SQL: Data Manipulation Language

Part 1

Warning!

Do not confuse

the *relational algebra* **SELECT** operator

with the *SQL* **SELECT** operator...

they are not the same!

SELECT Syntax

SELECT [DISTINCT] *column_list1*
    [INTO *table2*]
    FROM *table_list*
    [WHERE *condition1*]
    [GROUP BY *column_list2*]
    [HAVING *condition2*]
    [ORDER BY *column_list3*]

SELECT Syntax

SELECT [DISTINCT] *column_list1*
[INTO *table2*]
FROM *table_list*
[WHERE *condition1*]
[GROUP BY *column_list2*]
[HAVING *condition2*]
[ORDER BY *column_list3*]

column_list is one or more columns separated by commas

SELECT Syntax

SELECT [DISTINCT] *column_list1*
[INTO *table2*]
FROM *table_list*
[WHERE *condition1*]
[GROUP BY *column_list2*]
[HAVING *condition2*]
[ORDER BY *column_list3*]

table_list is one or more tables separated by commas

Clauses

- **SELECT** displays columns
- **DISTINCT** compresses out duplicate rows
- **FROM** specifies which tables to use
- **WHERE** chooses rows to include
- **GROUP BY** reorganizes data into bands
- **HAVING** chooses groups to include
- **ORDER BY** sorts the output

SELECT Syntax (more formally)

Subselect General Form:

```
SELECT [all|distinct] expression {, expression}
    FROM tablename [corr_name] {, tablename [corr_name]}
    [WHERE search_condition]
    [GROUP BY column {, column}]
    [HAVING search_condition]
```

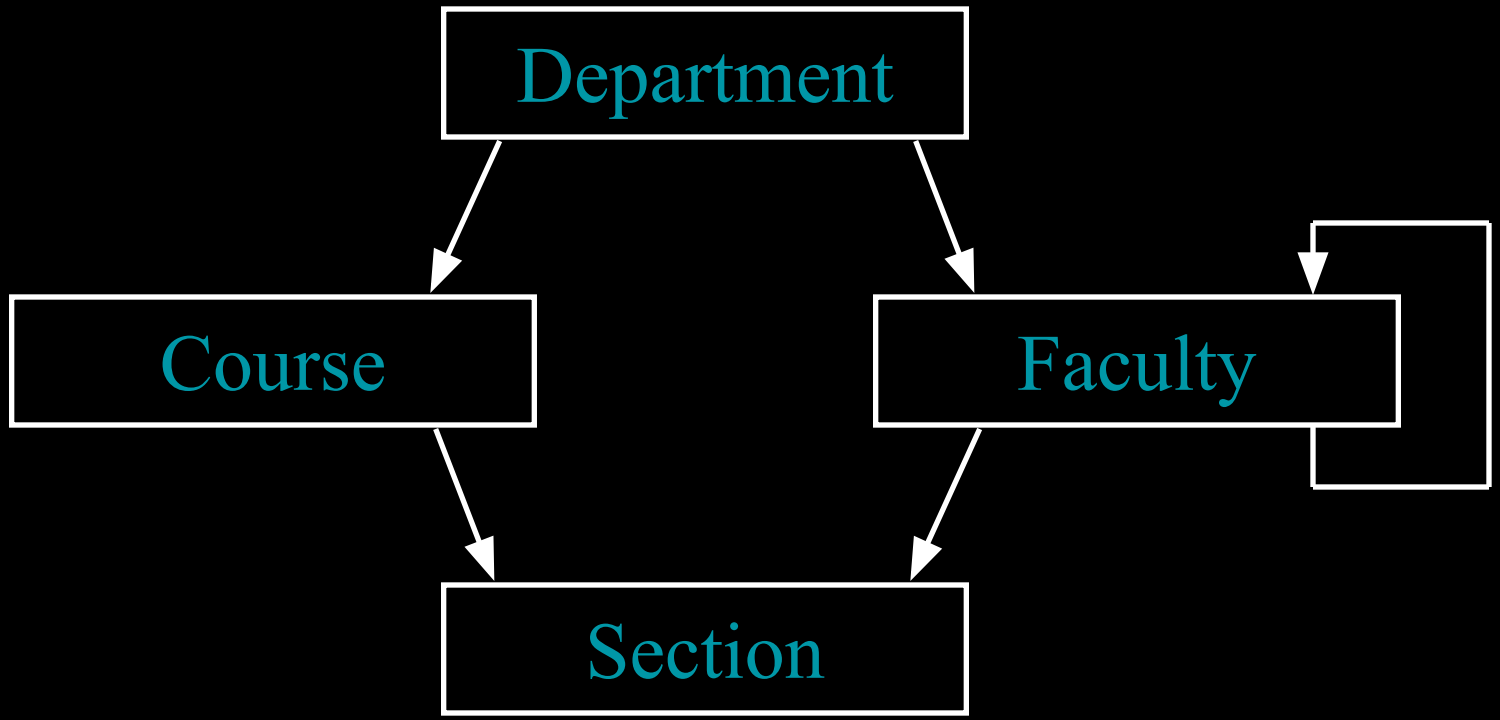
Full Select General Form:

```
Subselect
    {UNION [ALL] Subselect}
    [ORDER BY result_column [ ASC | DESC ] {, result_column [ ASC | DESC ]}]
```

Conceptual Order of Evaluation of a Select Statement

1. First the product of all tables in the **from** clause is formed.
2. The **where** clause is then evaluated to eliminate rows that do not satisfy the search_condition.
3. Next, the rows are grouped using the columns in the **group by** clause.
4. Then, Groups that do not satisfy the *search_condition* in the **having clause** are eliminated.
5. Next, the expressions in the **select** clause target list are evaluated.
6. If the **distinct** keyword is present in the select clause, duplicate rows are now eliminated.
7. The **union** is taken after each sub-select is evaluated.
8. Finally, the resulting rows are sorted according to the columns specified in the **order by** clause.

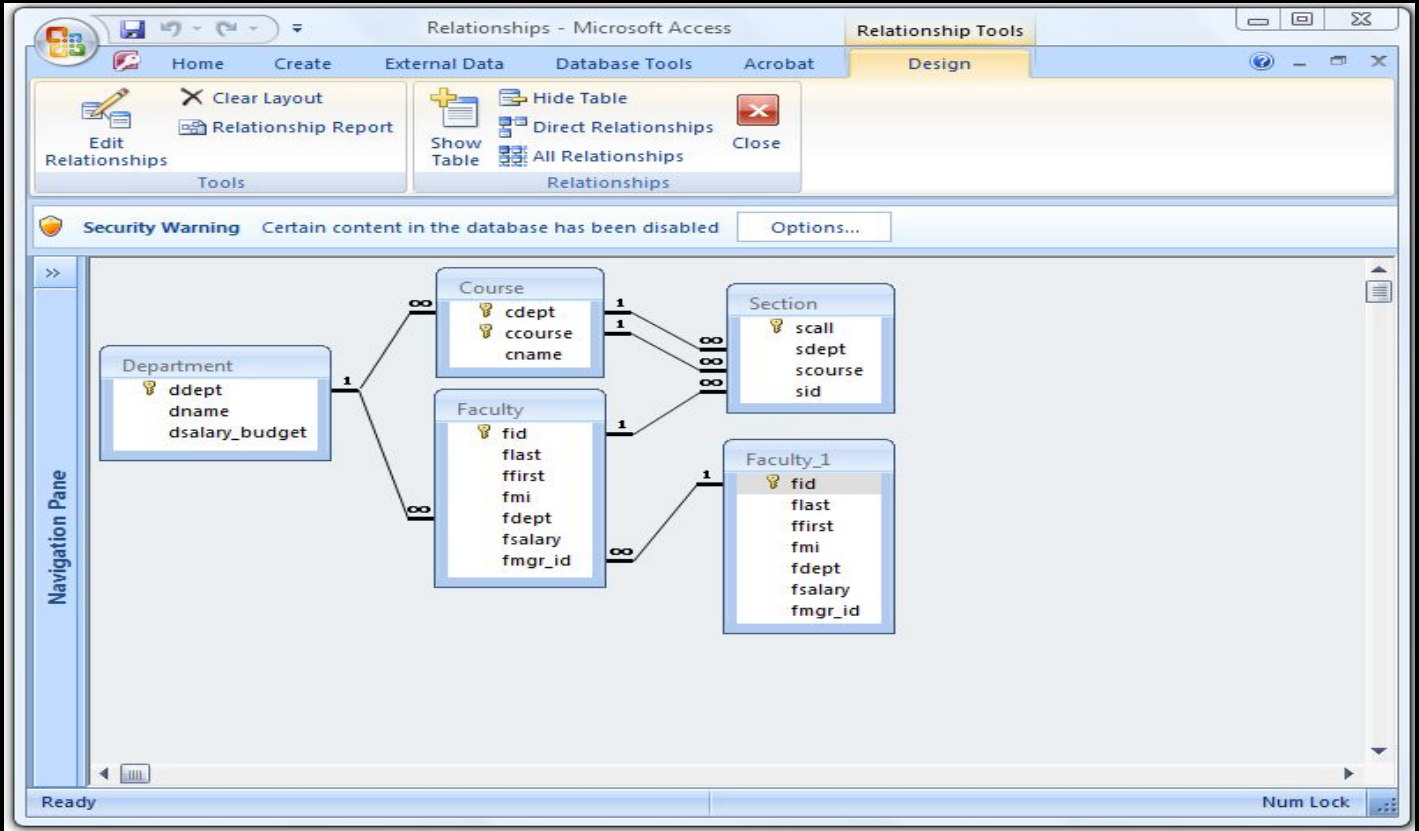
Sample Database



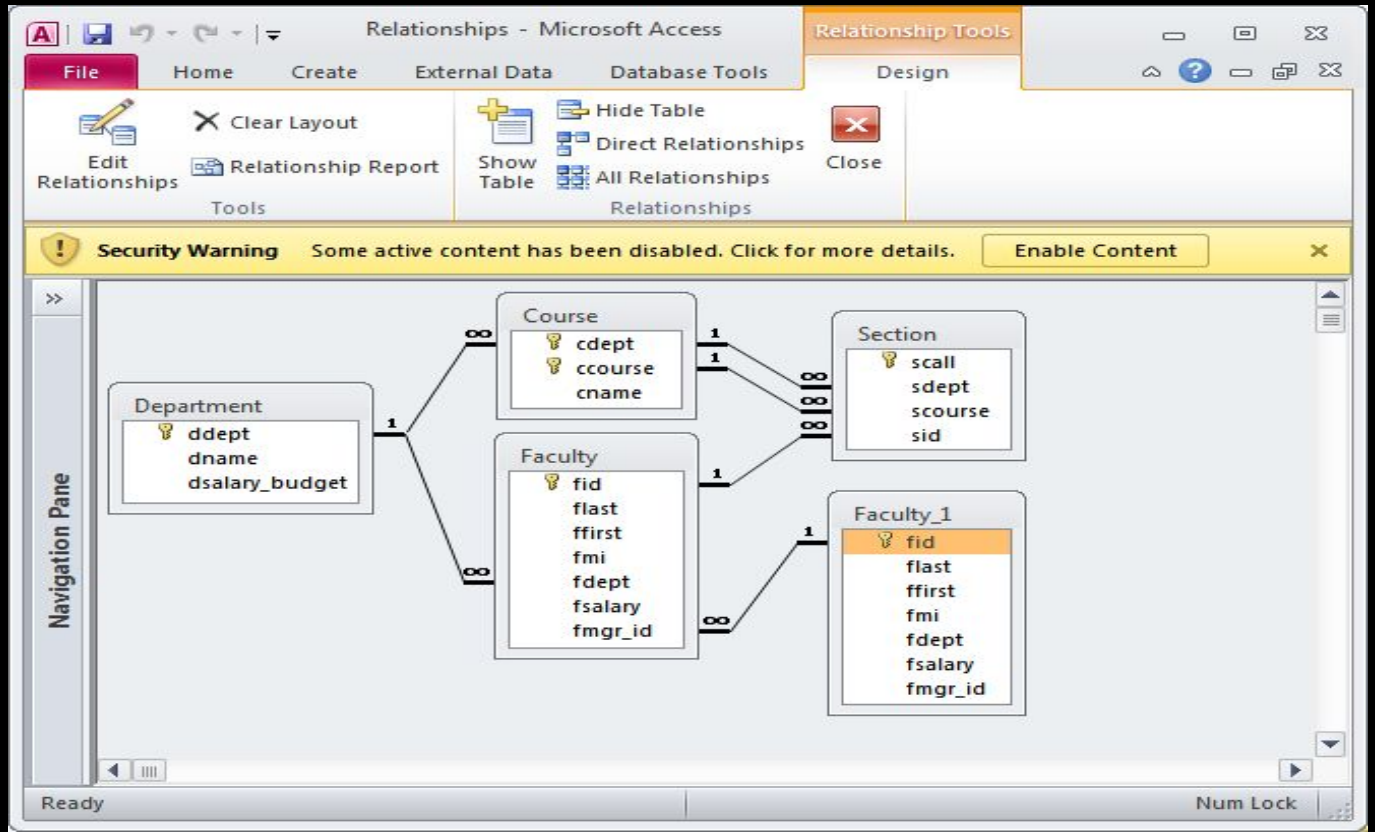
Relationships

Table	Primary Key	Foreign Key
Department	ddept	
Course	cdept & ccourse	cdept
Faculty	fid	fdept fmgr_id
Section	scall	sid sdept & scourse

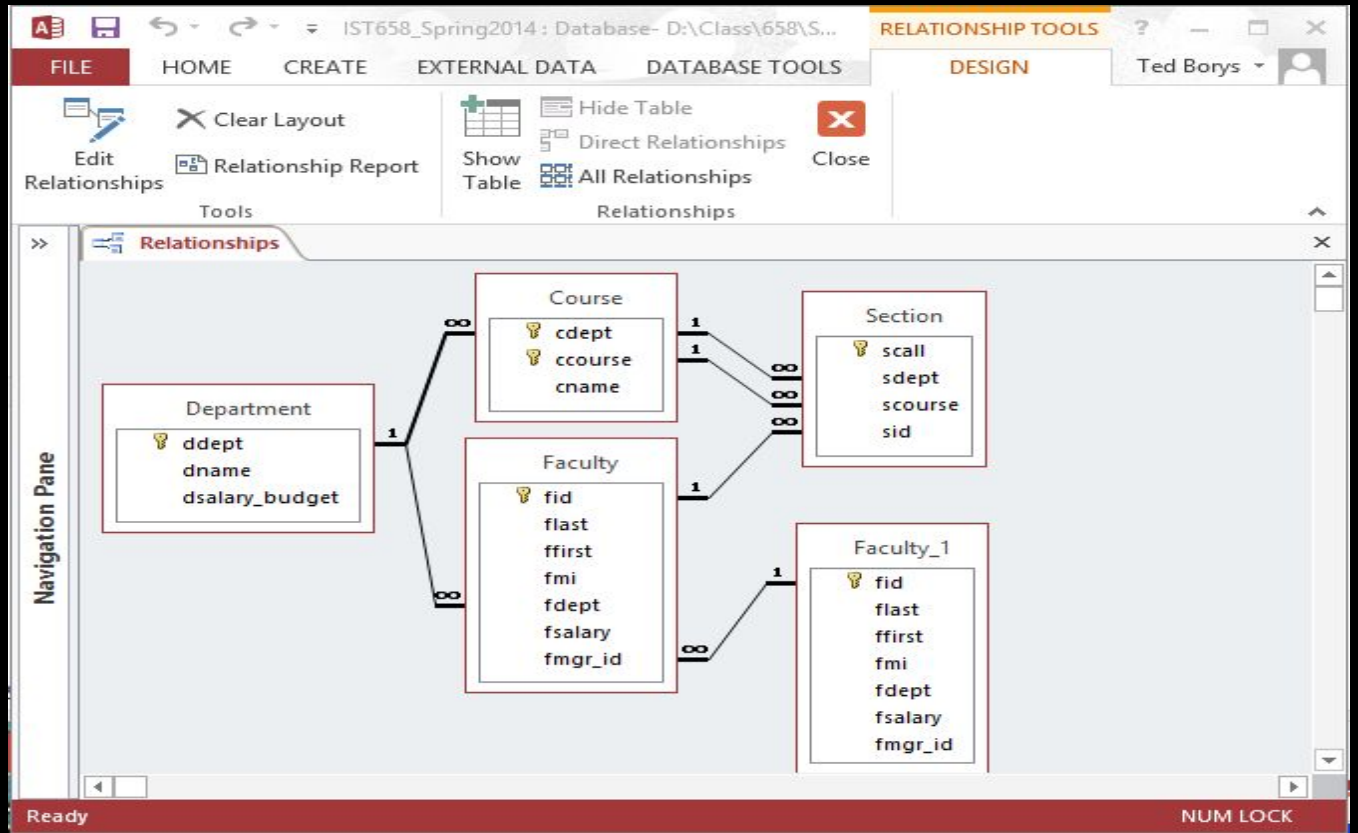
Access 2007


Relationships - Microsoft Access
Relationship Tools
Home
Create
External Data
Database Tools
Acrobat
Design
Edit Relationships
Clear Layout
Relationship Report
Tools
Show Table
Hide Table
Direct Relationships
All Relationships
Close
Relationships
Security Warning
Certain content in the database has been disabled
Options...
Navigation Pane
Department
ddept
dname
dsalary_budget
Course
cdept
ccourse
cname
Faculty
fid
flast
ffirst
fmi
fdept
fsalary
fmgr_id
Section
scall
sdept
scourse
sid
Faculty_1
fid
flast
ffirst
fmi
fdept
fsalary
fmgr_id
Ready
Num Lock

Access 2010


Relationships - Microsoft Access
Relationship Tools
File
Home
Create
External Data
Database Tools
Design
Edit Relationships
Clear Layout
Relationship Report
Tools
Show Table
Hide Table
Direct Relationships
All Relationships
Relationships
Close
Security Warning
Some active content has been disabled. Click for more details.
Enable Content
Navigation Pane
Department
ddept
dname
dsalary_budget
Course
cdept
ccourse
cname
Faculty
fid
flast
ffirst
fmi
fdept
fsalary
fmgr_id
Section
scall
sdept
scourse
sid
Faculty_1
fid
flast
ffirst
fmi
fdept
fsalary
fmgr_id
Ready
Num Lock

Access 2013


IST658_Spring2014 : Database- D:\Class\658\S...
RELATIONSHIP TOOLS
FILE
HOME
CREATE
EXTERNAL DATA
DATABASE TOOLS
DESIGN
Ted Borys
Edit Relationships
Clear Layout
Relationship Report
Tools
Show Table
Hide Table
Direct Relationships
All Relationships
Relationships
Close
Relationships
Navigation Pane
Course
cdept
ccourse
cname
Section
scall
sdept
scourse
sid
Department
ddept
dname
dsalary_budget
Faculty
fid
flast
ffirst
fmi
fdept
fsalary
fmgr_id
Faculty_1
fid
flast
ffirst
fmi
fdept
fsalary
fmgr_id
Ready
NUM LOCK

phpMySQL

phpMyAdmin

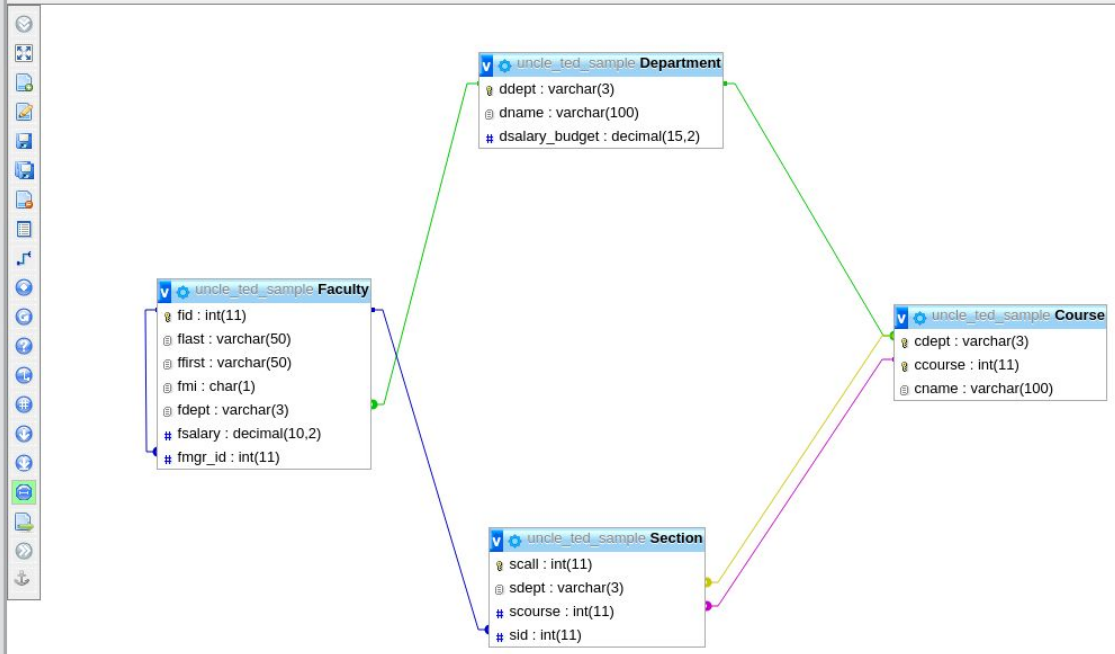
Recent Favorites

- information_schema
- learning_sql
- uncle_ted_sample
  - New
  - Course
  - Department
  - Faculty
  - Section

Server: mysql » Database: uncle_ted_sample

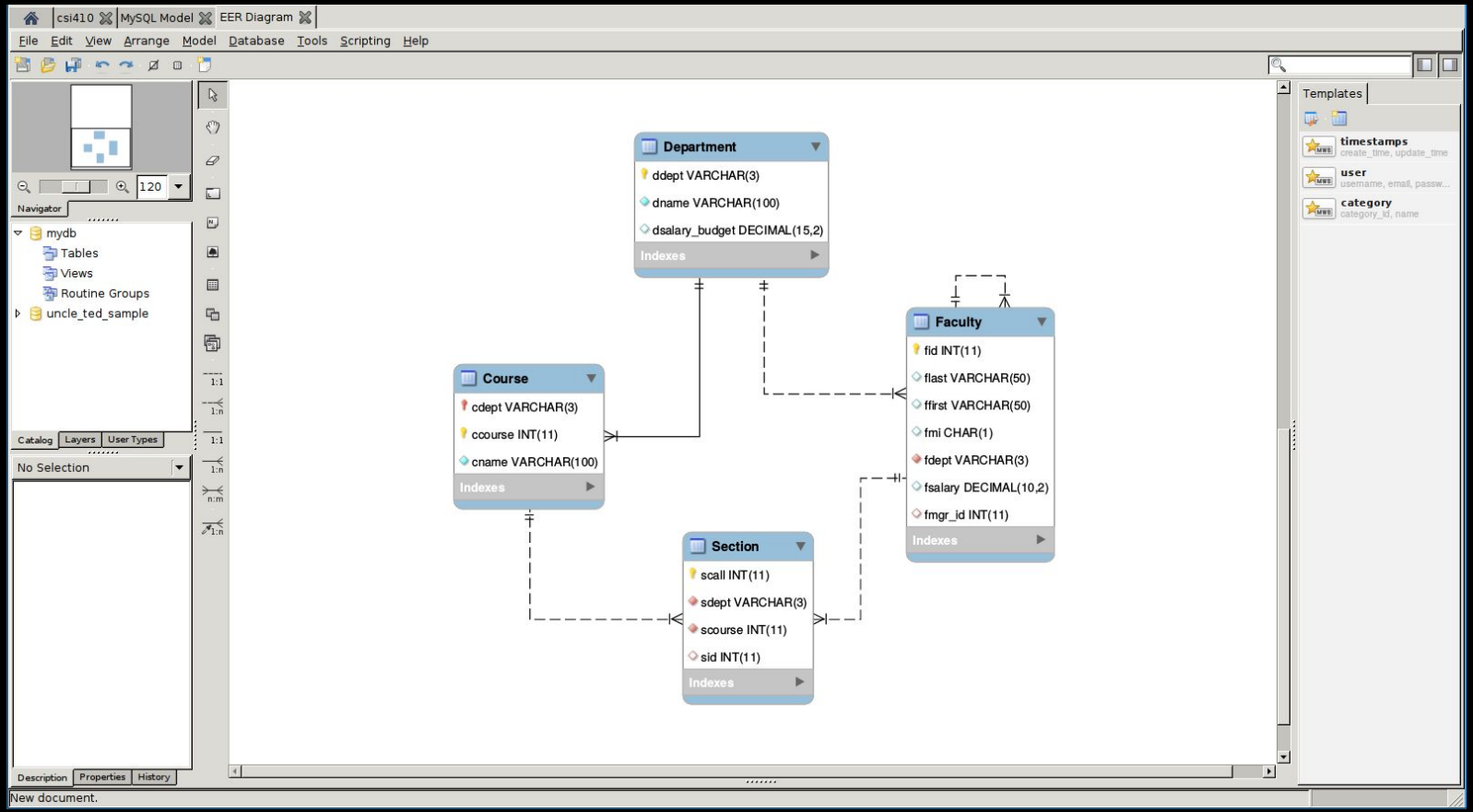
Structure | SQL | Search | Query | Export | Import | Operations | Routines | Designer

Untitled *



Console

MySQL Workbench



Department		
ddept	dname	dsalary_budget
ADM	Administration	200000
ATM	Atmospheric Science	90000
BIO	Biology	40000
CSI	Computer Science	180000
ENG	English	80000
SPN	Spanish	70000

Faculty

fid	flast	ffirst	fmi	fdept	fsalary	fmgr_id
12058	Borys	Ted	J	CSI	48000	22321
12206	Ryan	Alfred	C	ENG	48000	52110
21004	Perry	Bill	S	BIO	21800	31890
22321	Brady	Kathy	M	CSI	63400	52110
31890	Coulsen	Mary		BIO	21400	52110
32000	delBene	Bill	S	CSI	63500	22321
47862	Anders	John	P	ENG	33700	12206
52110	Smith	Alice		ADM	82000	*null*

Course		
cdept	ccourse	cname
ATM	408	Hydrometeorology
ATM	410	Dynamic Meteorology 1
BIO	205	Human Genetics
BIO	410	Human Physiology
CSI	205	C Language Programming
CSI	409	Automata & Formal Languages
CSI	410	Database Management Systems
ENG	427	The Victorian Period

Section			
scall	sdept	scourse	sid
102	ATM	408	22321
273	BIO	205	21004
285	BIO	410	22321
312	CSI	205	22321
313	CSI	205	47862
324	CSI	410	12058

Query 1

Display all the data in the Department table.

Query 1

Display all the data in the Department table.

```
SELECT ddept, dname, dsalary_budget
FROM   Department;
```

Query 1

Display all the data in the Department table

```
SELECT ddept, dname, dsalary_budget
FROM    Department;
```

or

```
SELECT *
FROM Department;
```

Query 1

Display all the data in the Department table

```
SELECT ddept, dname, dsalary_budget
FROM    Department;
```

or

```
SELECT *
FROM Department;
```

or

```
SELECT Department.*
FROM Department;
```

Query 2

Display the flast, ffirst, fdept, and fsalary columns from the Faculty table.

Query 2

Display the flast, ffirst, fdept, and fsalary columns from the Faculty table.

```
SELECT flast, ffirst, fdept, fsalary
FROM Faculty;
```

Query 3

Display all the first names in the Faculty table and remove duplicate rows.

Query 3

Display all the first names in the Faculty table and remove duplicate rows.

```
SELECT DISTINCT ffirst
FROM Faculty;
```

SQL Seduction 1

DISTINCT compresses out fully duplicated rows, i.e., it is applied to the combination of all the values of all the columns being displayed, not just the first column.

```
SELECT DISTINCT ffirst, flast
FROM Faculty;
```

Query 4

Display all the data in the Course table in *descending* cname *order*.

Query 4

Display all the data in the Course table in *descending* cname *order*.

```
SELECT *
FROM Course
ORDER BY cname DESC;
```

Query 5

Display the id, full name, and salary columns from the Faculty table and sort the output in telephone book order.

Query 5

Display the id, full name, and salary columns from the Faculty table and sort the output in telephone book order.

```
SELECT fid, ffirst, fmi, flast, fsalary
FROM Faculty
ORDER BY flast, ffirst, fmi;
```

How to Join Tables

- Follow the relationships
  - Equate owner's primary key to member's foreign key
  - Equate the primary key of one table to the primary key of another table (1:1 relationship)
- Intelligently joining on foreign key pairs is not likely
- Fight temptation to join on any pair of columns that appear similar
- Of course, there are exceptions …

Query 6

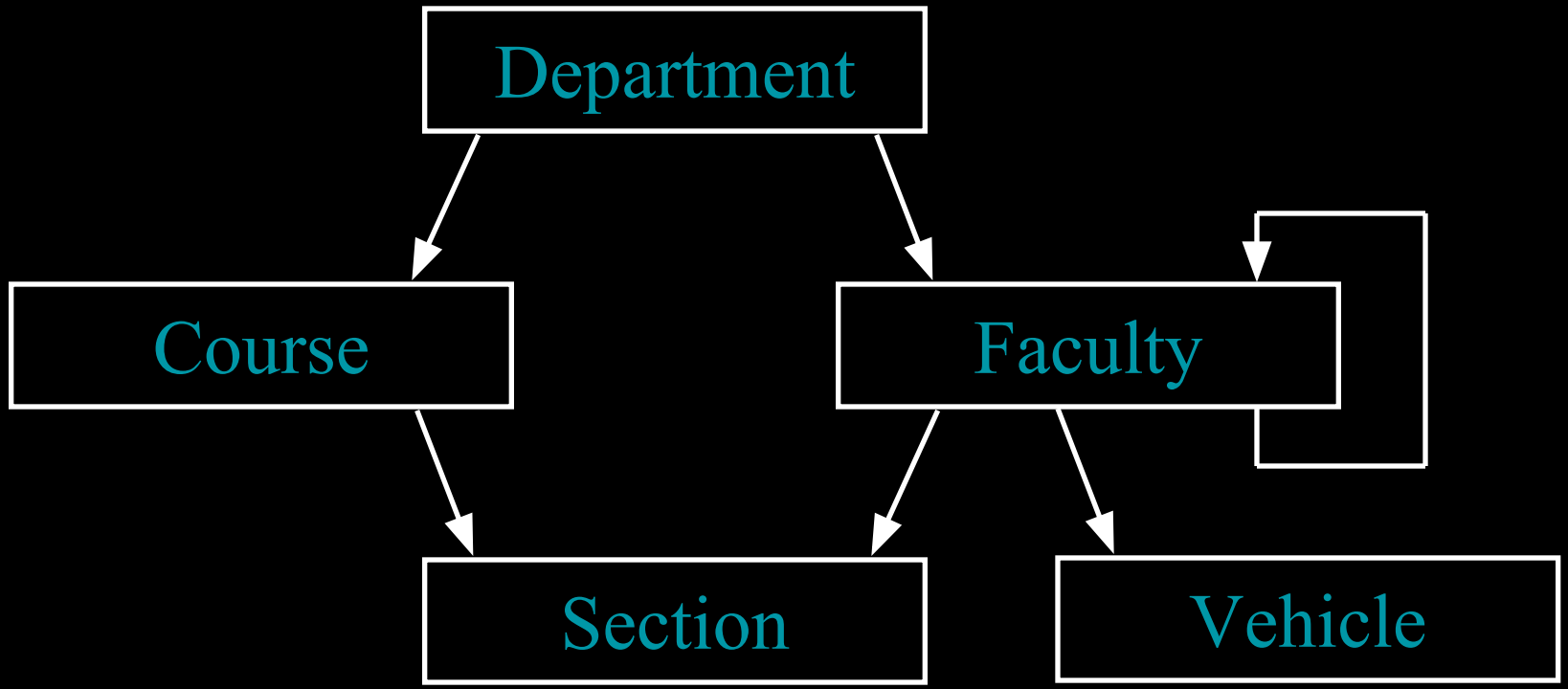
Join the Department and Faculty tables and display all the columns.

Query 6

Join the Department and Faculty tables and display all the columns.

```
SELECT *
FROM Department, Faculty
WHERE ddept = fdept;
```

Modified Sample Database



Continuing The Set Up…

- Eye color column added to Faculty: feye
- Vehicle table added
  - Primary key is vpermit#
  - Foreign key is vid
  - Vehicle has a color column: vcolor

SQL Seduction 2

Display all information from the Faculty and Vehicle tables for faculty whose eye color matches their vehicle’s color.

SQL Seduction 2

Display all information from the Faculty and Vehicle tables for faculty whose eye color matches their vehicle's color.

~~SELECT *~~
~~FROM Faculty, Vehicle~~
~~WHERE feye = vcolor;~~

Gotcha!

SQL Seduction 2

Display all information from the Faculty and Vehicle tables for faculty whose eye color matches their vehicle's color.

We need to use the foreign key!!!

```
SELECT *
FROM Faculty, Vehicle
WHERE (feye = vcolor) AND (fid = vid);
```

Query 7

Join the Faculty and Course tables.

Query 7

Join the Faculty and Course tables.

~~SELECT *~~
~~FROM Faculty, Course~~
~~WHERE fdept = cdept;~~

Gotcha again!

Query 7

Join the Faculty and Course tables.

```
SELECT Faculty.*, Course.*
FROM Faculty, Course, Section
WHERE (fid = sid)
    AND (cdept = fdept)
    AND (ccourse = scourse);
```

ANOTHER GOTCHA!!!

What about instructors teaching outside of their departments?

Query 7

Join the Faculty and Course tables.

```
SELECT Faculty.*, Course.*
FROM Faculty, Course, Section
WHERE (fid = sid)
    AND (cdept = sdept)
    AND (ccourse = scourse);
```

Query 7

This query shows us the rows missed when we assume instructors only teach within their dept.

```
SELECT Faculty.*, Course.*
FROM Faculty, Course, Section
WHERE (fid = sid)
    AND (ccourse = scourse)
    AND (cdept = sdept)
    AND (fdept <> cdept);
```

fid	flast	ffirst	fmi	fdept	fsalary	fmgr_id	cdept	ccourse	cname
22321	Brady	Kathy	M	CSI	63400	52110	ATM	408	Hydrometeorology
22321	Brady	Kathy	M	CSI	63400	52110	BIO	410	Human Physiology
47862	Anders	John	P	ENG	33700	12206	CSI	205	C Language Programming

NOTE: The table name prefixes for the column names were removed.

Query 8

Join the Department and Course tables.

Query 8

Join the Department and Course tables.

```
SELECT *
FROM Department, Course
WHERE ddept = cdept;
```

Query 9

Join the Course and Section tables.

Query 9

Join the Course and Section tables.

```
SELECT *
FROM Course, Section
WHERE (cdept=sdept)
    AND (ccourse=scourse);
```

Query 10

Do a 3-way join of the Department, Course, and Section tables.

Query 10

Do a 3-way join of the Department, Course, and Section tables.

```
SELECT *
FROM Department, Course, Section
WHERE (ddept = cdept)
    AND (cdept = sdept)
    AND (ccourse = scourse);
```

Query 11

Do a 3-way join of the Department, Faculty, and Section tables.

Query 11

Do a 3-way join of the Department, Faculty, and Section tables.

```
SELECT *
FROM Department, Faculty, Section
WHERE (ddept=fdept) AND (fid = sid);
```

Query 12

Incorrectly join the Department, Faculty, and Section tables (another SQL Seduction 2)

```
SELECT *
FROM Department, Faculty, Section
WHERE ddept=fdept AND fdept=sdept;
```

Query 12

Incorrectly join the Department, Faculty, and Section tables (another SQL Seduction 2)

```
SELECT *
FROM Department, Faculty, Section
WHERE (ddept = fdept)
   AND (fid = sid)
   AND (fdept = sdept);
```

Typical Subqueries

- WHERE *column_name* IN (*subquery*)
- WHERE EXISTS (*subquery*)
- **Standalone**
  - Can be executed by itself
  - Typically used with IN
- **Correlated**
  - Subquery uses column(s) from outer query's table
  - Can't be executed by itself
  - Typically used with EXISTS

Query 13

Find departments that offer courses.

Query 13

Find departments that offer courses.

```
SELECT *
FROM Department
WHERE ddept IN
    (SELECT cdept FROM Course);
```

How Standalone Subquery Works

- Subquery, inside parentheses, executes
  - Intermediate result generated once
  - Original statement reduces to:

```
SELECT *
FROM Department
WHERE ddept IN
  (“ATM”,“ATM”,“BIO”,“BIO”,“CSI”,“CSI”,“CSI”,“ENG”);
```

(SELECT cdept FROM Course)

How Standalone Subquery Works

- May want to use DISTINCT in subquery
  - This is only a potential performance issue
  - Correct result generated either way

Query 14

Another way to find departments that offer courses.

Query 14

Another way to find departments that offer courses.

```
SELECT *
FROM Department
WHERE EXISTS (
    SELECT cdept
    FROM Course
    WHERE cdept = ddept);
```

How Correlated Subquery Works

- EXISTS checks to see if rows are generated in subquery for each row examined in the main query.

- Subquery, inside parentheses, executes
  - Intermediate result generated for each row in Department.

Examining 1st Department Row

… the original statement internally reduces to:

```
SELECT *
FROM Department
WHERE EXISTS
   (SELECT cdept
    FROM Course
    WHERE cdept = “ADM” );
```

⇒

```
SELECT *
FROM Department
WHERE EXISTS
   ( );
```

Examining 1st Department Row

… the original statement internally reduces to:

```
SELECT *
FROM Department
WHERE EXISTS
   (SELECT cdept
    FROM Course
    WHERE cdept = “ADM” );
```

⇒

```
SELECT *
FROM Department
WHERE EXISTS
   (“ATM”, “ATM”);
```

IN versus EXISTS

- All queries that use IN can be re-written with EXISTS.
- Some queries that use EXISTS can be re-written with IN, **but not all can.**
- The theoretical reason why is beyond the scope of this class.

Query 15

And one more way to find departments that offer courses.

```
SELECT DISTINCT Department.*
FROM Department, Course
WHERE ddept = cdept;
```

Query 16

You can correlate this subquery. Does no logical harm, but adds *syntax clutter*.

```
SELECT *
FROM Department
WHERE ddept IN
    (SELECT cdept
     FROM Course
     WHERE cdept = ddept);
```

Query 17

Even this variation works correctly, despite the added *syntax clutter.*

```
SELECT *
FROM Department
WHERE ddept IN
    (SELECT cdept
     FROM Course, Department
     WHERE cdept = ddept);
```

SQL Seduction 3

But, this variation does not work. As long as there is at least one row in Course, every Department row will be displayed!

```
SELECT *
  FROM Department
  WHERE EXISTS
    (SELECT cdept
     FROM Course, Department
     WHERE cdept = ddept);
```

Query 18

Find departments that don’t offer courses.

Query 18

Find departments that don’t offer courses.

```
SELECT *
FROM Department
WHERE ddept NOT IN
    (SELECT cdept FROM Course);
```

Query 19

Another way to find departments that don't offer courses.

```
SELECT *
FROM Department
WHERE NOT EXISTS
    (SELECT cdept
     FROM Course
     WHERE cdept = ddept);
```

SQL Seduction 4

Find departments that don’t offer courses.

But this dog won’t hunt.
You can’t rewrite NOT IN as not equal.

```
SELECT DISTINCT Department.*
FROM Department, Course
WHERE ddept <> cdept;
```

Query 20

Find courses that have sections.

Query 20

Find courses that have sections.

```
SELECT DISTINCT Course.*
FROM Course, Section
WHERE (cdept = sdept)
    AND (ccourse = scourse);
```

Query 21

Another way to find courses that have sections.

```
SELECT *
FROM Course
WHERE cdept&ccourse IN
    (SELECT sdept&scourse FROM Section);
```

Use Access concatenation operator & to create single column.

Query 22

SQL99 standard lets you rewrite Query 21 like this, but Access 2007 disallows it.

```
SELECT *
FROM Course
WHERE (cdept, ccourse) IN
  (SELECT sdept, scourse FROM Section);
```

SQL Seduction 5

You can't replace the single, concatenated IN with two, singular INs (Hint: ATM 410).

```
SELECT *
FROM Course
WHERE cdept IN (SELECT sdept FROM Section)
  AND ccourse IN(SELECT scourse FROM Section);
```

Query 23

And one more way to find courses that have sections

```
SELECT *
  FROM Course
  WHERE EXISTS
    (SELECT * FROM Section
      WHERE (cdept = sdept)
        AND (ccourse = scourse));
```

Query 24

This is another incarnation of SQL Seduction 5

```
SELECT * FROM Course
WHERE EXISTS
   (SELECT * FROM Section WHERE cdept = sdept)
  AND EXISTS
   (SELECT * FROM Section WHERE ccourse = scourse);
```

Table Name Prefix

- Must be used when column names are not unique based on tables named in FROM clause
- Otherwise, can be used anytime
  - IMHO, adds syntax clutter

```
SELECT Course.cdept, Course.ccourse, Course.cname
FROM Course
ORDER BY Course.cname;
```

Table Name Alias

- Must be used when joining a table back on itself (also known as a *self-join*)
- Otherwise, can be used anytime
  - Useful when table name prefixing required, and table names are long
  - IMHO, when not needed, adds syntax clutter

```
SELECT C.cdept, C.ccourse, C.cname
FROM Course  C
ORDER BY C.cname;
```

Query 25

Join subordinates and managers… they both just happen to be in the same table: Faculty.

Query 25

Join subordinates and managers… they both just happen to be in the same table: Faculty.

```
SELECT *
FROM Faculty  Sub, Faculty  Mgr
WHERE Sub.fmgr_id = Mgr.fid;
```

Query 26

Find all the subordinates who make more than their managers.

Query 26

Find all the subordinates who make more than their managers.

```
SELECT *
FROM Faculty  Sub, Faculty  Mgr
WHERE (Sub.fmgr_id = Mgr.fid)
    AND (Sub.fsalary > Mgr.fsalary);
```

Query 27

List all the departments in Faculty and Course.

Query 27

List all the departments in Faculty and Course.

```
SELECT fdept FROM Faculty
UNION
SELECT cdept FROM Course;
```

Query 27

List all the departments in Faculty and Course.

```
SELECT fdept FROM Faculty
UNION
SELECT cdept FROM Course;
```

Note: duplicate rows eliminated

UNION ALL preserves duplicates

Query 28

Count the number of rows in Faculty.

Query 28

Count the number of rows in Faculty.

```
SELECT COUNT(*)
FROM Faculty;
```

Query 29

Count the number of *non-null* fid values in Faculty.

Query 29

Count the number of *non-null* fid values in Faculty.

```
SELECT COUNT(fid)
FROM Faculty;
```

Since fid is *primary key*, <u>never</u> null.

SQL Seduction 6

Counts non-null middle initial values; not necessarily the same as the number of rows.

~~SELECT COUNT(fmi)~~
~~FROM Faculty;~~

Query 30

Count the number of departments that offer courses.

```
SELECT COUNT(DISTINCT cdept)
FROM Course;
```

This syntax disallowed by Access.