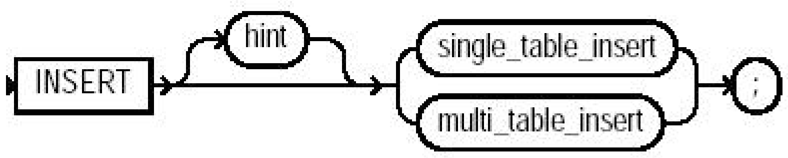
Session 6: SQL DML (CH-4)
CSCI-585, Cyrus Shahabi

(Some example queries, but you need to go read the book and do more exercise on your own, not everything is covered!)

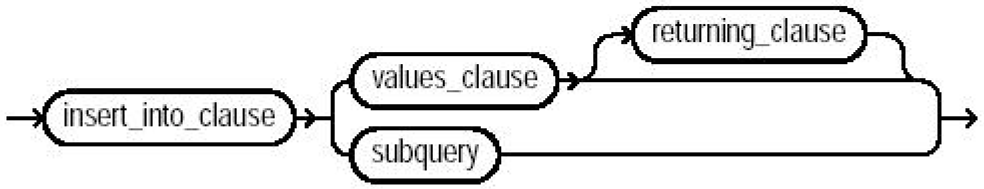
Emp (SS#, name, age, salary, dno)
Dept (dno, dname, floor, mgrSS#)

- SQL provides commands to change the state of database: **insert**, **delete**, and **update**.
- Insert has two different syntax:
  1. **insert into** rel-name **values** value list
  2. **insert into** rel-name **select**

insert::=



single_table_insert::=



To illustrate, assume the existence of two relations:
register(sid, sname, paid, course#) and CSCI585(sid,sname).
If Joe and Bob register for csci585 without having paid:

insert into register **values**

(666-66-6666, `Joe', No, 585)
(777-77-7777, `Bob', No, 585)

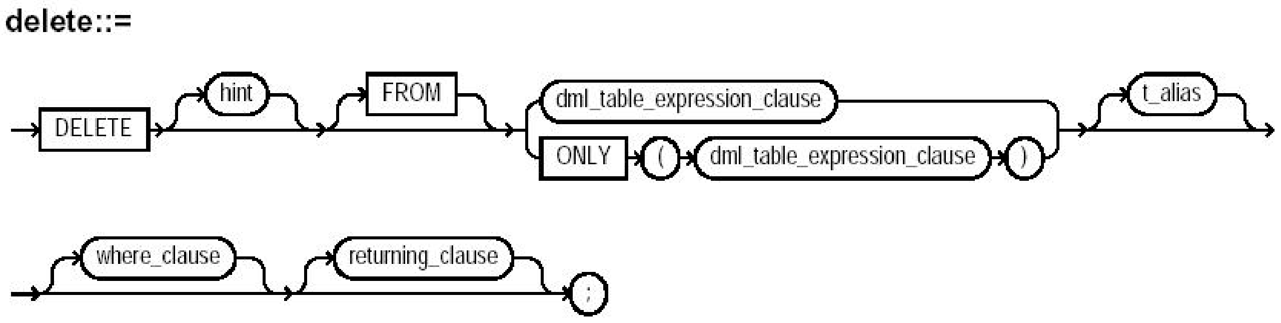
To insert all CSCI585 student into CSCI585 relation who have paid:

insert into CSCI585
select sid, name
from register r
where r.paid = `yes' and r.course#=585

Note that the target list of the select command must confirm to the schema of CSCI585

- Delete has the following syntax:

delete rel-name **where** qualification



Example: Fire all those employees whose salary is less than average.

delete Emp
where salary < (**select** avg(salary)
from Emp)

Problem: Average changes as we delete! Some versions disallow the above types of delete; some enforce the following semantic:

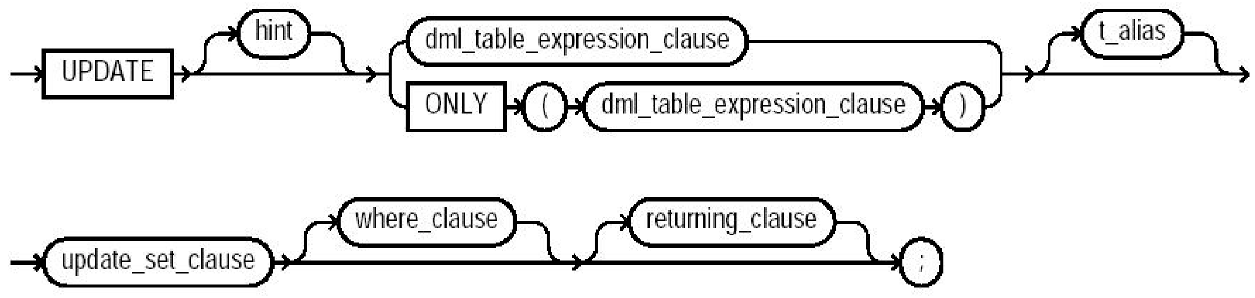
Step1: execute query:

select *
from rel-name
where qualification

Step 2: remove tuples found in Step 1 from rel-name

- **Update** command has the following syntax:
  update rel-name
  set target-list
  where qualification

update::=



- Example: Give a 10% raise to all employees in the toy department.

```
update Emp
set salary = 1.1 * salary
where SS# in (
        select e.SS#
        from Emp e, Dept d
        where e.dno = d.dno and d.dname = `Toy')
```

- What if we wanted to give a 10% raise to all employees who earn less than average (same discussion as delete)?

```
update Emp
set salary = 1.1 * salary
where salary < (select avg(salary)
                from  Emp)
```

- Hence, the semantic of update is as follows:

Step 1: Execute the following two queries:

> **insert into** del-temp
> **select** full-target-list
> **from** rel-name
> **where** qualification
>
> **insert** into app-temp
> **select** extended target list
> **from** rel-name
> **where** qualification
>
> *Extended target list in our example would be:*
> *(SS#, name, age, sal * 1.1, dno).*
> *Full target list in our example would be:*
> *(SS#, name, age, sal, dno).*

Step 2: Remove tuples in del-temp from rel-name
Step 3: Insert tuples in app-temp into rel-name

- **Order by**: To sort the results of a query.

  Example: List all employees in ascending order by age and descending order by salary (default is ascending)

  > **select** SS#, name
  > **from** Emp
  > **order by** age **asc**, salary **desc**

- **Views:** To provide a higher level of abstraction.
  Syntax: **create view** *v* **as** <*query expression*>
  Example: A view of all employees working in toy department

  > **create view** Toy-employee **as**

select SS#, name, salary
from Emp, Dept
where Emp.dno = Dept.dno and dname = ‘Toy’

A view name can appear in any place that a relation name may appear.

- Insertion to views: Use of null values!

- Updates on views: Works for views based on single relation where a candidate key of the base relation is included in the view attributes. Forbidden (ambiguous) on a view which is defined in terms of more than one relation, or on views with grouping and aggregate functions).

  create view AvgDeptSal (dno, dname, AvgSalary) **as**
  select d.dno, d.dname, avg(salary)
  from Emp e, Dept d
  where e.dno = d.dno
  group by dno

- Data definition:

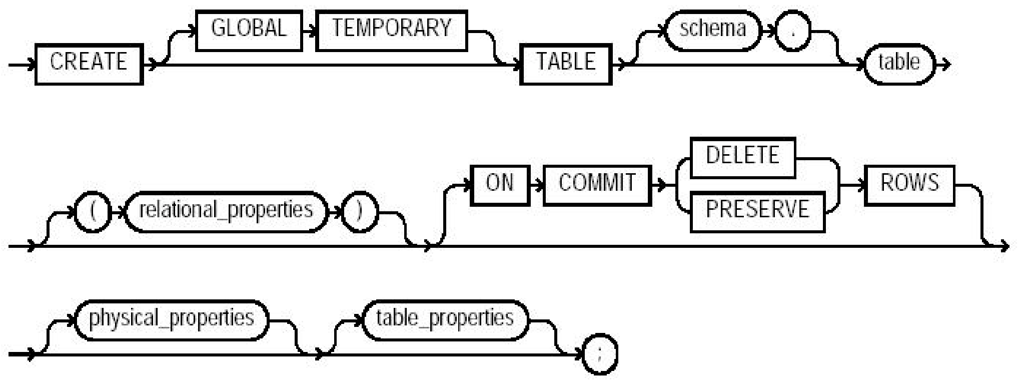
1. **create schema** s

Creates a schema!

2. **create table** r (A_1 D_1, A_2 D_2, …, A_n D_n): create relation or object tables.

A is the attribute name and D is its domain data type.
(Look into the book and Oracle manual for details. Different products have different syntaxes. You can define primary keys and foreign keys here as well.)

relational_table::=



Note: Each of the clauses following the table name is optional for any given relational table. However, for every table you must at least specify either column names and datatypes using the `relational_properties` clause or an `AS` *`subquery`* clause using the `table_properties` clause.

3. **drop table** r

Get rid of the entire relation r

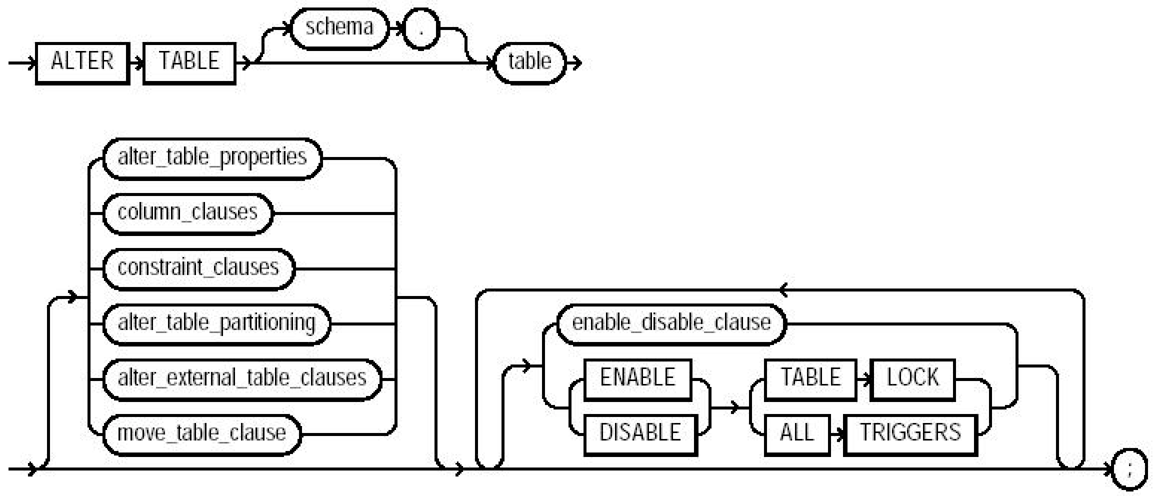
4. **delete** r

Only delete the tuples but keep the relation

5. **alter table** r **add** A D

   Only in some versions (modifies the database schema)

alter_table::=



Note: You must specify some clause after `table`. That is, none of the clauses after `table` are required, but you must specify at least one of them.