SQL

CPS352: Database Systems

Simon Miner
Gordon College
Last Revised: 1/28/15

Agenda

- Check-in
- Relational Algebra
- Introduction to the Course Database (DB2)
- SQL Overview
- Team Exercises
- More SQL
- Homework 2

Check-in

Datatabases

...of the Bible

A Plethora of Plagues

Exodus 8-12

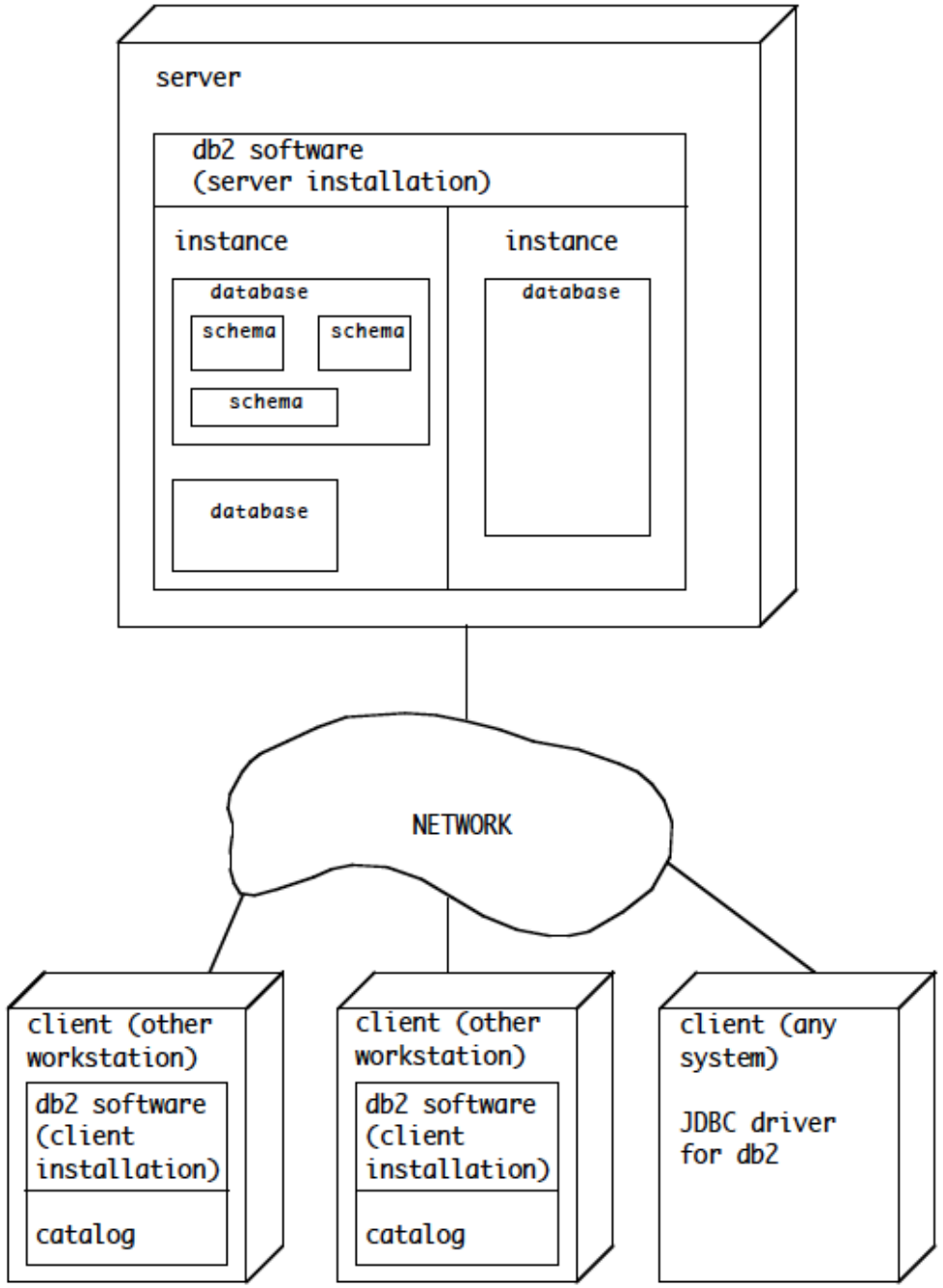
Introduction to the Course Database

DB2

DB2 Lives at KOSC.

- To be used for in class examples, homework assignments, and projects in this course
- DB2
  - Distributed by IBM
  - “Industrial strength” DBMS used all over the world

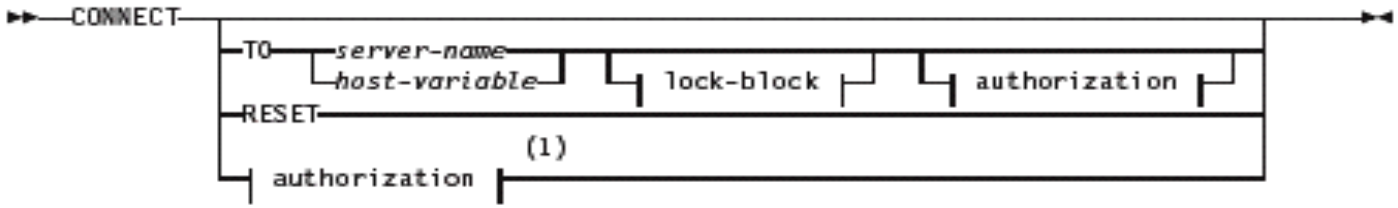
Course DB2 Architecture



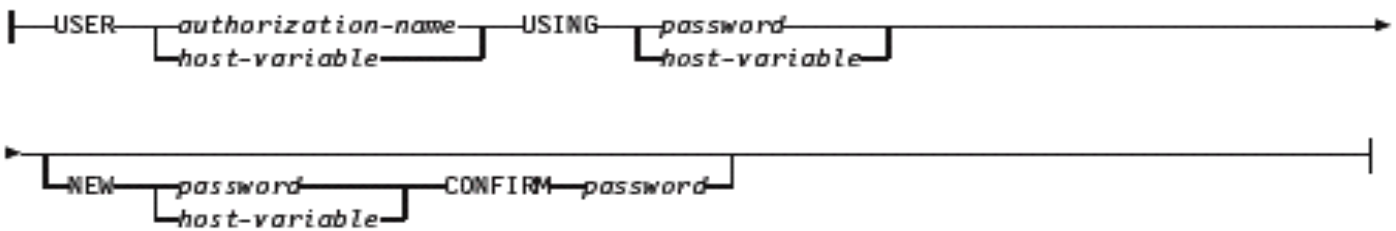
DB2 Resources are Available.

- Introduction to SQL Homework handout
- *DB2 LUW V9.7 Cookbook* by Graham Birchall
  - Most recent revision available online
  - Includes DDL for sample database in appendix
- *IBM DB2 Universal Database SQL Reference*
  - 2 volumes
  - Includes syntax diagrams for SQL statements
- These books are available on the course website.

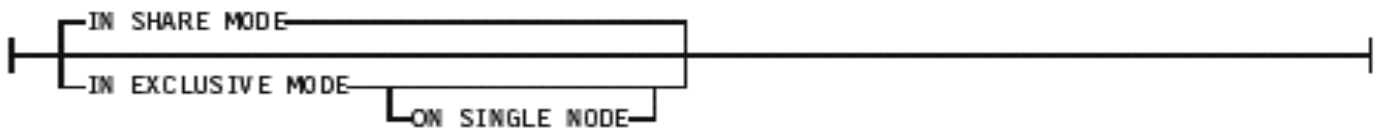
Here is an Example SQL Syntax Diagram.



authorization:



lock-block:



SQL Overview

There are several categories of SQL Statement.

- Select – querying the database
- DML – Changing the contents of the database
- DDL – Changing the database structure
- Database Integrity and Security Statements

Select Retrieves Data.

- Most fundamental and frequently used DML statement
  - Extensive variations – we will highlight a few in this session
  - Can be embedded in other SQL statements
- Example: List the names of all employees together with their supervisor's name
  - In Relational Algebra:
    - $\pi_{\text{e.last\_name, e.first\_name, s.last\_name, s.first\_name}}$ employee (ρ_{e} employee X $_{\theta\ \text{e..supervisor\_ssn = s.ssn}}$ ρ_{s} employee)
  - In SQL:
    - select e.last_name, e.first_name
      from employee as e join employee as s
      on e.supervisor_ssn = s.ssn
      where e.salary > s.salary;

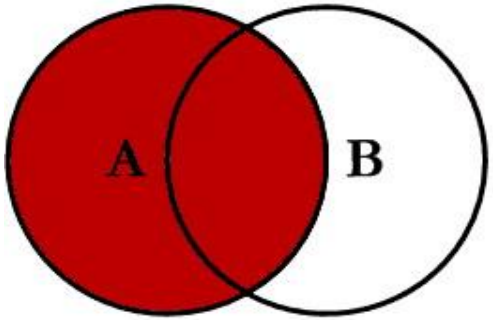
Select Statements can Have Several Clauses.

- Select – attributes to retrieve (projection operation)
  - Can include functions to compute based on attributes (or not)
  - Attributes and computed data can be renamed with "as" keyword (rename operation
- From – name of table(s) and/or views from which to fetch data
  - Can also be another select statement (subquery)
  - Join – names of tables and columns to join upon using "on" keyword (theta join)
- Where – criteria on data to fetch (selection operation)
  - Can be another select statement (subquery)
- Group by – aggregate data together (aggregation operations)
  - Having clause – limit aggregation
- Order by – criteria on how to sort matching records
- Limit – only display the first/last/range of X records
- Also union, intersect, and except (difference) keywords to join multiple selects

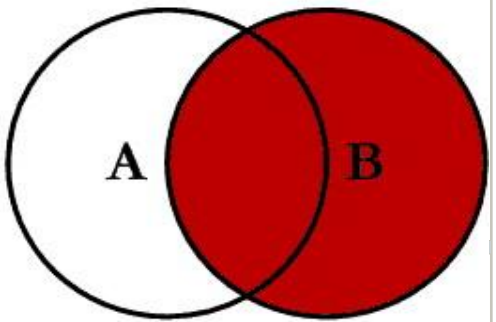
Joins

Join Type	Relational Algebra	SQL
Cartesian Product	$A \times B$	select * from A, B;
Natural Join	$A \mid\times\mid B$	select * from A natural join B; • not supported by many DBMSes (including DB2)
Theta Join	$A_{X\theta A(x) = B(y)}\ B$	select * from A join B on A.x = B.y; • this is how natural joins usually happen in SQL
Left Outer Join Right Outer Join Full Outer Join	$A \mathbin{⟕} B$ $A \mathbin{⟖} B$ $A \mathbin{⟗} B$	select * from A left/right/full join B on …;

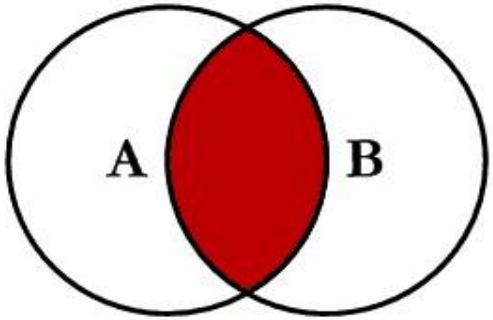
SQL JOINS



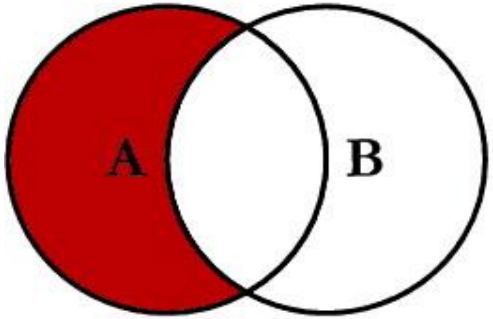

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



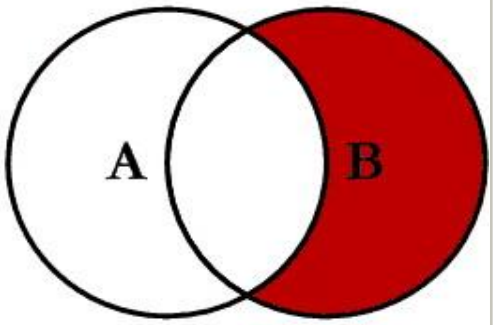

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



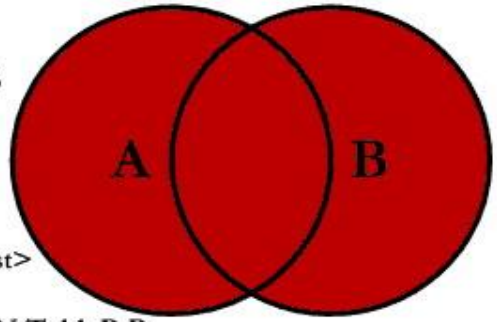

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



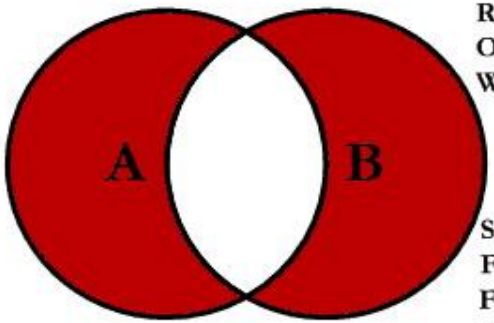

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

Subqueries let you embed one SQL query inside another.

- Embedding one SQL query within another
- Example: List the names and salaries of all employees earning more than the average salary for all employees.
- Need to first retrieve the average salary before we can compare it to that of each employee
  - Could be done as two separate queries
  - Or via a subquery
- select last_name, first_name, salary
  from employee
  where salary >
    (select avg(salary) from employee);

Subquery Result Sets

- Subqueries can return multiple results in a set
- “in” predicate – Does one of the subquery results match the criteria of the where clause?
- Example: List the names of all borrowers whose last name is the same as that of the author of a book.
- select last_name
  from borrower
  where last_name in (select author from book);

Subquery Results and Qualified Predicates

- "all" and "any" predicates
- Example: Print the name and salary of any employee earning more than all the employees in department E11.
  - select firstname, midinit, lastname, salary
    from employee
    where salary > all
      (select salary from employee where workdept = 'E11');
- Example: Print the name and salary of any employee earning more than some employee in department A00.
  - select firstname, midinit, lastname, salary
    from employee
    where salary > any
      (select salary from employee where workdept = 'A00');

Recursive Queries

- How can we select hierarchical data from a table?
  - Example: Select the name, SSN, and rank of the Chief Library Officer (the CLO -- who has no supervisor) and all employees who work under him.
- Subqueries and unions, but only to a point
  - CLO and direct reports (union)
  - CLO, direct reports, and their direct reports (another union with a subquery)
- Modern SQL supports recursive queries

How Recursive Queries Work.

- "with" clause to create a temporary table
  - Only exists for the duration of the SQL statement
- Temporary table is comprised a union of the following
  - The base case (i.e. the Chief Library Officer)
  - All of the employees who work under the CLO
  - "union all" used to connect these statements
- Select everything from the temporary table

Recursive Query Example

```
with emp_rank(ssn, last_name, first_name, supervisor_ssn, rank) as (
    select ssn, last_name, first_name, supervisor_ssn, 1
        from employee
        where supervisor_ssn is null
  union all
    select e.ssn, e.last_name, e.first_name, e.supervisor_ssn, s.rank+1
        from employee e join emp_rank er on e.supervisor_ssn = er.ssn
)
select * from emp_rank;
```

DML Statements Change the Contents of the Database.

- Change the contents stored within the database
- Insert
- Update
- Delete
- Commit and Rollback

Insert Creates One or More Records.

- Add a record to a table
- 3 forms
  - insert into *table*
    (*column1*, *column2*, *column3*, …)
    values (*value1*, *value2*, *value3*, …)
  - insert into *table*
    values (*value1*, *value2*, *value3*, …)
    - This form is error prone. (What if the table structure changes?) In general, avoid this form.
  - insert into *table*
    select *value1*, *value2, value3* from …
    - Embedded select statement

Update Changes One or More Records.

- Modify the data in a record in the table
- General form
  - update *table*
    set *column1* = *value1*, *column2* = *value2*, …
    where *condition(s)*
- Can include subqueries
- Example; Give all employees directly supervised by aardvark a 10% raise.
  - update employee
    set salary = salary * 1.1
    where supervisor_ssn =
    (select ssn from employee where last_name = 'aardvark');

Delete Removes One or More Records.

- Remove a record from the table
- General form
  - delete from *table* where *condition(s)*
- Example: Delete the employee entry for George Giraffe
  - delete from employee where last_name = ‘giraffe’;

Commit and Rollback Control Database Transactions.

- Statements to complete a transaction
  - Commit writes changes to the database
  - Rollback backs out the changes without applying them to the database.
- Database interfaces (both interactive and programmatic) often include a feature to turn on auto-commit.
  - In the DB2 shell, this is on by default
  - Disable it with the +c switch

Team Exercises

- Break into teams of 3-4.
- Complete exercises 3.11 and 3.12 (all parts) on pages 108-109 of *Database System Concepts*
- The university database schema diagram on page 47 (figure 2.8) of the text.
- We will go over these exercises and their answers after the break.

DDL Statements Change the Structure of the Database.

- Change the structure of the database
- Create … - create a new object (i.e. schema, table, view, etc.)
- Alter … - modify an existing object (i.e. table, view)
- Drop … - remove an existing object (i.e. table, view)

Create Table

- Statement to create a new table, specifying
  - Column names and data types
  - Integrity constraints (i.e. primary, referential, unique),
  - Domain constraints (i.e. not null)
  - Other objects and configurable parameters related to the new table
- A table can be created via an embedded select statement
  - create table employee_copy as select * from employee;

Domain Types in SQL

- **char(n).** Fixed length character string, with user-specified length *n*.
- **varchar(n).** Variable length character strings, with user-specified maximum length *n*.
- **int.** Integer (a finite subset of the integers that is machine-dependent).
- **smallint.** Small integer (a machine-dependent subset of the integer domain type).
- **numeric(p,d).** Fixed point number, with user-specified precision of *p* digits, with *n* digits to the right of decimal point.
- **real, double precision.** Floating point and double-precision floating point numbers, with machine-dependent precision.
- **float(n).** Floating point number, with user-specified precision of at least *n* digits.

Built-in Data Types in SQL

- **date:** Dates, containing a (4 digit) year, month and date
  - Example: **date** '2005-7-27'
- **time:** Time of day, in hours, minutes and seconds.
  - Example: **time** '09:00:30' **time** '09:00:30.75'
- **timestamp**: date plus time of day
  - Example: **timestamp** '2005-7-27 09:00:30.75'
- **interval:** period of time
  - Example: interval '1' day
  - Subtracting a date/time/timestamp value from another gives an interval value
  - Interval values can be added to date/time/timestamp values

Large-Object Types

- Large objects (photos, videos, CAD files, etc.) are stored as a *large object*:
  - **blob**: binary large object -- object is a large collection of uninterpreted binary data (whose interpretation is left to an application outside of the database system)
  - **clob**: character large object -- object is a large collection of character data
  - When a query returns a large object, a pointer is returned rather than the large object itself.

Database Integrity and Security

- Integrity – concerned with ensuring the accuracy and consistency of data
  - Protecting data from accidental inaccuracy
- Security – concerned with allowing only authorized access to data
  - Protecting against unauthorized reads and malicious writes (that damage or erase data)

Integrity Constraints

- Entity integrity constraints
  - Primary key
  - Unique
- Referential integrity constraints
  - Foreign key
- Domain integrity constraints
  - Built in domains and data types
  - Not null
  - User defined types
    - “create domain” or (in DB2) “create distinct type” statement
  - Check clause

Assertions and Triggers

- Assertion – an invariant enforced whenever data is modified in the database
- Trigger – statement(s) which gets executed ("triggered") whenever a certain type of database modification occurs
  - Specifies the following
    - Action on which the trigger should be fired (i.e. insert, update, delete)
    - Whether the trigger should fire before or after the action
    - Trigger body – statement(s) to execute

SQL Security Concepts

- User Accounts
  - Sometimes tied in with user and group security of underlying system
    - DB2 “Authorization IDs” on the Gordon system do this
- Object-level security
  - Protecting individual database objects (i.e. database, schema, table, view, column)
- Privileges
  - The right to perform a certain action on a given database object

Examples of Privileges

Type / Level	Examples
Administrative	SYSADM – Do anything on the system (a.k.a. root)
Database	DBADM – Administrative control of a given database CREATETAB – Create tables in a database
Schema	CREATEIN – Create objects in the schema ALTERIN – Modify objects in the schema DROPIN – Remove objects in the schema
Table	SELECT, INSERT, UPDATE, DELETE ALTER – Modify the structure of the table CONTROL – Ability to grant privileges on the table to other users
Column	UPDATE – Update the contents of a column

Granting and Revoking Privileges

- The creator of a database object is its owner
  - This user can grant privileges on that object to others
- Other users can also grant privileges on the object
  - User with SYSADM or DBADMIN privilege
  - User with CONTROL privilege on the object
- Grant statement
  - grant *privilege* on *object* to *recipient*;
  - Includes a "with grant option" clause which transfers the ability to grant the specified privilege to the recipient
  - Grants can be applied to groups of users or everyone (public)
- Revoke statement
  - Provides the ability to revoke a privilege from a given user

Views

- Structure providing read access to the results of a query
  - create view *view_name* as *query;*
  - View can be queried just like any other table
  - Created "on the fly" as queries are executed against it
- Provides a means of fine grained access control
  - Limit rows available for view
  - Limit columns available for view
- Provides simple to access to complex query results
- Provides selective ability to inserting, updating, or deleting data in the underlying table
  - Inserts lead to null values being added to columns not in the view
  - Disappearing rows – What happens if a row is updated so it no longer appears in the view?

Homework 2