MTN.BI.02

ORACLE SQL FOUNDATION

ORACLE SQL BASICS,
THE SELECT STATEMENT

Author: **Aliaksandr Chaika**
Senior Software Engineer
Certified Oracle Database SQL Expert
Aliaksandr_Chaika@epam.com

Objectives

- SQL ANSI Standard
- Oracle Human Resources (HR) Sample Schema
- The SELECT Statement
- Joins
- Set Operations
- Pseudocolumns

ANSI STANDARDS FOR SQL

ANSI Standards for SQL

Year	Standard Name (and Aliases)	Oracle Database
1986	SQL-86 / SQL-87	
1989	SQL-89 / FIPS 127-1	
1992	SQL-92 / SQL2 / FIPS 127-2	
1999	SQL:1999 / SQL3	
2003	SQL:2003	Oracle 10g Release 1 Oracle 10g Release 2 Oracle 11g Release 1
2006	SQL:2006	
2008	SQL:2008	Oracle 11g Release 2
2011	SQL:2011	

ANSI/ISO Standard Structure

Standard Part	Name	Content
ISO/IEC 9075-1:2011 Part 1	Framework (SQL/Framework)	Concepts
ISO/IEC 9075-2:2011 Part 2	Foundation (SQL/Foundation)	Language elements
ISO/IEC 9075-3:2008 Part 3	Call-Level Interface (SQL/CLI)	Interfacing components
ISO/IEC 9075-4:2011 Part 4	Persistent Stored Modules (SQL/PSM)	Procedural extensions
ISO/IEC 9075-9:2008 Part 9	Management of External Data (SQL/MED)	Foreign-data and Datalinks
ISO/IEC 9075-10:2008 Part 10	Object Language Bindings (SQL/OLB)	SQLJ
ISO/IEC 9075-11:2011 Part 11	Information and Definition Schemas (SQL/Schemata)	Self-describing objects
ISO/IEC 9075-13:2008 Part 13	SQL Routines and Types Using the Java Programming Language (SQL/JRT)	Using Java in the database
ISO/IEC 9075-14:2011 Part 14	XML-Related Specifications (SQL/XML)	Using XML

Core SQL Language Syntax and Semantic

ISO/IEC 9075-1:2008 Part 1: Framework (SQL/Framework)

Provides logical concepts.

ISO/IEC 9075-2:2008 Part 2: Foundation (SQL/Foundation)

Contains the most central elements of the language and consists of both mandatory and optional features.

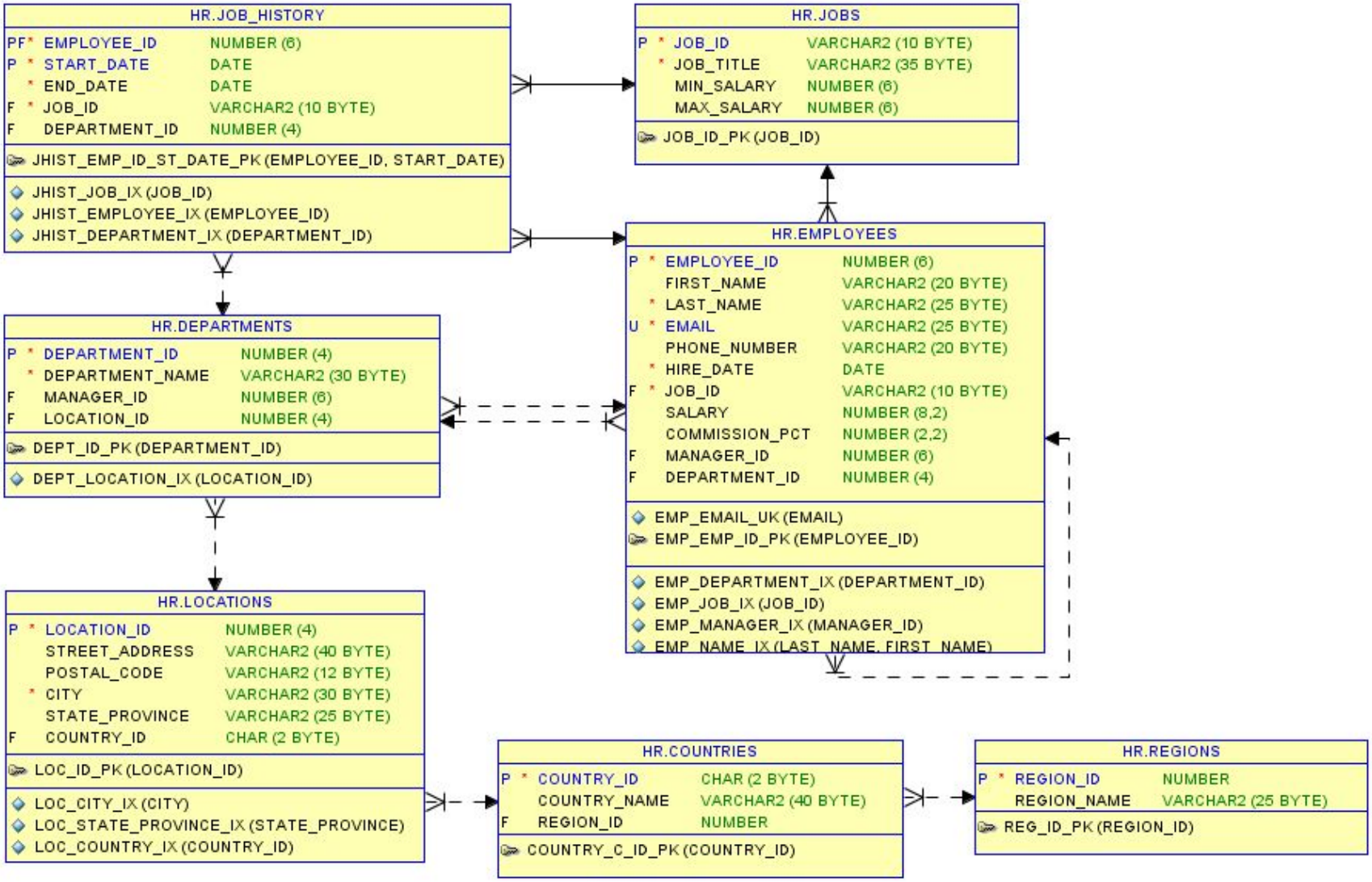
ISO/IEC 9075-11:2008 Part 11: Information and Definition Schemas (SQL/Schemata)

Defines the Information Schema and Definition Schema, providing a common set of tools to make SQL databases and objects self-describing.

Core SQL:2008

ORACLE HUMAN RESOURCES SAMPLE SCHEMA OVERVIEW

Oracle Sample Human Resources (HR) Schema



Retrieving all data from Employees table

Worksheet | Query Builder

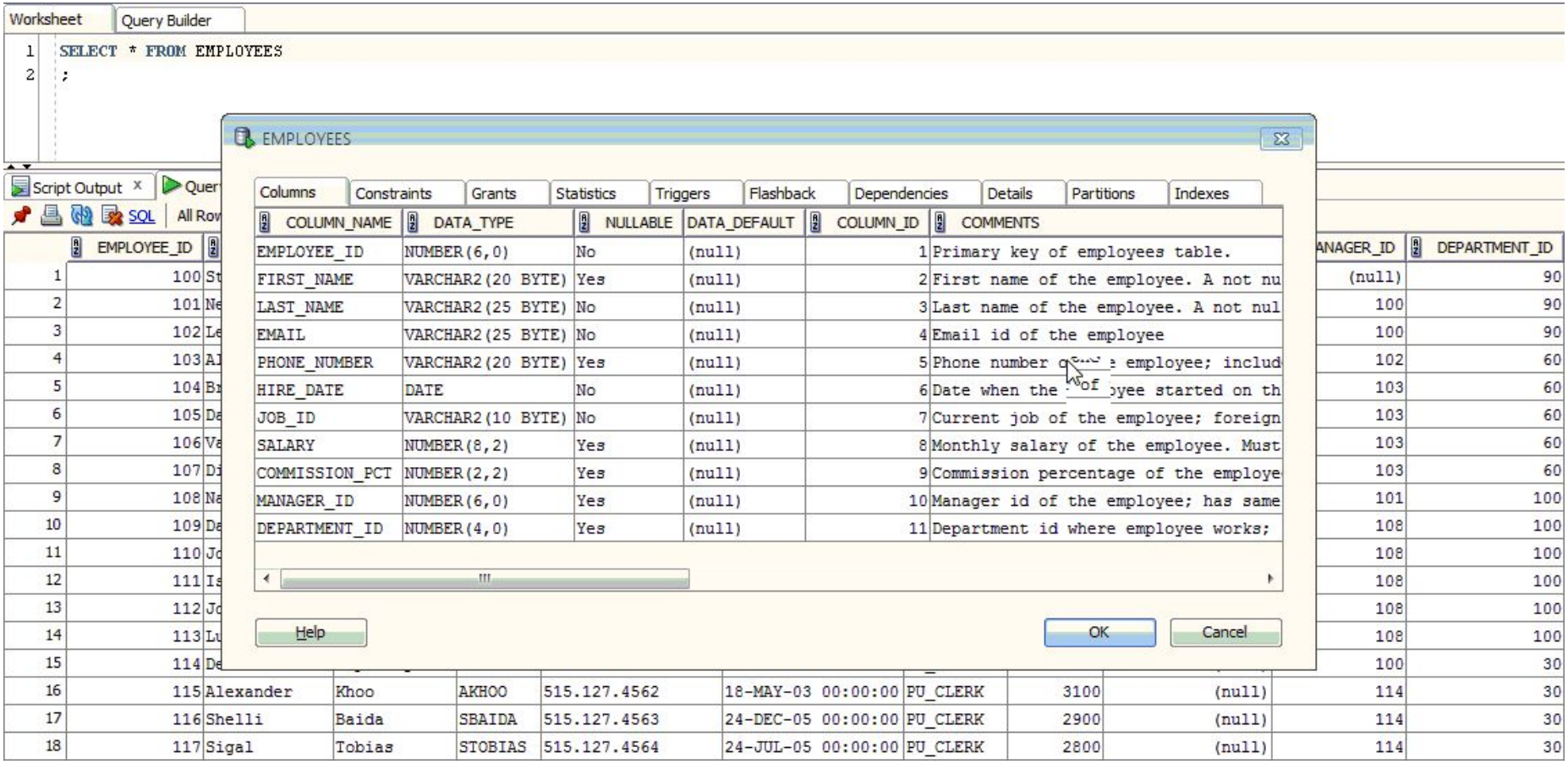
```
1  SELECT * FROM EMPLOYEES
2  ;
```

Script Output | Query Result

SQL | All Rows Fetched: 107 in 0.016 seconds

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
1	100	Steven	King	SKING	515.123.4567	17-JUN-03 00:00:00	AD_PRES	24000	(null)	(null)	90
2	101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-05 00:00:00	AD_VP	17000	(null)	100	90
3	102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-01 00:00:00	AD_VP	17000	(null)	100	90
4	103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-06 00:00:00	IT_PROG	9000	(null)	102	60
5	104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-07 00:00:00	IT_PROG	6000	(null)	103	60
6	105	David	Austin	DAUSTIN	590.423.4569	25-JUN-05 00:00:00	IT_PROG	4800	(null)	103	60
7	106	Valli	Pataballa	VPATABAL	590.423.4560	05-FEB-06 00:00:00	IT_PROG	4800	(null)	103	60
8	107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-07 00:00:00	IT_PROG	4200	(null)	103	60
9	108	Nancy	Greenberg	NGREENBE	515.124.4569	17-AUG-02 00:00:00	FI_MGR	12008	(null)	101	100
10	109	Daniel	Faviet	DFAVIET	515.124.4169	16-AUG-02 00:00:00	FI_ACCOUNT	9000	(null)	108	100
11	110	John	Chen	JCHEN	515.124.4269	28-SEP-05 00:00:00	FI_ACCOUNT	8200	(null)	108	100
12	111	Ismael	Sciarra	ISCIARRA	515.124.4369	30-SEP-05 00:00:00	FI_ACCOUNT	7700	(null)	108	100
13	112	Jose Manuel	Urman	JMURMAN	515.124.4469	07-MAR-06 00:00:00	FI_ACCOUNT	7800	(null)	108	100
14	113	Luis	Popp	LPOPP	515.124.4567	07-DEC-07 00:00:00	FI_ACCOUNT	6900	(null)	108	100
15	114	Den	Raphaely	DRAPHEAL	515.127.4561	07-DEC-02 00:00:00	PU_MAN	11000	(null)	100	30
16	115	Alexander	Khoo	AKHOO	515.127.4562	18-MAY-03 00:00:00	PU_CLERK	3100	(null)	114	30
17	116	Shelli	Baida	SBAIDA	515.127.4563	24-DEC-05 00:00:00	PU_CLERK	2900	(null)	114	30
18	117	Sigal	Tobias	STOBIAS	515.127.4564	24-JUL-05 00:00:00	PU_CLERK	2800	(null)	114	30
19	118	Guy	Himuro	GHIMURO	515.127.4565	15-NOV-06 00:00:00	PU_CLERK	2600	(null)	114	30
20	119	Karen	Colmenares	KCOLMENA	515.127.4566	10-AUG-07 00:00:00	PU_CLERK	2500	(null)	114	30

Employees Table Properties



Departments Table

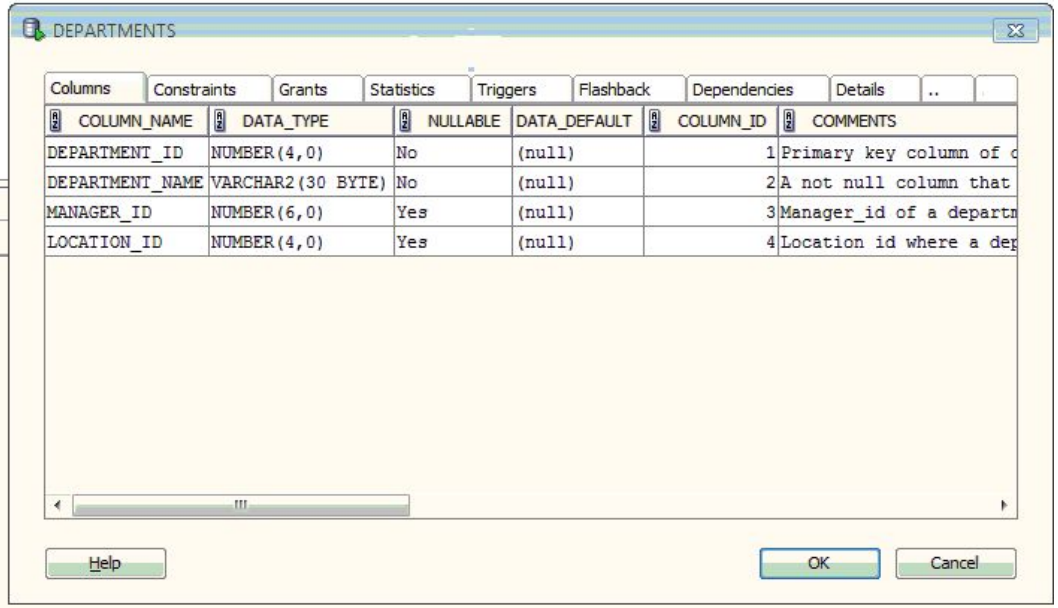
Worksheet | Query Builder

```
1  SELECT *
2  FROM DEPARTMENTS
3  ;
```

Query Result

SQL | All Rows Fetched: 27 in 0.004 seconds

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	30	Purchasing	114	1700
4	40	Human Resources	203	2400
5	50	Shipping	121	1500
6	60	IT	103	1400
7	70	Public Relations	204	2700
8	80	Sales	145	2500
9	90	Executive	100	1700
10	100	Finance	108	1700
11	110	Accounting	205	1700
12	120	Treasury	(null)	1700
13	130	Corporate Tax	(null)	1700
14	140	Control And Credit	(null)	1700
15	150	Shareholder Services	(null)	1700



Jobs Table

Worksheet | Query Builder

```
1  SELECT *
2  FROM JOBS
3  ;
4
5
```

Query Result

SQL | All Rows Fetched: 19 in 0.024 seconds

	JOB_ID	JOB_TITLE	MIN_SALARY	MAX_SALARY
1	AD_PRES	President	20080	40000
2	AD_VP	Administration Vice President	15000	30000
3	AD_ASST	Administration Assistant	3000	6000
4	FI_MGR	Finance Manager	8200	16000
5	FI_ACCOUNT	Accountant	4200	9000
6	AC_MGR	Accounting Manager	8200	16000
7	AC_ACCOUNT	Public Accountant	4200	9000
8	SA_MAN	Sales Manager	10000	20080
9	SA_REP	Sales Representative	6000	12008
10	PU_MAN	Purchasing Manager	8000	15000
11	PU_CLERK	Purchasing Clerk	2500	5500
12	ST_MAN	Stock Manager	5500	8500
13	ST_CLERK	Stock Clerk	2008	5000
14	SH_CLERK	Shipping Clerk	2500	5500
15	IT_PROG	Programmer	4000	10000
16	MK_MAN	Marketing Manager	9000	15000
17	MK_REP	Marketing Representative	4000	9000
18	HR_REP	Human Resources Representative	4000	9000
19	PR_REP	Public Relations Representative	4500	10500

Job_history Table

Worksheet | Query Builder

```
1  SELECT *
2  FROM JOBS
3  ;
4
5  SELECT *
6  FROM JOB_HISTORY
7  ;
8
9
```

Query Result

SQL | All Rows Fetched: 10 in 0.029 seconds

	EMPLOYEE_ID	START_DATE	END_DATE	JOB_ID	DEPARTMENT_ID
1	102	13-JAN-01 00:00:00	24-JUL-06 00:00:00	IT_PROG	60
2	101	21-SEP-97 00:00:00	27-OCT-01 00:00:00	AC_ACCOUNT	110
3	101	28-OCT-01 00:00:00	15-MAR-05 00:00:00	AC_MGR	110
4	201	17-FEB-04 00:00:00	19-DEC-07 00:00:00	MK_REP	20
5	114	24-MAR-06 00:00:00	31-DEC-07 00:00:00	ST_CLERK	50
6	122	01-JAN-07 00:00:00	31-DEC-07 00:00:00	ST_CLERK	50
7	200	17-SEP-95 00:00:00	17-JUN-01 00:00:00	AD_ASST	90
8	176	24-MAR-06 00:00:00	31-DEC-06 00:00:00	SA_REP	80
9	176	01-JAN-07 00:00:00	31-DEC-07 00:00:00	SA_MAN	80
10	200	01-JUL-02 00:00:00	31-DEC-06 00:00:00	AC_ACCOUNT	90

Locations Table

Worksheet | Query Builder

```
1  SELECT *
2  FROM LOCATIONS
3  ;
```

Query Result

SQL | All Rows Fetched: 23 in 0.06 seconds

	LOCATION_ID	STREET_ADDRESS	POSTAL_CODE	CITY	STATE_PROVINCE	COUNTRY_ID
1	1000	1297 Via Cola di Rie	00989	Roma	(null)	IT
2	1100	93091 Calle della Testa	10934	Venice	(null)	IT
3	1200	2017 Shinjuku-ku	1689	Tokyo	Tokyo Prefecture	JP
4	1300	9450 Kamiya-cho	6823	Hiroshima	(null)	JP
5	1400	2014 Jabberwocky Rd	26192	Southlake	Texas	US
6	1500	2011 Interiors Blvd	99236	South San Francisco	California	US
7	1600	2007 Zagora St	50090	South Brunswick	New Jersey	US
8	1700	2004 Charade Rd	98199	Seattle	Washington	US

Countries and Regions Tables

Worksheet | Query Builder

```
1  SELECT *
2  FROM LOCATIONS
3  ;
4
5  SELECT *
6  FROM COUNTRIES
7  ;
```

Query Result

SQL | All Rows Fetched: 25 in 0.036 seconds

	COUNTRY_ID	COUNTRY_NAME	REGION_ID
1	AR	Argentina	2
2	AU	Australia	3
3	BE	Belgium	1
4	BR	Brazil	2
5	CA	Canada	2
6	CH	Switzerland	1
7	CN	China	3
8	DE	Germany	1
9	DK	Denmark	1
10	EG	Egypt	4
11	FR	France	1
12	IL	Israel	4

Worksheet | Query Builder

```
5  SELECT *
6  FROM COUNTRIES
7  ;
8
9
10 SELECT *
11 FROM REGIONS
12 ;
```

Query Result

SQL | All Rows Fetched: 4 in 0.044 seconds

	REGION_ID	REGION_NAME
1	1	Europe
2	2	Americas
3	3	Asia
4	4	Middle East and Africa

Exploring data: Select Distinct Records

Worksheet | Query Builder

```
15
16  SELECT DISTINCT JOB_ID
17  FROM EMPLOYEES
18  ;
19
20
21
22
23
```

Script Output | Query...

SQL | All Rows Fetched: 19 in 0.002 seconds

	JOB_ID
1	AC_ACCOUNT
2	AC_MGR
3	AD_ASST
4	AD_PRES
5	AD_VP
6	FI_ACCOUNT
7	FI_MGR
8	HR_REP
9	IT_PROG
10	MK_MAN
11	MK_REP
12	PR_REP
13	PU_CLERK
14	PU_MAN
15	SA_MAN
16	SA_REP
17	SH_CLERK
18	ST_CLERK
19	ST_MAN

Worksheet | Query Builder

```
15
16  SELECT DISTINCT JOB_ID
17  FROM EMPLOYEES
18  ORDER BY 1
19  ;
20
21
22
23
```

Script Output | Query...

SQL | All Rows Fetched: 19 in 0.004 seconds

	JOB_ID
1	AC_ACCOUNT
2	AC_MGR
3	AD_ASST
4	AD_PRES
5	AD_VP
6	FI_ACCOUNT
7	FI_MGR
8	HR_REP
9	IT_PROG
10	MK_MAN
11	MK_REP
12	PR_REP
13	PU_CLERK
14	PU_MAN
15	SA_MAN
16	SA_REP
17	SH_CLERK
18	ST_CLERK
19	ST_MAN

Exploring data: Counting Records

Worksheet | Query Builder

```
13
14
15
16  SELECT COUNT(DISTINCT JOB_ID)
17  FROM EMPLOYEES
18  ORDER BY 1
19  ;
20
21
```

Script Output | Query...

SQL | All Rows Fetched: 1 in 0.004 seconds

	COUNT(DISTINCTJOB_ID)
1	19

Worksheet | Query Builder

```
14  ;
15
16  SELECT JOB_ID, COUNT(LAST_NAME)
17  FROM EMPLOYEES
18  GROUP BY JOB_ID
19  ORDER BY 2 DESC
20  ;
21
22
```

Script Output | Query Result

SQL | All Rows Fetched: 19 in 0.004 seconds

	JOB_ID	COUNT(LAST_NAME)
1	SA_REP	30
2	SH_CLERK	20
3	ST_CLERK	20
4	PU_CLERK	5
5	FI_ACCOUNT	5
6	SA_MAN	5
7	IT_PROG	5
8	ST_MAN	5
9	AD_VP	2
10	PU_MAN	1
11	PR_REP	1
12	MK_REP	1
13	MK_MAN	1
14	HR_REP	1
15	FI_MGR	1
16	AD_PRES	1
17	AD_ASST	1
18	AC_ACCOUNT	1
19	AC_MGR	1

Exploring data: Using COUNT Function

Worksheet | Query Builder

```
20  ;
21
22  SELECT JOB_ID, COUNT(1)
23  FROM EMPLOYEES
24  GROUP BY JOB_ID
25  ORDER BY 2 DESC
26  ;
27
28
```

Script Output | Query... | SQL | All Rows Fetched: 19 in 0.0

	JOB_ID	COUNT(1)
1	SA_REP	30
2	SH_CLERK	20
3	ST_CLERK	20
4	PU_CLERK	5
5	FI_ACCOUNT	5
6	SA_MAN	5
7	IT_PROG	5
8	ST_MAN	5
9	AD_VP	2
10	PU_MAN	1
11	PR_REP	1
12	MK_REP	1
13	MK_MAN	1
14	HR_REP	1
15	FI_MGR	1
16	AD_PRES	1
17	AD_ASST	1
18	AC_ACCOUNT	1
19	AC_MGR	1

EMPLOYEES

Columns | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
EMPLOYEE_ID	NUMBER(6,0)	No	(null)	1	Primary key of employees table.
FIRST_NAME	VARCHAR2(20 BYTE)	Yes	(null)	2	First name of the employee. A not null column.
LAST_NAME	VARCHAR2(25 BYTE)	No	(null)	3	Last name of the employee. A not null column.
EMAIL	VARCHAR2(25 BYTE)	No	(null)	4	Email id of the employee
PHONE_NUMBER	VARCHAR2(20 BYTE)	Yes	(null)	5	Phone number of the employee; includes country
HIRE_DATE	DATE	No	(null)	6	Date when the employee started on this job. A n
JOB_ID	VARCHAR2(10 BYTE)	No	(null)	7	Current job of the employee; foreign key to job
SALARY	NUMBER(8,2)	Yes	(null)	8	Monthly salary of the employee. Must be greater
COMMISSION_PCT	NUMBER(2,2)	Yes	(null)	9	Commission percentage of the employee; Only emp
MANAGER_ID	NUMBER(6,0)	Yes	(null)	10	Manager id of the employee; has same domain as
DEPARTMENT_ID	NUMBER(4,0)	Yes	(null)	11	Department id where employee works; foreign key

Help | OK | Cancel

Using COUNT Function

Worksheet | Query Builder

```
20  ;
21
22  SELECT JOB_ID, COUNT(1), COUNT(LAST_NAME), COUNT(COMMISSION_PCT)
23  FROM EMPLOYEES
24  GROUP BY JOB_ID
25  ORDER BY 2 DESC
26  ;
27
28
```

Script Output | Query...

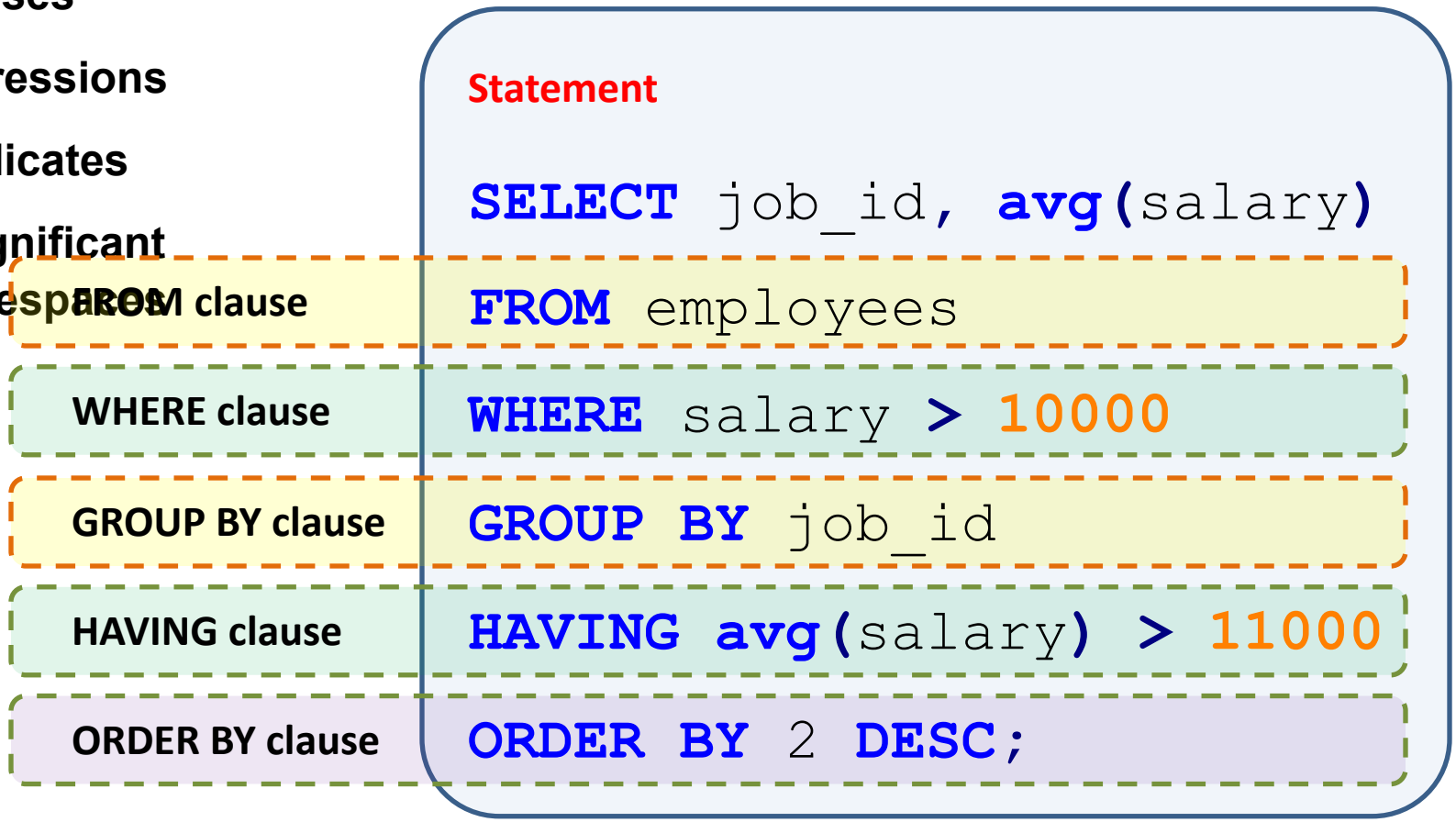
SQL | All Rows Fetched: 19 in 0.003 seconds

	JOB_ID	COUNT(1)	COUNT(LAST_NAME)	COUNT(COMMISSION_PCT)
1	SA_REP	30	30	30
2	ST_CLERK	20	20	0
3	SH_CLERK	20	20	0
4	FI_ACCOUNT	5	5	0
5	IT_PROG	5	5	0
6	ST_MAN	5	5	0
7	SA_MAN	5	5	5
8	PU_CLERK	5	5	0
9	AD_VP	2	2	0
10	MK_REP	1	1	0
11	AD_PRES	1	1	0
12	PR_REP	1	1	0
13	MK_MAN	1	1	0
14	FI_MGR	1	1	0
15	AD_ASST	1	1	0
16	PU_MAN	1	1	0
17	AC_ACCOUNT	1	1	0
18	HR_REP	1	1	0
19	AC_MGR	1	1	0

THE SELECT STATEMENT

Basic Language Elements

- Statements
- Queries
- Clauses
- Expressions
- Predicates
- Insignificant whitespaces



Tables Aliases

- Table aliases is optional mechanism to make queries easier to read, understand and maintain.
- **Aliases should be meaningful!**
- Aliases can be used with asterisk, like SELECT emp.*
- Optional AS keyword between table name and its alias throws error in Oracle (non-standard behavior).

```
SELECT emp.job_id, avg(emp.salary)
FROM employees emp
WHERE emp.salary > 10000
GROUP BY emp.job_id
HAVING avg(emp.salary) > 11000
ORDER BY avg(emp.salary) DESC;
```

	JOB_ID	AVG(EMP.SALARY)
1	AD_PRES	24000
2	AD_VP	17000
3	MK_MAN	13000
4	SA_MAN	12200
5	AC_MGR	12000
6	FI_MGR	12000

Field Aliases

Naming Rules:

- Must not exceed 30 characters.
- First character must be a letter
- The rest can be any combination of letters, numerals, dollar signs ($), pound signs (#), and underscores (_).
- Identifier enclosed by double quotation marks (") can contain any combination of legal characters, including spaces but excluding quotation marks.
- Identifiers are not case sensitive except within double quotation marks.

```
SELECT
  emp.job_id AS "Group by job",
  avg(emp.salary) "Salary, AVG"
FROM employees "EMP"
WHERE "EMP".salary > 10000
GROUP BY emp.job_id
HAVING avg(emp.salary) > 11000
ORDER BY -"Salary, AVG";
```

	Group by job	Salary, AVG
1	AD_PRES	24000
2	AD_VP	17000
3	MK_MAN	13000
4	SA_MAN	12200
5	AC_MGR	12000
6	FI_MGR	12000

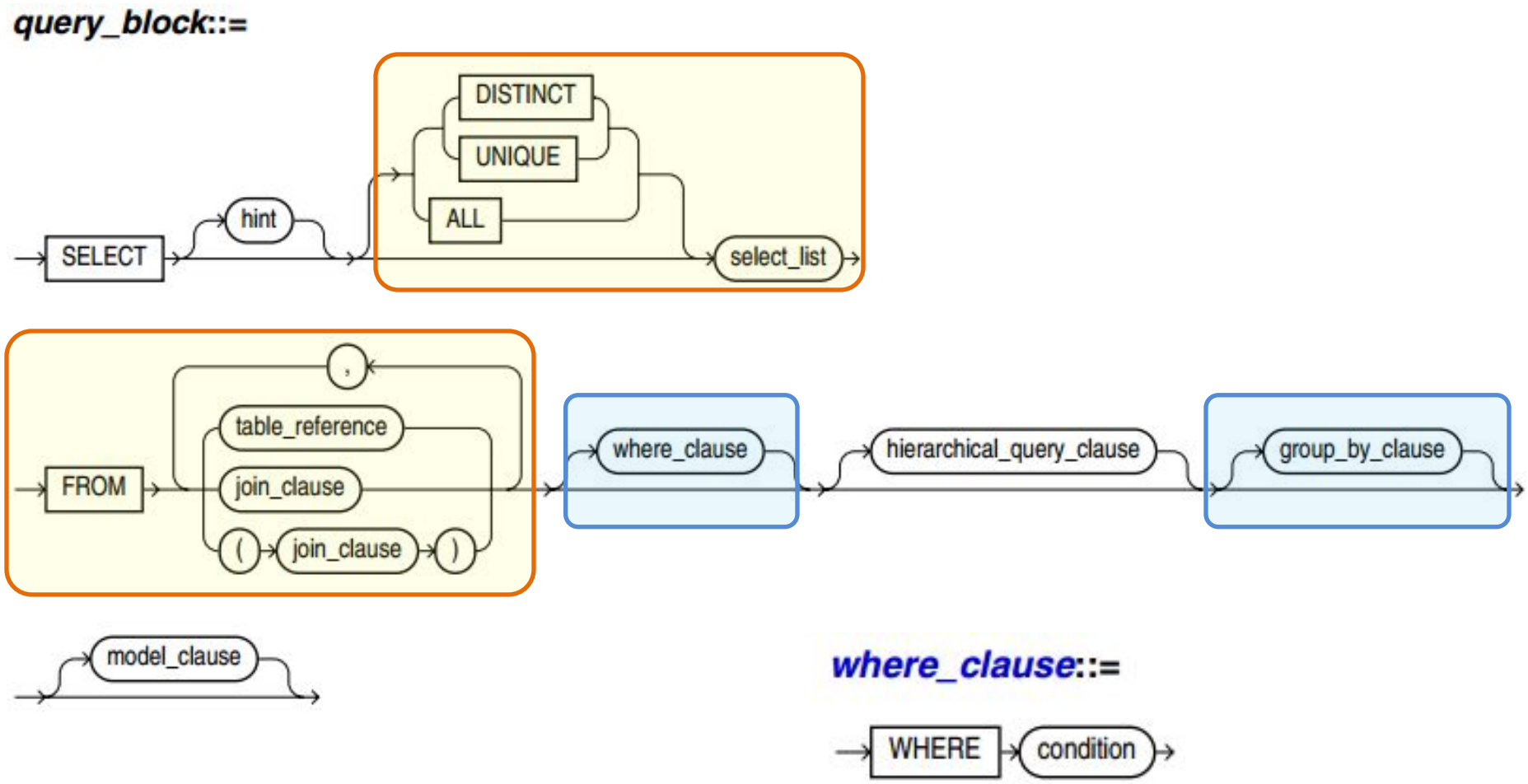
ORDER BY clause (NULLs Ordering)

- ASC | DESC
  Specify the ordering sequence. ASC is the default.
- NULLS FIRST | NULLS LAST
  Specify whether returned rows containing nulls should appear first or last in the ordering sequence.
- NULLS LAST is the default for ascending order, and NULLS FIRST is the default for descending order.

```
SELECT e.job_id AS "Group by job",
  avg(e.commission_pct) "Commission, AVG"
FROM employees e
WHERE "E".salary > 9000
GROUP BY e.job_id
--HAVING min(e.commission_pct) > 0
ORDER BY 2 DESC NULLS LAST;
```

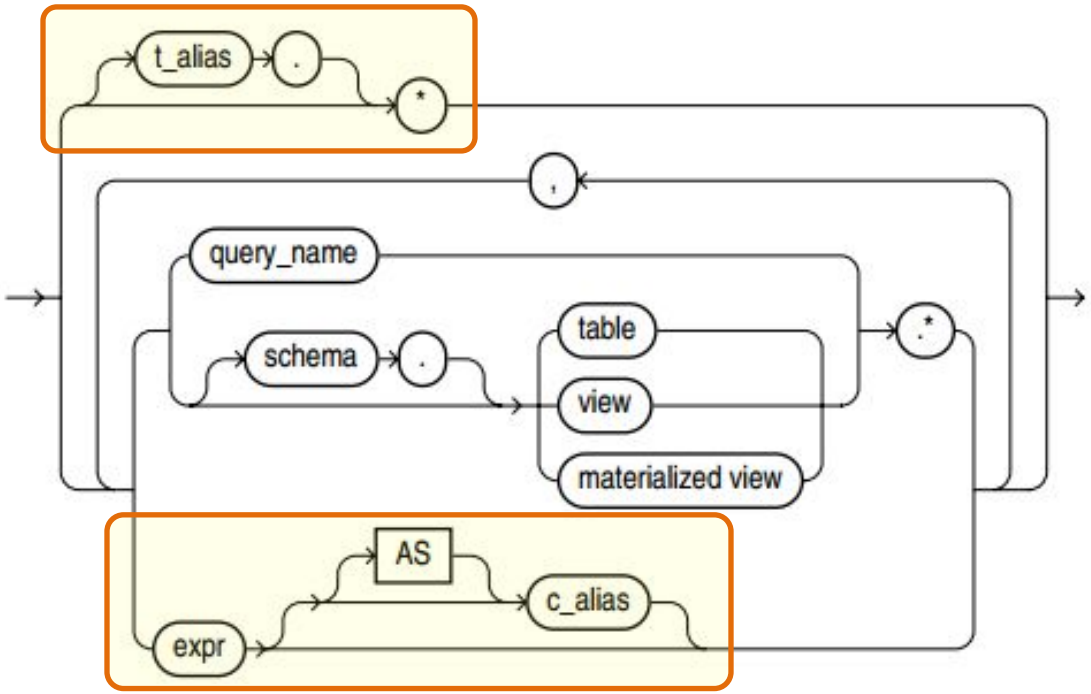
	Group by job	Commission, AVG
1	SA_MAN	0.3
2	SA_REP	0.26
3	PU_MAN	(null)
4	AD_VP	(null)
5	FI_MGR	(null)
6	MK_MAN	(null)
7	PR_REP	(null)
8	AD_PRES	(null)
9	AC_MGR	(null)

Oracle Query Block Structure and WHERE Clause



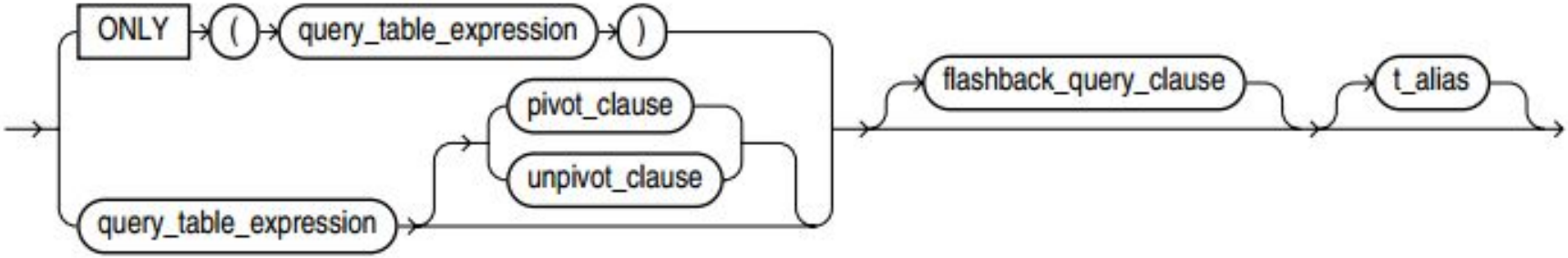
SELECT Columns List

***select_list*::=**

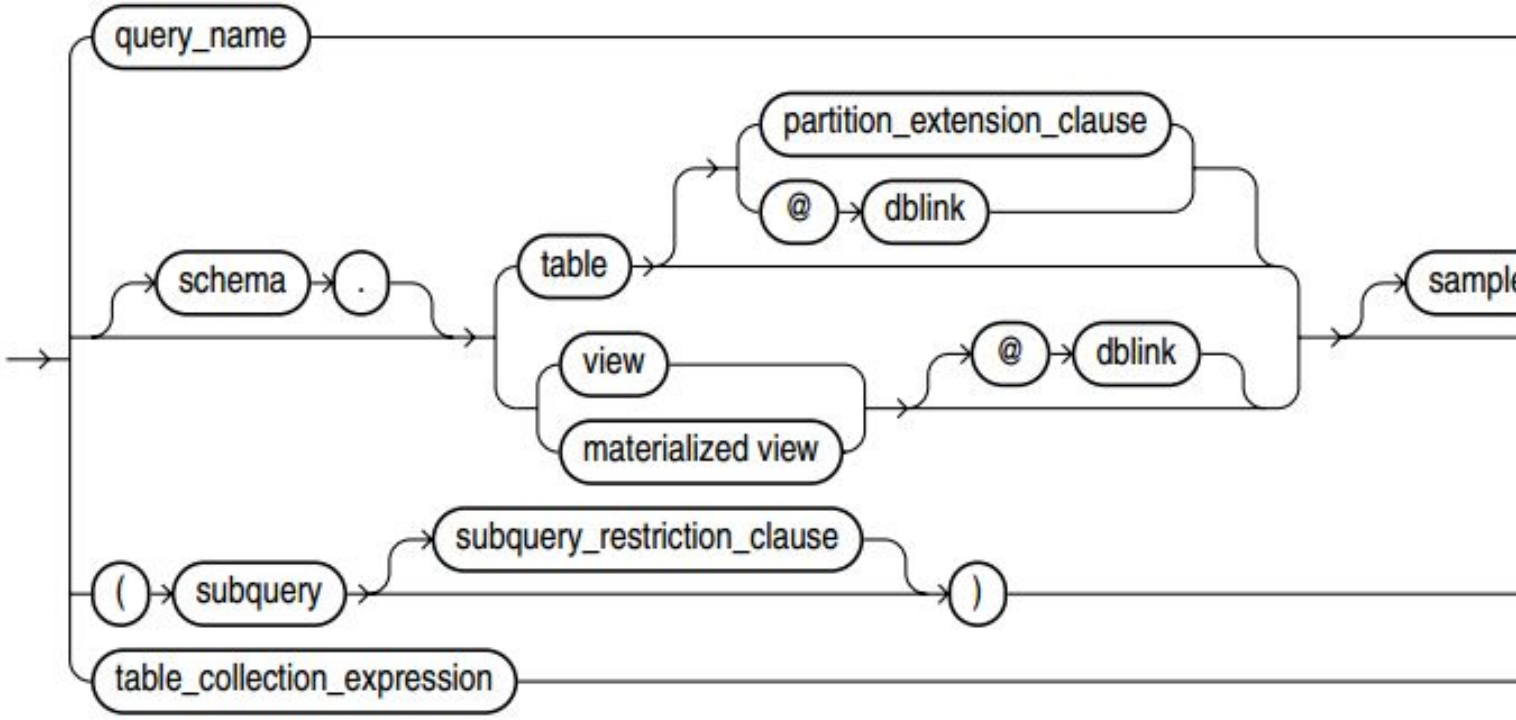


Tables References (simplified FROM clause)

***table_reference*::=**

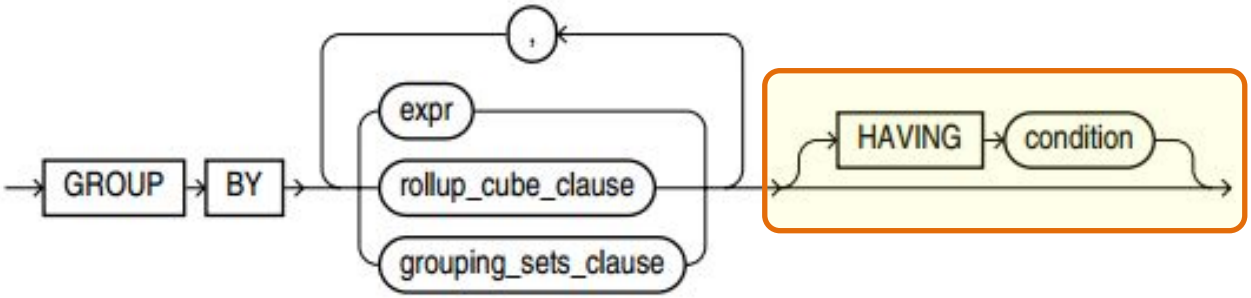


***query_table_expression*::=**

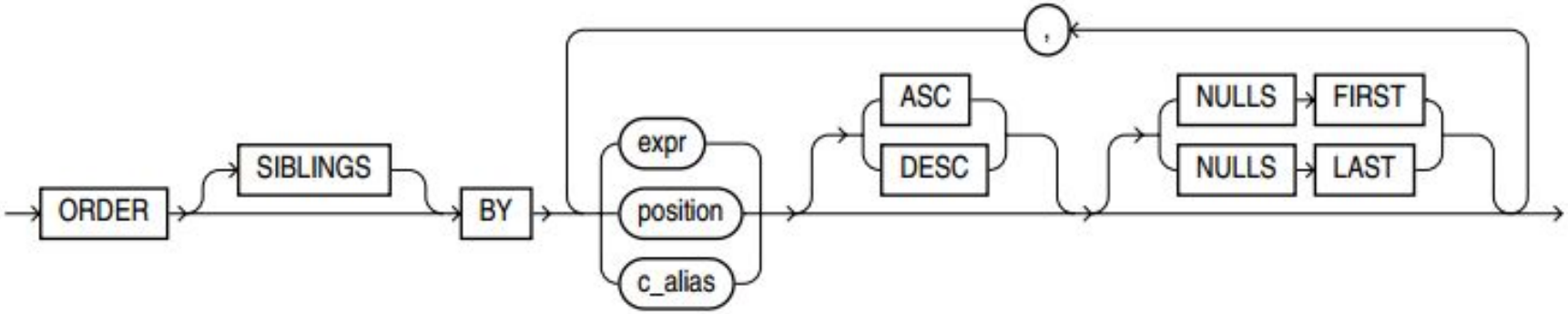


GROUP BY and HAVING clauses, ORDER BY clause

***group_by_clause*::=**

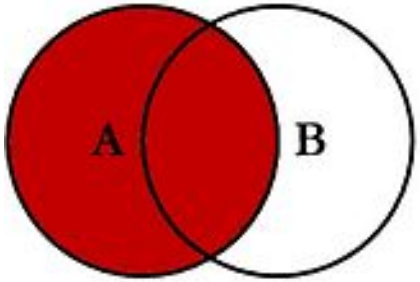


***order_by_clause*::=**

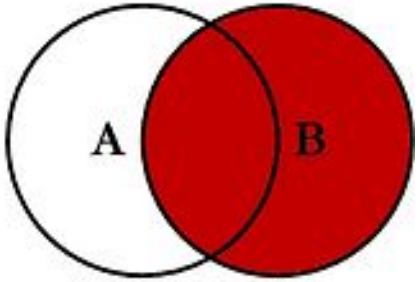


JOIN TABLES

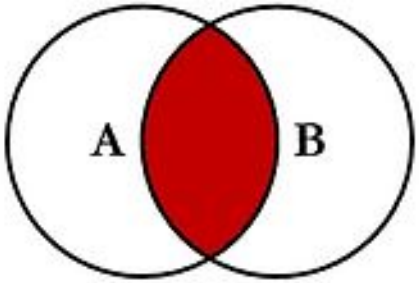
SQL Joins



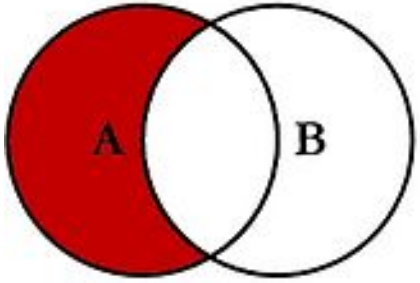

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



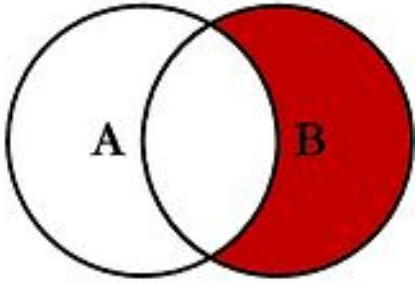

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



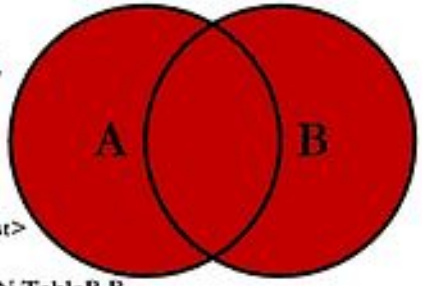

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



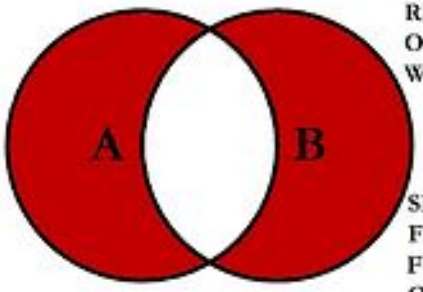

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

SQL Joins Classification

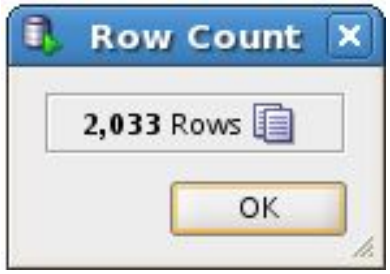
- Inner join
  - Equi-join
    - Natural join
- Outer joins
  - Left outer join
  - Right outer join
  - Full outer join
- Cross join
- Self-join

Qualified joins

Simple Join Example (cross join Employees and Jobs)

```
SELECT emp.first_name, emp.last_name,
  emp.job_id, emp.salary, jb.*
FROM employees emp, jobs jb;
```

```
SELECT emp.first_name, emp.last_name,
  emp.job_id, emp.salary, jb.*
FROM employees emp CROSS JOIN jobs jb;
```



	FIRST_NAME	LAST_NAME	JOB_ID	SALARY	JOB_ID_1	JOB_TITLE	MIN_SALARY	MAX_SALARY
1	Steven	King	AD_PRES	24000	AD_PRES	President	20000	40000
2	Neena	Kochhar	AD_VP	17000	AD_PRES	President	20000	40000
3	Lex	De Haan	AD_VP	17000	AD_PRES	President	20000	40000
4	Alexander	Hunold	IT_PROG	9000	AD_PRES	President	20000	40000
5	Bruce	Ernst	IT_PROG	6000	AD_PRES	President	20000	40000
6	David	Austin	IT_PROG	4800	AD_PRES	President	20000	40000
7	Valli	Pataballa	IT_PROG	4800	AD_PRES	President	20000	40000
8	Diana	Lorentz	IT_PROG	4200	AD_PRES	President	20000	40000
9	Nancy	Greenberg	FI_MGR	12000	AD_PRES	President	20000	40000
10	Daniel	Faviet	FI_ACCOUNT	9000	AD_PRES	President	20000	40000
11	John	Chen	FI_ACCOUNT	8200	AD_PRES	President	20000	40000

Prove Cross Join

```
SELECT count(*) AS cnt
FROM employees emp, jobs jb;
```

```
CNT
---
2033
```

```
SELECT count(*) AS cnt
FROM employees emp CROSS JOIN jobs jb;
```

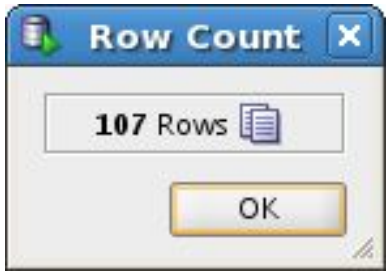
```
CNT
---
2033
```

```
SELECT
  (SELECT count(*) FROM employees emp)
  * (SELECT count(*) FROM jobs jb) cnt
FROM dual;
```

```
CNT
---
2033
```

Reducing Cartesian Product to get meaningful result

```
SELECT emp.first_name,  emp.last_name,
  emp.job_id, emp.salary, jb.*
FROM employees emp, jobs jb
WHERE emp.job_id = jb.job_id;
```

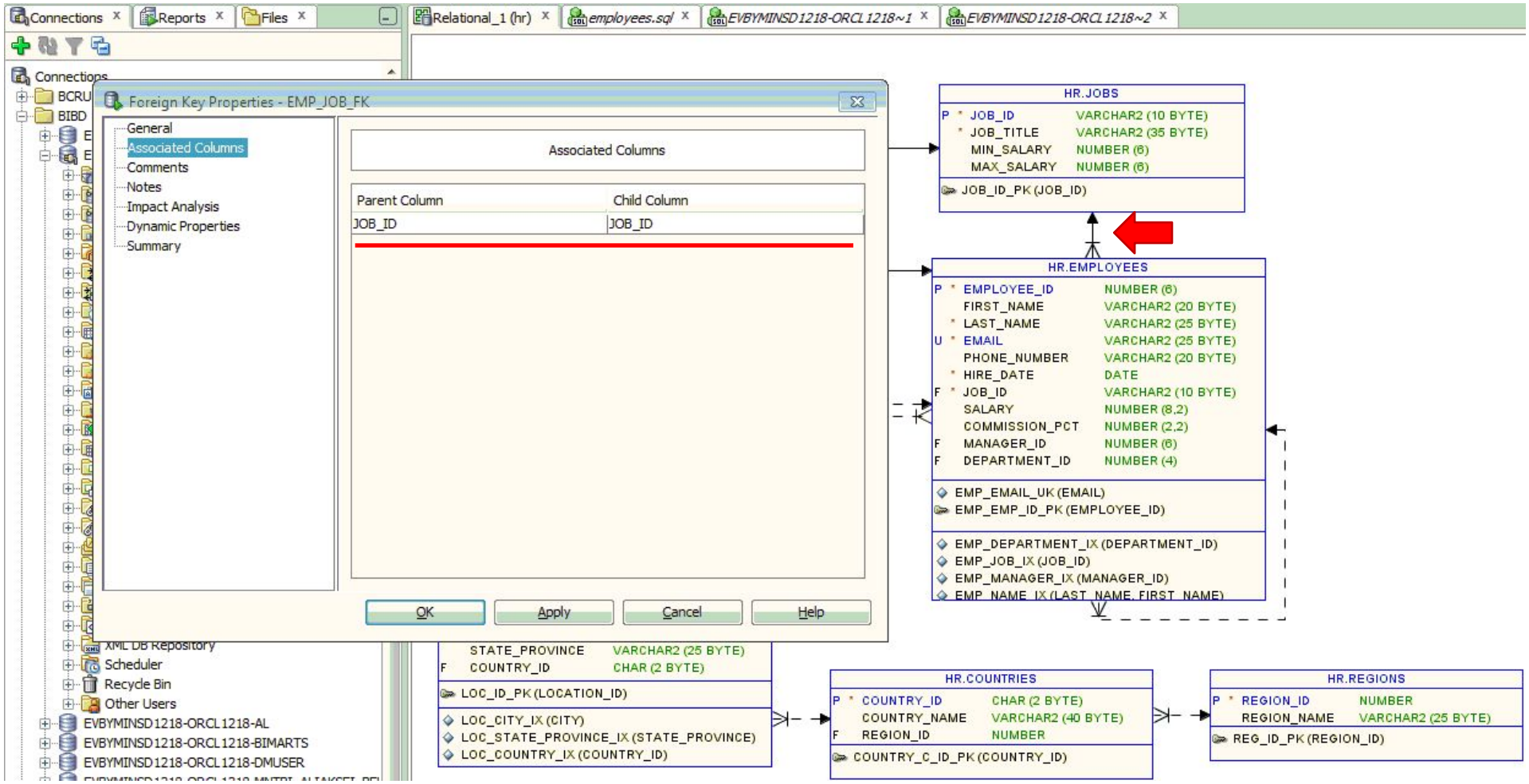



```
SELECT emp.first_name, emp.last_name,
  emp.job_id, emp.salary, jb.*
FROM employees emp CROSS JOIN jobs jb
WHERE emp.job_id = jb.job_id;
```

Senseless syntax

	FIRST_NAME	LAST_NAME	JOB_ID	SALARY	JOB_ID_1	JOB_TITLE	MIN_SALARY	MAX_SALARY
1	Steven	King	AD_PRES	24000	AD_PRES	President	20000	40000
2	Neena	Kochhar	AD_VP	17000	AD_VP	Administration Vice...	15000	30000
3	Lex	De Haan	AD_VP	17000	AD_VP	Administration Vice...	15000	30000
4	Alexander	Hunold	IT_PROG	9000	IT_PROG	Programmer	4000	10000
5	Bruce	Ernst	IT_PROG	6000	IT_PROG	Programmer	4000	10000
6	David	Austin	IT_PROG	4800	IT_PROG	Programmer	4000	10000
7	Valli	Pataballa	IT_PROG	4800	IT_PROG	Programmer	4000	10000
8	Diana	Lorentz	IT_PROG	4200	IT_PROG	Programmer	4000	10000
9	Nancy	Greenberg	FI_MGR	12000	FI_MGR	Finance Manager	8200	16000
10	Daniel	Faviet	FI_ACCOUNT	9000	FI_ACCOUNT	Accountant	4200	9000

Check Your Join (Using foreign keys)



Check Your Join (Nullable fields)

```
DESCRIBE employees
```

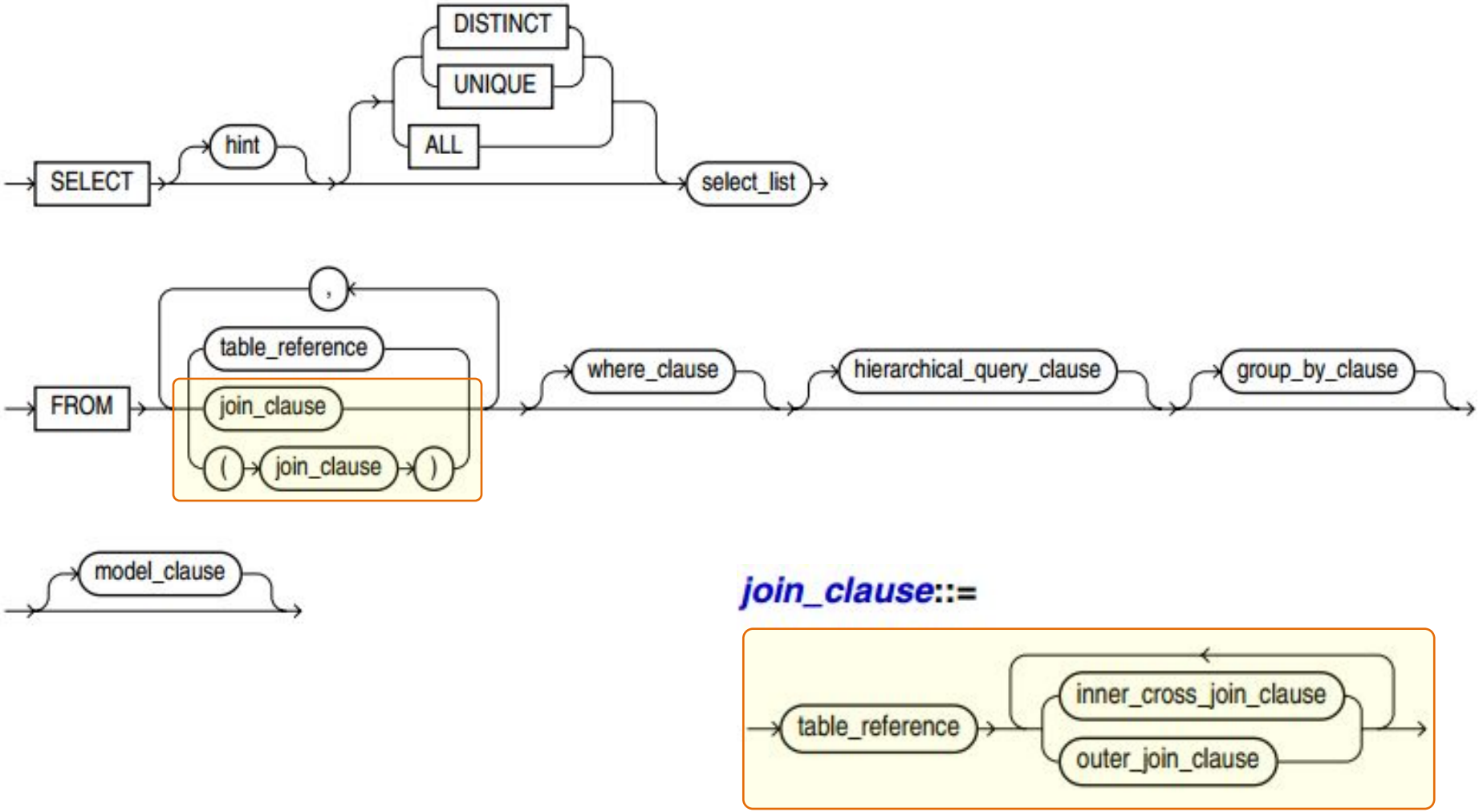
```
Name            Null     Type
--------------- -------- ------------
EMPLOYEE_ID     NOT NULL NUMBER(6)
FIRST_NAME               VARCHAR2(20)
LAST_NAME       NOT NULL VARCHAR2(25)
EMAIL           NOT NULL VARCHAR2(25)
PHONE_NUMBER             VARCHAR2(20)
HIRE_DATE       NOT NULL DATE
JOB_ID          NOT NULL VARCHAR2(10)
SALARY                   NUMBER(8,2)
COMMISSION_PCT           NUMBER(2,2)
MANAGER_ID               NUMBER(6)
DEPARTMENT_ID            NUMBER(4)
```

```
DESC jobs
```

```
Name       Null     Type
---------- -------- ------------
JOB_ID     NOT NULL VARCHAR2(10)
JOB_TITLE  NOT NULL VARCHAR2(35)
MIN_SALARY          NUMBER(6)
MAX_SALARY          NUMBER(6)
```

Join Syntax

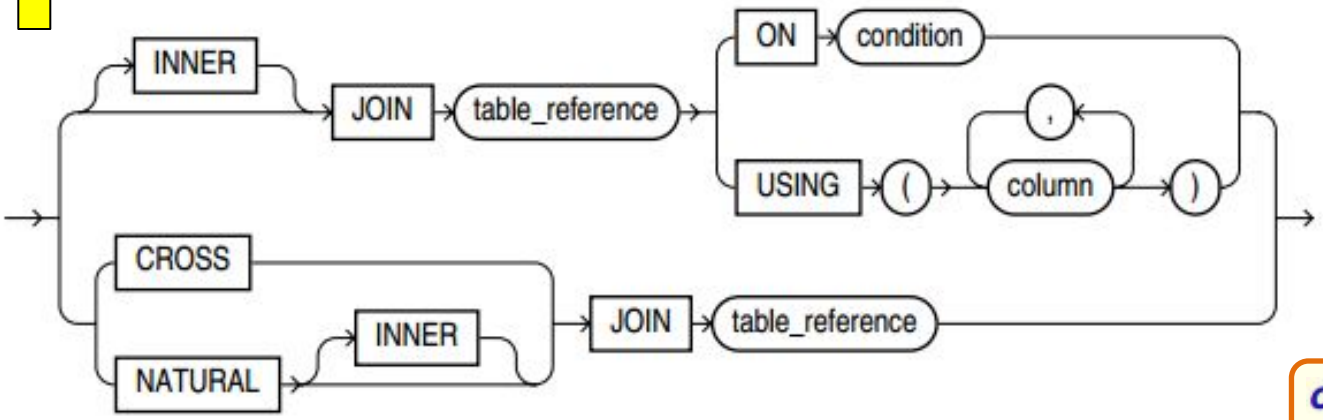
query_block::=



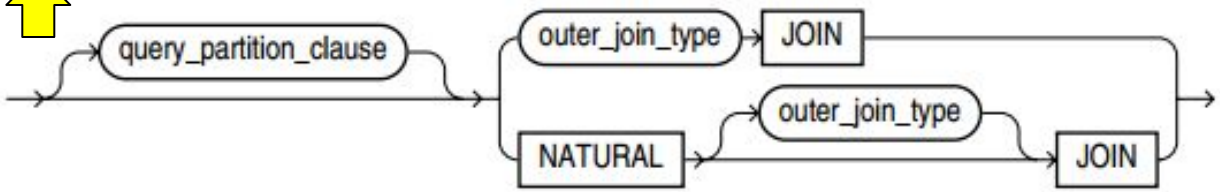
join_clause::=

Inner / Outer / Cross Joins Syntax

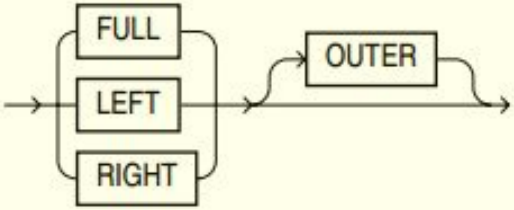
inner_cross_join_clause::=

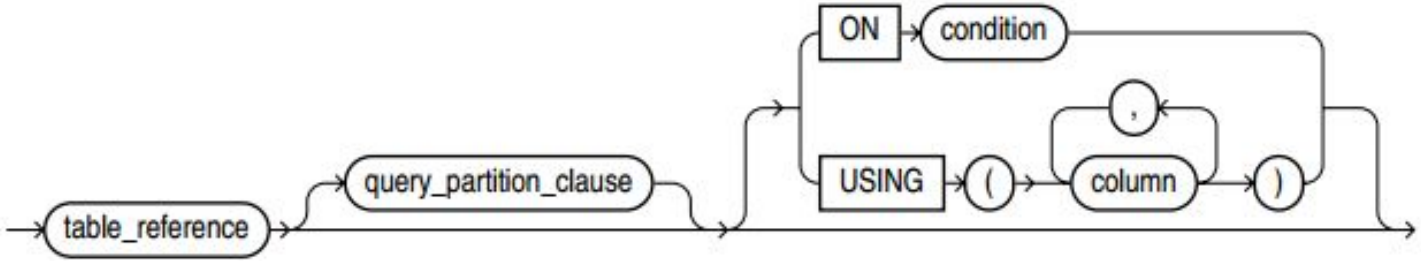


outer_join_clause::=



outer_join_type::=



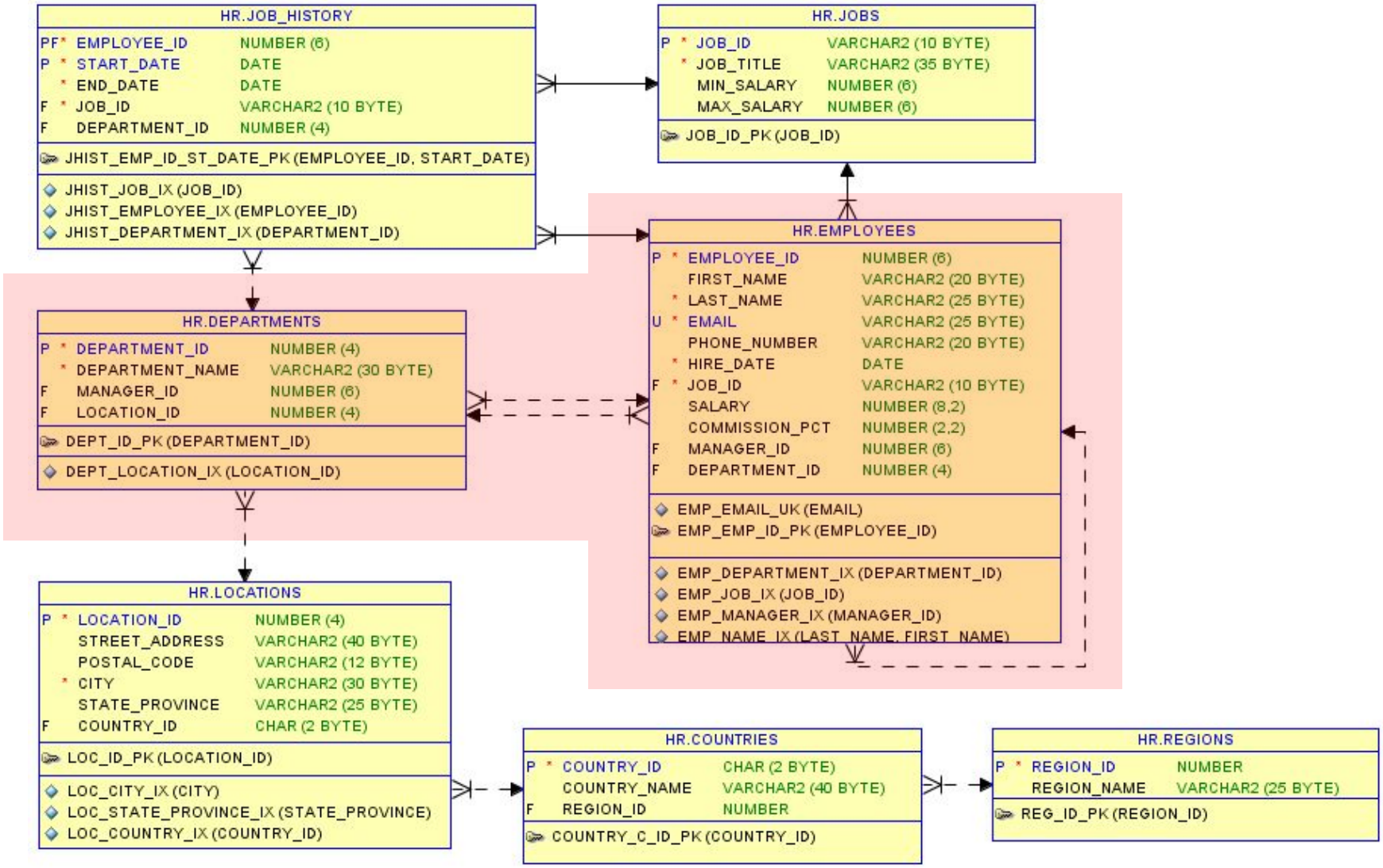


Inner Equi-joins

```
SELECT emp.first_name, emp.last_name, emp.salary, jb.*
FROM employees emp, jobs jb
WHERE emp.job_id = jb.job_id;

SELECT emp.first_name, emp.last_name, emp.salary,
  job_id, jb.job_title, jb.min_salary, jb.max_salary
FROM employees emp NATURAL JOIN jobs jb;

SELECT emp.first_name, emp.last_name, emp.salary,
  job_id, jb.job_title, jb.min_salary, jb.max_salary
FROM employees emp JOIN jobs jb USING(job_id);

SELECT emp.first_name, emp.last_name, emp.salary, jb.*
FROM employees emp JOIN jobs jb ON emp.job_id=jb.job_id;
```

	FIRST_NAME	LAST_NAME	SALARY	JOB_ID	JOB_TITLE	MIN_SALARY	MAX_SALARY
1	Steven	King	24000	AD_PRES	President	20000	40000
2	Neena	Kochhar	17000	AD_VP	Administration Vice President	15000	30000
3	Lex	De Haan	17000	AD_VP	Administration Vice President	15000	30000
4	Alexander	Hunold	9000	IT_PROG	Programmer	4000	10000
5	Bruce	Ernst	6000	IT_PROG	Programmer	4000	10000
6	David	Austin	4800	IT_PROG	Programmer	4000	10000
7	Valli	Pataballa	4800	IT_PROG	Programmer	4000	10000

Outer Equi-joins



Left Outer Equi-joins

```
SELECT emp.first_name, emp.last_name, emp.salary, dept.department_name
FROM employees emp, departments dept
WHERE emp.department_id = dept.department_id(+)
ORDER BY dept.department_name NULLS FIRST;
```

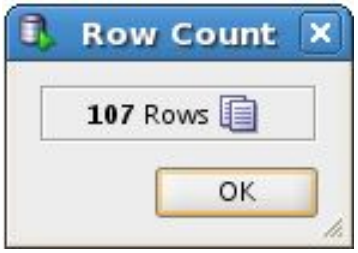
Old Oracle's syntax

```
SELECT emp.first_name, emp.last_name, emp.salary, dept.department_name
FROM employees emp NATURAL LEFT OUTER JOIN departments dept
ORDER BY dept.department_name NULLS FIRST;
```

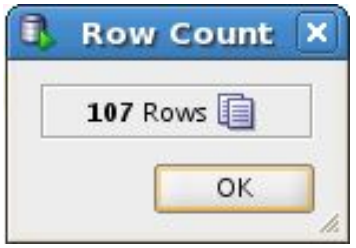
```
SELECT emp.first_name, emp.last_name, emp.salary, dept.department_name
FROM employees emp LEFT OUTER JOIN departments dept USING (department_id)
ORDER BY dept.department_name NULLS FIRST;
```

```
SELECT emp.first_name, emp.last_name, emp.salary, dept.department_name
FROM employees emp LEFT OUTER JOIN departments dept
  ON (emp.department_id = dept.department_id)
ORDER BY dept.department_name NULLS FIRST;
```

	FIRST_NAME	LAST_NAME	SALARY	DEPARTMENT_NAME
1	Kimberely	Grant	7000	(null)
2	William	Gietz	8300	Accounting
3	Shelley	Higgins	12000	Accounting
4	Jennifer	Whalen	4400	Administration
5	Steven	King	24000	Executive
6	Neena	Kochhar	17000	Executive



Typical Mistake with NATURAL JOIN




```
SELECT emp.first_name, emp.last_name,
  emp.salary, dept.department_name,
  department_id, manager_id
FROM employees emp NATURAL LEFT JOIN departments dept;
```

	FIRST_NAME	LAST_NAME	SALARY	DEPARTMENT_NAME	DEPARTMENT_ID	MANAGER_ID
1	Steven	King	24000	(null)	90	(null)
2	Neena	Kochhar	17000	Executive	90	100
3	Lex	De Haan	17000	Executive	90	100
4	Alexander	Hunold	9000	(null)	60	102
5	Bruce	Ernst	6000	IT	60	103
6	David	Austin	4800	IT	60	103
7	Valli	Pataballa	4800	IT	60	103
8	Diana	Lorentz	4200	IT	60	103
9	Nancy	Greenberg	12000	(null)	100	101
10	Daniel	Faviet	9000	Finance	100	108
11	John	Chen	8200	Finance	100	108
12	Ismael	Sciarra	7700	Finance	100	108

```
SELECT emp.first_name, emp.last_name,
  emp.salary, dept.department_name,
  department_id, manager_id
FROM employees emp LEFT OUTER JOIN departments dept
  USING (department_id, manager_id);
```

Do you really want this?

Right Outer Equi-joins

```
SELECT dept.department_name, max(emp.salary)
FROM employees emp, departments dept
WHERE emp.department_id(+) = dept.department_id
GROUP BY dept.department_name
HAVING count(emp.employee_id) > 0
ORDER BY dept.department_name NULLS FIRST;
```

Old Oracle's syntax

```
SELECT dept.department_name, max(emp.salary)
FROM employees emp NATURAL RIGHT JOIN departments dept
GROUP BY dept.department_name
HAVING count(emp.employee_id) > 0
ORDER BY dept.department_name NULLS FIRST;
```

Do you really want this?

```
SELECT dept.department_name, max(emp.salary)
FROM employees emp RIGHT OUTER JOIN departments dept
  USING (department_id)
GROUP BY dept.department_name
HAVING count(emp.employee_id) > 0
ORDER BY dept.department_name NULLS FIRST;
```

```
SELECT dept.department_name, max(emp.salary)
FROM employees emp RIGHT OUTER JOIN departments dept
  ON (emp.department_id = dept.department_id)
GROUP BY dept.department_name
HAVING count(emp.employee_id) > 0
ORDER BY dept.department_name NULLS FIRST;
```

	DEPARTMENT_NAME	MAX(EMP.SALARY)
1	Accounting	12000
2	Administration	4400
3	Executive	24000
4	Finance	12000
5	Human Resources	6500
6	IT	9000
7	Marketing	13000
8	Public Relations	10000
9	Purchasing	11000
10	Sales	14000
11	Shipping	8200

Full Outer Equi-joins

```
SELECT dept.department_name, max(emp.salary)
FROM employees emp, departments dept
WHERE emp.department_id(+) = ept.department_id(+)
GROUP BY dept.department_name
HAVING count(emp.employee_id) > 0
ORDER BY dept.department_name NULLS FIRST;
```

ORA-01468: a predicate may reference only one outer-joined table

```
SELECT dept.department_name, max(emp.salary)
FROM employees emp NATURAL FULL JOIN departments dept
GROUP BY dept.department_name
HAVING count(emp.employee_id) > 0
ORDER BY dept.department_name NULLS FIRST;
```

```
SELECT dept.department_name, max(emp.salary)
FROM employees emp FULL OUTER JOIN departments dept
  USING (department_id)
GROUP BY dept.department_name
HAVING count(emp.employee_id) > 0
ORDER BY dept.department_name NULLS FIRST;
```

```
SELECT dept.department_name, max(emp.salary)
FROM employees emp FULL OUTER JOIN departments dept
  ON (emp.department_id = dept.department_id)
GROUP BY dept.department_name
HAVING count(emp.employee_id) > 0
ORDER BY dept.department_name NULLS FIRST;
```

	DEPARTMENT_NAME	MAX(EMP.SALARY)
1	(null)	7000
2	Accounting	12000
3	Administration	4400
4	Executive	24000
5	Finance	12000
6	Human Resources	6500
7	IT	9000
8	Marketing	13000
9	Public Relations	10000
10	Purchasing	11000
11	Sales	14000
12	Shipping	8200

Self-join

```
SELECT emp.first_name, emp.last_name, emp.salary,
  mng.first_name manager_first_name, mng.last_name manager_last_name
FROM employees emp LEFT JOIN employees mng
  ON emp.manager_id = mng.employee_id;
```

```
SELECT emp.first_name, emp.last_name, emp.salary,
  mng.first_name manager_first_name, mng.last_name manager_last_name
FROM employees emp, employees mng
WHERE emp.manager_id = mng.employee_id(+);
```

	FIRST_NAME	LAST_NAME	SALARY	MANAGER_FIRST_NAME	MANAGER_LAST_NAME
1	Steven	King	24000	(null)	(null)
2	Neena	Kochhar	17000	Steven	King
3	Lex	De Haan	17000	Steven	King
4	Alexander	Hunold	9000	Lex	De Haan
5	Bruce	Ernst	6000	Alexander	Hunold
6	David	Austin	4800	Alexander	Hunold
7	Valli	Pataballa	4800	Alexander	Hunold
8	Diana	Lorentz	4200	Alexander	Hunold
9	Nancy	Greenberg	12000	Neena	Kochhar
10	Daniel	Faviet	9000	Nancy	Greenberg

Complex Join Example

```
SELECT dept.department_name "Dept",
  dept_mng.first_name || ' ' || dept_mng.last_name "Dept Manager",
  emp.first_name || ' ' || emp.last_name "Employee",
  emp_mng.first_name || ' ' || emp_mng.last_name "Emp Manager"
FROM departments dept
  LEFT OUTER JOIN employees dept_mng
    ON (dept.manager_id = dept_mng.employee_id)
  FULL OUTER JOIN employees emp
    ON (emp.department_id = dept.department_id)
  LEFT OUTER JOIN employees emp_mng
    ON (emp.manager_id=emp_mng.employee_id)
ORDER BY 1 NULLS FIRST, 2, 3, 4;
```

Resulting dataset contains 123 rows:

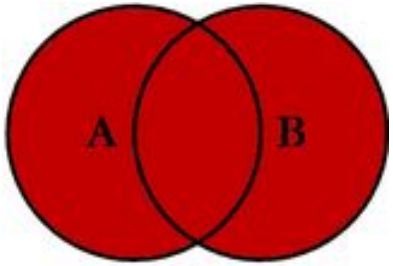
- 107 employees
- 16 empty departments

	Dept	Dept Manager	Employee	Emp Manager
1	(null)		Kimberely Grant	Eleni Zlotkey
2	Accounting	Shelley Higgins	Shelley Higgins	Neena Kochhar
3	Accounting	Shelley Higgins	William Gietz	Shelley Higgins
4	Administration	Jennifer Whalen	Jennifer Whalen	Neena Kochhar
5	Benefits			
6	Construction			
7	Contracting			
8	Control And...			
9	Corporate Tax			
10	Executive	Steven King	Lex De Haan	Steven King
11	Executive	Steven King	Neena Kochhar	Steven King
12	Executive	Steven King	Steven King	
13	Finance	Nancy Greenberg	Daniel Faviet	Nancy Greenberg

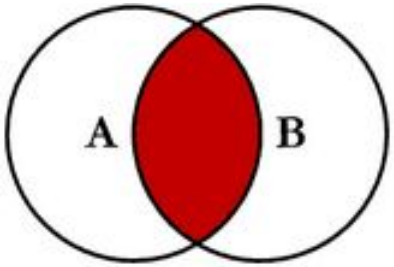
SET OPERATIONS

Set Operations

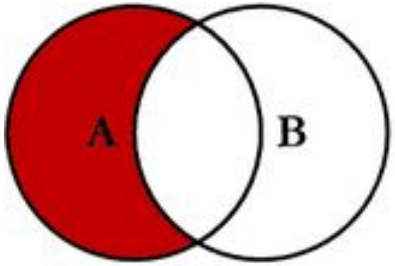
UNION



INTERSECT



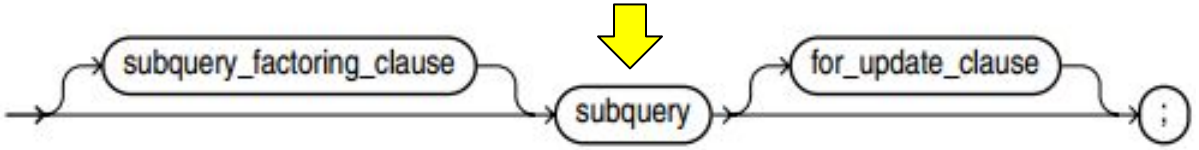
EXCEPT



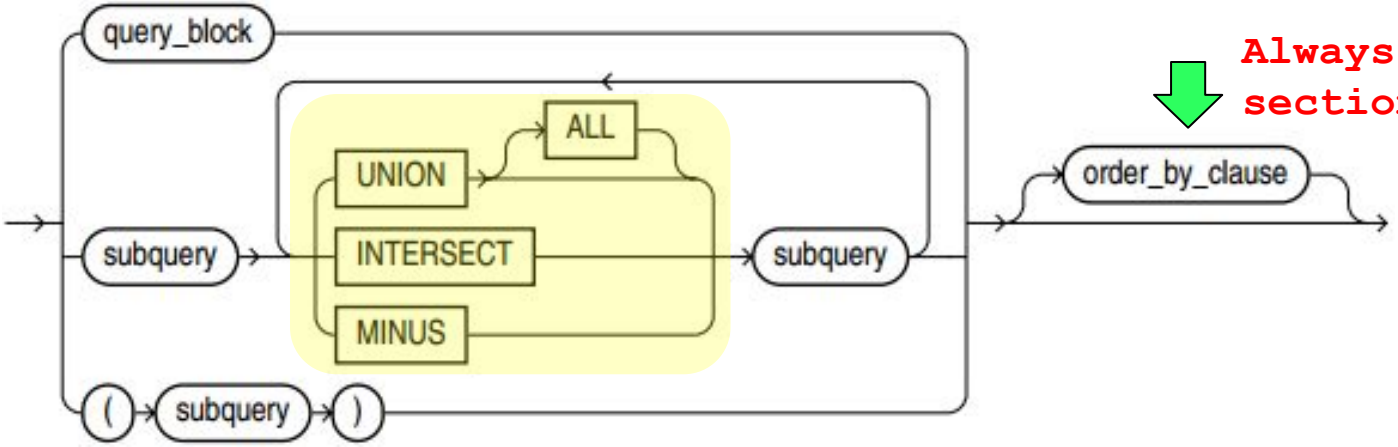
Operation	ANSI Standard	Oracle
UNION	UNION ALL	UNION ALL
	UNION DISTINCT	UNION
INTERSECT	INTERSECT ALL	
	INTERSECT DISTINCT	INTERSECT
EXCEPT	EXCEPT ALL	
	EXCEPT DISTINCT	MINUS

Set Operations Syntax

select::=

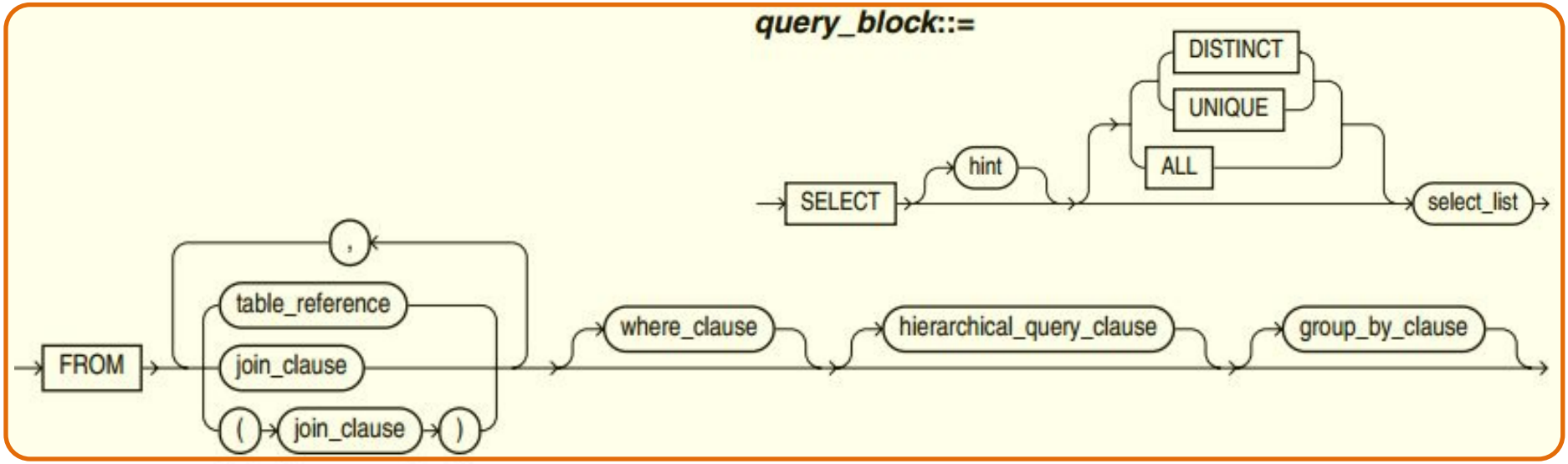


subquery::=



Always the last section

query_block::=



Union Operation

```
SELECT dept.department_name,
  max(emp.salary)
FROM employees emp, departments dept
WHERE
  emp.department_id(+) = dept.department_id
GROUP BY dept.department_name
HAVING count(emp.employee_id) > 0
UNION
SELECT dept.department_name, max(emp.salary)
FROM employees emp, departments dept
WHERE emp.department_id = dept.department_id(+)
GROUP BY dept.department_name
HAVING count(emp.employee_id) > 0
ORDER BY 1 NULLS FIRST;

SELECT dept.department_name, max(emp.salary)
FROM employees emp, departments dept
WHERE emp.department_id(+) = dept.department_id
GROUP BY dept.department_name
HAVING count(emp.employee_id) > 0
UNION
SELECT NULL, max(salary)
FROM employees emp
WHERE department_id IS NULL
ORDER BY 1 NULLS FIRST;
```

```
SELECT dept.department_name,
max(emp.salary)
FROM employees emp
  FULL OUTER JOIN departments dept
    USING (department_id)
GROUP BY dept.department_name
HAVING count(emp.employee_id) > 0
ORDER BY dept.department_name NULLS FIRST;
```

	DEPARTMENT_NAME	MAX(EMP.SALARY)
1	(null)	7000
2	Accounting	12000
3	Administration	4400
4	Executive	24000
5	Finance	12000
6	Human Resources	6500
7	IT	9000
8	Marketing	13000
9	Public Relations	10000
10	Purchasing	11000
11	Sales	14000
12	Shipping	8200

Minus Operation (Check datasets equivalence)

```
(
  SELECT dept.department_name, max(emp.salary)
  FROM employees emp FULL OUTER JOIN departments dept
    USING (department_id)
  GROUP BY dept.department_name
  HAVING count(emp.employee_id) > 0
)
MINUS
(
  SELECT dept.department_name, max(emp.salary)
  FROM employees emp, departments dept
  WHERE emp.department_id(+) = dept.department_id
  GROUP BY dept.department_name
  HAVING count(emp.employee_id) > 0
  UNION
  SELECT dept.department_name, max(emp.salary)
  FROM employees emp, departments dept
  WHERE emp.department_id = dept.department_id(+)
  GROUP BY dept.department_name
  HAVING count(emp.employee_id) > 0
);
```

Full Outer Join

Right Outer Join
Union
Left Outer join

DEPARTMENT_NAME	MAX(EMP.SALARY)

Minus Operation (Check datasets equivalence)

```
(
  SELECT dept.department_name, max(emp.salary)
  FROM employees emp, departments dept
  WHERE emp.department_id(+) = dept.department_id
  GROUP BY dept.department_name
  HAVING count(emp.employee_id) > 0
  UNION
  SELECT dept.department_name, max(emp.salary)
  FROM employees emp, departments dept
  WHERE emp.department_id = dept.department_id(+)
  GROUP BY dept.department_name
  HAVING count(emp.employee_id) > 0
)
MINUS
(
  SELECT dept.department_name, max(emp.salary)
  FROM employees emp FULL OUTER JOIN departments dept
    USING (department_id)
  GROUP BY dept.department_name
  HAVING count(emp.employee_id) > 0
);
```

Right Outer Join
Union
Left Outer join

Full Outer Join

DEPARTMENT_NAME	MAX(EMP.SALARY)

Intersect Operation

```
SELECT dept.department_name
FROM employees emp, departments dept
WHERE emp.department_id(+) = dept.department_id
GROUP BY dept.department_name
HAVING count(emp.employee_id) > 3
INTERSECT
SELECT dept.department_name
FROM employees emp, departments dept
WHERE emp.department_id(+) = dept.department_id
GROUP BY dept.department_name
HAVING MAX(emp.salary) > 9000;
```

	DEPARTMENT_NAME
1	Finance
2	Purchasing
3	Sales

```
SELECT dept.department_name
FROM employees emp, departments dept
WHERE emp.department_id(+) = dept.department_id
GROUP BY dept.department_name
HAVING count(emp.employee_id) > 3 and max(emp.salary) > 9000;
```

UNION ALL Operation

```
SELECT 'Dept' AS "Dept/Job",
  dept.department_name "Name",
  avg(emp.salary) "Avg Salary"
FROM employees emp
  JOIN departments dept
    USING (department_id)
GROUP BY department_id, dept.department_name
HAVING avg(emp.salary) > 9000
UNION ALL
SELECT 'Job',
  jb.job_title,
  avg(emp.salary)
FROM employees emp
  JOIN jobs jb
    USING (job_id)
GROUP BY job_id, jb.job_title
HAVING avg(emp.salary) > 9000
ORDER BY 1, 2, 3;
```

	Dept/Job	Name	Avg Salary
1	Dept	Accounting	10150
2	Dept	Executive	19333.33...
3	Dept	Marketing	9500
4	Dept	Public Relations	10000
5	Job	Accounting Manager	12000
6	Job	Administration Vice Pr...	17000
7	Job	Finance Manager	12000
8	Job	Marketing Manager	13000
9	Job	President	24000
10	Job	Public Relations Repre...	10000
11	Job	Purchasing Manager	11000
12	Job	Sales Manager	12200

PSEUDOCOLUMNS

Pseudocolumns

Oracle Pseudocolumns Overview

- Hierarchical Query Pseudocolumns
- Sequence Pseudocolumns
- Version Query Pseudocolumns
- COLUMN_VALUE Pseudocolumn
- OBJECT_ID Pseudocolumn
- OBJECT_VALUE Pseudocolumn
- ORA_ROWSCN Pseudocolumn
- **ROWID Pseudocolumn**
- **ROWNUM Pseudocolumn**
- XMLDATA Pseudocolumn

ROWNUM Pseudocolumn

```
SELECT ROWNUM, employee_id,
  first_name, last_name
FROM employees;
```

	ROWNUM	EMPLOYEE_ID	FIRST_NAME	LAST_NAME
1	1	174	Ellen	Abel
2	2	166	Sundar	Ande
3	3	130	Mozhe	Atkinson
4	4	105	David	Austin
5	5	204	Hermann	Baer
6	6	116	Shelli	Baida
7	7	167	Amit	Banda
8	8	172	Elizabeth	Bates
9	9	192	Sarah	Bell
10	10	151	David	Bernstein
11	11	129	Laura	Bissot
12	12	169	Harrison	Bloom

```
SELECT ROWNUM, employee_id,
  first_name, last_name
FROM employees
ORDER BY first_name, last_name;
```

	ROWNUM	EMPLOYEE_ID	FIRST_NAME	LAST_NAME
1	33	121	Adam	Fripp
2	104	196	Alana	Walsh
3	26	147	Alberto	Errazuriz
4	46	103	Alexander	Hunold
5	51	115	Alexander	Khoo
6	13	185	Alexis	Bull
7	68	158	Allan	McEwen
8	47	175	Alyssa	Hutton
9	7	167	Amit	Banda
10	14	187	Anthony	Cabrio
11	27	193	Britney	Everett
12	25	104	Bruce	Ernst

Isn't good idea if we need employee number into the list

ROWNUM Pseudocolumn

```
SELECT ROWNUM,
   first_name,
   last_name,
   salary
FROM employees
ORDER BY salary DESC;
```

	ROWNUM	FIRST_NAME	LAST_NAME	SALARY
1	1	Steven	King	24000
2	2	Neena	Kochhar	17000
3	3	Lex	De Haan	17000
4	46	John	Russell	14000
5	47	Karen	Partners	13500
6	102	Michael	Hartstein	13000
7	9	Nancy	Greenberg	12000
8	48	Alberto	Errazuriz	12000
9	106	Shelley	Higgins	12000
10	69	Lisa	Ozer	11500
11	75	Ellen	Abel	11000
12	49	Gerald	Cambrault	11000

```
SELECT ROWNUM, first_name,
   last_name,
   salary
FROM (
   SELECT first_name,
      last_name,
      salary
   FROM employees
   ORDER BY salary DESC
);
```

	ROWNUM	FIRST_NAME	LAST_NAME	SALARY
1	1	Steven	King	24000
2	2	Neena	Kochhar	17000
3	3	Lex	De Haan	17000
4	4	John	Russell	14000
5	5	Karen	Partners	13500
6	6	Michael	Hartstein	13000
7	7	Nancy	Greenberg	12000
8	8	Alberto	Errazuriz	12000
9	9	Shelley	Higgins	12000
10	10	Lisa	Ozer	11500
11	11	Ellen	Abel	11000
12	12	Gerald	Cambrault	11000

Limiting result set of SELECT query

```
SELECT ROWNUM, first_name, last_name, salary
FROM (
  SELECT first_name, last_name, salary
  FROM employees
  ORDER BY salary DESC
)
WHERE ROWNUM <= 5;
```

	ROWNUM	FIRST_NAME	LAST_NAME	SALARY
1	1	Steven	King	24000
2	2	Neena	Kochhar	17000
3	3	Lex	De Haan	17000
4	4	John	Russell	14000
5	5	Karen	Partners	13500

```
SELECT ROWNUM, first_name, last_name, salary
FROM (
  SELECT first_name, last_name, salary
  FROM employees
  ORDER BY salary DESC
)
WHERE ROWNUM BETWEEN 3 AND 5;
```

ROWNUM	FIRST_N...	LAST_N...	SALARY

ROWID Pseudocolumn

For each row in the database, the ROWID pseudocolumn returns the address of the row.

Oracle Database rowid values contain information necessary to locate a row:

- The data object number of the object
- The data block in the data file in which the row resides
- The position of the row in the data block (first row is 0)
- The data file in which the row resides (first file is 1). The file number is relative to the tablespace.

Rowid values have several important uses:

- *They are the fastest way to access a single row.*
- They can show you how the rows in a table are stored.
- They are unique identifiers for rows in a table.

ROWID Pseudocolumn

```
SELECT first_name,
  last_name,
  ROWID,
  DBMS_ROWID.ROWID_RELATIVE_FNO(ROWID) FILE_NO,
  DBMS_ROWID.ROWID_BLOCK_NUMBER(ROWID) BLOCK_NO,
  DBMS_ROWID.ROWID_ROW_NUMBER(ROWID) ROW_NO
FROM employees
ORDER BY 4, 5, 6;
```

Data file Block Row

	FIRST_NAME	LAST_NAME	ROWID	FILE_NO	BLOCK_NO	ROW_NO
1	Steven	King	AAAXPXAAEAAAAD1AAA	4	245	0
2	Neena	Kochhar	AAAXPXAAEAAAAD1AAB	4	245	1
3	Lex	De Haan	AAAXPXAAEAAAAD1AAC	4	245	2
4	Alexander	Hunold	AAAXPXAAEAAAAD1AAD	4	245	3
5	Bruce	Ernst	AAAXPXAAEAAAAD1AAE	4	245	4
6	David	Austin	AAAXPXAAEAAAAD1AAF	4	245	5
7	Valli	Pataballa	AAAXPXAAEAAAAD1AAG	4	245	6
8	Diana	Lorentz	AAAXPXAAEAAAAD1AAH	4	245	7
9	Nancy	Greenberg	AAAXPXAAEAAAAD1AAI	4	245	8
10	Daniel	Faviet	AAAXPXAAEAAAAD1AAJ	4	245	9

Locate Datafile where Table is stored

```
CONNECT SYSTEM

SELECT DISTINCT df.FILE_NAME
FROM hr.employees emp
  JOIN dba_data_files df
  ON (DBMS_ROWID.ROWID_RELATIVE_FNO(emp.ROWID)=df.RELATIVE_FNO)
ORDER BY 1;

DISCONNECT
```

```
Connected
FILE_NAME
--------------------------------------------------------------------------------
/home/oracle/app/oracle/oradata/orcl/users01.dbf

Connection created by CONNECT script command disconnected
```

How many blocks table actually occupies

```
SELECT
  COUNT(DISTINCT DBMS_ROWID.ROWID_BLOCK_NUMBER(ROWID)) BLOCKS_NUM
FROM employees;
```

	BLOCKS_NUM
1	2

```
CONNECT SYSTEM/oracle
SELECT df.file_name, ts.tablespace_name, ts.block_size,
  COUNT(DISTINCT DBMS_ROWID.ROWID_BLOCK_NUMBER(emp.ROWID)) BLOCKS_NUM,
  ts.block_size
   * COUNT(DISTINCT DBMS_ROWID.ROWID_BLOCK_NUMBER(emp.ROWID)) TBL_SIZE
FROM hr.employees emp
  JOIN dba_data_files df
    ON (DBMS_ROWID.ROWID_RELATIVE_FNO(emp.ROWID) = df.RELATIVE_FNO)
  JOIN dba_tablespaces ts
    ON (df.tablespace_name = ts.tablespace_name)
GROUP BY df.file_name, ts.tablespace_name, ts.block_size;
DISCONNECT
```

```
Connected
FILE_NAME                                          TABLESPACE_NAME BLOCK_SIZE BLOCKS_NUM TBL_SIZE
-------------------------------------------------- --------------- ---------- ---------- --------
/home/oracle/app/oracle/oradata/orcl/users01.dbf   USERS                 8192          2    16384

Connection created by CONNECT script command disconnected
```

MTN.BI.02

ORACLE SQL

Questions & Answers

Author: **Aliaksandr Chaika**
Senior Software Engineer
Certified Oracle Database SQL Expert
Aliaksandr_Chaika@epam.com