МегаФон

Oracle Database

Внимание, черный ящик!

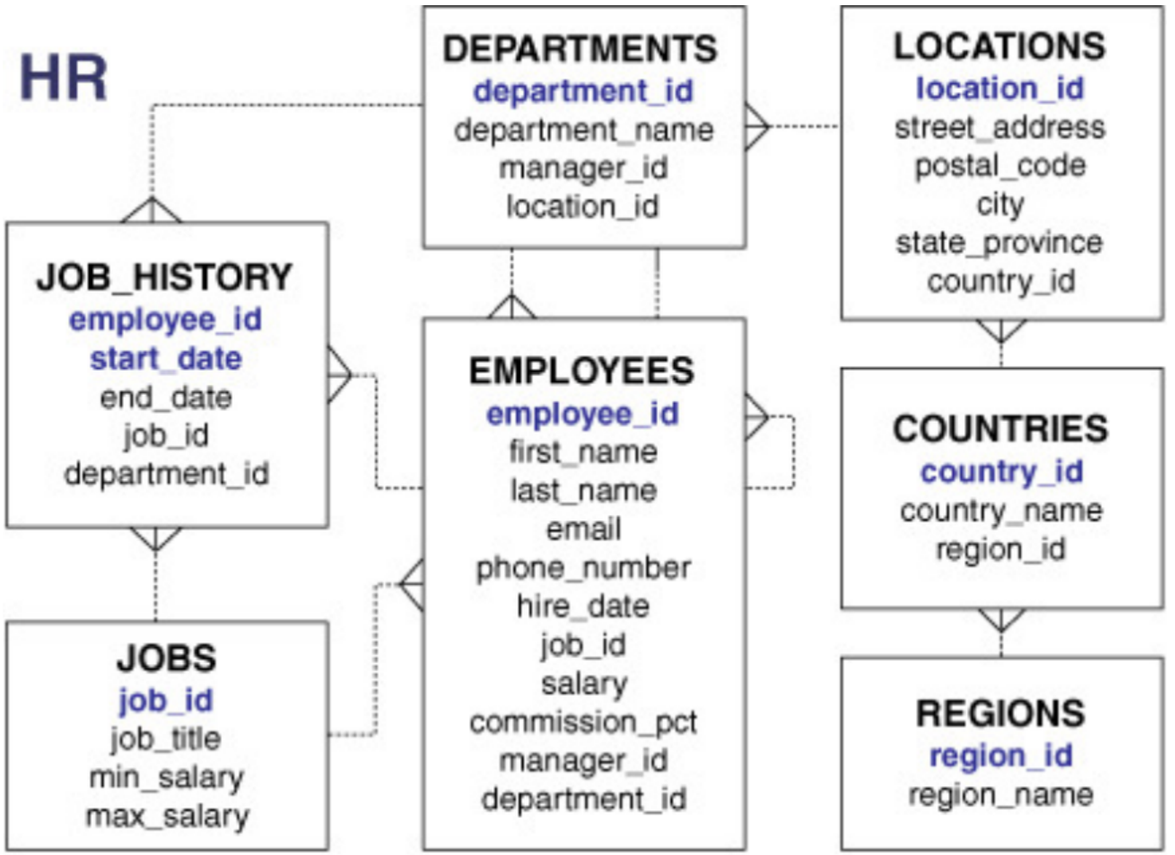
Что такое реляционная база данных?

Реляционные базы данных представляют собой **базы данных**, которые используются для хранения и предоставления доступа к взаимосвязанным элементам информации. Реляционные базы данных основаны на **реляционной модели** — интуитивно понятном, наглядном **табличном способе представления данных**. Каждая **строка**, содержащая в таблице такой базы данных, представляет собой запись с уникальным идентификатором, который называют **ключом**. **Столбцы** таблицы имеют атрибуты данных, а каждая запись обычно содержит значение для каждого атрибута, что дает возможность легко устанавливать взаимосвязь между элементами данных.

Реляционная модель

Реляционная модель представляет собой совокупность данных, состоящую из набора двумерных таблиц. *

*Определение не строгое, отношения (таблицы) являются абстракциями и не могут быть ни «двумерными», ни «не двумерными».


HR
DEPARTMENTS
department_id
department_name
manager_id
location_id
LOCATIONS
location_id
street_address
postal_code
city
state_province
country_id
JOB_HISTORY
employee_id
start_date
end_date
job_id
department_id
EMPLOYEES
employee_id
first_name
last_name
email
phone_number
hire_date
job_id
salary
commission_pct
manager_id
department_id
COUNTRIES
country_id
country_name
region_id
JOBS
job_id
job_title
min_salary
max_salary
REGIONS
region_id
region_name

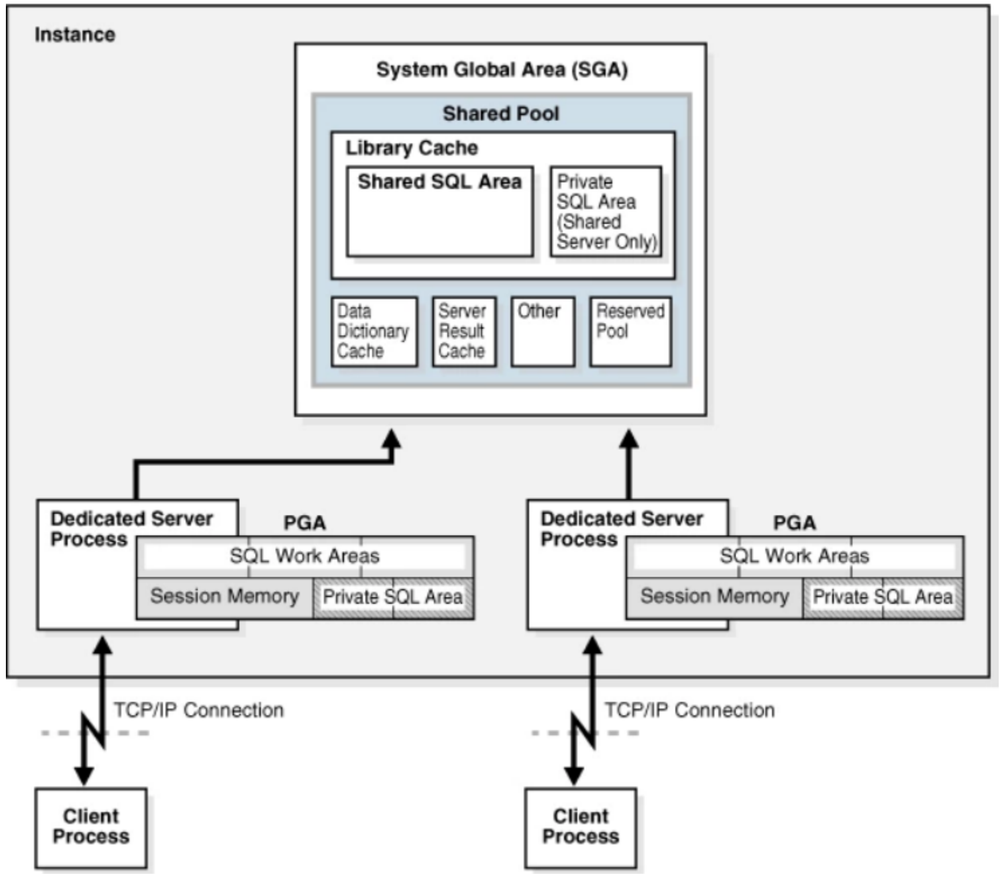
База данных

База данных — это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе.

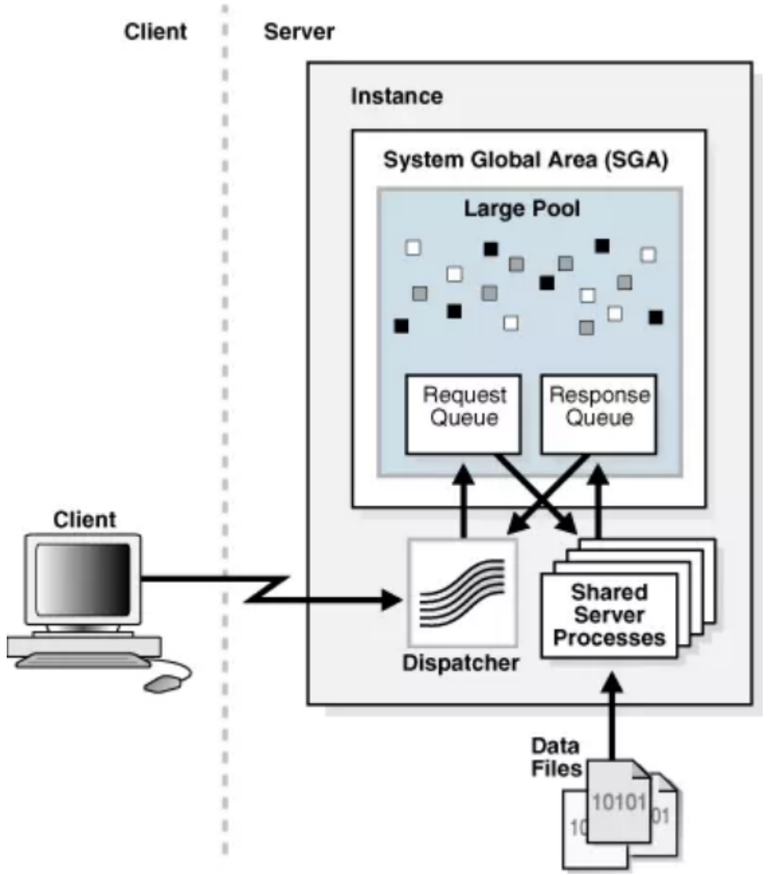
Архитектура Oracle

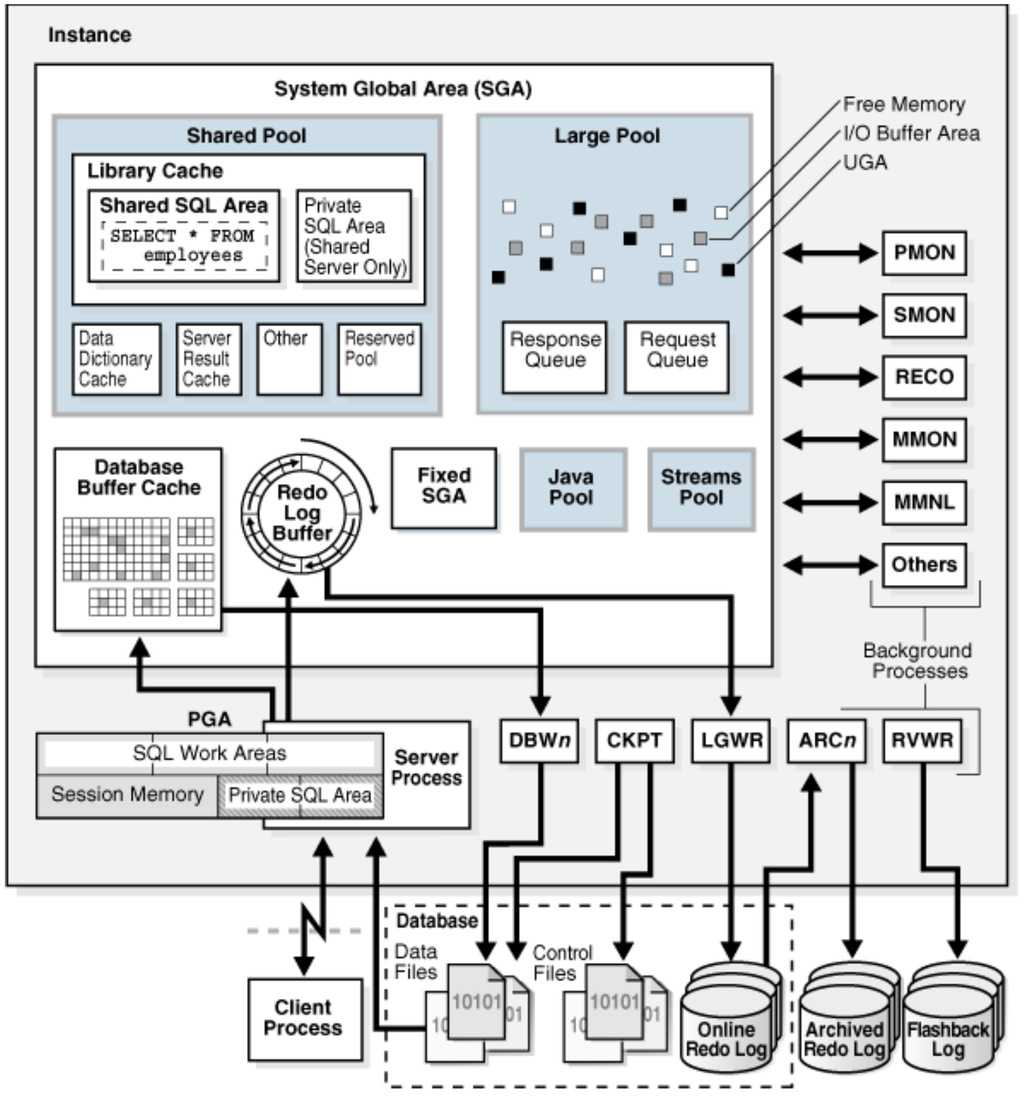
Подключение к БД

Выделенный (dedicated) сервер



Разделяемый (shared) сервер




Instance
System Global Area (SGA)
Shared Pool
Library Cache
Shared SQL Area
SELECT * FROM employees
Private SQL Area (Shared Server Only)
Data Dictionary Cache
Server Result Cache
Other
Reserved Pool
Large Pool
Response Queue
Request Queue
Free Memory
I/O Buffer Area
UGA
Database Buffer Cache
Redo Log Buffer
Fixed SGA
Java Pool
Streams Pool
PMON
SMON
RECO
MMON
MMNL
Others
Background Processes
PGA
SQL Work Areas
Session Memory
Private SQL Area
Server Process
DBWn
CKPT
LGWR
ARCn
RVWR
Database
Data Files
Control Files
Client Process
Online Redo Log
Archived Redo Log
Flashback Log

Экземпляр и База данных

База данных — это коллекция физических файлов или дисков операционной системы.

Экземпляр — представляет собой просто набор процессов операционной системы или один процесс с множеством потоков и определенной областью памяти.

Архитектура памяти

System global area (SGA)

Program global area (PGA)

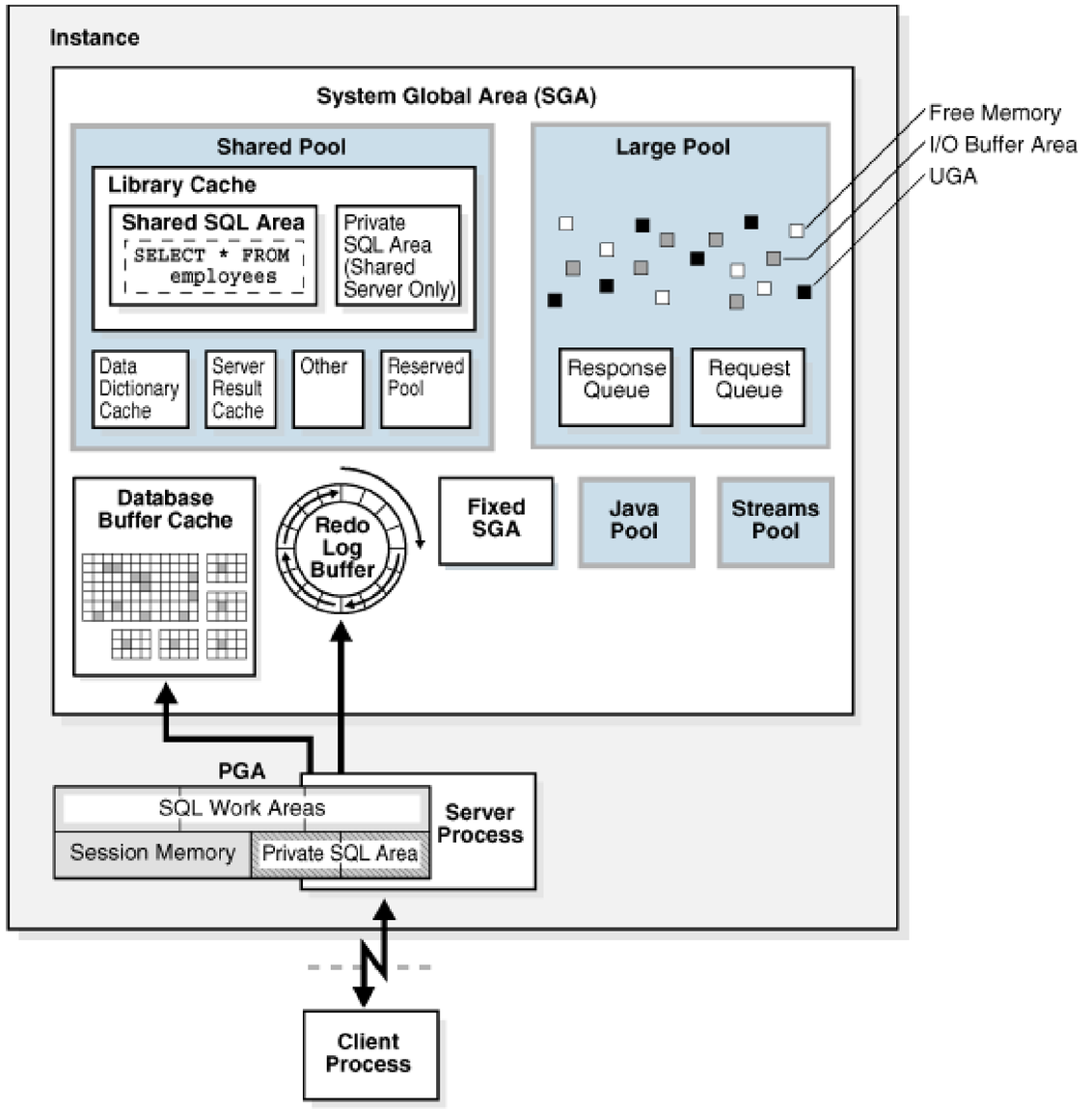
User global area (UGA)

NAME	KBYTES_WRITES	DIFF_KBYTES_WRITES
physical reads direct temporary tablespace	3000	3000
physical writes direct temporary tablespace	3000	3000
session pga memory	1196	320
session pga memory max	1260	384
session uga memory	654	320
session uga memory max	718	384

NAME	KBYTES_WRITES	DIFF_KBYTES_WRITES
physical reads direct temporary tablespace	0	0
physical writes direct temporary tablespace	0	0
session pga memory	1132	256
session pga memory max	11372	10496
session uga memory	654	320
session uga memory max	10631	10296

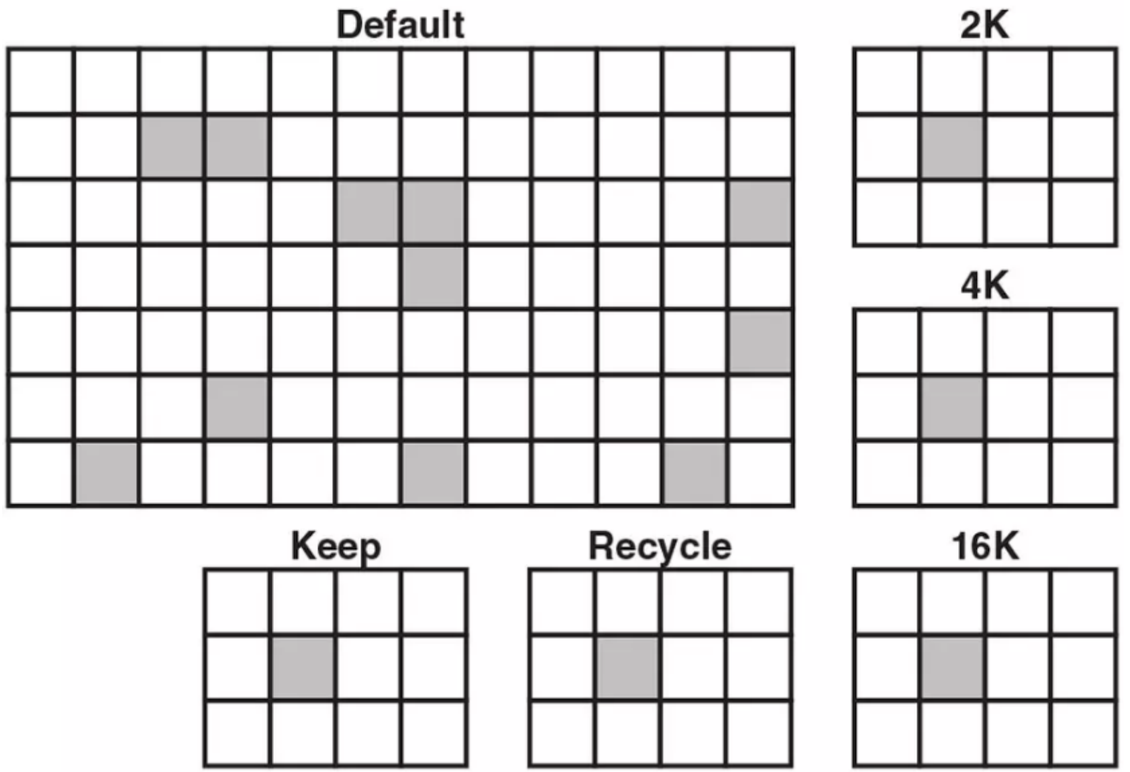
SGA

SGA — Сегмент совместно используемой памяти (формально: системная глобальная область (System Global Area), иногда ее называют разделяемой глобальной областью (Shared Global Area), но чаще просто используют аббревиатуру SGA) хранит массу разнообразной информации.

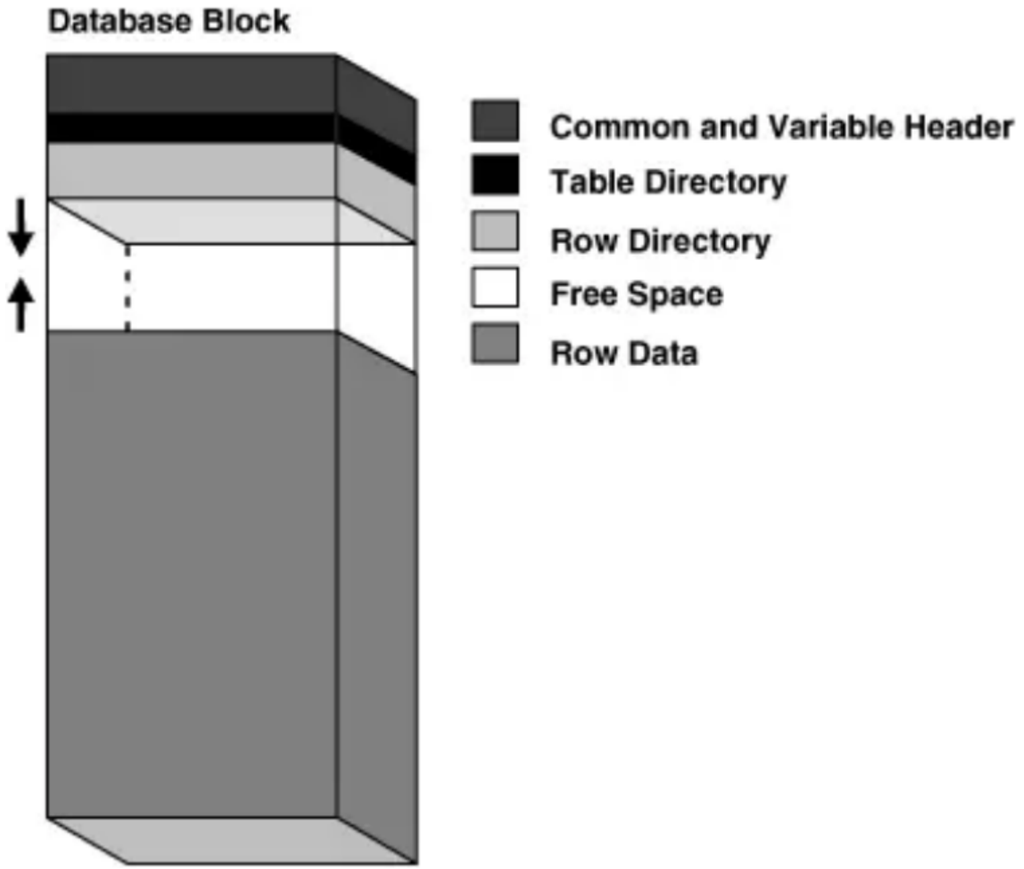

Instance
System Global Area (SGA)
Shared Pool
Library Cache
Shared SQL Area
SELECT * FROM employees
Private SQL Area (Shared Server Only)
Data Dictionary Cache
Server Result Cache
Other
Reserved Pool
Large Pool
Response Queue
Request Queue
Free Memory
I/O Buffer Area
UGA
Database Buffer Cache
Redo Log Buffer
Fixed SGA
Java Pool
Streams Pool
PGA
SQL Work Areas
Session Memory
Private SQL Area
Server Process
Client Process

- **Кэш блоков базы данных (Database Buffer Cache)**
- **Буфера журнала повторения (Redo log buffer)**
- **Пул Java (Java pool)**
- **Большой пул (Large pool)**
- **Разделяемый пул (Shared pool)**
- **Пул Streams (Streams pool)**
- **Фиксированный пул SGA (Fixed SGA)**

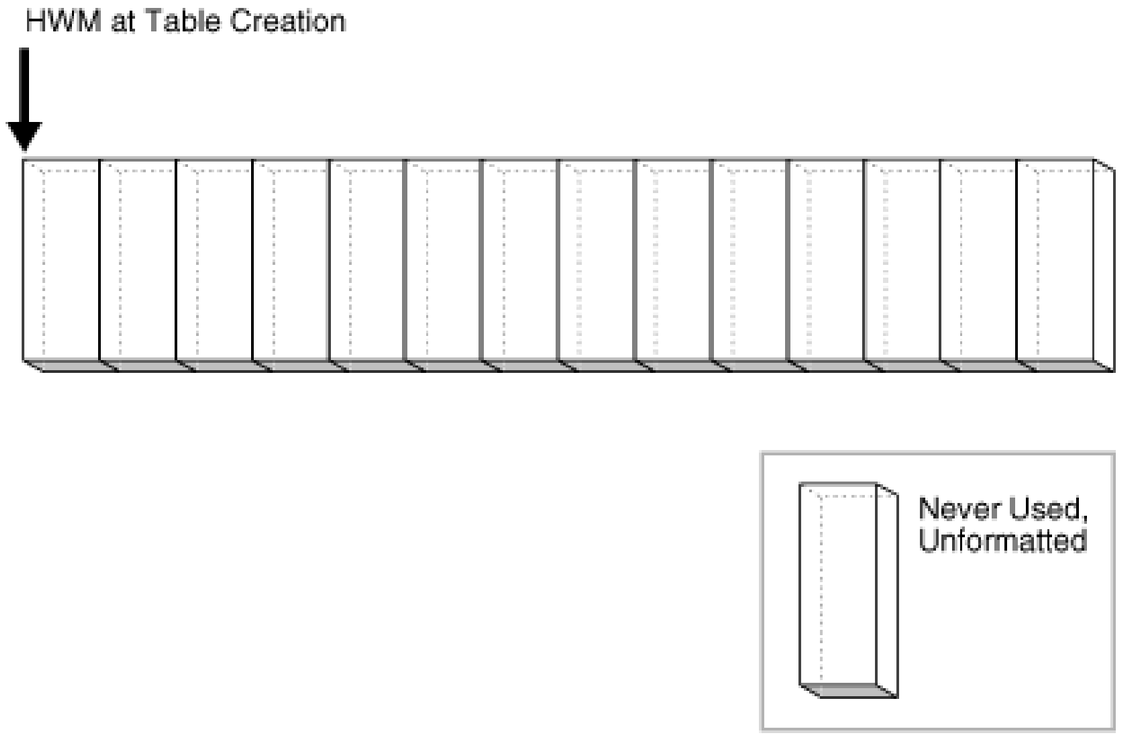
Кэш блоков базы данных (Database Buffer Cache)

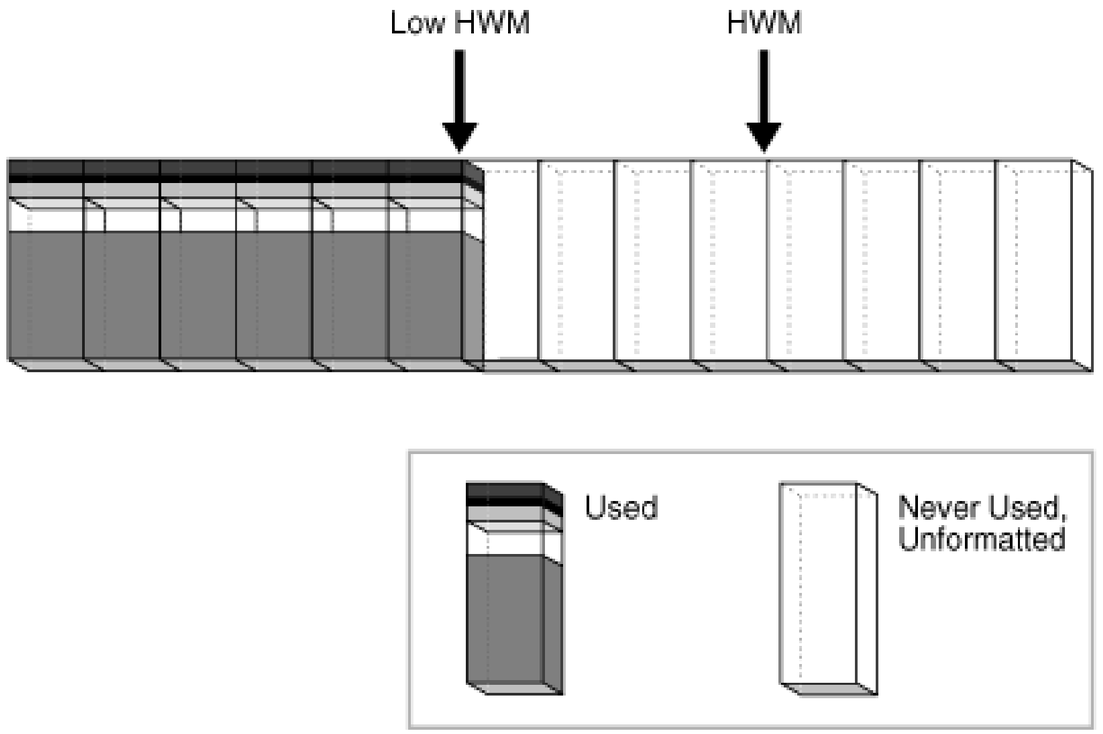

Default
2K
4K
Keep
Recycle
16K

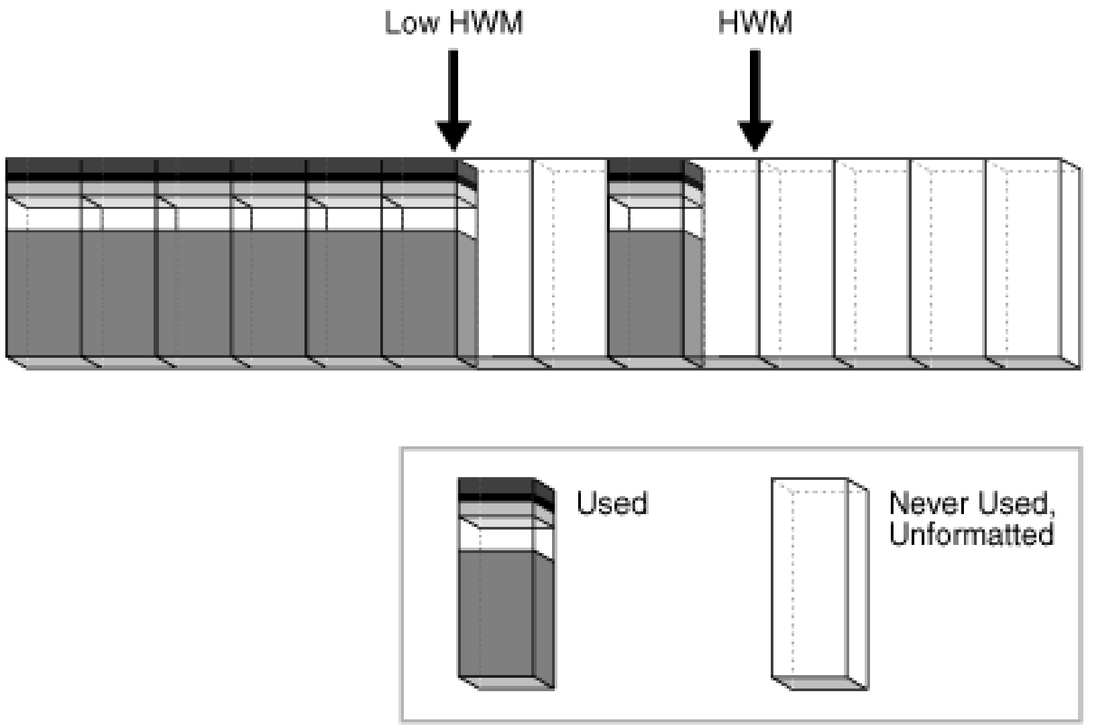
Физическая структура и Логическая структура

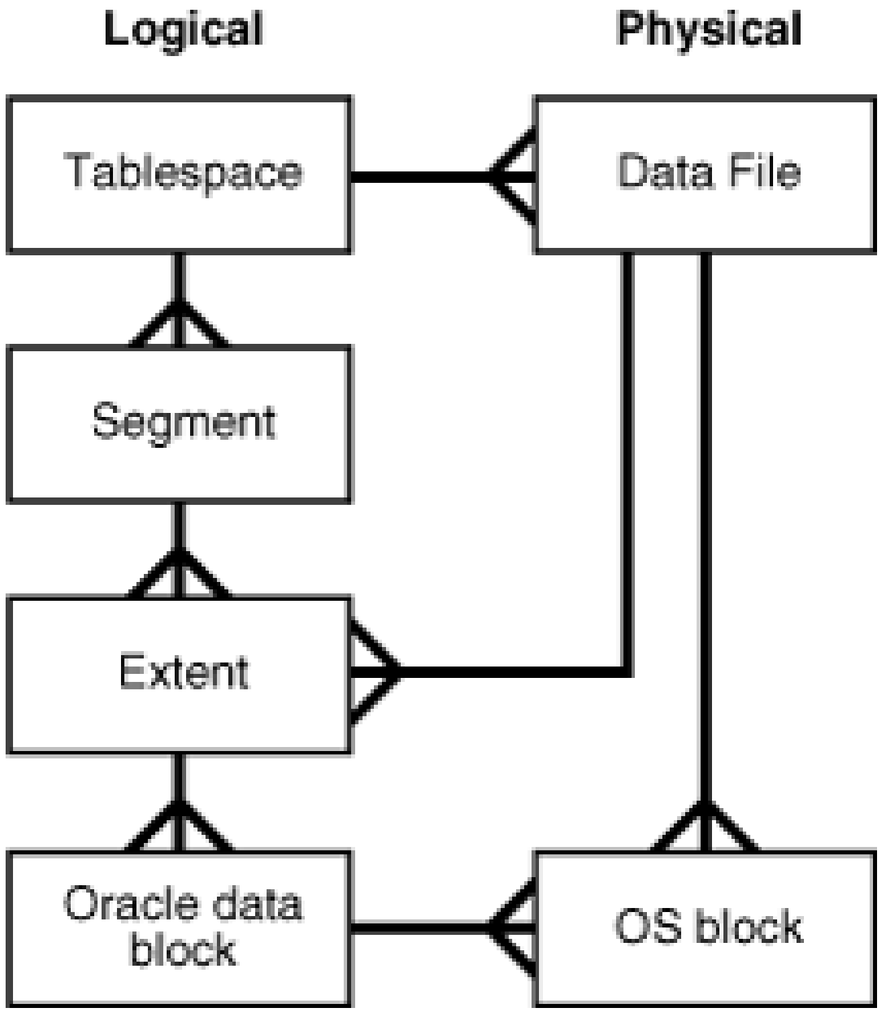

Database Block
Common and Variable Header
Table Directory
Row Directory
Free Space
Row Data

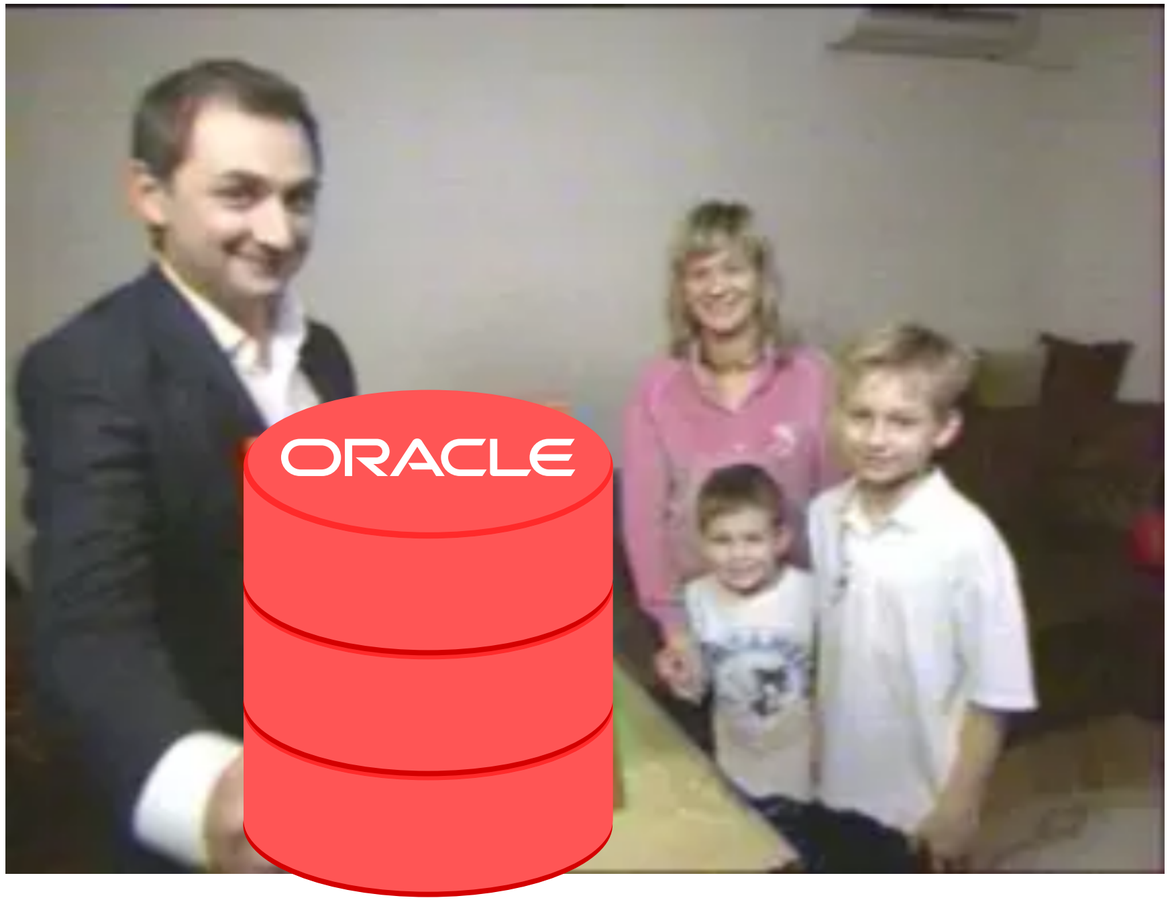
HIGH WATER MARK


HWM at Table Creation
Never Used,
Unformatted


Low HWM
HWM
Used
Never Used,
Unformatted


Low HWM
HWM
Used
Never Used,
Unformatted


Logical
Physical
Tablespace
Data File
Segment
Extent
Oracle data block
OS block


ORACLE

Блокировки

Блокировка (lock) — это механизм, применяемый для регулирования параллельного доступа к разделяемому ресурсу.

Типы блокировок

- **Блокировки DML.**
- **Блокировки DDL.**
- **Внутренние блокировки и защелки.**

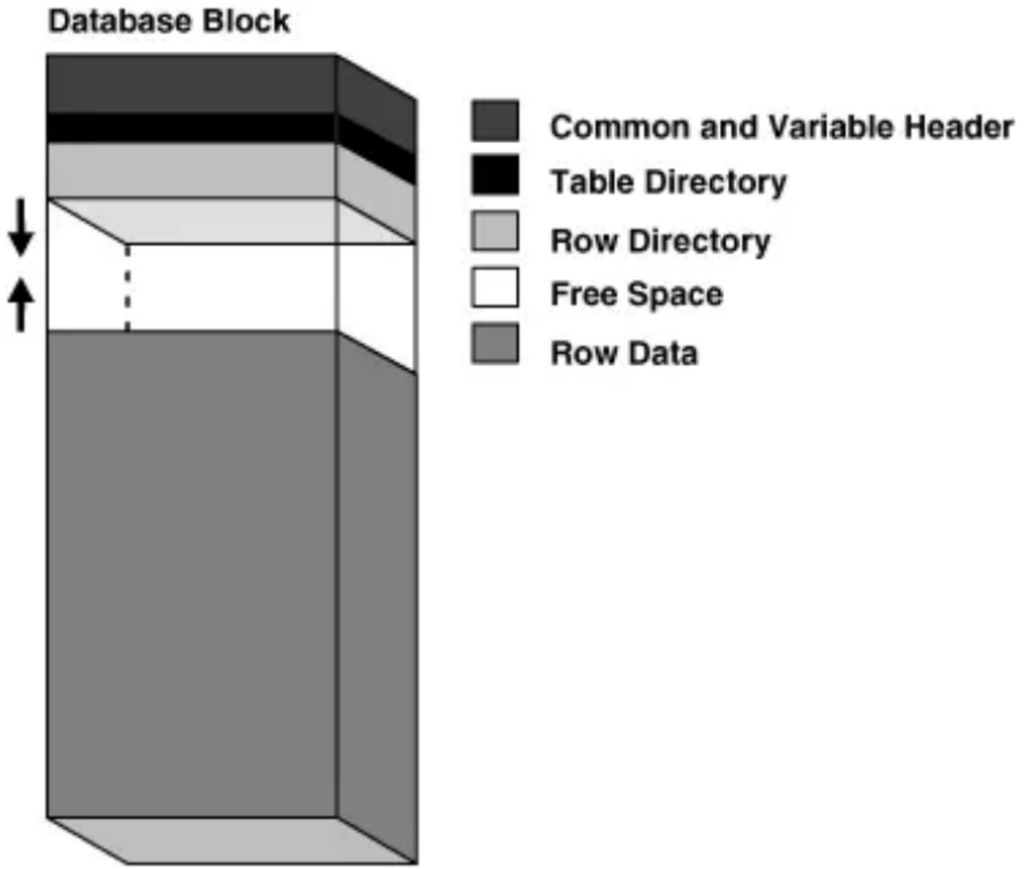
Блокировки DML

БД с традиционным менеджером блокировок

- Найти адрес строки, которую нужно заблокировать.
- Установить связь с диспетчером блокировок (требуется сериализация, т.к. он представляет собой общую структуру в памяти).
- Заблокировать список.
- Выполнить поиск в списке, чтобы выяснить, не блокирует ли эту строку кто-то другой.
- Создать в списке новую запись, свидетельствующую том, что вы блокируете данную строку.
- Разблокировать список.
- ВЫПОЛНЯЕМ ИЗМЕНЕНИЕ СТРОКИ И ФИКСИРУЕМ ИХ.
- Снова установить связь с диспетчером блокировок.
- Заблокировать список.
- Просмотреть список и освободить все свои блокировки.
- Разблокировать список.

ORACLE

- Найти адрес строки , которую нужно заблокировать.
- Перейти к этой строке.
- Заблокировать строку прямо здесь - по местоположению строки, а не в большом списке где-то в другом месте (с ожиданием завершения блокирующей транзакции, если строка уже заблокирована, если только не применяется опция NOWAIT).
- ВЫПОЛНЯЕМ ИЗМЕНЕНИЕ СТРОКИ И ФИКСИРУЕМ ИХ.


Database Block
Common and Variable Header
Table Directory
Row Directory
Free Space
Row Data

Блокировки DDL

```
Begin
    Commit;
    ОПЕРАТОР DDL
    Commit;
Exception
    Иначе выполнить откат;
End;
```

Защелки (latch)

Защелка (latch) — это легковесный механизм сериализации, используемый для координации многопользовательского доступа к разделяемым структурам данных, объектам и файлам.

Цикл

для значения i из диапазона 1 . . 2000

Цикл

Попытаться получить защелку

Если защелка получена , выполнить возврат

Если i = 1, то счетчик неудач = счетчик неудач + 1

Конец цикла

ИНКРЕМЕНТИРОВАТЬ СЧЕТЧИК ОЖИДАНИЯ

Бездействовать

ДОБАВИТЬ ВРЕМЯ ОЖИДАНИЯ

Конец цикла;

Без переменных привязки

```
declare
  order_rec orders%rowtype;
begin
  for i in 1 .. 50000 loop
    begin
      execute immediate
        'select * from orders where order_id = ' || i
        into order_rec;
    exception
      when no_data_found then null;
    end;
  end loop;
end;
```

Elapsed: 00:01:33.94

С переменными привязки

```
declare
  order_rec orders%rowtype;
begin
  for i in 1 .. 50000 loop
    begin
      execute immediate
        'select * from orders where order_id = :i'
        into order_rec
        using i;
    exception
      when no_data_found then null;
    end;
  end loop;
end;
```

Elapsed: 00:00:03.49

Транзакции

Транзакция — это логическая единица работы в базе данных Oracle, состоящая из одного или более операторов SQL. Транзакция начинается с первого исполняемого оператора SQL и завершается, когда вы фиксирует или отказываете транзакцию.

ACID

- **Атомарность (atomicity) -** выполняются либо все действия, произведенные в транзакции, либо ни одного;
- **Согласованность (consistency)** - транзакция перемешает базу данных из одного согласованного состояния в другое;
- **Изоляция (isolation)** - результаты транзакции могут быть не видны другим транзакциям до тех пор, пока она не будет зафиксирована;
- **Постоянство (durability)** - как только транзакция зафиксирована, она остается постоянной.

Атомарность (atomicity)

SQL> create table t2 (cnt int);

Table T2 created.

SQL>
SQL> insert into t2 values (0);

1 row inserted.

SQL>
SQL> commit;

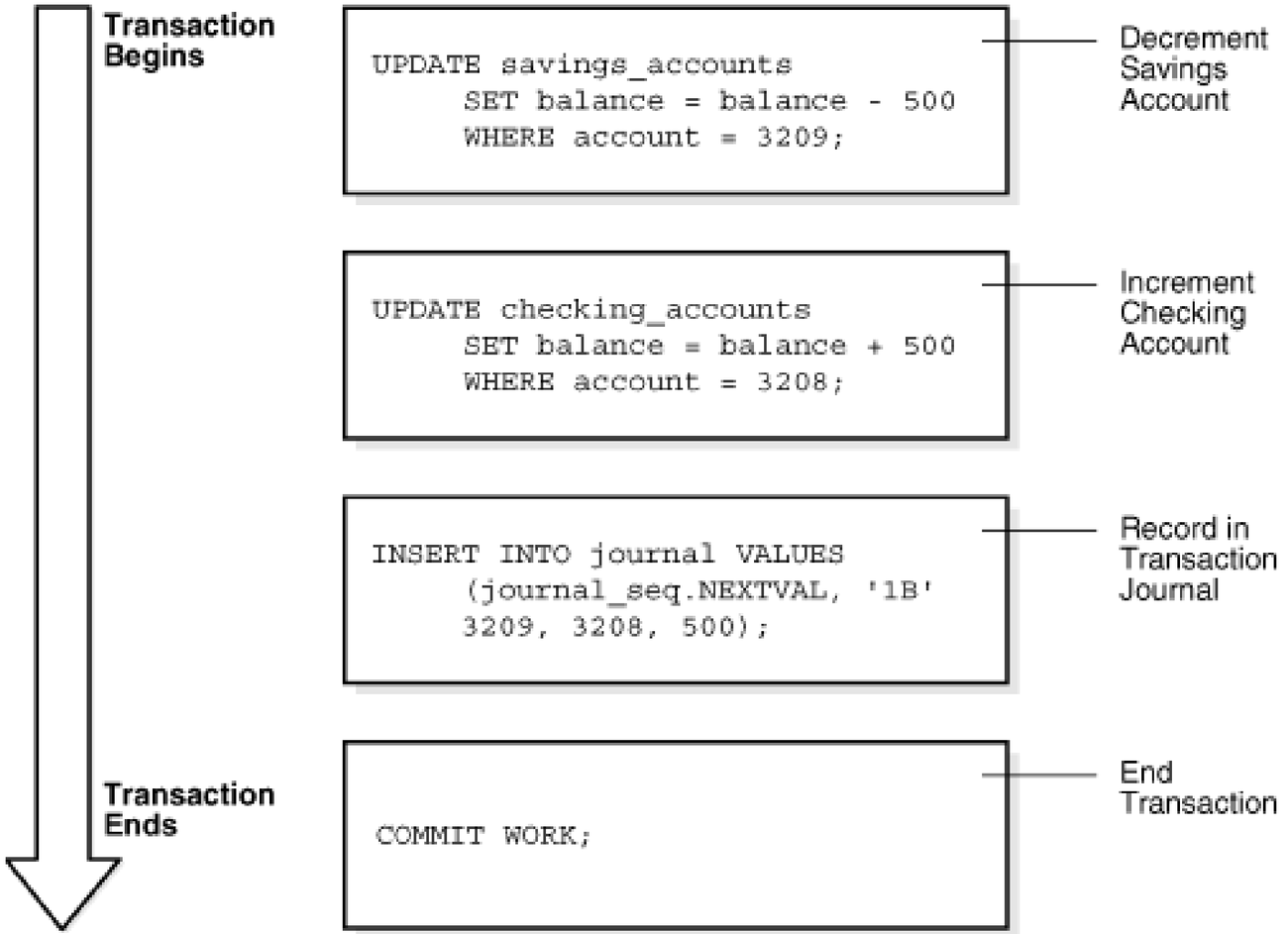
Commit complete.

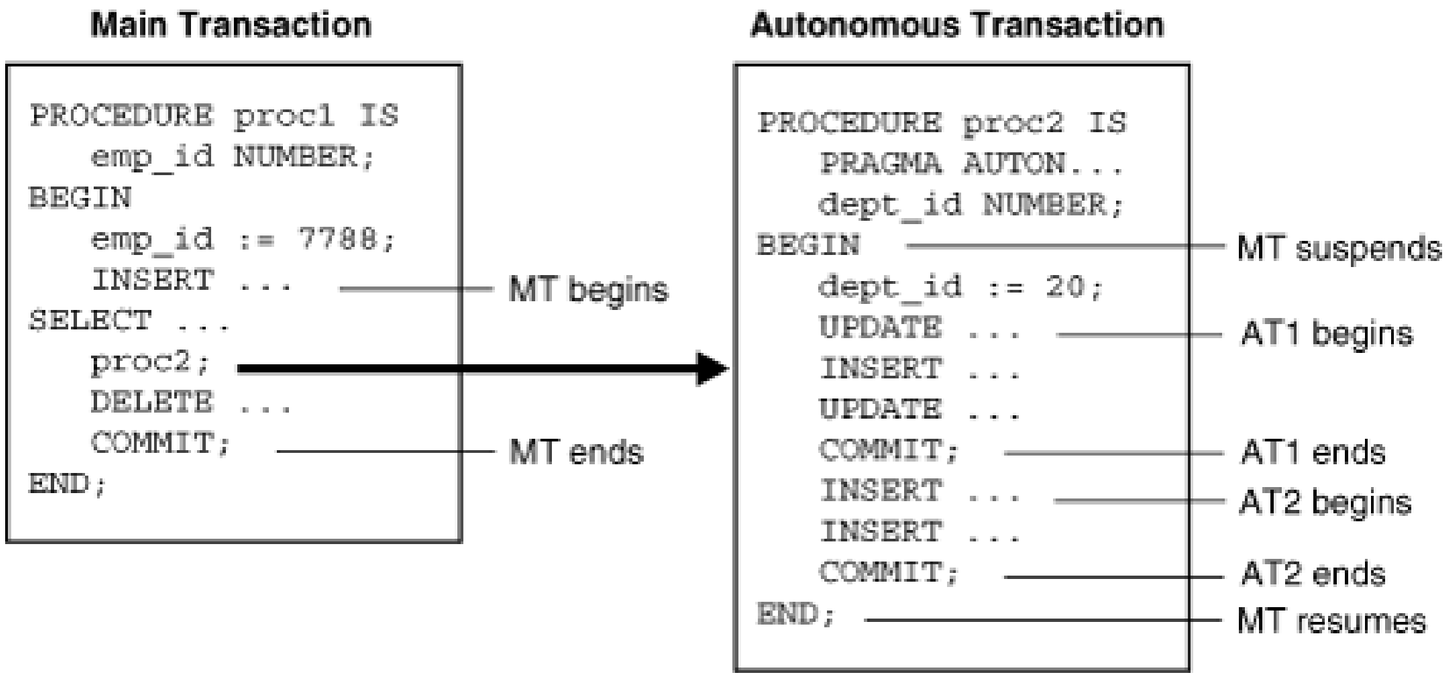
SQL>
SQL> create table t (x int check (x>0));

Table T created.

```
SQL>
SQL> create trigger t_trigger
  2    before insert or delete on t for each row
  3    begin
  4      if ( inserting ) then
  5           update t2 set cnt = cnt +1;
  6      else
  7           update t2 set cnt = cnt -1;
  8      end if;
  9      dbms_output.put_line( 'I fired and updated '   ||
 10                                 sql%rowcount || ' rows' );
 11    end;
 12  /

Trigger T_TRIGGER compiled
```

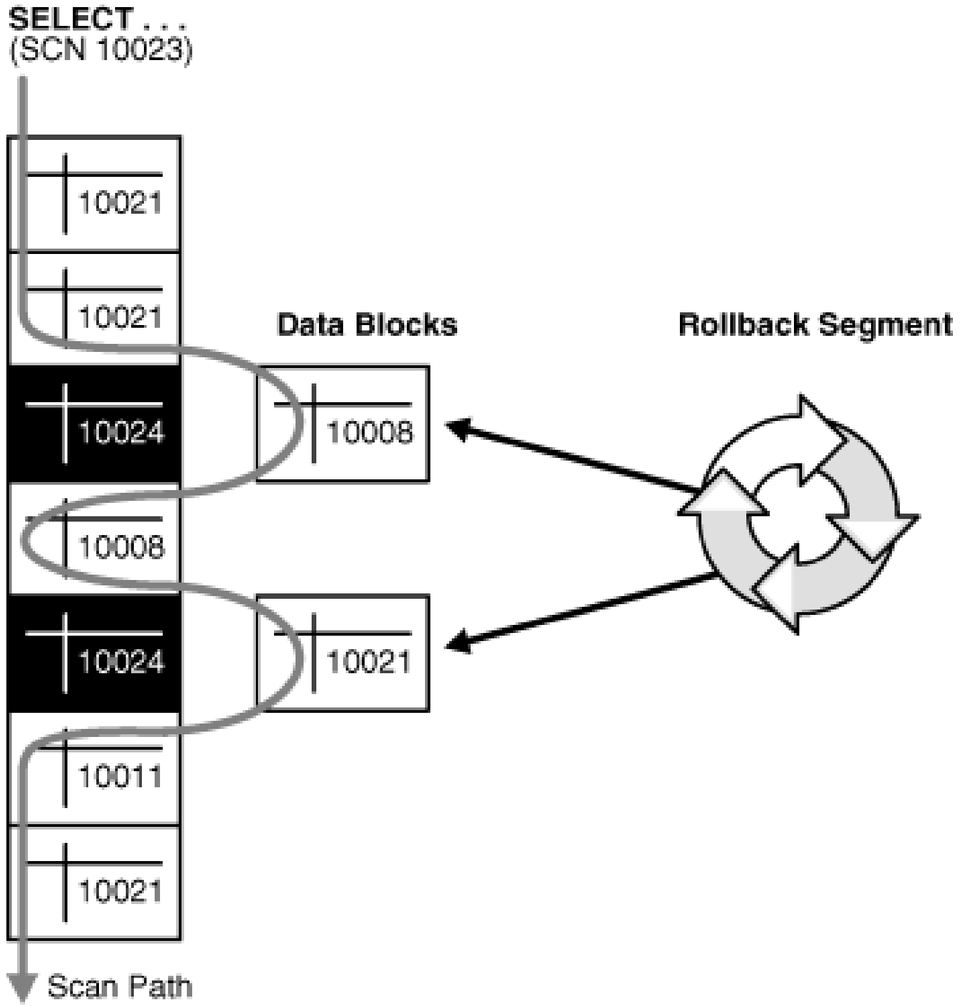
```
SQL>
SQL> set serveroutput on
SQL> insert into t values (1);
I fired and updated 1 rows

1 row inserted.

SQL>
SQL> insert into t values(-1);
I fired and updated 1 rows

Error starting at line : 32 in command -
insert into t values(-1)
Error report -
ORA-02290: нарушено ограничение целостности CHECK(MEGAFON.SYS_C0010981)

SQL>
SQL> select * from t2;

       CNT
----------
         1
```


Transaction Begins
UPDATE savings_accounts
SET balance = balance - 500
WHERE account = 3209;
Decrement Savings Account
UPDATE checking_accounts
SET balance = balance + 500
WHERE account = 3208;
Increment Checking Account
INSERT INTO journal VALUES
(journal_seq.NEXTVAL, '1B'
3209, 3208, 500);
Record in Transaction Journal
COMMIT WORK;
End Transaction
Transaction Ends


Main Transaction
Autonomous Transaction
PROCEDURE proc1 IS
emp_id NUMBER;
BEGIN
emp_id := 7788;
INSERT ...
SELECT ...
proc2;
DELETE ...
COMMIT;
END;
MT begins
MT ends
PROCEDURE proc2 IS
PRAGMA AUTON...
dept_id NUMBER;
BEGIN
dept_id := 20;
UPDATE ...
INSERT ...
UPDATE ...
COMMIT;
INSERT ...
INSERT ...
COMMIT;
END;
MT suspends
AT1 begins
AT1 ends
AT2 begins
AT2 ends
MT resumes

Согласованность (consistency)

Многоверсионное согласованное чтение (Multiversion Read Consistency)


SELECT . . .
(SCN 10023)
10021
10021
10024
10008
10024
10011
10021
Data Blocks
10008
10021
Rollback Segment
Scan Path

- В исходной таблице используется триггер для поддержки столбца LAST_UPDATED
- Для начального наполнения таблицы в DWX они запоминают текущее время, выбирая SYSDATE в исходной системе. Например, предположим, что прямо сейчас ровно 9:00.
- Затем извлекаются все строки из транзакционной системы, т.е. выполняют полный запрос SELECT * FROM <ТАБЛИЦА>, чтобы провести начальное заполнение хранилища данных.
- Для обновления хранилища данных они опять запоминают текущее время. Например, предположим, что прошел час и теперь в исходной системе 10:00. Они запоминают этот факт. Затем они извлекают все изменения, накопившиеся после 9:00 (момента перед началом первого извлечения), и объединяют их.

Согласованность записи

update t set x = 2 where y = 5

Изоляция (isolation)

- **Грязное чтение (dirty read).**
- **Невоспроизводимое чтение (nonrepeatable read).**
- **Фантомное чтение (phantom read).**

Уровни изоляции ANSI

	DIRTY READ	NONREPEATABLE READ	PHANTOM READ
READ UNCOMMITTED	+	+	+
READ COMMITTED	-	+	+
REPEATABLE READ	-	-	+
SERIALIZABLE	-	-	-

Уровни изоляции ORACLE

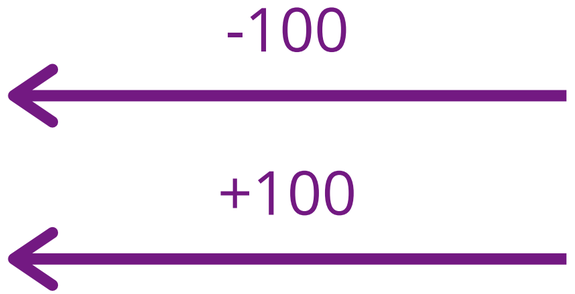

READ COMMITTED
SERIALIZABLE
READ ONLY

ACCOUNT_NUMBER	ACCOUNT_BALANCE
123	500
456	240
987	100

SUM_ACCOUNT_BALANCE
840

READ UNCOMMITTED

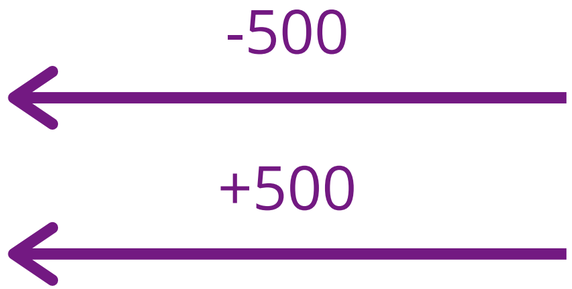
ACCOUNT_NUMBER	ACCOUNT_BALANCE
123	500
456	240
987	100



SUM_ACCOUNT_BALANCE
840

READ COMMITTED

ACCOUNT_NUMBER	ACCOUNT_BALANCE
123	500
456	240
987	100



SUM_ACCOUNT_BALANCE
840

REPEATABLE READ

ACCOUNT_NUMBER	ACCOUNT_BALANCE
123	500
456	240
987	100

UPDATE 123 на -500

COMMIT

UPDATE 123 на -500 и 987 на +500

SUM_ACCOUNT_BALANCE
840

ACCOUNT_NUMBER	ACCOUNT_BALANCE
123	500
456	240
987	100

UPDATE 987 на +500 и 123 на -500

Не можем прочитать 987 (DEADLOCK)

SUM_ACCOUNT_BALANCE
840

SERIALIZABLE И READ ONLY

SQL

DML

SELECT
INSERT
UPDATE
DELETE
MERGE

DDL

CREATE
ALTER
DROP
RENAME
TRUNCATE
COMMENT

DCL

GRANT
REVOKE

TCL

COMMIT
ROLLBACK
SAVEPOINT

SELECT

SELECT *|{[DISTINCT] column|expression [alias],...} **FROM** table;

SELECT last_name, salary, (salary + 100) as new_salary FROM hr.employees

LAST_NAME	SALARY	NEW_SALARY
King	24000	24100
Kochhar	17000	17100
De Haan	17000	17100
Hunold	9000	9100
Ernst	6000	6100
Austin	4800	4900
Pataballa	4800	4900
Lorentz	4200	4300
Greenberg	12008	12108
Faviet	9000	9100
Chen	8200	8300

```
SELECT first_name || '_' || last_name FROM  hr.employees
```

```
FIRST_NAME||'_'||LAST_NAME
------------------------------------------
Ellen_Abel
Sundar_Ande
Mozhe_Atkinson
David_Austin
Hermann_Baer
Shelli_Baida
Amit_Banda
Elizabeth_Bates
Sarah_Bell
David_Bernstein
Laura_Bissot
```

```
SELECT DISTINCT department_id FROM employees;
```

DEPARTMENT_ID
100
30
90
20
70
110
50
80
40
60

SELECT *|{[DISTINCT] column|expression [alias],...} FROM table [WHERE condition(s)];

```
SELECT DISTINCT department_id FROM employees WHERE department_id = 20;
```

```
DEPARTMENT_ID
-------------
           20
```

```
SELECT first_name FROM employees WHERE first_name LIKE 'A%';
```

```
FIRST_NAME
--------------------
Amit
Alexis
Anthony
Alberto
Adam
Alexander
Alyssa
Alexander
Allan
Alana
```

```
SELECT first_name FROM employees WHERE department_id IS NULL;
```

```
FIRST_NAME
--------------------
Kimberely
```

```
SELECT *|{[DISTINCT] column|expression [alias],...} FROM table;
[WHERE condition(s)][ORDER BY {column, expr, alias} [ASC|DESC]]
```

```
SELECT first_name FROM employees
WHERE department_id IS NOT NULL
ORDER BY salary DESC;
```

```
FIRST_NAME
--------------------
Steven
Neena
Lex
John
Karen
Michael
Nancy
Shelley
Alberto
Lisa
Den
```

```
SELECT first_name FROM employees
WHERE lower(first_name) = 'nancy'
ORDER BY salary DESC;
```

```
FIRST_NAME
--------------------
Nancy
```

```
SELECT first_name FROM employees
WHERE lower(first_name) = 'nancy'
ORDER BY salary DESC;
```

```
FIRST_NAME
--------------------
Nancy
```

CHR
CONCAT
INITCAP
LOWER
LPAD
LTRIM
NLS_INITCAP
NLS_LOWER
NLSSORT
NLS_UPPER
REGEXP_REPLACE
REGEXP_SUBSTR
REPLACE
RPAD
RTRIM
SOUNDEX
SUBSTR
TRANSLATE
TREAT
TRIM
UPPER

ADD_MONTHS
CURRENT_DATE
CURRENT_TIMESTAMP
DBTIMEZONE
EXTRACT (datetime)
FROM_TZ
LAST_DAY
LOCALTIMESTAMP
MONTHS_BETWEEN
NEW_TIME
NEXT_DAY
NUMTODSINTERVAL
NUMTOYMINTERVAL
ROUND (date)
SESSIONTIMEZONE
SYS_EXTRACT_UTC
SYSDATE
SYSTIMESTAMP
TO_CHAR (datetime)
TO_TIMESTAMP
TO_TIMESTAMP_TZ
TO_DSINTERVAL
TO_YMINTERVAL
TRUNC (date)
TZ_OFFSET

```
SELECT hire_date, TRUNC(hire_date, 'MM') as hire_month_date,
ADD_MONTHS(TRUNC(hire_date, 'IW'), -2) as hire_date_min2months
FROM employees;
```

HIRE_DAT	HIRE_MON	HIRE_DAT
13.01.08	01.01.08	07.11.07
17.09.03	01.09.03	15.07.03
17.02.04	01.02.04	16.12.03
17.08.05	01.08.05	15.06.05
07.06.02	01.06.02	03.04.02
07.06.02	01.06.02	03.04.02
07.06.02	01.06.02	03.04.02
07.06.02	01.06.02	03.04.02

```
SELECT hire_date, TO_CHAR(hire_date, 'fmDD Month YYYY') AS
HIREDATE FROM employees;
```

```
HIRE_DAT HIREDATE
-------- --------------------
13.01.08 13 Январь 2008
17.09.03 17 Сентябрь 2003
17.02.04 17 Февраль 2004
17.08.05 17 Август 2005
07.06.02 7 Июнь 2002
07.06.02 7 Июнь 2002
07.06.02 7 Июнь 2002
07.06.02 7 Июнь 2002
```

SELECT employee_id, salary, commission_pct, nvl(commission_pct, 0), coalesce(commission_pct, 0) FROM employees;

EMPLOYEE_ID	SALARY	COMMISSION_PCT	NVL(COMMISSION_PCT,0)	COALESCE(COMMISSION_PCT,0)
199	2600		0	0
200	4400		0	0
201	13000		0	0
202	6000		0	0
203	6500		0	0
204	10000		0	0
205	12008		0	0
206	8300		0	0

SELECT employee_id, salary, commission_pct, nvl(commission_pct, 0), coalesce(commission_pct, 0) FROM employees;

EMPLOYEE_ID	SALARY	COMMISSION_PCT	NVL(COMMISSION_PCT,0)	COALESCE(COMMISSION_PCT,0)
199	2600		0	0
200	4400		0	0
201	13000		0	0
202	6000		0	0
203	6500		0	0
204	10000		0	0
205	12008		0	0
206	8300		0	0

```
CASE expr
      WHEN comparison_expr1 THEN return_expr1
      [WHEN comparison_expr2 THEN return_expr2
      WHEN comparison_exprn THEN return_exprn
      ELSE else_expr]
END
```

```
SELECT
    employee_id, job_id, salary as old_salary,
     CASE job_id
        WHEN 'IT_PROG' THEN salary * 300000
        ELSE salary * 100
    END as new_salary
FROM employees;
```

EMPLOYEE_ID	JOB_ID	OLD_SALARY	NEW_SALARY
100	AD_PRES	24000	2400000
101	AD_VP	17000	1700000
102	AD_VP	17000	1700000
103	IT_PROG	9000	2700000000
104	IT_PROG	6000	1800000000
105	IT_PROG	4800	1440000000
106	IT_PROG	4800	1440000000
107	IT_PROG	4200	1260000000
108	FI_MGR	12008	1200800
109	FI_ACCOUNT	9000	900000
110	FI_ACCOUNT	8200	820000