Midterm Exam Review
BLONDIE YOUNG & GERSHER

I'VE GOT A MATH EXAM TOMORROW

AND IT'S GONNA BE A REAL TOUGH ONE!

ARE YOU STUDYING FOR IT?

THAT'S WHY IT'S GONNA BE SO TOUCHING
Review All Homework, Exercises and Quizzes
Review the following SQL from Class

Types of Joins:

- **Inner Joins** - the following 3 are essentially the same except for syntax:
  - `SELECT * FROM emp e, dept d WHERE e.deptno = d.deptno`
  - `SELECT * FROM emp e JOIN dept d ON e.deptno = d.deptno`
  - `SELECT * FROM emp e INNER JOIN dept d ON e.deptno = d.deptno`

- **Outer Joins**:
  - `SELECT * FROM emp e RIGHT OUTER JOIN dept d ON e.deptno = d.deptno`
  - `SELECT * FROM emp e LEFT OUTER JOIN dept d ON e.deptno = d.deptno`
  - `SELECT * FROM emp e FULL OUTER JOIN dept d ON e.deptno = d.deptno`

- **Self Join**:
  - `SELECT e1.*, 'Manager is: ' || e2.ename
    FROM EMP e1
    LEFT OUTER JOIN EMP e2
    ON e2.empno = e1.mgr`


**Views**:

- `CREATE VIEW emp_v1 AS
  SELECT ename, job, mgr FROM emp
  SELECT * FROM emp_v1`

**Subqueries and Group By**:

- `SELECT e.*, (SELECT AVG(sal) FROM emp WHERE mgr = e.empno) AS avg_sal
  FROM (SELECT empno, mgr, sal FROM emp) e
  WHERE empno IN (SELECT mgr FROM emp)
  ORDER BY 1;`

- `SELECT mgr, AVG(sal) FROM emp
  WHERE sal >= 3000
  GROUP BY mgr
  HAVING AVG(sal) > 3000
  ORDER BY 1;`

**Interesting Queries**:

- `SELECT e.*, NVL(sal * comm, 0) FROM emp e`
Know how to construct Relationships and do Queries in the following Data Models

**Schema-first (i.e., a Relational Model is required) DBMS**

Oracle emp/dept (scott/tiger) **Relational database:**
Cut and paste from this link into SQLDeveloper to create the emp and dept tables in your Oracle user account.

**Schema-later DBMSs**

Contrived Hierarchical emp/dept **HTML databases:**
Load these into Chrome and then turn on the JavaScript Console (option-command-j on the Mac) to view the Elements structure.

- HTML DBMS #1
- HTML DBMS #2

emp/dept **Neo4j database:**
The emp/dept Neo4j DDL can be viewed at this link. You can cut and paste the sql from this link into SQLDeveloper to see how I generated the neo4j emp/dept DDL from the Relational emp/dept database.

emp/dept **SIM database:**
Download a prototype SIM DBMS named WDB from here.
Download the emp/dept WDB example at this link into the top level folder of WDB. Then follow the instructions at the top of this example file.
A WDB document can be found at these two links, Bo's Thesis, Saurabh's Thesis.

**Oracle emp/dept RDF/OWL database:**
The emp/dept RDF/OWL DDL can be viewed at this link.

**Oracle emp/dept JSON database:**
The emp/dept json DDL can be viewed at this link. You can cut and paste the sql from this link into SQLDeveloper to see how I generated the json emp/dept DDL from the Relational emp/dept database. This material was derived from the following link.

However, it’s best to have an insert for each dept/emp pairing. To do this, see this link.

Extra Credit – see the last page(s) of these notes for details.
Why is the SIM Data Model better than the Relational Data Model?

Using SIM to build an application, would you need a,

• Logical Model
• Relational Model
• a ddl2 file
• a ddl3 file
• a ddl4 file

The answer to all of these is No, why?
Basic Data Model Building Blocks

- **Entity (should really be called and Relation)** - anything about which data are to be collected and stored
- **Attribute (should really be called and Domain)** - a characteristic of an Entity
- **Relationship** - describes an association among Entities
  - One-to-many (1:M) relationship
  - Many-to-many (M:N or M:M) relationship
  - One-to-one (1:1) relationship
  - Inheritance
- **Constraint** - a restriction placed on the data
Basic Data Model Building Blocks

- Be able to draw a logical model from DDL
  - A Foreign Key which has a Uniqueness Constraint ➔ One-to-One Relationship
  - A Table with a compound key where each component of the compound key is a Foreign Key and the Referential Integrity is Cascade Delete ➔ Many-to-Many Relationship
  - Other Foreign Keys ➔ One-to-Many Relationships
  - If a Foreign Key is a Primary Key or part of a Primary Key, then it is Identifying

- Be able to write the DDL for a logical model

See the next three pages for details.
(Note: these details are a bit different from what I taught you for building an Apex application, for instance, the many-to-many table in these details has a compound primary key.)
SQLDeveloper Data Modeler - Logical Model

```
<table>
<thead>
<tr>
<th>Air_city</th>
</tr>
</thead>
<tbody>
<tr>
<td>P * city_id</td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>airport_code</td>
</tr>
<tr>
<td>F * city_id</td>
</tr>
<tr>
<td>AIR_city PK (city_id)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Air_airplane</th>
</tr>
</thead>
<tbody>
<tr>
<td>P * airplane_id</td>
</tr>
<tr>
<td>seats</td>
</tr>
<tr>
<td>F * airline_id</td>
</tr>
<tr>
<td>AIR_airplane PK (id)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Air_airline</th>
</tr>
</thead>
<tbody>
<tr>
<td>P * airline_id</td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>AIR_airline PK (id)</td>
</tr>
</tbody>
</table>
```
SQLDeveloper Data Modeler - Relational Model
CREATE TABLE AIR_airline(
    airline_id INTEGER NOT NULL,
    name VARCHAR2 (4000) ) ;
ALTER TABLE AIR_airline
ADD CONSTRAINT "AIR_airline PK"
PRIMARY KEY ( airline_id ) ;

CREATE TABLE AIR_airplane(
    airplane_id INTEGER NOT NULL,
    seats INTEGER,
    airline_id INTEGER ) ;
ALTER TABLE AIR_airplane
ADD CONSTRAINT "AIR_airplane PK"
PRIMARY KEY ( airplane_id ) ;

CREATE TABLE AIR_city(
    city_id INTEGER NOT NULL,
    name VARCHAR2 (100 BYTE),
    airport_code VARCHAR2 (10 BYTE),
    city_id1 INTEGER ) ;
CREATE UNIQUE INDEX AIR_city__IDX ON AIR_city (city_id1 ASC ) ;
ALTER TABLE AIR_city
ADD CONSTRAINT "AIR_city PK"
PRIMARY KEY ( city_id ) ;

CREATE TABLE AIR_services(
    AIR_city_city_id INTEGER NOT NULL,
    AIR_airplane_airplane_id INTEGER NOT NULL ) ;
ALTER TABLE AIR_services
ADD CONSTRAINT AIR_services__IDX
PRIMARY KEY ( AIR_city_city_id, AIR_airplane_airplane_id ) ;
ALTER TABLE AIR_airplane
ADD CONSTRAINT AIR_Relation_4 FOREIGN KEY(airline_id)
REFERENCES AIR_airline (airline_id) ;
ALTER TABLE AIR_city
ADD CONSTRAINT AIR_Sister FOREIGN KEY (city_id1)
REFERENCES AIR_city (city_id) ;
ALTER TABLE AIR_services
ADD CONSTRAINT FK_ASS_3 FOREIGN KEY (AIR_city_city_id)
REFERENCES AIR_city (city_id)
ON DELETE CASCADE ;
ALTER TABLE AIR_services
ADD CONSTRAINT FK_ASS_4 FOREIGN KEY (AIR_airplane_airplane_id)
REFERENCES AIR_airplane (airplane_id)
ON DELETE CASCADE ;
Web-based Relational Application Development

- Know what parser1 and parser 2 produce, i.e., what’s in the ddl2 and ddl3 files? Also, what’s in the ddl4 file?
- Apex – Homework 5 and 6
Introducing the Database Management System (DBMS)

To See Your Data Dictionary - SELECT * FROM DICT WHERE TABLE_NAME LIKE 'USER_%' ORDER BY TABLE_NAME;
This is the Database Management System (DBMS) Architecture that we will be studying in this course. See the following articles for a detailed discussion of the Oracle Architecture:

1. See Lesson 1 of "Oracle Admin I Vol 1 D62541.pdf"
2. Oracle Architecture Overview
3. Oracle Instance Architecture
4. Oracle Architecture
5. Oracle DBMS Concepts
You need to know this process for create, insert and update statements.

For Create Table:
4. The Server Process then causes the next free 8K Block on the disk containing the data dictionary to be ...
5. The table DDL is inserted into the data dictionary.
6. Changes as appropriate.

For Insert:
4. The Server Process then causes the next free 8K Block on the disk from the tablespace containing the emp table to be ...
5. The row is inserted.
6. Changes as appropriate.

11. [10 Points] Assuming that an Oracle instance has just been started and the following SQL is issued for it to run:
   
   update emp set sal = 70000 where empno = 7369;
   commit;

   Fill in the blanks in the statements below that describe the steps followed by Oracle to execute this SQL.

1. The User Process contacts a **Listener** on a **port** on the Oracle Server. The **Listener** creates a **Serve Process** and establishes a connection between the **Server Process** and the User Process.
2. The **User Process** sends the SQL to the **Server Process**.
3. The **Server Process** creates an optimized execution plan for the SQL and stores this plan in the **Library Cache** with a unique hash code.
4. The **Server Process** then causes the **8K Byte Block** on the disk containing the emp **tuple** with empno equal to 7369 to be brought into the **Database Buffer Cache** in the **System Global Area**.
5. The value is modified.
6. The fact that the block has been modified and how it was modified is written into the **Redo Log Buffer**.
7. In addition, the next free **Undo Block** is brought into the **Database Buffer Cache** and information to undo the sql change is recorded in it.
8. The fact that the **Undo Block** has been modified and how it was modified is written into the **Redo Log Buffer** which, after the commit, gets flushed to the **Online Redo Log** by the LGWR Process.
9. One of the **DBW Processes** eventually writes the modified 8K Byte Blocks to disk.
Transactions, Concurrency Control, Query Optimization, and Indexing

- Transactions and Concurrency Control (Review my demos)
- Redo Log Operation
- Relational Algebra (be able to convert SQL to Relational Algebra)
- Query Optimization (be able to draw execution trees from Relational Algebra and do simple optimization of them)
- B-Trees

Details are on the next pages.
## Oracle Concurrency Control – read committed

<table>
<thead>
<tr>
<th>T1 (starts first)</th>
<th>T2 (starts second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>commit ;</td>
<td>commit ;</td>
</tr>
<tr>
<td>set transaction isolation level</td>
<td>set transaction isolation level</td>
</tr>
<tr>
<td><strong>read committed</strong> ;</td>
<td><strong>read committed</strong> ;</td>
</tr>
<tr>
<td>select sal from emp where empno = 7369 ;</td>
<td>select sal from emp where empno = 7369 ;</td>
</tr>
<tr>
<td>update emp set sal = sal + 1 where empno = 7369 ;</td>
<td>update emp set sal = sal + 10 where empno = 7369 ;</td>
</tr>
<tr>
<td>pause &quot;Press Enter to continue&quot;</td>
<td>pause &quot;Press Enter to continue&quot;</td>
</tr>
<tr>
<td>update emp set sal = sal + 1 where empno = 7369 ;</td>
<td>update emp set sal = sal + 10 where empno = 7369 ;</td>
</tr>
<tr>
<td>select sal from emp where empno = 7369 ;</td>
<td>select sal from emp where empno = 7369 ;</td>
</tr>
<tr>
<td>commit ;</td>
<td>commit ;</td>
</tr>
<tr>
<td>select sal from emp where empno = 7369 ;</td>
<td>select sal from emp where empno = 7369 ;</td>
</tr>
</tbody>
</table>
Oracle Concurrency Control – *read committed*

<table>
<thead>
<tr>
<th>Output from T1 (started first)</th>
<th>Output from T2 (started second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>committed.</td>
<td>committed.</td>
</tr>
<tr>
<td>transaction ISOLATION succeeded.</td>
<td>transaction ISOLATION succeeded.</td>
</tr>
<tr>
<td>SAL</td>
<td>SAL</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>910</td>
<td>910</td>
</tr>
<tr>
<td>1 rows updated.</td>
<td>1 rows updated.</td>
</tr>
<tr>
<td>1 rows updated.</td>
<td>1 rows updated.</td>
</tr>
<tr>
<td>SAL</td>
<td>SAL</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>912</td>
<td>932</td>
</tr>
<tr>
<td>committed.</td>
<td>committed.</td>
</tr>
<tr>
<td>SAL</td>
<td>SAL</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>912</td>
<td>932</td>
</tr>
</tbody>
</table>
### Oracle Concurrency Control – *serializable*

<table>
<thead>
<tr>
<th>T1 (starts first)</th>
<th>T2 (starts second)</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>commit ;</code></td>
<td><code>commit ;</code></td>
</tr>
<tr>
<td><code>set transaction isolation level serializable ;</code></td>
<td><code>set transaction isolation level serializable ;</code></td>
</tr>
<tr>
<td><code>select sal from emp where empno = 7369 ;</code></td>
<td><code>select sal from emp where empno = 7369 ;</code></td>
</tr>
<tr>
<td><code>update emp set sal = sal + 1</code></td>
<td><code>update emp set sal = sal + 10</code></td>
</tr>
<tr>
<td><code>where empno = 7369 ;</code></td>
<td><code>where empno = 7369 ;</code></td>
</tr>
<tr>
<td><code>pause &quot;Press Enter to continue&quot;</code></td>
<td><code>pause &quot;Press Enter to continue&quot;</code></td>
</tr>
<tr>
<td><code>update emp set sal = sal + 1</code></td>
<td><code>update emp set sal = sal + 10</code></td>
</tr>
<tr>
<td><code>where empno = 7369 ;</code></td>
<td><code>where empno = 7369 ;</code></td>
</tr>
<tr>
<td><code>select sal from emp where empno = 7369 ;</code></td>
<td><code>select sal from emp where empno = 7369 ;</code></td>
</tr>
<tr>
<td><code>commit ;</code></td>
<td><code>commit ;</code></td>
</tr>
<tr>
<td><code>select sal from emp where empno = 7369 ;</code></td>
<td><code>select sal from emp where empno = 7369 ;</code></td>
</tr>
</tbody>
</table>
### Output from T1 (started first)

| committed.  
| transaction ISOLATION succeeded.  
| SAL  
| -------  
| 934  

1 rows updated.  

| 1 rows updated.  
| SAL  
| -------  
| 936  

### Output from T2 (started second)

| committed.  
| transaction ISOLATION succeeded.  
| SAL  
| -------  
| 934  

Error starting at line 5 in command:
update emp set sal = sal + 10  
where empno = 7369  
Error report:
SQL Error: ORA-08177: can't serialize access for this transaction  
08177. 00000 - "can't serialize access for this transaction"  
*Cause: Encountered data changed by an operation that occurred after the start of this serializable transaction.  
*Action: In read/write transactions, retry the intended operation or transaction.

Error starting at line 8 in command:
update emp set sal = sal + 10  
where empno = 7369  
Error report:
SQL Error: ORA-08177: can't serialize access for this transaction  
08177. 00000 - "can't serialize access for this transaction"  
*Cause: Encountered data changed by an operation that occurred after the start of this serializable transaction.  
*Action: In read/write transactions, retry the intended operation or transaction.

| SAL  
| -------  
| 936  

| committed.  
| SAL  
| -------  
| 934  

| committed.  
| SAL  
| -------  
| 936
ACID Transactions

commit;
set transaction isolation level serializable;
select empno, ename, sal from emp
  where empno = 7839;
update emp set sal = sal + 10
  where empno = 7839;
select empno, ename, sal from emp
  where empno = 7839;
pause "Press Enter to continue";
select empno, ename, sal from emp
  where empno = 7839;
commit;
set serverout on
select empno, ename, sal from emp
  where empno = 7499;
select empno, ename, sal from emp
  where empno = 7839;
begin
commit;
set transaction isolation level serializable;
update emp set sal = sal + 1
  where empno = 7839;
update emp set sal = sal + 1
  where empno = 7499;
update emp set sal = sal + 1
  where empno = 7839;
commit;
dbms_output.put_line("*** Commit done.");
exception
  when others
    then rollback;
dbms_output.put_line("*** Rollback
done.");
end;
/
select empno, ename, sal from emp
  where empno = 7499;
select empno, ename, sal from emp
  where empno = 7839;
Review pages 13 to the End, in particular, what is a:

- phantom, pages 13-16
- ACID Transaction properties, pages 17-20
- schedule, page 20, and 34
- serial schedule, page 22
- serializable schedule, page 23
- serializability, pages 25-30
- recoverable schedule, pages 31-33
- cascading rollback, page 32
- strict and rigorous two-phase locking (and what are the advantages of these), pages 34-42
- pitfalls of lock-based protocols
- deadlock
- pitfalls of timestamp-based protocols, pages 43-44
- multiversion two-phase locking, pages 45-48
Mon. September 28, and Wed. September 30 - Transactions - Concurrency Control, the Oracle Architecture, Redo Logs, and Recovery

1. Overview of Subqueries and Group By SQL statements. Also see the "Class SQL" tab in the middle section of this web page.
2. Transactions and Concurrency Control
   1. Class notes.
3. The Oracle Architecture
   1. Click here to see a larger image. Also see the "Oracle Architecture" tab in the middle section of this web page.
   2. See chapter one of the Oracle Admin I Vol 1 D62541.pdf at this link.
4. Redo Logs and Recovery
   1. Class notes.
   2. See "An Overview of the Mechanisms of Oracle RDBMS Transactions and Logs" at this link.
Example Query Optimization

select d.name, r.name, w.city from s_dept d, s_region r, s_warehouse w
where d.region_id = r.id and r.id = w.region_id
and w.country = 'US'

\[ \prod_{d.name, r.name, w.city} (\sigma_{d.region_id = r.id \text{ and } d.country=US} (\sigma_{r.id = w.region_id} (\rho_r (s_region) \times \rho_w (s_warehouse)) \times \rho_d (s_dept))) \]
B-Tree Example

Step 9
Insert 7839:3

New

7369:1 7499:1 7521:2 7698:2 7782:1 7788:2 7839:3 7844:2 7902:1
Extra Credit

select ename, dname from emp e join dept d on(e.deptno = d.deptno);

SELECT v1 "ename", v2 "dname"
FROM TABLE(SEM_MATCH(''
SELECT * WHERE {
    GRAPH <emp_SCHEMA> { ?s1 rdf:type :emp }
    GRAPH <dept_SCHEMA> { ?s2 rdf:type :dept }
    OPTIONAL { ?s1 :ename ?v1 }
    OPTIONAL { ?s2 :dname ?v2 }
} ,
SEM_MODELS('EMP_CS370_ONTOLOGIES'), null, SEM_ALIASES( SEM_ALIAS("", '#'), null )
);

SPARQL is in Red
(Know how to write the SPARQL for an SQL query)