



CMU SCS

**Carnegie Mellon Univ.
Dept. of Computer Science
15-415 - Database Applications**

C. Faloutsos
Lecture#7 (cont'd): *Rel. model - SQL part3*



CMU SCS

General Overview - rel. model

- Formal query languages
 - rel algebra and calculi
- Commercial query languages
 - SQL
 - QBE, (QUEL)

Faloutsos CMU SCS 15-415 #2



CMU SCS

Overview - detailed - SQL

- DML
 - select, from, where, renaming, ordering,
 - aggregate functions, nested subqueries
 - insertion, deletion, update
- other parts: DDL, authorization, triggers
- embedded SQL

Faloutsos CMU SCS 15-415 #3

 CMU SCS

Reminder: our Mini-U db

STUDENT		
Ssn	Name	Address
123	smith	main str
234	jones	forbes ave

CLASS		
c-id	c-name	units
15-413	s.e.	2
15-412	o.s.	2

TAKES		
SSN	c-id	grade
123	15-413	A
234	15-413	B

Faloutsos CMU SCS 15-415 #4

 CMU SCS

DML - insertions etc

insert into student
values ("123", "smith", "main")

insert into student(ssn, name, address)
values ("123", "smith", "main")

Faloutsos CMU SCS 15-415 #5

 CMU SCS

DML - insertions etc

bulk insertion: how to insert, say, a table of
 'foreign-student's, in bulk?

Faloutsos CMU SCS 15-415 #6



CMU SCS

DML - insertions etc

bulk insertion:

```
insert into student  
select ssn, name, address  
from foreign-student
```

Faloutsos CMU SCS 15-415 #7



CMU SCS

DML - deletion etc

delete the record of 'smith'

Faloutsos CMU SCS 15-415 #8



CMU SCS

DML - deletion etc

delete the record of 'smith':

```
delete from student  
where name='smith'
```

(careful - it deletes ALL the 'smith's!)

Faloutsos CMU SCS 15-415 #9



CMU SCS

DML - update etc

record the grade 'A' for ssn=123 and course 15-415

update takes
set grade="A"
where ssn="123" and c-id="15-415"

(will set to "A" ALL such records)

Faloutsos CMU SCS 15-415 #10



CMU SCS

DML - view update

consider the db-takes view:
create view db-takes as
(**select** * **from** takes **where** c-id="15-415")

view updates are tricky - typically, we can only
update views that have no joins, nor aggregates
even so, consider changing a c-id to 15-222...

Faloutsos CMU SCS 15-415 #11



CMU SCS

DML - joins

so far: 'INNER' joins, eg:

select ssn, c-name
from takes, class
where takes.c-id = class.c-id

Faloutsos CMU SCS 15-415 #12

 CMU SCS

DML - joins

Equivalently:

```
select ssn, c-name
from takes join class on takes.c-id = class.c-id
```

FaloutsosCMU SCS 15-415#13

 CMU SCS

Joins

```
select [column list]
from table_name
[inner | {left | right | full} outer ] join
table_name
on qualification_list
where...
```

FaloutsosCMU SCS 15-415#14

 CMU SCS

Reminder: our Mini-U db

STUDENT		
Ssn	Name	Address
123	smith	main str
234	jones	forbes ave

CLASS		
c-id	c-name	units
15-413	s.e.	2
15-412	o.s.	2

TAKES		
SSN	c-id	grade
123	15-413	A
234	15-413	B

FaloutsosCMU SCS 15-415#15

CMU SCS

Inner join

TAKES		
SSN	c-id	grade
123	15-413	A
234	15-413	B

CLASS		
c-id	c-name	units
15-413	s.e.	2
15-412	o.s.	2

SSN	c-name
123	s.e
234	s.e

o.s.: gone!

Faloutsos CMU SCS 15-415 #16

CMU SCS

Outer join

TAKES		
SSN	c-id	grade
123	15-413	A
234	15-413	B

CLASS		
c-id	c-name	units
15-413	s.e.	2
15-412	o.s.	2

SSN	c-name
123	s.e
234	s.e.
null	o.s.

←

Faloutsos CMU SCS 15-415 #17

CMU SCS

Outer join

select ssn, c-name
from takes **right outer join** class **on** takes.c-id=class.c-id

SSN	c-name
123	s.e
234	s.e.
null	o.s.

←

Faloutsos CMU SCS 15-415 #18



CMU SCS

Outer join

- left outer join
- right outer join
- full outer join
- natural join

Faloutsos CMU SCS 15-415 #19



CMU SCS

Null Values

- **null** -> unknown, or inapplicable, (or ...)
- Complications:
 - 3-valued logic (true, false and *unknown*).
 - **null = null** : false!!

Faloutsos CMU SCS 15-415 #20



CMU SCS

Overview - detailed - SQL

- DML
 - select, from, where, renaming, ordering,
 - aggregate functions, nested subqueries
 - insertion, deletion, update
- other parts: **DDL**, authorization, triggers
- embedded SQL

Faloutsos CMU SCS 15-415 #21



CMU SCS

Data Definition Language

```
create table student  
(ssn char(9) not null,  
  name char(30),  
  address char(50),  
  primary key (ssn) )
```

Faloutsos CMU SCS 15-415 #22



CMU SCS

Data Definition Language

```
create table r( A1 D1, ..., An Dn,  
  integrity-constraint1,  
  ...  
  integrity-constraint-n)
```

Faloutsos CMU SCS 15-415 #23



CMU SCS

Data Definition Language

Domains:

- **char**(n), **varchar**(n)
- **int**, **numeric**(p,d), **real**, **double precision**
- **float**, **smallint**
- **date**, **time**

Faloutsos CMU SCS 15-415 #24



CMU SCS

Data Definition Language

delete a table: difference between
drop table student

delete from student

Faloutsos CMU SCS 15-415 #25



CMU SCS

Data Definition Language

modify a table:
alter table student **drop** address

alter table student **add** major char(10)

Faloutsos CMU SCS 15-415 #26



CMU SCS

Data Definition Language

integrity constraints:

- **primary key**
- **foreign key**
- **check(P)**

Faloutsos CMU SCS 15-415 #27



Data Definition Language

create table takes
 (ssn **char**(9) **not null**,
 c-id **char**(5) **not null**,
 grade **char**(1),
primary key (ssn, c-id),
check grade in (“A”, “B”, “C”, “D”, “F”))

Faloutsos CMU SCS 15-415 #28



Referential Integrity constraints

‘foreign keys’ - eg:
create table takes(
 ssn **char**(9) **not null**,
 c-id **char**(5) **not null**,
 grade **integer**,
primary key(ssn, c-id),
foreign key ssn **references** student,
foreign key c-id **references** class)

Faloutsos CMU SCS 15-415 #29



Referential Integrity constraints

...
foreign key ssn **references** student,
foreign key c-id **references** class)

Effect:

- expects that ssn to exist in ‘student’ table
- blocks ops that violate that - how??
 - insertion?
 - deletion/update?

Faloutsos CMU SCS 15-415 #30



CMU SCS

Referential Integrity constraints

...

foreign key ssn **references** student
on delete cascade
on update cascade,

...

- -> eliminate all student enrollments
- other options (set to null, to default etc)

Faloutsos CMU SCS 15-415 #31



CMU SCS

Overview - detailed - SQL

- DML
 - select, from, where, renaming, ordering,
 - aggregate functions, nested subqueries
 - insertion, deletion, update
- other parts: DDL, authorization, **triggers**
- embedded SQL

Faloutsos CMU SCS 15-415 #32



CMU SCS

Weapons for IC:

- assertions
 - **create assertion** <assertion-name> **check** <predicate>
- triggers (~ assertions with 'teeth')
 - on operation, if condition, then action

Faloutsos CMU SCS 15-415 #33



CMU SCS

Triggers - example

define trigger zerograde **on update** takes
(**if new** takes.grade < 0
 then takes.grade = 0)

Faloutsos CMU SCS 15-415 #34



CMU SCS

Triggers - discussion

- more complicated: “managers have higher salaries than their subordinates” - a trigger can automatically boost mgrs salaries
- triggers: tricky (infinite loops...)

Faloutsos CMU SCS 15-415 #35



CMU SCS

Overview - detailed - SQL

- DML
 - select, from, where, renaming, ordering,
 - aggregate functions, nested subqueries
 - insertion, deletion, update
- other parts: DDL, **authorization**, triggers
- embedded SQL

Faloutsos CMU SCS 15-415 #36



CMU SCS

Authorization

- **grant** <priv.-list> **on** <table-name> **to** <user-list>
- privileges for tuples: read / insert / delete / update
- privileges for tables: create, drop, index

Faloutsos CMU SCS 15-415 #37



CMU SCS

Authorization – cont'd

- variations:
 - **with grant option**
 - **revoke** <priv.-list> **on** <t-name> **from** <user_ids>

Faloutsos CMU SCS 15-415 #38



CMU SCS

Overview - detailed - SQL

- DML
 - select, from, where, renaming, ordering,
 - aggregate functions, nested subqueries
 - insertion, deletion, update
- other parts: DDL, authorization, triggers
- **embedded SQL**; application development

Faloutsos CMU SCS 15-415 #39



CMU SCS

Embedded SQL

from within a 'host' language (eg., 'C', 'VB')

```
EXEC SQL <emb. SQL stmt> END-EXEC
```

Q: why do we need embedded SQL??

Faloutsos CMU SCS 15-415 #40



CMU SCS

Embedded SQL

SQL returns sets; host language expects a tuple - impedance mismatch!

solution: 'cursor', ie., a 'pointer' over the set of tuples.

example:

Faloutsos CMU SCS 15-415 #41



CMU SCS

Embedded SQL

```
main(){  
...  
EXEC SQL  
  declare c cursor for  
  select * from student  
END-EXEC  
...  
}
```

Faloutsos CMU SCS 15-415 #42



CMU SCS

Embedded SQL - ctn'd

```
...  
EXEC SQL open c END-EXEC  
...  
while( !sqlerror ){  
    EXEC SQL fetch c into :cssn, :cname, :cad  
    END-EXEC  
    fprintf( ... , cssn, cname, cad);  
}
```

Faloutsos CMU SCS 15-415 #43



CMU SCS

Embedded SQL - ctn'd

```
...  
EXEC SQL close c END-EXEC  
...  
} /* end main() */
```

Faloutsos CMU SCS 15-415 #44



CMU SCS

Dynamic SQL

```
main(){ /* set all grades to user's input */  
...  
char *sqlcmd=" update takes set grade = ?";  
EXEC SQL prepare dynsql from :sqlcmd ;  
char inputgrade[5]="a";  
EXEC SQL execute dynsql using :inputgrade;  
...  
} /* end main() */
```

Faloutsos CMU SCS 15-415 #45



CMU SCS

Overview - detailed - SQL

- DML
 - select, from, where, renaming, ordering,
 - aggregate functions, nested subqueries
 - insertion, deletion, update
- other parts: DDL, authorization, triggers
- embedded SQL; **application development**

Faloutsos CMU SCS 15-415 #46



CMU SCS

Overview

- concepts of SQL programs
- walkthrough of Create.java
- walkthrough of showAll.java

Faloutsos CMU SCS 15-415 #47

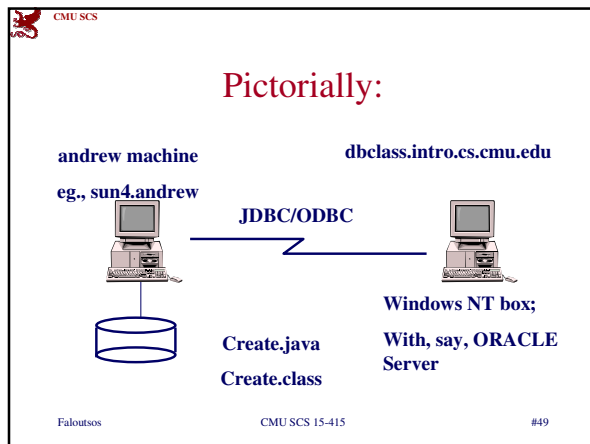


CMU SCS

Outline of an SQL application

- establish connection with db server
- authenticate (user/password)
- execute SQL statement(s)
- process results
- close connection

Faloutsos CMU SCS 15-415 #48



CMU SCS

Create.java

- Purpose: to load the parent-child table

legend:

⇒ interesting observation

➡ very important point

Faloutsos CMU SCS 15-415 #50

CMU SCS

Walk-through Create.java

```
import java.io.*;
import java.util.*;
import java.sql.*;

public class Create {

    static final String DbURL =
        "jdbc:oracle:thin:@dbclass.intro.cs.cmu.edu:1521:dbintro";
    //Oracle server at cs.cmu
    static final String OraDriver = "oracle.jdbc.driver.OracleDriver";
    //Oracle driver

    static final String User = "your-andrew-id";
    static final String Passwd = "your-oracle-password";
}
```

Faloutsos CMU SCS 15-415 #51

CMU SCS

Walk-through Create.java

```
static final String Passwd = "your-oracle-password";  
static final String fileName="PC.txt";  
//file name for text data  
public static void main(String[] args) {  
    Connection con = null;  
  
    try {  
        // Load the Oracle Driver  
        Class.forName(OraDriver);  
  
        // Get a Connection to the database  
        con = DriverManager.getConnection(DbURL, User,  
        Passwd);  
  
        // Create a Statement object  
        Statement stmt = con.createStatement();
```

Faloutsos CMU SCS 15-415 #52

CMU SCS

Walk-through Create.java

```
// Create a table named as PC (varchar2(10), varchar2(10));  
String sqlSt =  
    "CREATE TABLE PC (parent varchar2(10), child varchar2(10))";  
stmt.executeQuery(sqlSt);
```

Faloutsos CMU SCS 15-415 #53

CMU SCS

Walk-through Create.java

rest of program:

- read input file
- insert one tuple at a time
- close connection

Faloutsos CMU SCS 15-415 #54



CMU SCS

Walk-through Create.java

```
while ((line = in.readLine()) != null) {  
    // read in the names into 'parent' and 'child'  
  
    // Execute a SQL - insert statement  
    sqlSt = "INSERT INTO PC (parent, child) VALUES ('"  
        + parent + "', '" + child + "')";  
    System.out.println("=== " + (i++) + "===>" + sqlSt);  
    stmt.executeQuery(sqlSt);  
}  
in.close();  
con.commit();  
}
```

Faloutsos CMU SCS 15-415 #55



CMU SCS

Overview

- concepts of SQL programs
- walkthrough of Create.java
- walkthrough of showAll.java

Faloutsos CMU SCS 15-415 #56



CMU SCS

Walk-through showAll.java

- purpose: print all (parent, child) pairs

Faloutsos CMU SCS 15-415 #57

CMU SCS

Walk-through showAll.java

→ `// after opening the connection ...
String sqlSt = "SELECT * FROM PC";`

Faloutsos CMU SCS 15-415 #58

CMU SCS

Walk-through showAll.java

→ `ResultSet rs = stmt.executeQuery(sqlSt);
while (rs.next()) {
 System.out.println(rs.getString("parent") +
 ";" + rs.getString("child"));
}`

Faloutsos CMU SCS 15-415 #59

CMU SCS

Conclusions

Outline of an SQL application:

- establish connection with db server
- authenticate (user/password)
- execute SQL statement(s)
- process results
- close connection

Faloutsos CMU SCS 15-415 #60
