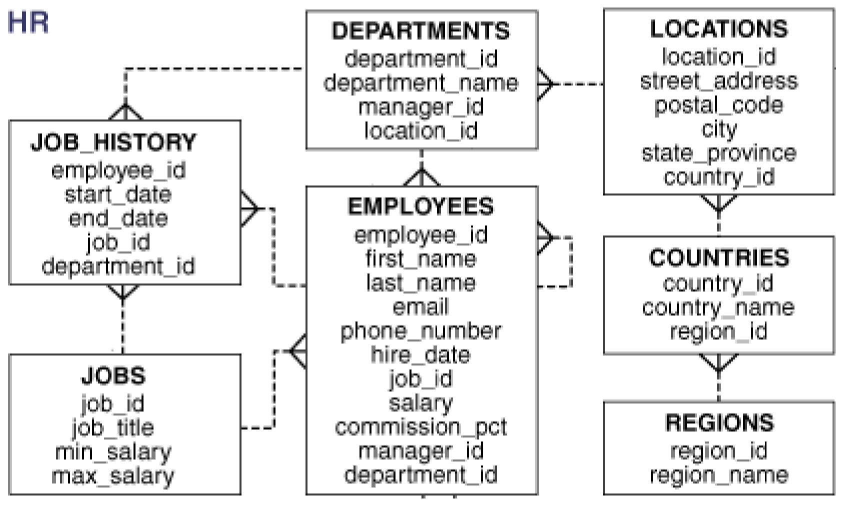
Part 1. Basic SELECT statement

HR Schema structure



COUNTRIES table.

Column name	Null ?	Data type
COUNTRY_ID	NOT NULL	CHAR(2)
COUNTRY_NAME		VARCHAR2(40)
REGION_ID		NUMBER

DEPARTMENTS table.

Column name	Null ?	Data type
DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

EMPLOYEES table.

Column name	Null ?	Data type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(20)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

JOBS table.

Column name	**Null ?**	**Data type**
JOB_ID	NOT NULL	VARCHAR2(10)
JOB_TITLE	NOT NULL	VARCHAR2(35)
MIN_SALARY		NUMBER(6)
MAX_SALARY		NUMBER(6)

JOB_HISTORY table.

Column name	**Null ?**	**Data type**
EMPLOYEE_ID	NOT NULL	NUMBER(6)
START_DATE	NOT NULL	DATE
END_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
DEPARTMENT_ID		NUMBER(4)

LOCATIONS table.

Column name	**Null ?**	**Data type**
LOCATION_ID	NOT NULL	NUMBER(4)
STREET_ADDRESS		VARCHAR2(40)
POSTAL_CODE		VARCHAR2(12)
CITY	NOT NULL	VARCHAR2(30)
STATE_PROVINCE		VARCHAR2(25)
COUNTRY_ID		CHAR(2)

REGIONS table.

Column name	**Null ?**	**Data type**
REGION_ID	NOT NULL	NUMBER
REGION_NAME		VARCHAR2(25)

Database connection details

- Username: your domain account (without @epam.com).
- Password: your domain account (without @epam.com, Capitalized).
- Database: evuakyisd0228:1521/orcl

Operator Precedence

1. `+, -` (as unary operators)
2. `*, /`
3. `+, -` (as binary operators), `||`
4. SQL conditions are evaluated after SQL operators

Condition Precedence

1. `=, !=, <, >, <=, >=`
2. `IS [NOT] NULL, LIKE, [NOT] BETWEEN, [NOT] IN, EXISTS`
3. `NOT`
4. `AND`
5. `OR`

NULL

- Means ‘Unknown’ or ‘Unavailable’
- Any arithmetic operation with NULL gives NULL;
- Ternary logic. Any condition in Oracle SQL can be TRUE, FALSE or NULL (unknown);
- Any condition with NULL (except IS [NOT] NULL) is unknown;
- Empty string (‘’) in Oracle is identical to NULL.

Useful links

- Oracle Database 11g documentation
- SQL Reference
- 2 Day Developer Guide
- Concepts
- Error messages
- http://asktom.oracle.com
- www.sql.ru

Practice

1. Select all information from REGIONS table;
2. Find all unique employees’ last names;
3. Find all departments with names starting with "IT";
4. Find full names and salaries of employees who yearn from 8000 to 12000;
5. Find all employees phone numbers that contain substring '123' anywhere in the number;
6. For each employee calculate gross income (salary + commission_pct*salary) and form the string like "<Name> <Surname> earns <sum> USD";
7. Evaluate this logical condition: "not (1=2 or 'dumy' = null) ". It’s not necessary to write the query, but you can use query to check your answer;
8. Evaluate this logical condition: "1 > null and 0 <> 0". It’s not necessary to write the query, but you can use query to check your answer.

Part 2. Dates and functions

Date literals

- Implicit conversion from string ('01.01.2012') is environment-dependent => unreliable;
- ANSI literals (DATE'2012-12-31') are OK. Don't allow time part.
- TO_DATE function is the most flexible and commonly used.

SQL functions

Case transformation functions

LOWER('Oracle SQL')	oracle sql
UPPER('Oracle SQL')	ORACLE SQL
INITCAP('Oracle SQL')	Oracle Sql

Character functions

LPAD('Oracle SQL',15,'*')	*****Oracle SQL
RPAD('Oracle SQL',15,'*')	Oracle SQL*****
LTRIM('==> Oracle SQL <==','=<> ')	Oracle SQL <==
RTRIM('==> Oracle SQL <==','=<> ')	==> Oracle SQL
TRIM('=' FROM '==> Oracle SQL <==')	> Oracle SQL <
REPLACE('Oracle SQL', 'Oracle', 'ANSI')	ANSI SQL
SUBSTR('Oracle SQL', 1, 6)	Oracle
TRANSLATE('Introduction to Oracle SQL!', ' !','_')	Introduction_to_Oracle_SQL
INSTR('Introduction to Oracle SQL', 'o', 10, 2)	15
LENGTH('Oracle SQL')	10

Numeric functions

ABS(-10)	10
MOD(10,3)	1
ROUND(3.14159,4)	3.1416
TRUNC(3.14159,4)	3.1415

Date functions

SYSDATE	Current DB server date
ADD_MONTHS	Add calendar months
NEXT_DAY	Nearest specified day of the week
LAST_DAY	Last day of the month
ROUND	Round date up to given component
TRUNC	Truncate date up to given component
EXTRACT	Extracts date component (year, month etc.) from date

NULL-related functions

NVL(expr1, expr2)	if <expr1> is not null
	  Return <expr1>
	else
	  Return <expr2>

NVL2(expr1, expr2, expr3)	if <expr1> is not null
	  Return <expr2>
	else
	  Return <expr3>

COALESCE(expr1,…,exprN)	Return first non-NULL argument

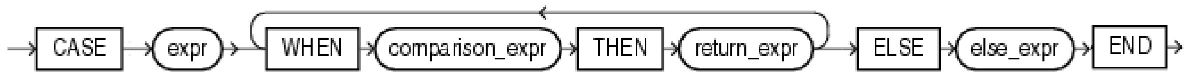
LNNVL(boolean expr)	If <boolean expr>
	  Return False
	else
	  Return True

Conversion functions

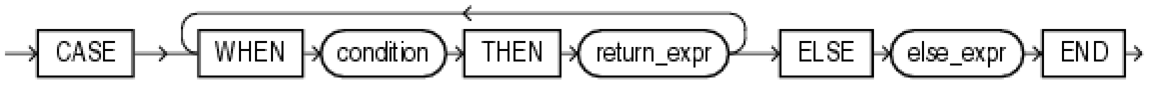
TO_DATE('string','format')	Converts string to date using specified format.
TO_CHAR(<argument>, 'format')	Converts number or date to string using specified format.
TO_NUMBER(<string>,'format')	Converts string to number using specified format.
CAST(<argument> as <type>)	Converts argument to specified type.

Conditional expressions

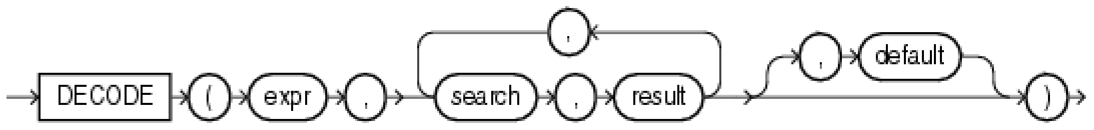
Simple CASE expression:



Searched CASE expression:



DECODE:



Useful links

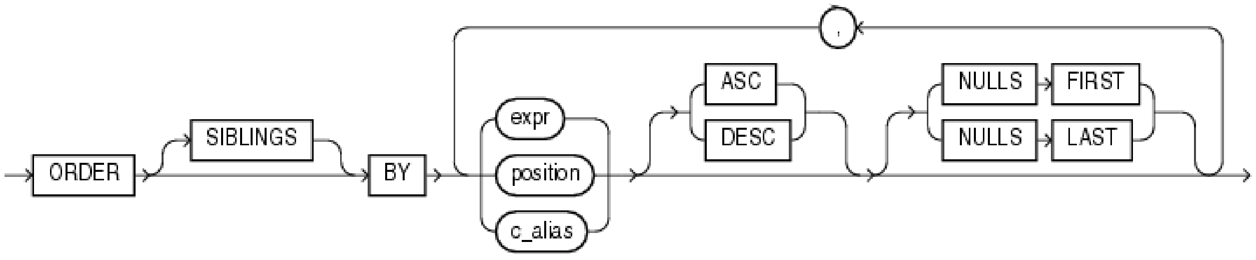
- Datetime literals
- Single-row functions
- CASE expressions

Practice

1. Select ID and FIRST_NAME of all employees with first names starting with "JA". Search should be case-insensitive.
2. For each employee form the string like "Person #<ID> has/hasn't commission";
3. List all "valuable" employees with one query. Employee is “valuable” if both conditions below are true for him:
    a. He is hired before 2007;
    b. He has salary from 7000 to 10000 or his JOB_ID starts with ‘IT’;
4. Find number of days to New Year. Note: no hardcoded dates are allowed;
5. Calculate how many seconds have passed since month start;
6. Find the second Sunday of next year. Note: no hardcoded dates are allowed;
7. For all employees hired in June (any year) print string like 'John Doe was hired on 01.01.2006';
8. List employees full names and their phone numbers in "encoded" format. Format is:
    a. no dots, just numbers;
    b. each digit should be replaced according to the formula F(d) = MOD(d+5,10) . For example, F(1) = 6, F(7) = 2.

Part 3. Ordering data.

ORDER BY clause



Query without ORDER BY clause returns rows in virtually random order.

ROWNUM

- Pseudocolumn;
- Row number in query result set;
- Used for top-N and pagination queries;
- Assined after WHERE predicates evaluation and before ORDER BY clause.
- Remember: rownum > 1 is always false.

Pagination query structure

- Subquery 1 – with ORDER BY clause;
- Subquery 2 – with “SELECT …, rownum as rn”
- Main query – with “WHERE rn between…”

Useful links

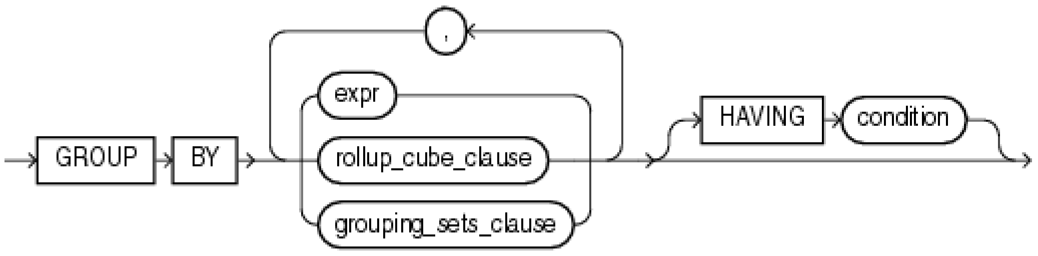
- ORDER BY clause
- ROWNUM
- ROWNUM by Tom Kyte

Part #3 practice

1. List all cities from LOCATIONS table in alphabetical order;
2. Select top 3 longest job periods from JOB_HISTORY;
3. With one query list all employees in the following order:
    a. first - employees with empty MANAGER_ID or DEPARTMENT_ID;
    b. then - other employees ordered by MANAGER_ID and DEPARTMENT_ID, both in descending order;
4. Your application shows top-paid employees, 10 persons per page. Write a query for the second page;
5. With one query create list of employees with full name, hire date and salary. Use the following ordering:
    a. first part - people with salary > 10000 or hired before '01.01.2007'. Order people by full name inside this list;
    b. second part - all other people. Again, order the list by full name.

Part 4. Grouping data.

GROUP BY clause



Most-used aggregate functions

- COUNT
- SUM
- MIN, AVG, MAX
- LISTAGG

All aggregate functions but count(*) ignore nulls.

All aggregate functions but “count” return NULL for empty data set. Count returns 0.

Aggregate functions can be used with DISTINCT keyword. Duplicated values are ignored in this case.

Aggregate functions can be nested. However, only 2 levels and only with GROUP BY clause is allowed. Use subqueries for other situations.

Useful links

- Aggregate functions
- GROUP BY clause

Part #4 practice

1. Calculate maximum, minimum and average salary for each department;
2. Select number of unique JOB_IDs in EMPLOYEES table;
3. List departments having more than 10 employees or summary salary > 30000;
4. Find 5 most-paid (in average) JOB_IDs;
5. Find average number of employees in department;
6. Show list of DEPARTMENT_IDs having at least one employee with salary > 8000. Show total salary for each such department.
7. For each department in EMPLOYEES table calculate:
    a. Number of people with commission;
    b. Number of people without commission.
8. For each JOB_ID show list of employees on this position with their DEPARTMENT_IDs and SALARIES (one row per JOB_ID). List of employees should be ordered by salary in descending order.

Part 5. JOINs

JOIN means attaching rows from one table to rows from another table [using some condition].

Examples data:

select * from dep

Dep_ID	Dep_Name
10	Accounting
20	IT
30	Useless
40	Security

select * from emp

Last_Name	Dep_ID
King	10
Brooks	10
Bush	20
Smith	30
Baker	

Cartezian (cross) JOIN

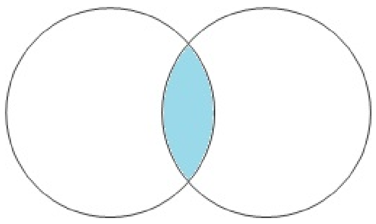
	Accounting 10	IT 20	Useless 30	Security 40
King 10	🟦	🟦	🟦	🟦
Brooks 10	🟦	🟦	🟦	🟦
Bush 20	🟦	🟦	🟦	🟦
Smith 30	🟦	🟦	🟦	🟦
Baker NULL	🟦	🟦	🟦	🟦

```
select
  e.dep_id
 ,e.last_name
 ,d.dep_id
 ,d.dep_name
from emp e
  cross join dep d
```

Inner JOIN

	Accounting 10	IT 20	Useless 30	Security 40
King 10	🟦			
Brooks 10	🟦			
Bush 20		🟦		
Smith 30			🟦	
Baker NULL				

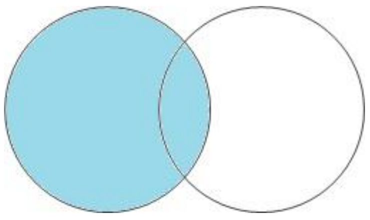
```
select
  e.dep_id
 ,e.last_name
 ,d.dep_id
 ,d.dep_name
from emp e
  inner join dep d
  on e.dep_id = d.dep_id
```



Left [outer] JOIN

	Accounting 10	IT 20	Useless 30	Security 40
King 10	🟦			
Brooks 10	🟦			
Bush 20		🟦		
Smith 30			🟦	
Baker NULL				

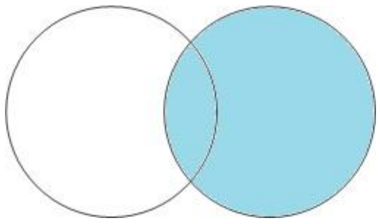
```
select
  e.dep_id
 ,e.last_name
 ,d.dep_id
 ,d.dep_name
from emp e
  left outer join dep d
  on e.dep_id = d.dep_id
```



Right [outer] JOIN

	Accounting 10	IT 20	Useless 30	Security 40
King 10	🟦			
Brooks 10	🟦			
Bush 20		🟦		
Smith 30			🟦	
Baker NULL				

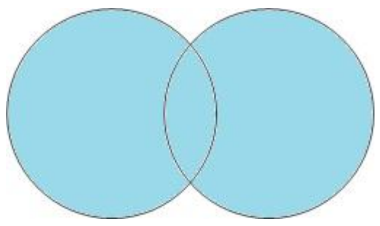
```
select
  e.dep_id
 ,e.last_name
 ,d.dep_id
 ,d.dep_name
from emp e
  right outer join dep d
  on e.dep_id = d.dep_id
```



Full [outer] JOIN

	Accounting 10	IT 20	Useless 30	Security 40
King 10	🟦			
Brooks 10	🟦			
Bush 20		🟦		
Smith 30			🟦	
Baker NULL				

```
select
   e.dep_id
  ,e.last_name
  ,d.dep_id
  ,d.dep_name
from emp e
   full outer join dep d
   on e.dep_id = d.dep_id
```

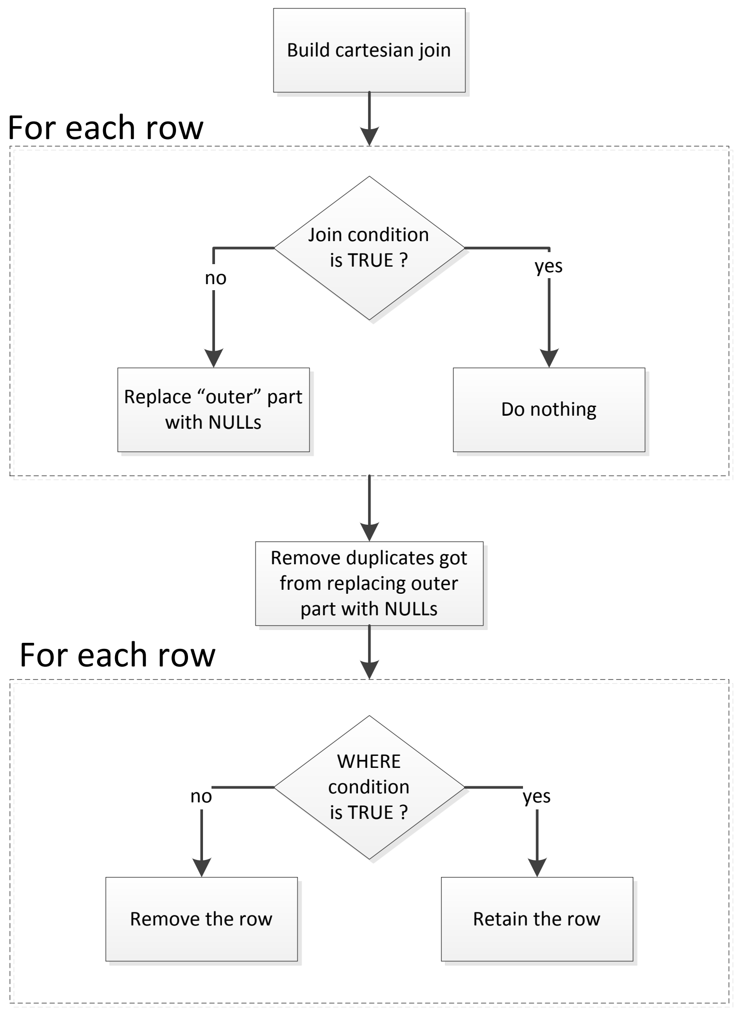


Native Oracle JOIN syntax

- coma-separated list of tables in FROM clause;
- join conditions in WHERE clause;
- (+) syntax for outer joins.

```
select
   e.dep_id
  ,e.last_name
  ,d.dep_id
  ,d.dep_name
from emp e, dep d
where e.dep_id = d.dep_id(+)
```

Outer join processing model



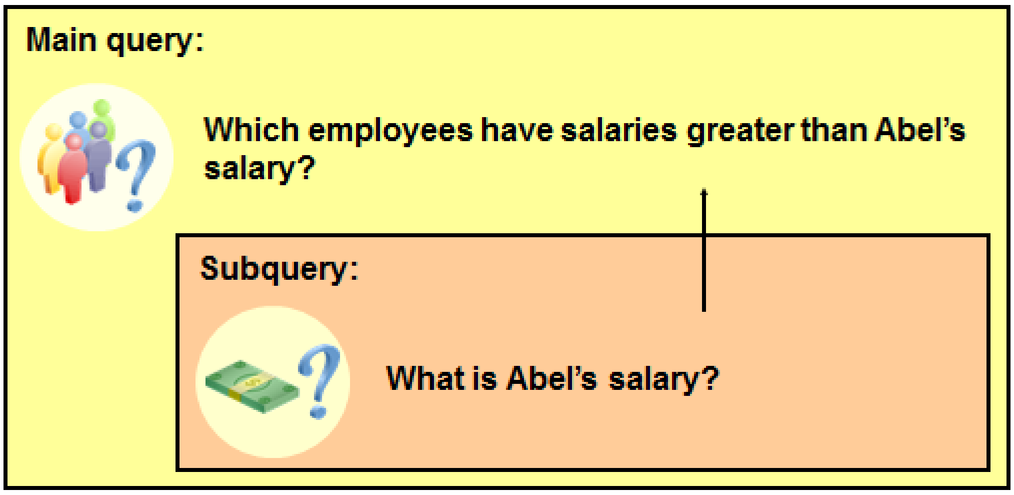
Useful links

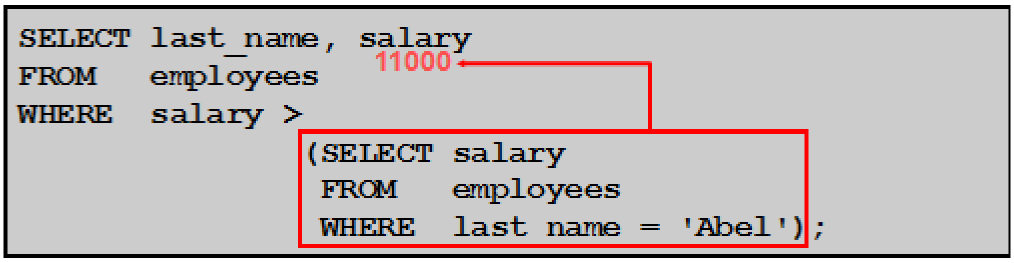
- Joins in documentation
- Join condition VS where condition
- Full outer join with Oracle syntax

Practice

1. List DEPARTMENT_NAMEs for departments located in UK.
2. Create list of all employees. Output should include employee LAST_NAME, JOB_TITLE and CITY where employee is located.
3. Show departments' names with number of employees having salary over 9000 for each department (note: departments without such employees should be included too). Show departments with max. number of such people first.
4. For each department print DEPARTMENT_NAME, LAST_NAME of manager and budget (total salary);
5. Print full names of employees who worked (and is not working anymore) as "Public Accountant" on '01.12.2000'.
6. Find the name of employee who worked as "Administration Assistant" and then as "Public Accountant". Both jobs are in the past (JOB_HISTORY table).

Part 6. Subqueries





```
SELECT last_name, salary
FROM   employees
WHERE  salary >
                (SELECT salary
                 FROM   employees
                 WHERE  last_name = 'Abel');
```

Scalar subquery

- One column, one (or zero) row.
- Used almost anywhere in SELECT statement
- Used with =, <>, >, < etc.
- Zero rows => NULL value;
- More than 1 row => error;

Multi-column subquery

- Many columns, one (or zero) row.
- Used in WHERE and HAVING clauses;
- Used with = and <> operators;

Multi-row subquery

- One or many columns, many row.
- Used in WHERE and HAVING clauses;
- Used with [NOT] IN, [NOT] EXISTS, ALL, ANY operators;

Correlated subquery – subquery that has a reference to parent query column(s).

Inline view

- Subquery in FROM clause.
- Used as row source for parent query.
- Can't be correlated (use join condition if correlation is needed).

WITH subquery

- Known as “subquery factoring clause”
- Created named subquery for further reference in parent query;
- Useful if same subquery is needed more than ones;
- Useful for test data sets;
- Since 11g can be used for recursive queries;

Useful links

- Using Subqueries
- AskTom EXISTS vs IN
- AskTom NOT EXISTS vs NOT IN
- 9i optimizer operations

Practice

1. List all employees having the same manager as Sarah Bell;
2. Print full names of employees who have namesakes in company. Order list by first names. Note: namesake - person with the same name.
3. Find all employees who had at least two other positions before;
4. List all departments having no employees currently. Do it in two different ways.
5. For each employee find his full name, salary and salary rank in his department. Order list by DEPARTMENT_ID and salary rank.

Part 7. Set operations

Database set operations introduce basic mathematical set operations against datasets. They are:

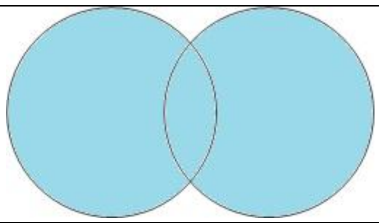
- UNION and UNION ALL
- MINUS
- INTERSECT

Both query blocks in set operation must have the same structure:

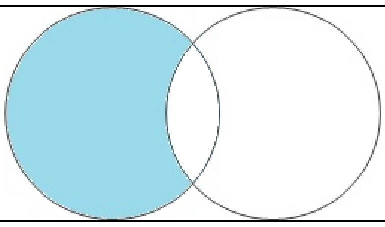
- Same number of columns;
- Same types of corresponding columns.

Column names for set operation result is defined by column names of **first** query block.

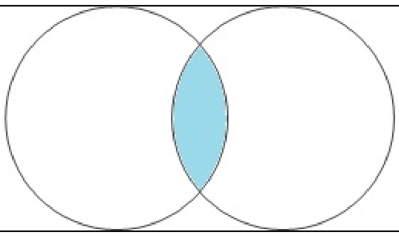
UNION, UNION ALL

T1 UNION T2 – includes rows from T1 and rows from T2. Duplicates are deleted. T1 UNION ALL T2 – includes ALL rows from T1 and T2.	

MINUS

T1 MINUS T2 – includes rows from T1 which are not present in T2. Duplicates are deleted.	

INTERSECT

T1 INTERSECT T2 – includes rows present in both from T1 and T2. Duplicates are deleted.	

Set operations order of precedence

- from left to right by default;
- can be changed with parentheses.

ORDER BY with set operations

- can be used in the last query block only;
- can use column aliases from the first query block.

Useful links

- UNION [ALL], INTERSECT, MINUS operators

Practice

1. List full names of people who worked or is currently working as "Programmer". Specify “Currently” or “In the past” in a separate column.
2. Find the most popular month for hiring people in our company. Take into account currently working people and history.
3. Find all departments that are either located in Canada or have not less than 10 people.
4. For each employee print full name and number of different positions (jobs) he worked. Take into account current positions and JOB_HISTORY.
5. List names of departments that have employees currently and hadn't fired employees in the past.

Part 8. DML statements.

Insert statement

```
INSERT INTO   table [(column [, column...])]
VALUES        (value [, value...]);
```

```
INSERT INTO sales_reps(id, name, salary, commission_pct)
  SELECT employee_id, last_name, salary, commission_pct
  FROM   employees
  WHERE  job_id LIKE '%REP%';

4 rows inserted
```

Update statement

```
UPDATE        table
SET           column = value [, column = value, ...]
[WHERE        condition];
```

Delete statement

```
DELETE [FROM]   table
[WHERE          condition];
```

Merge statement

- Insert and update statements together;

```
MERGE
   INTO  table [ t_alias ]
   USING { table | subquery } [ t_alias ]
   ON ( condition )
   WHEN MATCHED THEN
     UPDATE SET column = { expr | DEFAULT }
             [, column = { expr | DEFAULT } ]...
   WHEN NOT MATCHED THEN
     INSERT [ (column [, column ]...) ]
     VALUES ({ expr | DEFAULT }
          [, { expr | DEFAULT } ]...)
```

Truncate statement

- fast table cleanup;
- DDL statement (commits transaction and can’t be rolled back);
- will fail if table is referenced by any foreign key constraint;

Useful links

- INSERT statement
- UPDATE statement
- AskTom. UPDATE based on other table
- DELETE statement
- MERGE statement

Practice

1. Add yourself to EMPLOYEES table.
2. Update all emails in EMPLOYEES table adding "@epam.com" to the end.
3. Set all employees’ salary to maximum salary available for their position (see JOBS table).
4. Delete records with even EMPLOYEE_ID from JOB_HISTORY table.
5. Delete current records from JOB_HISTORY table. Record is considered "current" if combination of (employee_id, start_date, job_id and department_id) is present in EMPLOYEES table.
6. Modify you own record in EMPLOYESS (added in Task #1) to have NULL in DEPARTMENT_ID.
7. Modify "MERGE INTO emp_directory" statement shown in class. For employees without department DEPARTMENT_ID and CITY fields should be set to 'Unknown' in both insert and update part of statement. Execute MERGE statement and ensure it works correctly.

Part 9. Transactions

Transaction and ACID

Transaction in RDBMS – unit of work having the following properties (aka ACID properties):

- **Atomicity**. Transaction is executed successfully up to the end or is not executed at all.
- **Consistency**. Transaction moves DB from one consistent state to another. Note: database may be inconsistent inside transaction.
- **Isolation**. Many definitions can be found. In fact, it means RDBM should provide some mechanisms to isolate transaction from changes done in other transactions (see "Isolation levels" below).
- **Durability.** Being committed, transaction should be preserved forever despite all possible failures (RDBMS, OS, power outage etc.)

Oracle architectural features for ACID

UNDO data – image of data before modification. Is used for:

- **Atomicity** – allows to rollback changes in case of error.
- **Isolation** – allows "read consistency" – reconstruction of data past-images.

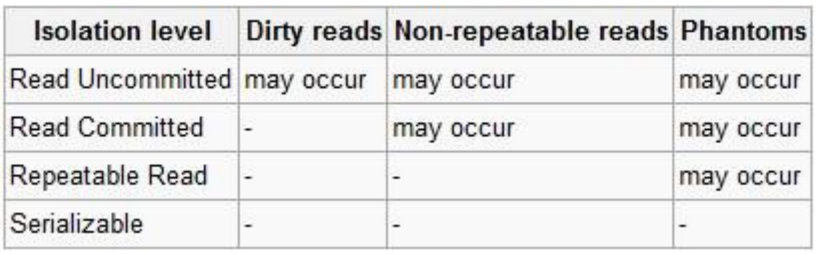
REDO data – log of database changes that allows to "repeat" changes applied to database. Is used for:

- **Durability –** all changes done by commited transaction are available in redo logs on disk. This allows RDBMS to reconstruct database blocks in case of failure (and loss of in-memory data copy).

Isolation problems

- **Dirty read**. Transaction reads uncommitted data;
- **Non-repeatable read**. Transaction reads the same rows twice and rows are changed between these two reads.
- **Phantom reads**. Transaction reads data according to some criteria twice. New rows meeting criteria appears between these two reads.

ANSI isolation levels



Isolation level	Dirty reads	Non-repeatable reads	Phantoms
Read Uncommitted	may occur	may occur	may occur
Read Committed	-	may occur	may occur
Repeatable Read	-	-	may occur
Serializable	-	-	-

Oracle Isolation levels

Oracle Isolation Level	ANSI analogue
READ COMMITTED (default)	For statement - serializable For transaction - read committed
SERIALIZABLE	Serializable

Transaction control statements

- COMMIT – end transaction and “save” changes;
- SAVEPOINT – “remember” current state to rollback here later;
- ROLLBACK TO SAVEPOINT – decline all changes done since savepoint;
- ROLLBACK – end transaction and decline all changes;
- SET TRANSACTION – modify transaction parameters (ro/rw, isolation level etc.)

Useful Links

- Transactions concepts
- "The Transaction Concept" by Jim Gray
- «Оракл для профессионалов» Тома Кайта, глава 4.
- Когда начинается транзакция?

Part 10. Data dictionary, data types and CREATE/ALTER/DROP table statements.

Schema

- Just a set of objects which belongs to one user. Not a separate object!
- Schema name = user name
- Qualified object name: SCHEMA.OBJECT
- Unqualified object name – name without schema. By default is resolved using current user's schema.

Data dictionary

Oracle data dictionary – set of views with information about database objects, current state and operational statistics. Data dictionary can be accessed with usual SQL statements.

List of data dictionary views can be found:

- In documentation (see Useful links);
- In special view called "dictionary" or "dict": select * from dict;

Two big groups of views:

1. **Data dictionary** itself. "Static" information about database objects (files, users, tables, columns, indexes etc.). Many of those views are available with 3 prefixes:
    - Prefix "user_" (like user_tables) – objects in current user's schema.
    - Prefix "all_" (like all_tables) – objects current user can access. (including objects from "user_" view).
    - Prefix "dba_" (like dba_tables) – all database objects.
2. **Dynamic performance views**. Provide information about current database state and cumulative statistics since last startup. Are prefixed with "v$". Examples: v$session (current DB sessions), v$sql (cached SQL statements), v$session_longops (long-running DB operations), v$sysstat (system statistics).

Simple CREATE TABLE statement

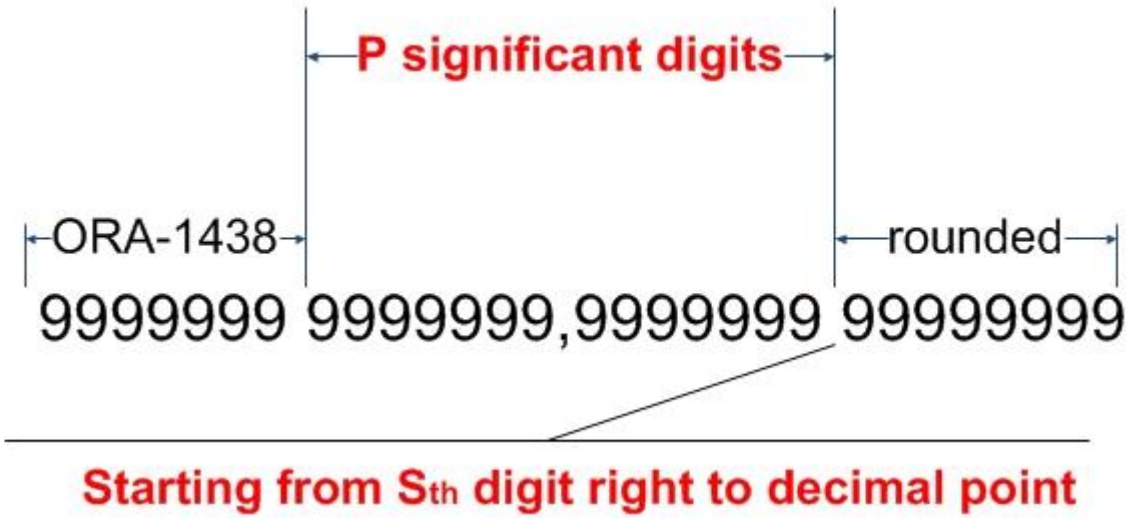
```
create table person(
       person_id  number(10)
      ,full_name  varchar2(60)
      ,birth_date date default SYSDATE
      ,sex        char(1)
);
```

Identifiers in Oracle

- must be 1-30 characters long;
- must begin with a letter;
- may contain alphnumeric characters, _, $, #;
- must not be Oracle reserved word (v$reserved_words view);
- identifier can break these rules if it's "quoted". Not recommended.

Number(p,s) type

- P – precision, number of **stored** decimal digits. 38 maximum.
- S – scale, number of digits after (before if negative) decimal point.
- Is automatically rounded to Scale.
- Returns error if precision is not enough.
- Number(p) = number(p,0) – integer with P decimal digits.



Character types

	Database charset	National charset
Fixed length, right-padded with spaces	Char(N)	NChar(N)
Variable length	Varchar2(N)	NVarchar2(N)

(N) – number of bytes (default) or characters. Semantics can be given directly as “varchar2(10 char/byte)”. 4000 bytes is maximum.

Date type

- Just date without fractions of second and without timezone information.

Other data types

- BINARY_FLOAT, BINARY_DOUBLE – floating point numbers in machine format.
- CLOB, BLOB, BFILE – large **C**haracter or **B**inary objects.
- LONG, LONG RAW – deprecated data types for characted and binary large objects.
- RAW – small (2000 bytes)binary objects.
- TIMESTAMP, TIMESTAMP WITH TIMEZONE, TIMESTAMP WITH LOCAL TIMEZONE – date and time with fractions of second and timezone info (with some differences).
- INTERVAL YEAR TO MONTH, INTERVAL DAY TO SECOND – time intervals.
- ROWID – physical address of row in its table.

DEFAULT values

- used if no value have been provided in insert statement.
- can be any expression with literals and SQL functions.
- cannot contain references to table columns.

Create Table As Select

- Creates table in fills it with data from given query.
- Data types are defined automatically.

ALTER TABLE

- Add/drop/modify/rename column;
- Add/drop/modify/rename constraints;
- Rename table
- And much more.

DROP TABLE – drops the table ☺

Useful links

- Database reference index
- Codd's rules
- CREATE TABLE statement
- Object naming rules
- Data types
- ALTER TABLE statement

Practice

- Using **DBA_OBJECTS** view count number of objects in the database;
- Using **USER_TAB_COLUMNS** view create two-column report. First column should contain table name, second column – comma separated list of table's columns.
- Using **DBA_USERS** view create report showing user name, date of account creation and current account status.
- Using **DBA_USERS** and **v$session** view find number of currently connected sessions for each user in the database. Order list by number of sessions.
- Using **v$sql** view (and documentation ☺) find out how much time your query from previous task took to execute.
- Create table COMPANIES to store the following information:
    - ID of company. Integer number, as large as possible.
    - Company name. String, max. 40 characters length.
    - Full name of company. String, 255 characters length.
    - Date of company foundation. If not provided – should be set to current date.
    - Country of residence code. Two characters.
    - Yearly revenue. Should be automatically rounded to $1000.

Part 11. Integrity constraints

Integrity constraints - rules or limitations enforcing data accuracy and consistency. Database engine guarantee that data don't break any created (and enabled and validated) constraint.

NOT NULL

- Column-level constraint.
- Does not allow NULLs in column.

UNIQUE

- Table-level constraint. Can include one or more column(s).
- Does not allow duplicated values (duplicated combinations of values) in column(s).
- Consider several NULLs as different values.
- Several unique constraints can be created for table.

PRIMARY KEY

- Table-level constraint. Can include one or more column(s).
- Like "NOT NULL + UNIQUE" – prohibits nulls and duplicates in column(s).
- Only one primary key per table is allowed.

FOREIGN KEY

- Table-level constraint. Can include one or more column(s).
- Ensures that value from child table is present in parent table.
    - Does not allow to insert into child table value not present in parent;
    - Does not allow to delete from parent table value present in child (By default. This can be changed with ON DELETE CASCADE or ON DELETE SET NULL);
- Several foreign keys can be created for table (and even for column).

CHECK

- Row-level constraint.
- Ensures than row does not break some condition – i.e. condition is TRUE or NULL for each row.
- Can reference literals, columns and deterministic SQL functions;
- Several check constraint per table and per column can be created.

Useful links

- Integrity constraints description
- Integrity constrains syntax

Practice

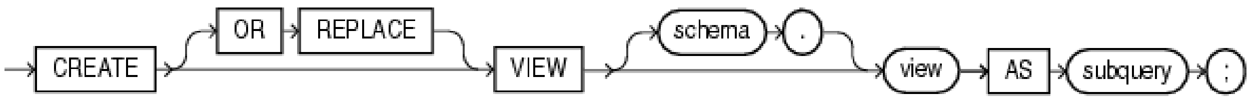
1. Modify table COMPANIES created at the previous lesson:
    a. Make ID of company necessary and unique.
    b. Make company name necessary.
    c. Enforce the following rule: if full name of company is present, it should be unique.
    d. Enforce the following rule: date of company foundation should be JUST date, without time.
    e. Enforce the following rule: country of residence code should be in UPPERCASE.
2. Modify EMPLOYEES table:
    a. Add necessary field COMPANY_ID.
    b. Ensure that value of this field is one of company id's from COMPANY table;
    c. Ensure that company can't be deleted if it has at least one employee.

Part 12. Views, sequences and synonyms

View

- Named query stored in the database;
- Query text is stored, not query result;
- Used in SQL statements just like tables.

Simplified create view syntax:



Views are useful for:

- access restriction to column and/or rows;
- queries simplification;
- code reuse;
- application isolation from DB structure.

DML statements on views

- are allowed with some restrictions;
- are transformed to DML on view base table automatically;
- in general, restriction is: view query should allow one-to-one mapping to table you want to modify;
- query clauses that makes view non-updatable: GROUP BY, DISTINCT, set operations, some joins etc.
- Note: INSTEAD OF triggers might be used to allow DML on non-updateable views.

Sequences

- special objects for unique numbers generation

```
CREATE SEQUENCE sequence
  [INCREMENT BY n]   -- increment step. Might be negative.
  [START WITH n]     -- start value
  [{MAXVALUE n | NOMAXVALUE}] -- upper bound
  [{MINVALUE n | NOMINVALUE}] -- upper bound
  [{CYCLE | NOCYCLE}]  -- stop when bound reached or restart?
  [{CACHE n | NOCACHE}] -- in-memory cache size
  [{ORDER|NOORDER}];    -- values strictly ordered in cluster?
```

select sequence.nextval from dual - next value of sequence.

select sequence.curval from dual – last generated value **in this session**

Nextval and currval

Can be used in:

- The select list of a SELECT statement that is not contained in a subquery, materialized view, or view
- The select list of a subquery in an INSERT statement
- The VALUES clause of an INSERT statement (+MERGE)
- The SET clause of an UPDATE statement (+MERGE)

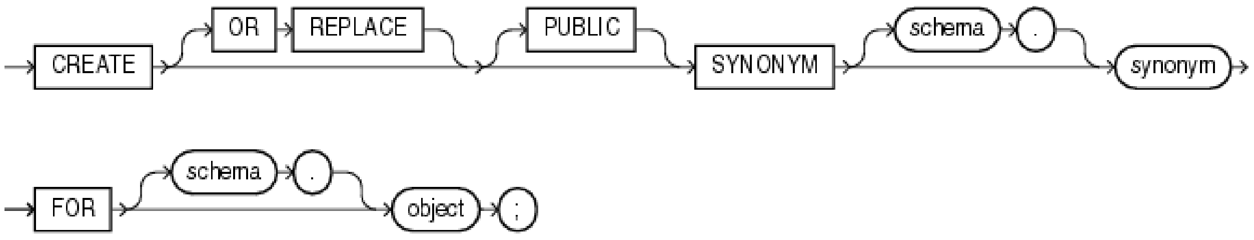
Nextval is incremented ones for each row:

- returned by SELECT statement (individual, insert...select, multitable insert, CTAS);
- inserted by single-row insert;
- updated by UPDATE;
- merged by MERGE;

Synonym

- Alias for database object.

Simplified CREATE SYNONYM syntax



Synonyms usage:

- simplify queries;
- hide real object name and schema.

Useful links

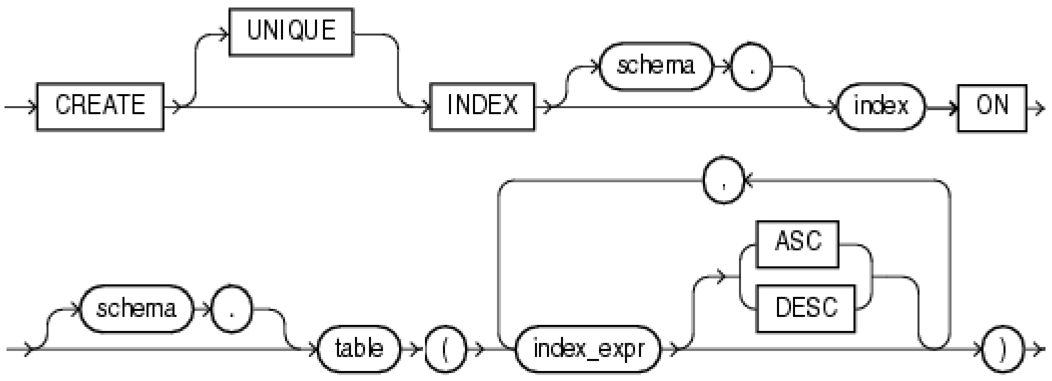
- Concepts: views, sequences, synonyms
- CREATE SEQUENCE
- Nextval and Currval
- CREATE SYNONYM

Part 13. Indexes

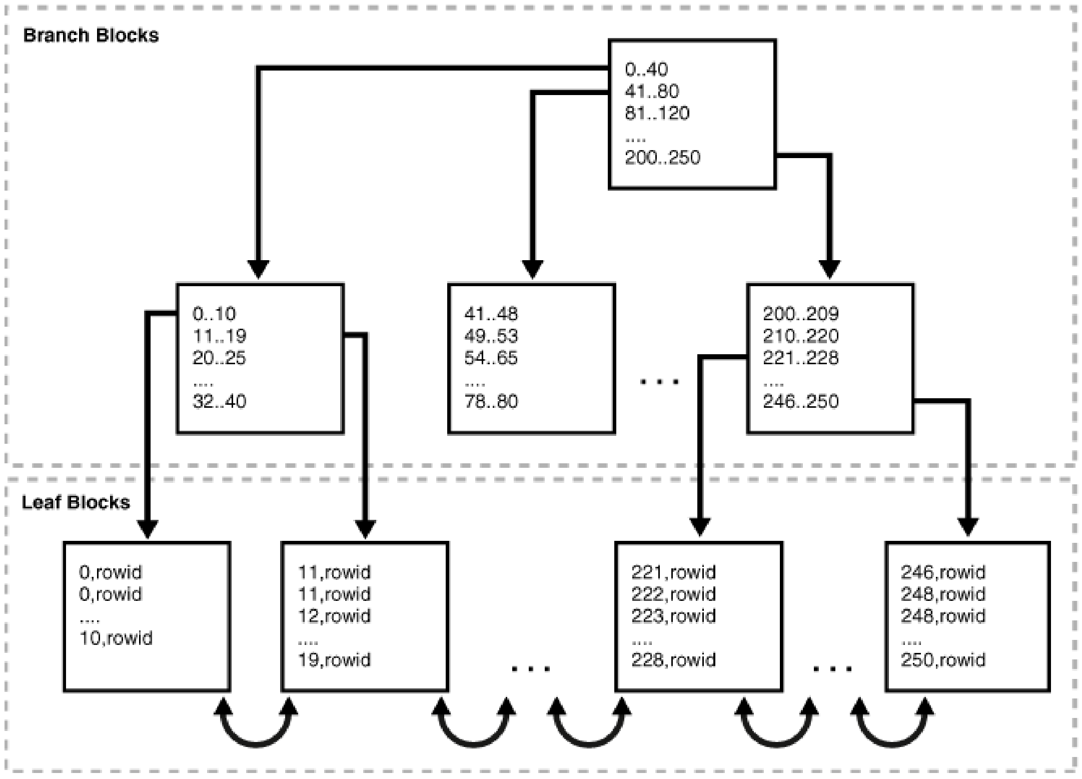
Indexes

- auxiliary data structures for fast data access;
- most popular type of index – B-tree index – is something like sorted key values with references to table rows.

Simplified CREATE INDEX syntax



B-tree index structure



Simple index creation considerations

- for column used in FOREIGN KEY constraints.
- for column used in WHERE and JOIN clauses;

Remember, index benefits also depend on:

- table size;
- data distribution;
- search/join criteria;
- search/join criteria selectivity;
- and many other factors...

Not covered

- unique/reverse key/descending b-tree indexes;
- function-based indexes;
- bitmap and bitmap join indexes;
- application-domain indexes.

Useful links

- Concepts: indexes
- CREATE INDEX
- "Why isn't Oracle using my index?" by Jonathan Lewis
- "Основные причины ошибок CBO" by Андрей Киселев
- "Cost-based Oracle fundamentals" by Johathan Lewis (book).