Where clause

Operator	Meaning
AND	Returns true if both conditions are true
OR	Returns true if either condition is true
NOT	Returns true if the condition is false

Operator	Becomes
IN	NOT IN
LIKE	NOT LIKE
BETWEEN…AND	NOT BETWEEN …AND
IS NULL	IS NOT NULL

```
SELECT last_name, job_id
    FROM employees
    WHERE job_id NOT IN (‘AD_PRES’, ‘AD_VP’,‘SA_MAN’);
```

Date functions

```
SELECT last_name, SYSDATE + 1 as tomorrow, (SYSDATE - hire_date)/7
    AS weeks_on_the_job
    FROM employees;
```

```
SELECT MONTHS_BETWEEN(sysdate, hire_date)
    FROM employees;
```

```
SELECT hire_date, ADD_MONTHS(hire_date, 6)
    FROM employees;
```

```
SELECT hire_date, NEXT_DAY(hire_date, 'Friday')
    FROM employees;
```

```
SELECT hire_date, LAST_DAY(hire_date)
    FROM employees;
```

```
SELECT hire_date, ROUND(hire_date, 'year')
    FROM employees;
```

TO_CHAR function with dates

```
SELECT TO_CHAR(start_date, ‘DD Month, yyyy’) AS start_date
    FROM job_history;
```

Element	Decription
YYYY	Four digit year
YEAR	Year spelled out – case sensitive
MONTH	Month spelled out – case sensitive
MON	Three letter month – case sensitive
MM	Two digit month
DAY	Day of the week spelled out – case sensitive
DY	Three letter day – case sensitive
DD	Two digit day of the month

TO_CHAR function with numbers

```
SELECT TO_CHAR(salary, '$999,999.99') AS salary
    FROM employees
    WHERE department_id = 20;
```

Element	Decription
9	Represents one digit
0	Forces a zero to be displayed
$	Displays a floating dollar sign
.	Displays a decimal point
,	Displays a comma
L	Floating local currency symbol
G	Local group separator
D	Local decimal symbol

CASE expression

```
SELECT last_name, job_id, salary,
   CASE job_id   WHEN 'AD_ASST'  THEN 1.1 * salary
                 WHEN 'MK_REP'    THEN 1.15 * salary
                 WHEN 'HR_REP'    THEN 1.2 * salary
                 ELSE salary END AS revised_salary
   FROM employees;
```

DECODE

```
SELECT last_name, job_id,
   DECODE (job_id,   'ASD_ASST' ,  1.1 * salary,
                     'MK_REP'   ,  1.15 * salary,
                     'HR_REP'   ,  1.2 * salary,
                  salary) AS revised_salary
   FROM employees;
```

Types of group functions

```
SELECT COUNT(manager_id)
    FROM departments;
```

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

```
SELECT COUNT(DISTINCT manager_id)
    FROM employees;
```

```
SELECT MIN(hire_date), MAX(hire_date)
    FROM employees;
```

```
SELECT SUM(salary), AVG(salary)
    FROM employees;
```

```
SELECT AVG(NVL(commission_pct, 0))
    FROM employees;
```

```
SELECT department_id, job_id, AVG(salary)
    FROM employees
    GROUP BY department_id, job_id
    ORDER BY department_id;
```

```
SELECT department_id, MAX(salary)
        FROM employees
        WHERE commission_pct IS NULL
        GROUP BY  department_id
        HAVING MAX(salary) >= 10000
        ORDER BY 1;
```

Join

- NATURAL JOIN Joins tables based on all columns with the same name and datatype

```
SELECT department_id, department_name, location_id, city
    FROM departments NATURAL JOIN locations;
```

- Joins tables USING the common column name that you specify

```
SELECT employee_id, last_name, department_id, department_name
    FROM employees JOIN departments
    USING (department_id);
```

- More than two related tables can be joined

```
SELECT employee_id, last_name, department_name, city
    FROM employees JOIN departments
    ON employee.department_id = departments.department_id
    JOIN locations
    ON departments.location_id = locations.location_id;
```

- All rows from the table to the left of the JOIN will be selected as well as any matching rows from the right tabe

```
SELECT last_name, department_name
    FROM employees e LEFT OUTER JOIN departments d
    ON e.department_id = d.department_id;
```

Subqueries in the where clause (IN, ANY, ALL)

```
SELECT last_name, job_id, salary
    FROM employees                SA_REP
    WHERE job_id = (SELECT job_id
                                  FROM employees
                                  WHERE employee_id = 152)
    AND salary > (SELECT salary   9000
                          FROM employees
                          WHERE employee_id = 152);
```

```
SELECT last_name, job_id, salary
    FROM employees
    WHERE salary < ANY (SELECT salary 4200 4800 6000 9000
                          FROM employees
                                  WHERE job_id = 'IT_PROG')
    AND job_id != 'IT_PROG';
```

```
SELECT department_id, MIN(salary)
    FROM employees
    GROUP BY department_id              2100
    HAVING MIN(salary) > (SELECT MIN(salary)
                          FROM employees);
```

Union, Intersect, Minus

```
SELECT first_name, last_name
    FROM employees
UNION
SELECT first_name, last_name
    FROM consultants;
```

```
SELECT first_name, last_name
    FROM employees
INTERSECT
SELECT first_name, last_name
    FROM consultants;
```

```
SELECT first_name, last_name
    FROM employees
MINUS
SELECT first_name, last_name
    FROM consultants;
```

```
SELECT location_id, department_name, NULL AS city
    FROM departments
UNION
SELECT location_id, NULL, city
    FROM locations;
```

Create table

Datatype	Description	Size Limit
VARCHAR2(size)	Variable length character data	4000 characters
CHAR(size)	Fixed length character data	2000 characters
NUMBER(p)	Variable length integer data	38 digits
NUMBER(p,s)	Variable length Floating point data	38 digits
DATE	Date/time data	N/A
TIMESTAMP	Date/Time with fractional seconds	N/A

Constraint	Description
Primary Key	Uniquely identifies each row with a non-null value
Foreign Key	Establishes Referential Integrity between the column and a unique identifier, usually in another table, so the values in the two columns match
Unique	The values must be unique in each row
Not Null	The column cannot contain nulls
Check	Specifies which values are allowed

```
CREATE TABLE trades
(
trade_id            NUMBER(9)    CONSTRAINT  trade_id_pk PRIMARY KEY  ,
trade_office        CHAR(2)      NOT NULL                              ,
trade_city      VARCHAR2(25)     NOT NULL                                 ,
trade_timestamp TIMESTAMP    DEFAULT LOCALTIMESTAMP                    ,
trade_settlement DATE            DEFAULT SYSDATE + 3                   ,
trade_amt       NUMBER(11,2)NOT NULL CONSTRAINT sales_amt_ck
                                 CHECK(trade_amt > 0)                  ,
trade_broker    VARCHAR2(15)CONSTRAINT trade_broker_fk
    REFERENCES salespersons(salesperson_id)
);
```

DML – Insert & Delete

```
INSERT INTO departments(department_id, department_name,
                                    manager_id, location_id)
   VALUES(280, 'Operations', 101, 1800);
```

```
INSERT INTO sales_reps(id, lname, salary, comm)
   SELECT employee_id, last_name, salary, commission_pct
       FROM employees
       WHERE job_id = 'SA_REP';
```

```
DELETE FROM employees
   WHERE department_id  =                    70
                    (SELECT department_id
                        FROM departments
                        WHERE department_name = 'Public_Relations');
```

```
DELETE FROM emps_copy;
```

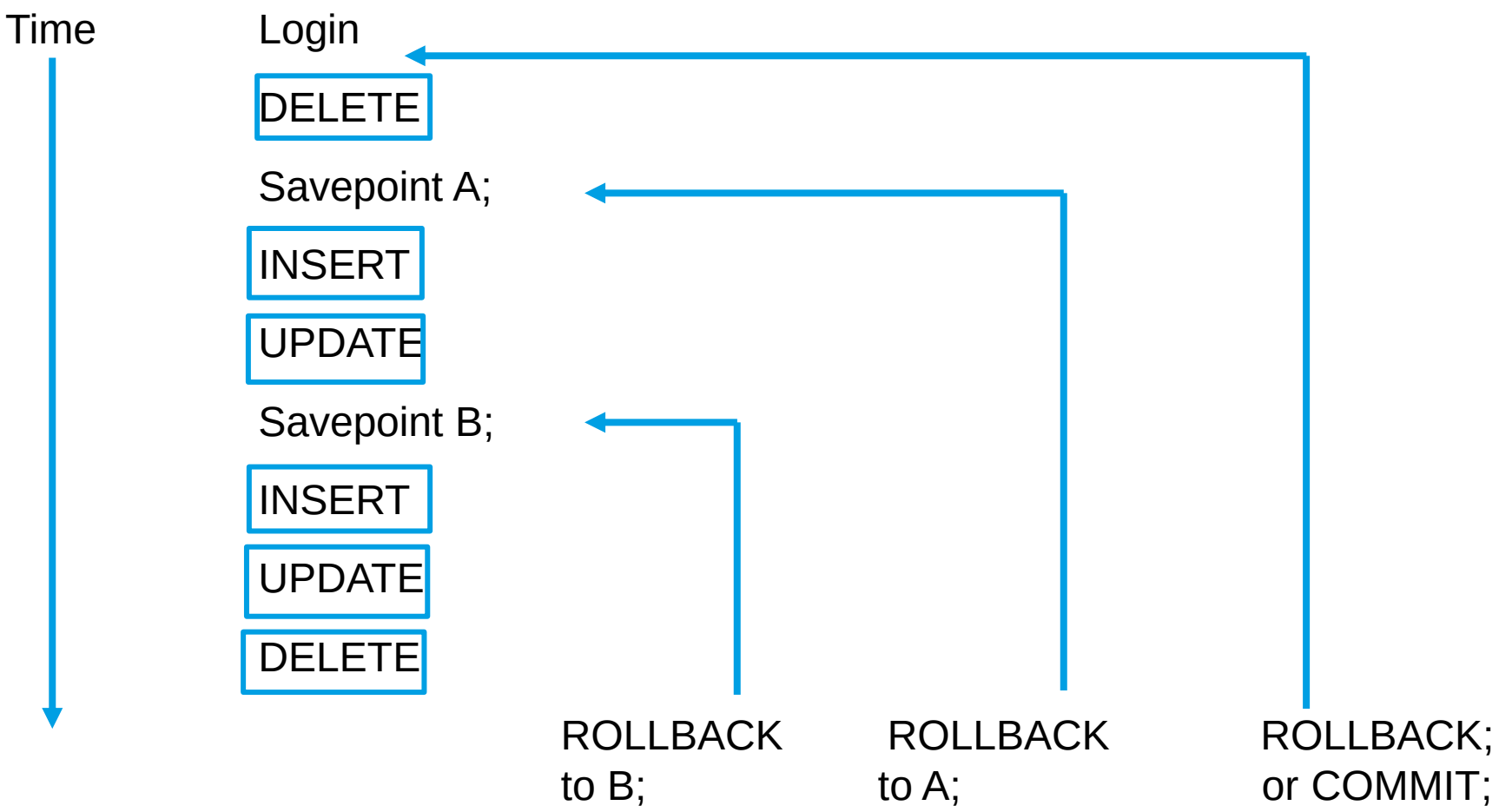
DML - Update

```
UPDATE employees
    SET job_id = 'SA_MAN', salary = 11000
    WHERE employee_id = 150;
```

```
UPDATE employees                          AC_MGR    12000
    SET job_id = (SELECT job_id
                  FROM employees
                  WHERE employee_id = 205),
    salary = (SELECT salary
                  FROM employees
                  WHERE employee_id = 205)

    WHERE employee_id = 113;
```

Transactions



Views

```
CREATE OR REPLACE VIEW yearly_pay_vu
   AS SELECT employee_id, last_name, salary * 12 AS yearly_pay
          FROM employees;
```

```
SELECT *
   FROM yearly_pay_vu;
```

```
CREATE OR REPLACE VIEW emps50_vu (empid, fname, lname, sal, mgr, deptid)
   AS SELECT employee_id, first_name, last_name, salary, manager_id,
          department_id
          FROM employees
          WHERE department_id = 50;
```

```
DROP VIEW yearly_pay_vu;
```

Subqueries in FROM, JOIN

```
SELECT AVG(dept_tot)
     FROM (SELECT manager_id, sum(salary) AS dept_tot
          FROM employees
          GROUP BY manager_id) a;
```

```
SELECT d.department_name, c.city
  FROM departments d JOIN
                (SELECT city, location_id
                     FROM locations join countries
                     USING (country_id)
                     JOIN regions
                     USING (region_id)
                     WHERE region_id IN (3,4) ) c
    ON d.location_id = c.location_id
    ORDER BY 1;
```

Subqueries WITH clause

```
WITH a AS (
          SELECT department_name, sum(salary)  AS totpay
               FROM employees e JOIN departments d
               ON e.department_id = d.department_id
               GROUP BY department_name),
       b AS (
          SELECT avg(totpay) AS depts_avg
               FROM a)
SELECT department_name, tot_pay
     FROM a
     WHERE  totpay > (SELECT depts_avg
                          FROM b)
     ORDER BY 1;
```