Query Optimization

Background of problems :
The data stored in the database is getting bigger both size and volume, so it needs to be supported with access speed.

The reliability of the database system (DBMS) can be known from the way its optimizer works in processing SQL statements.

The query optimization is process of selecting the most efficient query-evaluation plan from the many strategies for processing the query, especially if the query is complex.

query optimization aims to :

1. Reducing as many as possible unneeded tuples.
2. Improving the evaluation strategy of query to make the evaluation more effective.
3. Finding the most efficient access path to minimize the total time of query process.

Query Optimization

Some simple attempts to improve DBMS effectiveness :

1. **Creating Index**
   Index can increase the speed on search records. In creating the index, index key fields are specified based on fields that are often accessed by clauses : *Where, Join, Order By, Group By.*

   Create Index index_name on table_name (key_field)

   Example :
   Query : Find the name of employee ‘Smith’

   Sql : Select Ename from emp where ename=‘smith’;

   Creating the index with key field is ename :

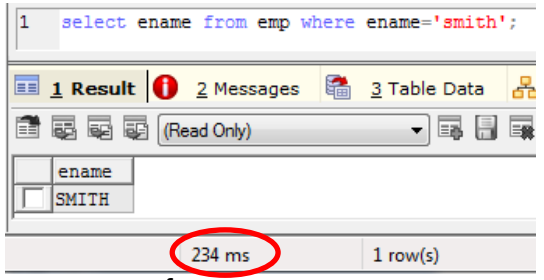
   Create index idx_ename on emp (ename);

   Furthermore, the above sql is accessed again.

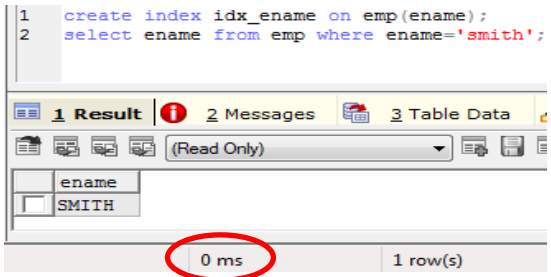
   Sql : Select Ename from emp where ename=‘smith’;

The execution result of the sql after creating the index can increase the speed. access time can be seen in the bottom line :

Before :



after :



Query Optimization

2. **Determining the Right Data Type**
   For example using data type of char and varchar. Both types of data can accommodate characters. The **char** provides a fixed storage size (fixed-length), while the **varchar** provides the storage size in accordance with the contents of the data (variable-length).

   The **char** data type is used for fields that have a consistent data length, such as postal code. While **varchar** is used for fields that have a variable length of data.

3. **Avoiding Fields of Null Value**
   The null value is sometimes confused in the programmer's interpretation and may result in programming logic errors. In addition, null-valued fields consume more memory thereby increasing the load on the access query.

Query Optimization

4. **Avoiding the SELECT * operations**
   If you only want to access a particular field, it is better to write the field you want to access only, so the query becomes SELECT field1, field2, field3, fieldn. It will impact the reduction of network traffic load and locking the table, especially if the table has multiple fields .

5. **Restrict the ORDER BY operation**
   The use of the ORDER BY operations that serves to sort the data. It has consequences to increase the query load, because it will add one more process, namely the sorting process.

6. Using ***Subquery*** *or* ***JOIN***
   It is recommended to prioritize the use of JOIN operations, because in the general case it will result in faster performance. But if in certain cases can only be solved with subquery, then use subquery.

Query Optimization

7. Using ***WHERE in SELECT***
WHERE clause is used as a condition to filter records so that it can minimize the load of data traffic, because not all records are accessed.

8. **Consider the operator's access speed**
The fastest processed operator sequence is =, >, >=, <, <=, Like, <>.

Example :
use the logic expression of WHERE 1 = 1 rather than WHERE 0 <> 1 for a logical expression to be true.

Query Optimization

the following steps are the query optimization process that only consider the most minimum cost (the most minimum number of tuples and columns) :

1. Determining Query Plan
   Query plan is written by using relational algebra. Creation of a query plan is based on a query tree (expression tree) that contains a sequence of relational operations.

2. Transform the query plan by using equivalence rules
   Rules equivalence is used to transform the query plan into the optimize query :
   a. Creating the query tree (expression tree)
   b. Determining the relational algebra
   c. Evaluating the output and the number of tuples and columns involved
   d. Repeating step a until the optimize query is obtained

Query Optimization

Equivalence Rules

An equivalence rule says that expressions of two forms are equivalent. We can replace an expression of the first form by an expression of the second form or vise versa-- since the two expressions generate the same result on any valid database.

1. Conjunctive selection operations can be deconstructed into a sequence of individual selections

$$\sigma_{\theta_1 \wedge \theta_2}(E) = \sigma_{\theta_1}(\sigma_{\theta_2}(E))$$

2. Selection operations are **commutative**

$$\sigma_{\theta_1}(\sigma_{\theta_2}(E)) = \sigma_{\theta_2}(\sigma_{\theta_1}(E))$$

3. Only the final operations in a sequence of projection operations are needed; the others can be omitted

$$\Pi_{L_1}(\Pi_{L_2}(\ldots(\Pi_{L_n}(E))\ldots)) = \Pi_{L_1}(E)$$

Query Optimization

4. Selections can be combined with Cartesian products and theta joins
   a. $\sigma_\theta(E_1 \times E_2) = E_1 \bowtie_\theta E_2$

      This expression is just the definition of the theta join.

   b. $\sigma_{\theta_1}(E_1 \bowtie_{\theta_2} E_2) = E_1 \bowtie_{\theta_1 \wedge \theta_2} E_2$

5. Theta-join operations are **commutative**

$$E_1 \bowtie_\theta E_2 = E_2 \bowtie_\theta E_1$$

6. Natural-join operations are **associative**

$$(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$$

7. The set operations union and intersection are **commutative**.

$$E_1 \cup E_2 = E_2 \cup E_1$$

$$E_1 \cap E_2 = E_2 \cap E_1$$

Set difference is not commutative.

Query Optimization

8. Set union and intersection are **associative**.

$$(E_1 \cup E_2) \cup E_3 = E_1 \cup (E_2 \cup E_3)$$

$$(E_1 \cap E_2) \cap E_3 = E_1 \cap (E_2 \cap E_3)$$

9. The selection operation distributes over the union, intersection, and set difference operations.

$$\sigma_P(E_1 - E_2) = \sigma_P(E_1) - \sigma_P(E_2)$$

9. The projection operation distributes over the union operation.

$$\Pi_L(E_1 \cup E_2) = (\Pi_L(E_1)) \cup (\Pi_L(E_2))$$

Query Optimization

Example :

Teaches

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

Instructor

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Course

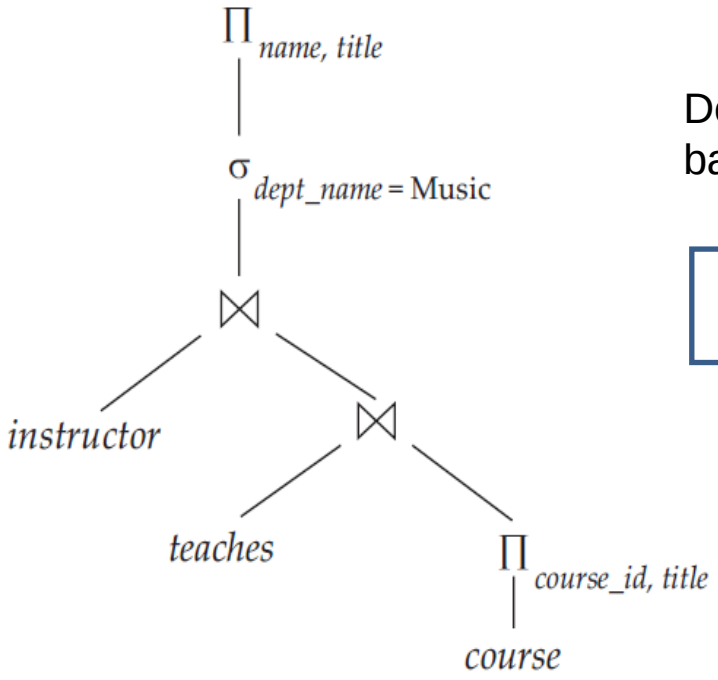
course_id	*title*	*dept_name*	*credits*
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

Query :

“Find the names of all instructors in the Music department together with the course title of all the courses that the instructors teach.”

Query Optimization

Step 1 :
Prepare the query tree (expression tree) as the basis for determining the query plan



Initial expression tree

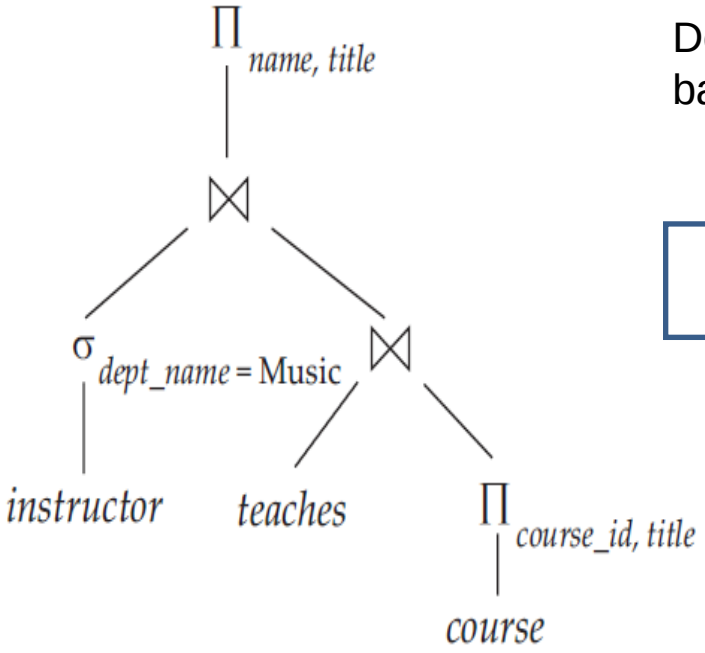
Determining the query plan by using relational algebra based on expression tree

Relational Algebra :

$\pi_{name,\ title}(\sigma_{dept\_name = \text{“Music”}}(instructor \bowtie (teaches \bowtie \pi_{course\_id,title}(course))))$

Query Optimization

Step 2 :
Transform the query plan into the optimize query by using Equivalence Rules.
In this cases, the query plan can be optimized using equivalence rules number 6 (Natural-join operations are **associative**).



Transformed expression tree

Determining the query plan by using relational algebra based on expression tree

Relational Algebra :

π *name, title* ((σ *dept_name =“Music”* *(instructor)*) ⋈ (*teaches* ⋈ $_{\pi \text{ course\_id,title}}$ *(course)*))

The next step :
output and process analysis of query plan and optimize query.

Query Optimization

Output and process analysis of the query plan and the optimize query

Query plan :

$\pi_{name,\ title}(\sigma_{dept\_name = \text{“Music”}}(instructor \bowtie (teaches \bowtie \pi_{course\_id,title}(course))))$

Output :

$\pi_{course\_id,title}(course)$

course_id	title
BIO-101	Intro to Biology
BIO-301	Genetics
BIO-399	Computational Biology
CS-101	Intro to Computer Science
CS-190	Game Design
CS-315	Robotics
CS-319	Image Processing
CS-347	Database System Concepts
EE-181	Intro to Digital System
FIN-201	Invesment Banking
HIS-351	World History
MU-199	Music Video Production
PHY-101	Physical Principles

$teaches \bowtie \pi_{course\_id,title}(course)$

course_id	ID	sec_id	semester	year	title
CS-101	10101	1	Fall	2009	Intro to Computer Science
CS-101	45565	1	Spring	2010	Intro to Computer Science
CS-190	83821	1	Spring	2009	Game Design
CS-190	83821	2	Spring	2009	Game Design
CS-315	10101	1	Spring	2010	Robotics
CS-319	45565	1	Spring	2010	Image Processing
CS-319	83821	2	Spring	2010	Image Processing
CS-347	10101	1	Fall	2009	Database System Concepts
EE-181	98345	1	Spring	2009	Intro to Digital System
FIN-201	12121	1	Spring	2010	Invesment Banking
HIS-351	32343	1	Spring	2010	World History
MU-199	15151	1	Spring	2010	Music Video Production
PHY-101	22222	1	Fall	2009	Physical Principles

$(instructor \bowtie (teaches \bowtie \pi_{course\_id,title}(course)))$

ID	name	dept_name	salary	course_id	sec_id	semester	year	title
76766	Crick	Biology	72000.00	BIO-101	1	Summer	2009	Intro to Biology
76766	Crick	Biology	72000.00	BIO-301	1	Summer	2010	Genetics
10101	Srinivasan	Comp. Sci.	65000.00	CS-101	1	Fall	2009	Intro to Computer Science
45565	Katz	Comp. Sci.	75000.00	CS-101	1	Spring	2010	Intro to Computer Science
83821	Brandt	Comp. Sci.	92000.00	CS-190	1	Spring	2009	Game Design
83821	Brandt	Comp. Sci.	92000.00	CS-190	2	Spring	2009	Game Design
10101	Srinivasan	Comp. Sci.	65000.00	CS-315	1	Spring	2010	Robotics
45565	Katz	Comp. Sci.	75000.00	CS-319	1	Spring	2010	Image Processing
83821	Brandt	Comp. Sci.	92000.00	CS-319	2	Spring	2010	Image Processing
10101	Srinivasan	Comp. Sci.	65000.00	CS-347	1	Fall	2009	Database System Concepts
98345	Kim	Elec. Eng.	80000.00	EE-181	1	Spring	2009	Intro to Digital System
12121	Wu	Finance	90000.00	FIN-201	1	Spring	2010	Invesment Banking
32343	El Said	History	60000.00	HIS-351	1	Spring	2010	World History
15151	Mozart	Music	40000.00	MU-199	1	Spring	2010	Music Video Production
22222	Einstein	Physics	95000.00	PHY-101	1	Fall	2009	Physical Principles

$\pi_{name,\ title}(\sigma_{dept\_name = \text{“Music”}}(instructor \bowtie (teaches \bowtie \pi_{course\_id,title}(course))))$

name	title
Mozart	Music Video Production

Query Optimization

Output and process analysis of the query plan and the optimize query

The Optimize Query :

π *name, title* ((σ *dept_name =“Music” (instructor)*) ⋈ (*teaches* ⋈ $\pi_{course\_id,title}$ *(course)*))

Output :

$\pi_{course\_id,title}$ *(course)*

course_id	title
BIO-101	Intro to Biology
BIO-301	Genetics
BIO-399	Computational Biology
CS-101	Intro to Computer Science
CS-190	Game Design
CS-315	Robotics
CS-319	Image Processing
CS-347	Database System Concepts
EE-181	Intro to Digital System
FIN-201	Invesment Banking
HIS-351	World History
MU-199	Music Video Production
PHY-101	Physical Principles

σ *dept_name =“Music” (instructor)*

	ID	name	dept_name	salary
☐	15151	Mozart	Music	40000.00

((σ *dept_name =“Music” (instructor)*) ⋈ (*teaches* ⋈ $\pi_{course\_id,title}$ *(course)*))

ID	name	dept_name	salary	course_id	sec_id	semester	year	title
15151	Mozart	Music	40000.00	MU-199	1	Spring	2010	Music Video Production

teaches ⋈ $\pi_{course\_id,title}$ *(course)*

course_id	ID	sec_id	semester	year	title
CS-101	10101	1	Fall	2009	Intro to Computer Science
CS-101	45565	1	Spring	2010	Intro to Computer Science
CS-190	83821	1	Spring	2009	Game Design
CS-190	83821	2	Spring	2009	Game Design
CS-315	10101	1	Spring	2010	Robotics
CS-319	45565	1	Spring	2010	Image Processing
CS-319	83821	2	Spring	2010	Image Processing
CS-347	10101	1	Fall	2009	Database System Concepts
EE-181	98345	1	Spring	2009	Intro to Digital System
FIN-201	12121	1	Spring	2010	Invesment Banking
HIS-351	32343	1	Spring	2010	World History
MU-199	15151	1	Spring	2010	Music Video Production
PHY-101	22222	1	Fall	2009	Physical Principles

π*name, title* ((σ *dept_name =“Music” (instructor)*) ⋈ (*teaches* ⋈ $\pi_{course\_id,title}$ *(course)*))

name	title
Mozart	Music Video Production

Query Optimization

analysis results :

Output analysis :

The query plan and the optimize query produces the same output

Process analysis :

The natural join operation of the optimize query is producing a smaller number of tuples than natural join operation of the query plan.

Query plan

$(instructor \bowtie (teaches \bowtie_{\pi course\_id,title} (course)))$

ID	name	dept_name	salary	course_id	sec_id	semester	year	title
76766	Crick	Biology	72000.00	BIO-101	1	Summer	2009	Intro to Biology
76766	Crick	Biology	72000.00	BIO-301	1	Summer	2010	Genetics
10101	Srinivasan	Comp. Sci.	65000.00	CS-101	1	Fall	2009	Intro to Computer Science
45565	Katz	Comp. Sci.	75000.00	CS-101	1	Spring	2010	Intro to Computer Science
83821	Brandt	Comp. Sci.	92000.00	CS-190	1	Spring	2009	Game Design
83821	Brandt	Comp. Sci.	92000.00	CS-190	2	Spring	2009	Game Design
10101	Srinivasan	Comp. Sci.	65000.00	CS-315	1	Spring	2010	Robotics
45565	Katz	Comp. Sci.	75000.00	CS-319	1	Spring	2010	Image Processing
83821	Brandt	Comp. Sci.	92000.00	CS-319	2	Spring	2010	Image Processing
10101	Srinivasan	Comp. Sci.	65000.00	CS-347	1	Fall	2009	Database System Concepts
98345	Kim	Elec. Eng.	80000.00	EE-181	1	Spring	2009	Intro to Digital System
12121	Wu	Finance	90000.00	FIN-201	1	Spring	2010	Invesment Banking
32343	El Said	History	60000.00	HIS-351	1	Spring	2010	World History
15151	Mozart	Music	40000.00	MU-199	1	Spring	2010	Music Video Production
22222	Einstein	Physics	95000.00	PHY-101	1	Fall	2009	Physical Principles

Optimize query

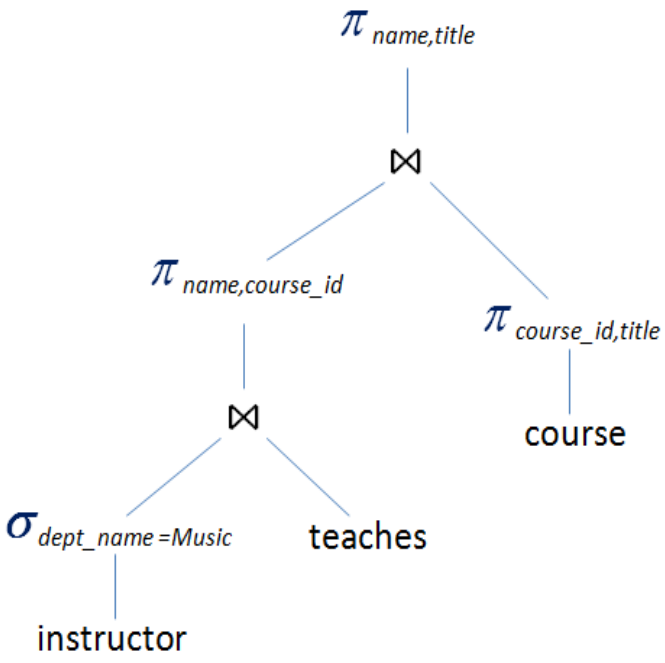
$((\sigma dept\_name = \text{“Music”} (instructor)) \bowtie (teaches \bowtie_{\pi course\_id,title} (course)))$

ID	name	dept_name	salary	course_id	sec_id	semester	year	title
15151	Mozart	Music	40000.00	MU-199	1	Spring	2010	Music Video Production

Result of the query above needs the optimize process by reducing the columns are not required (dept_name, salary, etc).

Query Optimization

The next step :
the query optimization process is used to reduce the columns are not required. Therefore, the expression tree is formulated as follows :



Transformed expression tree

Determining the optimized query by using relational algebra based on expression tree

Relational Algebra :

$\pi_{name,\ title}((\pi_{name,course\_id}((\sigma_{dept\_name = \text{“Music”}}(instructor)) \bowtie teaches) \bowtie \pi_{course\_id,title}(course))$

The next step :
output and process analysis of the previous optimized query and the optimize query.

Query Optimization

Output and process analysis of the optimized query

π name, title ((π name,course_id((σ dept_name ="Music" (instructor)) ⋈ teaches) ⋈ $_{\pi course\_id,title}$ (course))

Output :

$\pi_{course\_id,title}$ (course)

course_id	title
BIO-101	Intro to Biology
BIO-301	Genetics
BIO-399	Computational Biology
CS-101	Intro to Computer Science
CS-190	Game Design
CS-315	Robotics
CS-319	Image Processing
CS-347	Database System Concepts
EE-181	Intro to Digital System
FIN-201	Invesment Banking
HIS-351	World History
MU-199	Music Video Production
PHY-101	Physical Principles

σ dept_name ="Music" (instructor)

	ID	name	dept_name	salary
☐	15151	Mozart	Music	40000.00

π name,course_id((σ dept_name ="Music" (instructor)) ⋈ teaches)

name	course_id
Mozart	MU-199

((π name,course_id((σ dept_name ="Music" (instructor)) ⋈ teaches) ⋈ $_{\pi course\_id,title}$ (course))

course_id	name	title
MU-199	Mozart	Music Video Production

π name, title ((π name,course_id((σ dept_name ="Music" (instructor)) ⋈ teaches) ⋈ $_{\pi course\_id,title}$ (course))

name	title
Mozart	Music Video Production

Query Optimization

analysis results :

Output analysis :

the previous optimized query and the optimized query produces the same output

Process analysis :

The natural join operation of the optimized query is producing a smaller number of columns than natural join operation of the previous optimized query.

The previous optimized query

$((\sigma_{dept\_name = \text{“Music”}} (instructor)) \bowtie (teaches \bowtie \pi_{course\_id,title} (course)))$

ID	name	dept_name	salary	course_id	sec_id	semester	year	title
15151	Mozart	Music	40000.00	MU-199	1	Spring	2010	Music Video Production

the columns are not required

Result of the optimized query

$((\pi_{name,course\_id}((\sigma_{dept\_name = \text{“Music”}} (instructor)) \bowtie teaches) \bowtie \pi_{course\_id,title} (course))$

course_id	name	title
MU-199	Mozart	Music Video Production

the columns are not required

Query Optimization

Exercise (Using the schema diagram of university database) :

Write the most optimal query by using the query optimization process steps:

1. Realtional Algebra :

   π *name* (σ *title =“Game Design”* (*Student* ⋈ *takes* ⋈ course))

2. Find the names of all instructors in the Music department together with the course title of all the courses that the instructors teach in 2009.
3. Find the names of all students and the course title in the Comp. Sci. department.
4. Find the names of all students, the course title and the name of building in the Comp. Sci. Department.