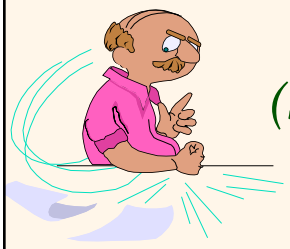
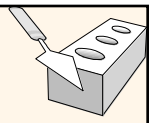
Introduction to Data Management



Lecture #18
(SQL, the Final Chapter…)

Instructor: Mike Carey
mjcarey@ics.uci.edu



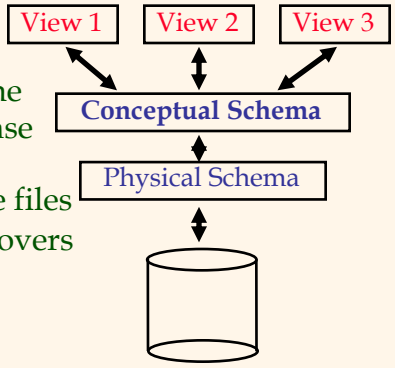
Announcements

- Grading progress has been made…!
  - HWs 2 & 3 are now graded
  - Midterm #1 is also graded (!)
- HW #6 coming out tonight, due next **Monday**
  - The usual one **late day** will be available as an option
  - The solution will be posted on Tuesday after 5PM
- Midterm #2 will be **Wednesday (5/22)** in class
  - Relational languages (see syllabus)
  - A sample from last year is available online
- ***So – let's finish up relational languages!***

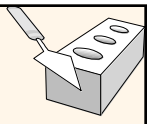
Midterm #1 Stats

Layers of Schemas: Brief "Re-View"

- Many *views* of one *conceptual (logical) schema* and an underlying *physical schema*
  - Views describe how different users see the data.
  - Conceptual schema defines the logical structure of the database
  - Physical schema describes the files and indexes used under the covers



*A Simple **View** Example (MySQL)*

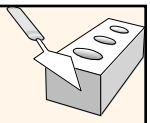


```
CREATE VIEW YoungSailorsView (yid, yname, yage, yrating)
  AS
SELECT sid, sname, age, rating
FROM Sailors
WHERE age < 18;
```

```
SELECT * FROM YoungSailorsView;
```

```
SELECT yname, yrating, yage
FROM YoungSailorsView
WHERE yrating >= 9;
```


Another View Example (MySQL)

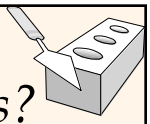


```
CREATE VIEW ActiveSailors (sid, sname, rating)
  AS
SELECT S.sid, S.sname, S.rating
FROM Sailors S WHERE EXISTS
   (SELECT * FROM Reserves R WHERE R.sid = S.sid);
```

```
SELECT * FROM ActiveSailors;
```

```
UPDATE ActiveSailors
SET rating = 11
WHERE sid = 22;
```

So What About Views & Updates?



Ex:

```
CREATE VIEW SailsBoats AS
SELECT DISTINCT S.*, B.*
FROM Sailors S, Boats B, Reserves R
WHERE S.sid = R.sid and R.bid = B.bid;
```

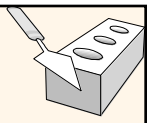
Q: What if we now try...

```
UPDATE SailsBoats
SET rating = 12
WHERE sid = 22 AND bid = 101;
```

(?)

*This view is **not** updatable since there is no update to the real (**stored**) tables that would have (**just**) the asked-for effect – see next slide...!*

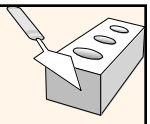
... Views & Updates (Cont'd.)?



```
UPDATE SailsBoats  SET rating = 12
WHERE sid = 22 AND bid = 101;
```

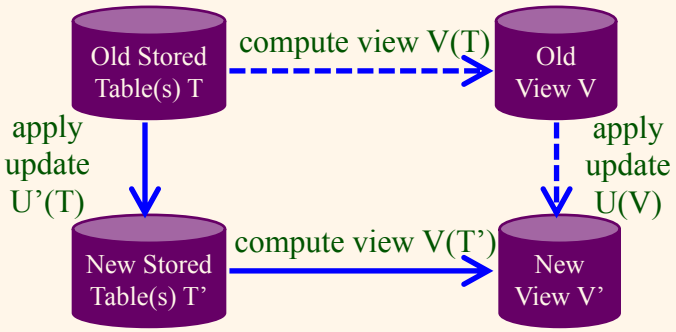
Result Grid Filter Rows: Search

sid	sname	rating	age	bid	bname	color
22	Dustin	7	45.0	101	Interlake	blue
64	Horatio	7	35.0	101	Interlake	blue
22	Dustin	7	45.0	102	Interlake	red
31	Lubber	8	55.5	102	Interlake	red
64	Horatio	7	35.0	102	Interlake	red
22	Dustin	7	45.0	103	Clipper	green
31	Lubber	8	55.5	103	Clipper	green
74	Horatio	9	35.0	103	Clipper	green
22	Dustin	7	45.0	104	Marine	red
31	Lubber	8	55.5	104	Marine	red

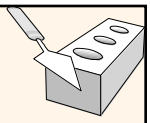


... Views & Updates? (Cont'd.)

- A legal update U to view V must be translatable into an equivalent update U' on the underlying table(s) T, *i.e.*:

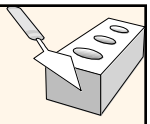


- If this isn't possible, the system will reject the update
- Systems differ in how well they do this and err on the conservative side (*i.e.*, declining more view updates)



SQL Access Control

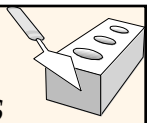
- Based on the concept of access rights or privileges for objects (tables, views, stored procedures, ...) and mechanisms for giving users privileges (and revoking privileges).
- Creator of a database object automatically gets all privileges on it.
  - DBMS keeps track of who subsequently gains and loses privileges, and it ensures that only requests from users who have the necessary privileges (at the time the request is issued) are allowed to execute.



GRANT Command

GRANT privileges ON object TO users [WITH GRANT OPTION]

- The following privileges can be specified:
  - SELECT: Can read all columns (including those added later via ALTER TABLE command).
  - INSERT(col-name): Can insert tuples with non-null or non-default values in this column.
    - INSERT means same right with respect to *all* columns.
  - DELETE: Can delete tuples.
  - REFERENCES (col-name): Can define foreign keys (in other tables) that refer to this column.
- If a user has a privilege with the GRANT OPTION, can pass privilege on to other users (with or without passing on the GRANT OPTION).
- Only the owner can execute CREATE, ALTER, or DROP.



GRANT and REVOKE of Privileges

- GRANT INSERT, SELECT ON Sailors TO Horatio
  - Horatio can query Sailors or insert tuples into it.
- GRANT DELETE ON Sailors TO Yuppy WITH GRANT OPTION
  - Yuppy can delete tuples *and* can authorize others to do so.
- GRANT UPDATE (*rating*) ON Sailors TO Dustin
  - Dustin can update (only) the *rating* field of Sailors tuples.
- GRANT SELECT ON ActiveSailors TO Guppy, Yuppy
  - This does *NOT* allow the 'uppies to query Sailors *directly*!
- REVOKE: When a privilege is revoked from X, it is also revoked from all users who got it *solely* from X.

MySQL授权命令grant的使用方法

本文实例，运行于 MySQL 5.0 及以上版本。

MySQL 赋予用户权限命令的简单格式可概括为：

```
grant 权限 on 数据库对象 to 用户
```

一、grant 普通数据用户，查询、插入、更新、删除 数据库中所有表数据的权利。

```
grant select on testdb.* to common_user@'%'
grant insert on testdb.* to common_user@'%'
grant update on testdb.* to common_user@'%'
grant delete on testdb.* to common_user@'%'
```

或者，用一条 MySQL 命令来替代：

```
grant select, insert, update, delete on testdb.* to common_user@'%'
```

二、grant 数据库开发人员，创建表、索引、视图、存储过程、函数。。。等权限。

grant 创建、修改、删除 MySQL 数据表结构权限。

```
grant create on testdb.* to developer@'192.168.0.%';
grant alter on testdb.* to developer@'192.168.0.%';
grant drop on testdb.* to developer@'192.168.0.%';
```

grant 操作 MySQL 外键权限。

```
grant references on testdb.* to developer@'192.168.0.%';
```

grant 操作 MySQL 临时表权限。

```
grant create temporary tables on testdb.* to developer@'192.168.0.%';
```

grant 操作 MySQL 索引权限。

```
grant index on testdb.* to developer@'192.168.0.%';
```

grant 操作 MySQL 视图、查看视图源代码 权限。

```
grant create view on testdb.* to developer@'192.168.0.%';
grant show view on testdb.* to developer@'192.168.0.%';
```

grant 操作 MySQL 存储过程、函数 权限。

```
grant create routine on testdb.* to developer@'192.168.0.%'; -- now, can show procedure status
grant alter routine on testdb.* to developer@'192.168.0.%'; -- now, you can drop a procedure
grant execute on testdb.* to developer@'192.168.0.%';
```

三、grant 普通 DBA 管理某个 MySQL 数据库的权限。

```
grant all privileges on testdb to dba@'localhost'
```

其中，关键字 “privileges” 可以省略。

四、grant 高级 DBA 管理 MySQL 中所有数据库的权限。

```
grant all on *.* to dba@'localhost'
```

五、MySQL grant 权限，分别可以作用在多个层次上。

1. grant 作用在整个 MySQL 服务器上：

```
grant select on *.* to dba@localhost; -- dba 可以查询 MySQL 中所有数据库中的表。
grant all on *.* to dba@localhost; -- dba 可以管理 MySQL 中的所有数据库
```

2. grant 作用在单个数据库上：

```
grant select on testdb.* to dba@localhost; -- dba 可以查询 testdb 中的表。
```

3. grant 作用在单个数据表上：

```
grant select, insert, update, delete on testdb.orders to dba@localhost;
```

这里在给一个用户授权多张表时，可以多次执行以上语句。例如：

```
grant select(user_id,username) on smp.users to mo_user@'%' identified by '123345';
grant select on smp.mo_sms to mo_user@'%' identified by '123345';
```

4. grant 作用在表中的列上：

```
grant select(id, se, rank) on testdb.apache_log to dba@localhost;
```

5. grant 作用在存储过程、函数上：

```
grant execute on procedure testdb.pr_add to 'dba'@'localhost'
grant execute on function testdb.fn_add to 'dba'@'localhost'
```

六、查看 MySQL 用户权限

查看当前用户（自己）权限：

```
show grants;
```

查看其他 MySQL 用户权限：

```
show grants for dba@localhost;
```

七、撤销已经赋予给 MySQL 用户权限的权限。

revoke 跟 grant 的语法差不多，只需要把关键字 “to” 换成 “from” 即可：

```
grant all on *.* to dba@localhost;
revoke all on *.* from dba@localhost;
```

八、MySQL grant、revoke 用户权限注意事项

1. grant, revoke 用户权限后，该用户只有重新连接 MySQL 数据库，权限才能生效。

2. 如果想让授权的用户，也可以将这些权限 grant 给其他用户，需要选项 “grant option“

```
grant select on testdb.* to dba@localhost with grant option;
```

这个特性一般用不到。实际中，数据库权限最好由 DBA 来统一管理。

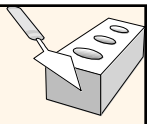
数据查询 SELECT（查询）

数据定义 CREATE（创建表、视图。。。）、DROP（删除表、视图。。。）、ALTER（修改表、视图。。。）、

数据操作 INSERT（添加数据）、UPDATE（修改数据）、DELETE（删除数据）

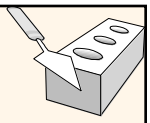
数据控制 GRANT（为用户授权）、REVOKE（废除权限）

GRANT/REVOKE on Views



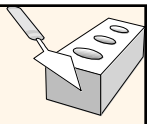
- Great combination to enforce restrictions on *data visibility* for various users/groups
- If view creator loses the SELECT privilege on an underlying table, the view is dropped!
- If view creator loses a privilege held with the grant option on an underlying table, (s)he loses it on the view as well – and so do users who were granted the privilege on the view!

Views & Security



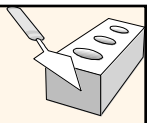
- Views can be used to present just the necessary information (or a summary) while hiding some details of the underlying relation(s):
  - Given *ActiveSailors*, but not Sailors or Reserves, we can find sailors who have a reservation, but not the *bid*'s of boats that have been reserved.
- *Creator* of a view has a privilege on the view if (s)he has the privilege on all underlying tables.
- Used together with GRANT/REVOKE commands, views are a very powerful access control tool.
- Stored procedures can be utilized similarly!

SQL Summary (I)



- SQL was a big factor in the early acceptance of the relational model; users found it more natural than earlier, procedural query languages.
- SQL is relationally complete; it has significantly more expressive power than the relational algebra.
- Queries that can be expressed in rel. alg. can often be expressed more naturally in SQL. (Ex: max ☺)
- Many alternative ways to write a query; optimizer will look for the most efficient evaluation plan.
  - In practice, expert users are aware of how queries are optimized and evaluated. (Optimizers are imperfect.)

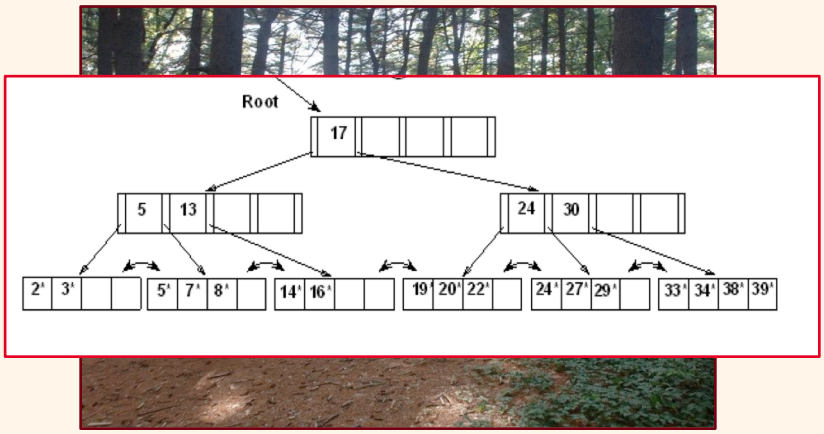
SQL Summary (II)



- NULL for unknown field values brings many complications (as well as a SQL specification divergence for Oracle *w.r.t.* VARCHAR data).
- Allows specification of rich integrity constraints (real RDBMSs implement just some of SQL IC spec).
- Triggers can respond to changes in the database (and make up the difference when the set of available integrity features falls short).
- Stored procedures (and CALL) are also available.
- Views and authorization are both useful features, and can be especially powerful in combination. (!)

That's it for SQL!

- ANY LINGERING QUESTIONS...?



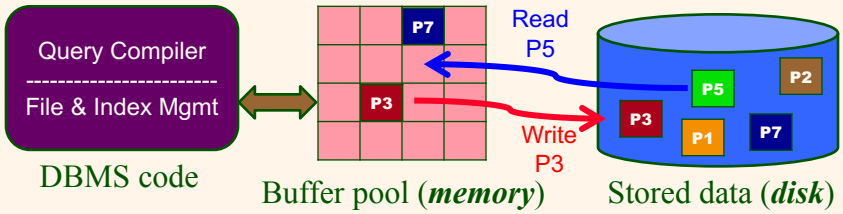
Roadmap Check…

Syllabus

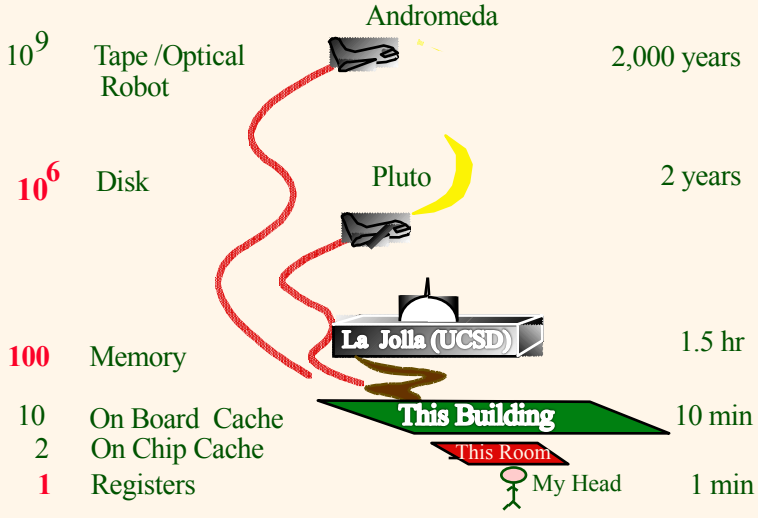
Topic	Reading
Databases and DB Systems	Ch. 1
Entity-Relationship (E-R) Data Model	Ch. 2.1-2.5, 2.8
Relational Data Model	Ch. 3.1-3.2
E-R to Relational Translation	3.5
Relational Design Theory	Ch. 19.1-19.6, 20.8
Midterm Exam 1	*Mon, Apr 29* (in class)
Relational Algebra	Ch. 4.1-4.2
Relational Calculus	Ch. 4.3-4.4
SQL Basics (SPJ and Nested Queries)	Ch. 3.4, 5.1-5.3
SQL Analytics (Aggregation, Nulls, and Outer Joins)	Ch. 5.4-5.6
Advanced SQL Goodies (Constraints, Triggers, Views, and Security)	Ch. 3.3, 3.6, 5.7-5.9, 21.1-21.3, 21.7
Midterm Exam 2	*Wed, May 22* (in class)
Tree-Based Indexing	Ch. 9.1, 8.1-8.3, 10.1-10.2
Hash-Based Indexing	Ch. 10.3-10.8, 11.1
Physical DB Design	Ch. 8.5, 20.1-20.7
Semistructured Data Management (*a.k.a.* NoSQL)	AsterixDB SQL++ Primer, Couchbase SQL++ Book
Basics of Transactions	Ch. 16 and Lecture Notes
Endterm Exam	*Fri, Jun 7* (in class)

Disks and Files

- DBMSs store data on *secondary storage*.
- This has major implications for DBMS design!
  - READ: xfer data from ***disk*** (or flash) to ***memory*** (RAM).
  - WRITE: xfer data from RAM to disk (or flash).
  - Both are high-cost operations, relative to in-memory operations, so must be considered carefully!



Storage Hierarchy & Latency (Jim Gray in the mid 1990's): How Far Away is my Data?

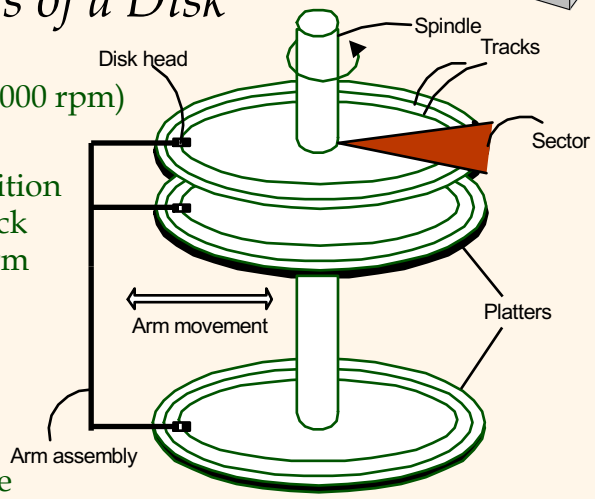


Why Not Store Data in Main Memory?

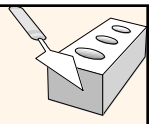
- *Main memory (RAM) costs too much! Roughly:*
  - RAM: $22/GB [Dell @ 05/2019]
  - SSD: $1/GB (22x cheaper than RAM)
  - Disk: $0.16/GB (138x cheaper than RAM, 5.5x *vs.* SSD)
- *Main memory is **volatile**.* We want our data to be saved between runs. (Obviously...!)
- Your typical (basic) storage hierarchy:
  - Main memory (RAM) for currently used data
  - *(SSD often sits here now as well...)*
  - Disk for the main database (secondary storage)
  - Tapes for archiving the data (tertiary storage)

Components of a Disk

- The platters spin (10,000 rpm)
- The arm assembly is moved in or out to position a head on a desired track Tracks under heads form a *cylinder* (imaginary!)
- Only one head reads/writes at any one time.
- *Block size* is a multiple of *sector size* (which is fixed)



Accessing a Disk Page



- Time to access (read/write) a disk block:
  - *Seek time* (moving arms to position disk head on track)
  - *Rotational delay* (waiting for block to rotate under head)
  - *Transfer time* (actually moving data to/from disk surface)
- Seek time and rotational delay dominate!
  - Seek time varies from about 1 to 20msec
  - Rotational delay varies from 0 to 10msec
  - Transfer rate is < 1 msec per 4KB page
  - Key to lowering I/O cost: Reduce seek/rotation delays! → **Bottom line:** ***Random vs. sequential I/O***