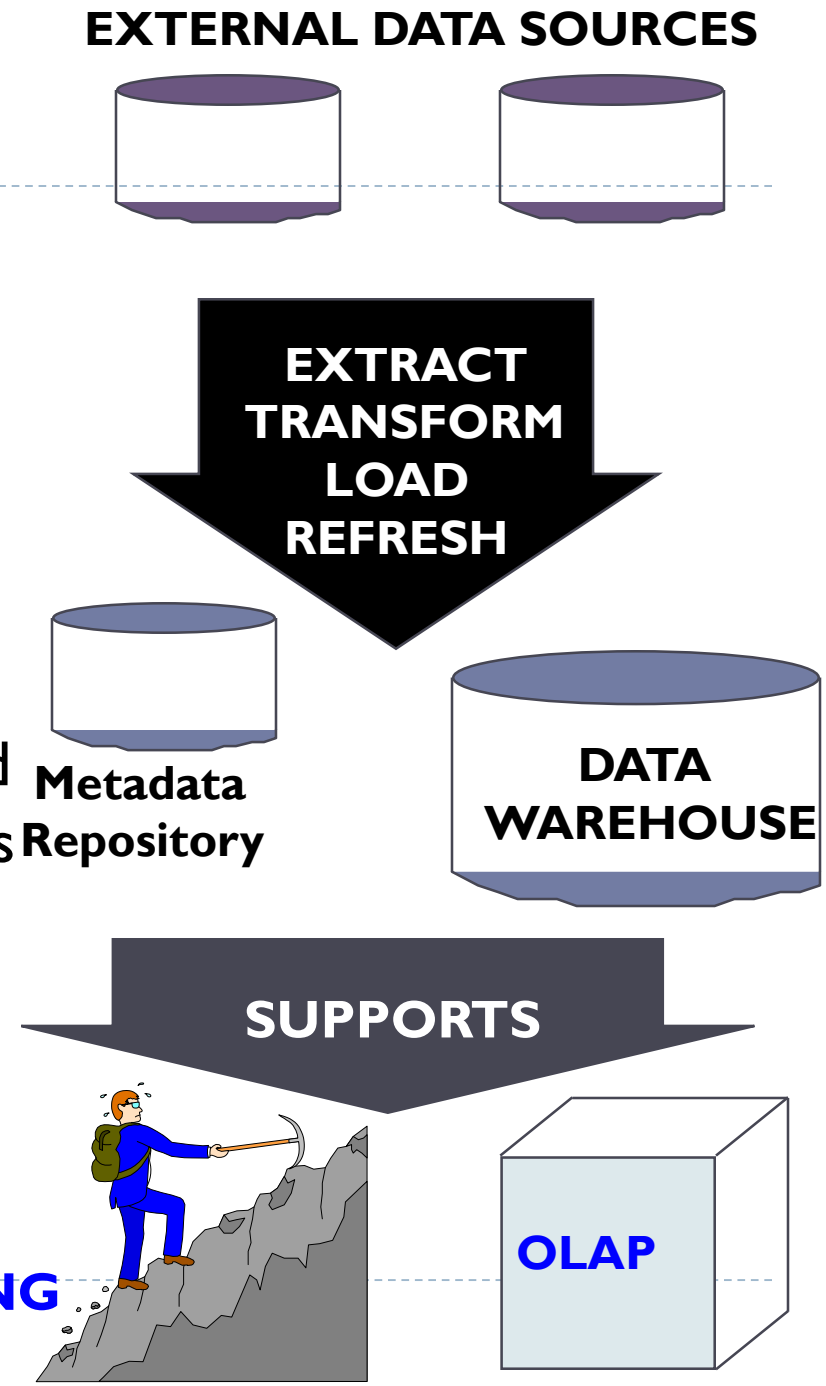
- Query Optimization
  - Multi-operator queries: pipelined evaluation
  - Alternative plans
  - Using indexes
  - Estimating plan costs

Transaction Management

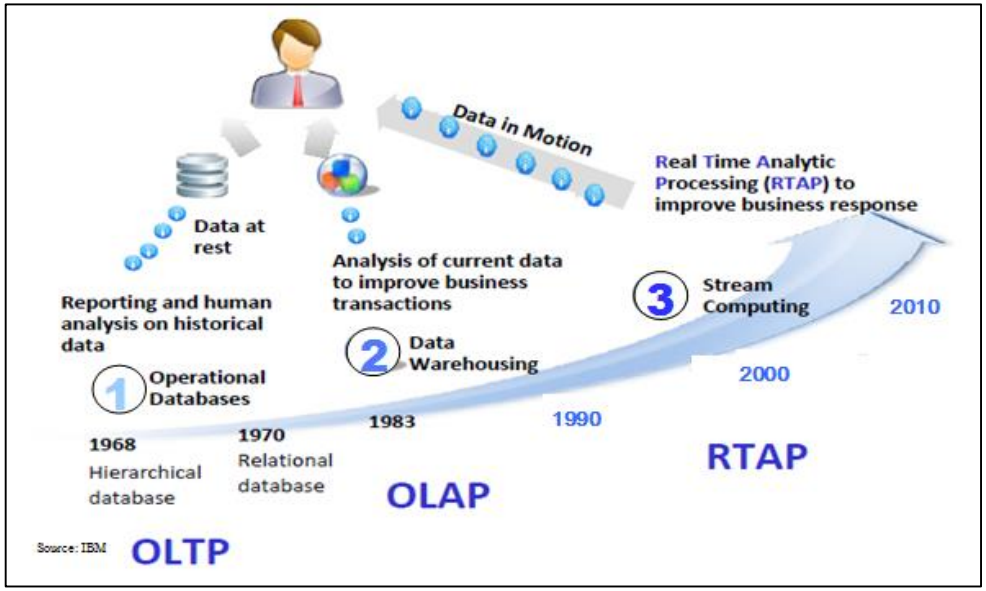
- Transaction = unit of work (sequence of operations)
  - Concurrency control: multiple transactions running simultaneously (updates are the issue)
  - Failure Recovery: what if system crashes during execution?
- ACID properties
  - A = Atomicity
  - C = Consistency
  - I = Isolation
  - D = Durability
- Synchronization protocols – serializable schedule

Data Warehousing

- Integrated data spanning long time periods, often augmented with summary information.
- Several gigabytes to terabytes common, now petabytes too.
- Interactive response times expected for complex queries; ad-hoc updates uncommon.
- Read-mostly data



On to Big Data



- **OLTP:** Online Transaction Processing (DBMSs)
- **OLAP:** Online Analytical Processing (Data Warehousing)
- **RTAP:** Real-Time Analytics Processing (Big Data Architecture & technology)

Review: Foreign Keys

Defined in Sec. 3.2.2 without mentioning nulls

First example: nice not-null foreign key column (because it's part of the primary key):

```
create table enrolled(
        studid char(20),
        cid  char(20),
        grade char(10),
        primary key(studid,cid),
        foreign key(studid) references Students
        );
```

This FK ensures that there's a real student record for the studid listed in this row. The students table is assumed to have a primary key of type compatible with studid's.

Review: Foreign Keys, etc.

```
create table enrolled(
        studid char(20),
        cid  char(20),
        grade char(10),
        primary key(studid,cid),  -- so both these cols are non-null
        foreign key(studid) references Students
        );
```

- Note the "Students" table name. Table names, column names, etc. are caseless in standard SQL. So this can also be written "students".
- **primary key**(studid,cid): This ensures that both studid and cid are non-null, as pointed out on pg. 77, top. So we don't have to write "cid char(20) not null", but it doesn't hurt to do so.
- "grade char(10)": this column may have null values, since there is no "not null" column constraint on it.

Review: Foreign Keys, etc.

```
create table enrolled(
        studid char(20),
        cid  char(20),
        grade char(10),
        primary key(studid,cid),  -- so both these cols are non-null
        foreign key(studid) references Students
        );
```

- We would usually expect a foreign key constraint on cid as well.
- MySQL: use "references students(sid)"
- More on this next time: read Sec. 3.2

Important SQL Standards

SQL-92: third and most important standard
Early enough to affect Oracle, DB2, other important commercial databases, so the real common ground.

SQL-2003 (also sometimes called SQL-99, a stepping-stone to it), revised 2008

SQL 2003 Data Types, from

http://www.w3resource.com/sql/data-type.php, with notes in color

CHARACTER(n) or CHAR(n)	Character string, fixed length n. A string of text in an implementer-defined format. The size argument is a single nonnegative integer that refers to the maximum length of the string. Values for this type must enclosed in single quotes. Character sets: another topic.
CHARACTER VARYING(n) or VARCHAR(n)	Variable length character string, maximum length n.
BINARY(n)	Fixed length binary string, maximum length n. Not in SQL-92, but BIT(n) there.
BOOLEAN	Stores truth values - either TRUE or FALSE. Not in SQL-92
BINARY VARYING(n) or VARBINARY(n)	Variable length binary string, maximum length n. BIT VARYING in SQL-92.

SQL 2003 Data Types

INTEGER(p)	Integer numerical, precision p. Not in SQL-92 with (p). MySQL: p means display size, not precision
SMALLINT	Integer numerical precision 5. SQL-92: precision is implementation dependent.
INTEGER	Integer numerical, precision 10. It is a number without decimal point with no digits to the right of the decimal point, that is, with a scale of 0. SQL-92: precision is implementation dependent.
BIGINT	Integer numerical, precision 19. Not in SQL-92.

SQL 2003 Data Types

DECIMAL(p, s)	Exact numerical, precision p, scale s. A decimal number, that is number that can have a decimal point in it. The size argument has two parts : precision and scale. The scale can not exceed the precision. Precision comes first, and a comma must separate from the scale argument. How many digits the number is to have - a precision indicates that and maximum number of digits to the right of decimal point have, that indicates the scale.
NUMERIC(p, s)	Exact numerical, precision p, scale s. (Same as DECIMAL).

SQL 2003 Data Types

FLOAT(p)	Approximate numerical, mantissa precision p. A floating number in base 10 exponential notation. The size argument for this type consists of a single number specifying the minimum precision.
REAL	Approximate numerical mantissa precision 7. Better to use FLOAT.
FLOAT	Approximate numerical mantissa precision 16. Usually IEEE Standard floating point, but not guaranteed by the SQL standard. Oracle uses NUMBER for FLOAT, use BINARY_DOUBLE for IEEE format, like Java double.
DOUBLE PRECISION	Approximate numerical mantissa precision 16. Same as FLOAT.

SQL 2003 Data Types

DATE TIME TIMESTAMP	Composed of a number of integer fields, representing an absolute point in time, depending on sub-type.
INTERVAL	Composed of a number of integer fields, representing a period of time, depending on the type of interval.
COLLECTION (ARRAY, MULTISET) Not in SQL-92	ARRAY(offered in SQL99) is a set-length and ordered collection of elements, MULTISET (added in SQL2003) is a variable-length and unordered collection of elements. Both the elements must be of a predefined datatype.
XML Not in SQL-92	Stores XML data. It can be used wherever a SQL datatype is allowed, such as a column of a table.