Oracle Direct Seminar

ORACLE®

ここからはじめよう！ Oracle SQL入門

日本オラクル株式会社

Oracle Direct

ORACLE 11g DATABASE

はじめに

- 本セミナーは、これからSQLを使った開発を始める方に向け、SQL文の基本的な構文をご紹介する初心者向けセミナーです

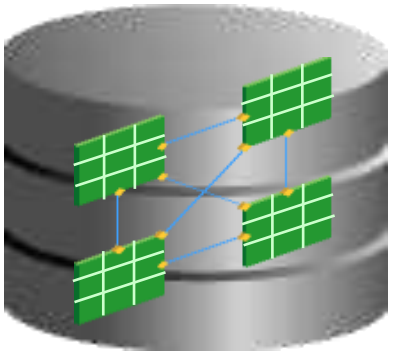
- アジェンダ
  - リレーショナル・データベースの特徴
  - 基本的なSELECT文
  - データを加工するSELECT文
  - データ集計を行うSELECT文
  - 複数の表の結合
  - 副問合せ

アジェンダ

- リレーショナル・データベースの特徴
- 基本的なSELECT文
- データを加工するSELECT文
- データ集計を行うSELECT文
- 複数の表の結合
- 副問合せ

リレーショナル・データベースの特徴

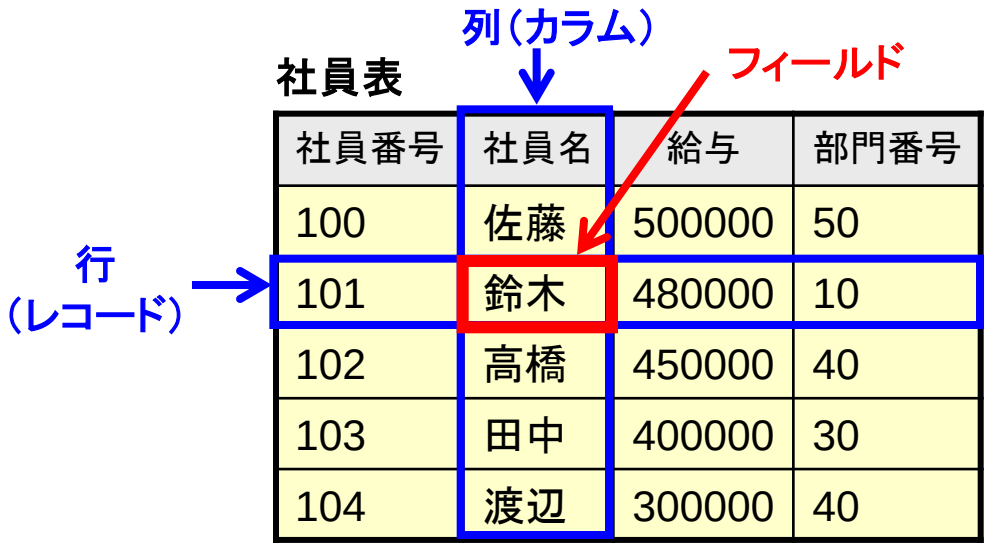
- 代表的な商用データベース
- リレーショナル・データベースの特徴
  ①データを「行」と「列」から構成される2次元の表形式で表す
  ②関連する複数の表にデータを格納する
  ③SQL（構造化問い合わせ言語）により、データ操作をする



リレーショナル・データベースの特徴

2次元の表でデータを管理

- データを「行」と「列」から構成される2次元の表形式で表す
  - 行は1件のデータ(レコード)を表す
  - 列(カラム)は各項目を表す
  - 行と列の交差部分を「フィールド」と呼ぶ



社員番号	社員名	給与	部門番号
100	佐藤	500000	50
101	鈴木	480000	10
102	高橋	450000	40
103	田中	400000	30
104	渡辺	300000	40

リレーショナル・データベースの特徴
関連する複数の表にデータを格納

- 複数の表を作成し、データを格納
- 表の間には関係（リレーション）がある

社員表

社員番号	社員名	給与	部門番号
100	佐藤	500000	50
101	鈴木	480000	10
102	高橋	450000	40
103	田中	400000	30
104	渡辺	300000	40

関係
（リレーション）

部門表

部門番号	部門名
10	営業
20	経理
30	人事
40	IT企画
50	経営企画

<補足>正規化と非正規化

社員番号	社員名	給与	部門名
100	佐藤	500000	経営企画
101	鈴木	480000	営業
102	高橋	450000	IT企画
103	田中	400000	人事
104	渡辺	300000	IT企画

重複データ
非正規化

正規化
一般的にリレーショナル・データベースでは、繰り返しデータを持たないように、複数表に分けてデータを格納
・データ量を節約するため
・更新時の負荷を少なくするため

正規化の詳細は「いまさら聞けない！？Oracle Database設計」で

リレーショナル・データベースの特徴
SQLによりデータを操作

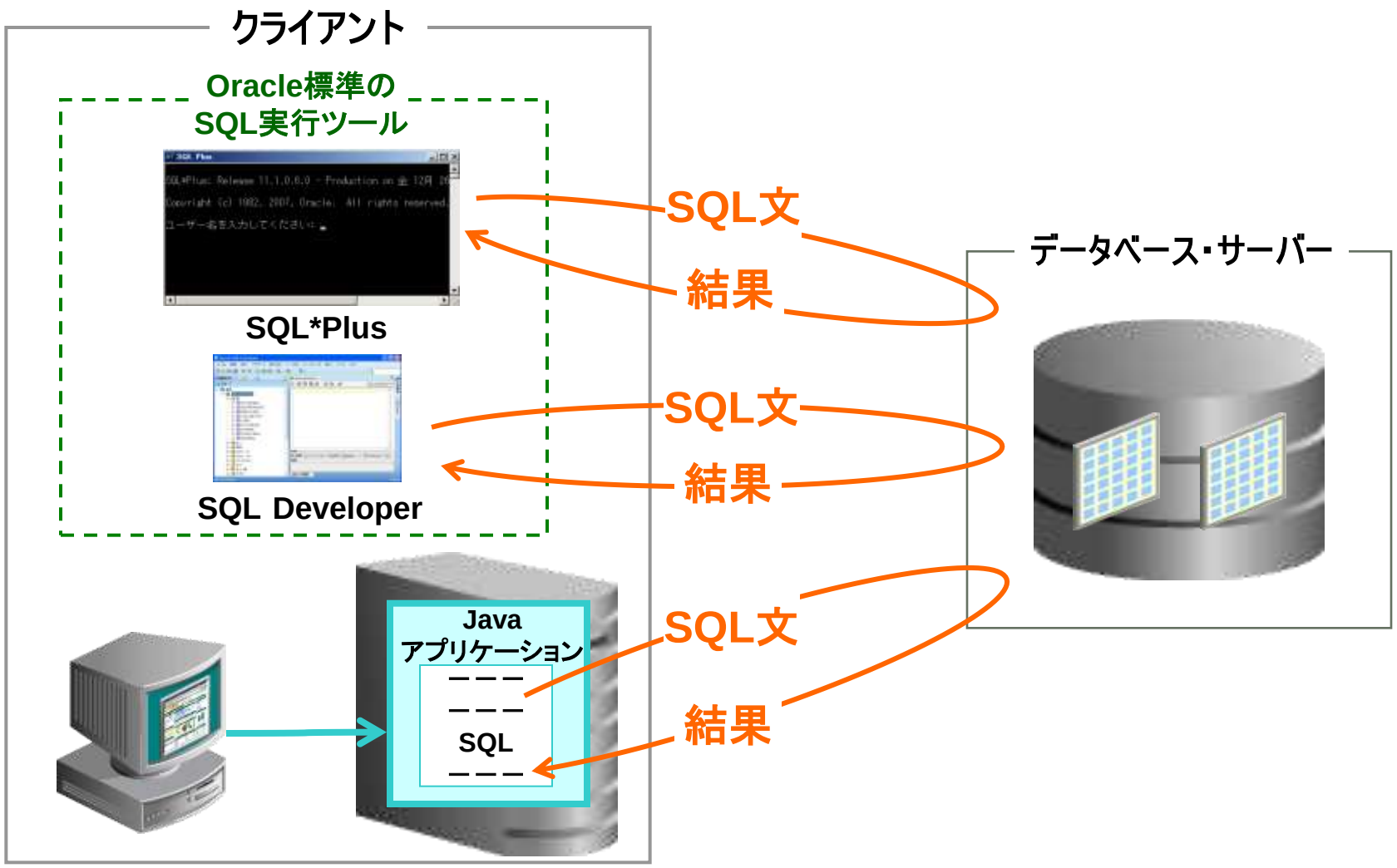
- **SQL（Structured Query Language：構造化問合せ言語）を使って操作**
  - **リレーショナル・データベースの定義や操作などを実現するための言語**
  - **ANSI（米国規格協会）やISO（国際標準化機構）によって、規格化**
  - **処理によって、いくつかの種類に分類される**

分類	SQL文	SQL文の説明
データ操作言語（DML）	SELECT INSERT UPDATE DELETE	データの検索 データの追加 データの更新 データの削除
データ定義言語（DDL）	CREATE DROP ALTER GRANT REVOKE	オブジェクト（表など）の作成 オブジェクト（表など）の削除 オブジェクト（表など）の変更 ユーザー権限の付与 ユーザー権限の削除
データ制御言語（DCL）	COMMIT ROLLBACK	更新処理の確定 更新処理の取り消し

SQL:2003(標準規格)
ANSI(米国規格協会)
ISO(国際標準化機構)
に準拠

SQLの実行環境

様々な環境から実行可能



セミナーで使用するサンプル表
EMP表とDEPT表

- 社員表（EMP）と部門表（DEPT）を使って、社員のデータを検索

EMP（社員表）

EMP_ID	NAME	HIRE_DATE	SALARY	DEPT_ID
100	佐藤	00-04-01	500000	50
101	鈴木	00-10-01	480000	10
102	高橋	02-01-01	450000	40
103	田中	03-12-01	400000	30
104	渡辺	07-04-01	300000	40
105	伊藤	04-12-01	350000	30
106	山本	01-04-01	450000	20
107	中村	06-11-01	300000	10
108	小林	05-04-01	360000	20
109	吉田	08-04-01	280000	

DEPT（部門表）

DEPT_ID	DEPT_NAME
10	営業
20	経理
30	人事
40	IT企画
50	経営企画
60	環境対策

アジェンダ

- リレーショナル・データベースの特徴
- 基本的なSELECT文
- データを加工するSELECT文
- データ集計を行うSELECT文
- 複数の表の結合
- 副問合せ

基本的なSELECT文

表から全てのデータを取り出すには

```
SELECT    *（全ての列を）
FROM      表名（どの表から）;
```

Point

SELECT句の指定
＊・・・全ての列
列名・・・複数列の場合は
カンマ区切りで指定

実行するときは;を指定

EMP（社員表）

EMP_ID	NAME	HIRE_DATE	SALARY	DEPT_ID
100	佐藤	00-04-01	500000	50
101	鈴木	00-10-01	480000	10
102	高橋	02-01-01	450000	40
103	田中	03-12-01	400000	30
104	渡辺	07-04-01	300000	40
105	伊藤	04-12-01	350000	30
106	山本	01-04-01	450000	20
107	中村	06-11-01	300000	10
108	小林	05-04-01	360000	20
109	吉田	08-04-01	280000	

社員表から全ての列のデータを検索

書くのはココ

```
SQL> SELECT   *
  2  FROM     EMP（社員表）;

    EMP_ID NAME       HIRE_DAT     SALARY    DEPT_ID
---------- ---------- -------- ---------- ----------
       100 佐藤       00-04-01     500000         50
       101 鈴木       00-10-01     480000         10
       102 高橋       02-01-01     450000         40
       103 田中       03-12-01     400000         30
       104 渡辺       07-04-01     300000         40
       105 伊藤       04-12-01     350000         30
       106 山本       01-04-01     450000         20
       107 中村       06-11-01     300000         10
       108 小林       05-04-01     360000         20
       109 吉田       08-04-01     280000
```

基本的なSELECT文

特定の列の検索

```
SELECT   列名,列名・・・(どの項目を)
FROM     表名(どの表から);
```

Point

SELECT句の指定
＊・・・全ての列
列名・・・複数列の場合は
カンマ区切りで指定

実行するときは;を指定

EMP(社員表)

EMP_ID	NAME	HIRE_DATE	SALARY	DEPT_ID
100	佐藤	00-04-01	500000	50
101	鈴木	00-10-01	480000	10
102	高橋	02-01-01	450000	40
103	田中	03-12-01	400000	30
104	渡辺	07-04-01	300000	40
105	伊藤	04-12-01	350000	30
106	山本	01-04-01	450000	20
107	中村	06-11-01	300000	10
108	小林	05-04-01	360000	20
109	吉田	08-04-01	280000	

社員表から社員名(name)、入社日(HIRE_DATE)、給与(SALARY)のデータを検索

```
SQL>SELECT name, hire_date,
  2        salary
  3 FROM   emp;

name       HIRE_DAT      SALARY
---------- -------- ----------
佐藤       00-04-01      500000
鈴木       00-10-01      480000
高橋       02-01-01      450000
田中       03-12-01      400000
渡辺       07-04-01      300000
伊藤       04-12-01      350000
.....
```

基本的なSELECT文
エラーの解決

- 構文が正しくない場合や表名、列名などが違う場合はエラーが発生

社員表から社員名(NAME)、入社日(HIRE_DATE)、給与(SALARY)のデータを検索

```
SQL>SELECT name, hiredate, salary
  2 FROM   emp;
```

```
SELECT name, hiredate, salary
*
行1でエラーが発生しました。:
ORA-00904: "HIREDATE": 無効な識別子です。
```

エラーメッセージで
エラー箇所に*が表示

```
SQL>SELECT name, hire_date, salary
 2   FROM   enp;
```

```
FROM   enp
       *
行2でエラーが発生しました。:
ORA-00942: 表またはビューが存在しません。
```

基本的なSELECT文

行の選択（数値データの検索）

```
SELECT    */列名（どの項目を）
FROM      表名（どの表から）
WHERE     列名＝値（条件（どの列がどんな値の））;
```

Point

特定の行を指定するには
WHERE句でどんな値を
検索するかを指定

EMP（社員表）

EMP_ID	NAME	HIRE_DATE	SALARY	DEPT_ID
100	佐藤	00-04-01	500000	50
101	鈴木	00-10-01	480000	10
102	高橋	02-01-01	450000	40
103	田中	03-12-01	400000	30
104	渡辺	07-04-01	300000	40
105	伊藤	04-12-01	350000	30
106	山本	01-04-01	450000	20
107	中村	06-11-01	300000	10
108	小林	05-04-01	360000	20
109	吉田	08-04-01	280000	

社員表から部門番号（dept_id）が10である社員の社員名(NAME)、入社日(HIRE_DATE)のデータを検索

```
SQL>SELECT name,hire_date,
  2        dept_id
  3 FROM   emp
  4 WHERE  dept_id=10 ;

NAME       HIRE_DAT    DEPT_ID
---------- -------- ----------
鈴木       00-10-01         10
中村       06-11-01         10
```

基本的なSELECT文

行の選択（文字データの検索）

Point
文字を指定する場合は
単一引用符‘’で囲む

社員表から社員名(NAME)が「佐藤さん」のデータを検索

```
SQL>SELECT name,hire_date,dept_id
  2 FROM    emp
  3 WHERE   name=佐藤;
WHERE   name=佐藤
                  *
行3でエラーが発生しました。:
ORA-00904: "佐藤": 無効な識別子です。
```

```
SQL>SELECT name,hire_date,dept_id
  2 FROM emp
  3 name='佐藤';

   NAME    HIRE_DAT     DEPT_ID
---------- -------- ----------
佐藤       00-04-01          50
```

EMP（社員表）

EMP_ID	NAME	HIRE_DATE	SALARY	DEPT_ID
100	佐藤	00-04-01	500000	50
101	鈴木	00-10-01	480000	10
102	高橋	02-01-01	450000	40
103	田中	03-12-01	400000	30
104	渡辺	07-04-01	300000	40
105	伊藤	04-12-01	350000	30
106	山本	01-04-01	450000	20
107	中村	06-11-01	300000	10
108	小林	05-04-01	360000	20
109	吉田	08-04-01	280000	

基本的なSELECT文

行の選択（日付データの検索）

Point

日付を指定する場合は単一引用符‘’で囲み
Oracleに認識できる日付書式で指定する
日本・・・RR（年下2桁）-MM（月）-DD（日）
米・・・DD（日）-MON（月略称）-RR（年下2桁）

社員表から入社日(HIRE_DATE)が「2008年4月1日」のデータを検索

```
SQL>SELECT name,hire_date,dept_id
  2 FROM   emp
  3 WHERE  hire_date='2008年4月1日';
```

```
WHERE  hire_date='2008年04月01日'
                 *
行3でエラーが発生しました。:
ORA-01861: リテラルが書式文字列と一致しません。
```

```
SQL>SELECT name,hire_date,dept_id
  2 FROM   emp
  3 WHERE  hire_date='08-04-01';

name       HIRE_DAT    DEPT_ID
---------- -------- ----------
吉田       08-04-01
```

EMP（社員表）

EMP_ID	NAME	HIRE_DATE	SALARY	DEPT_ID
100	佐藤	00-04-01	500000	50
101	鈴木	00-10-01	480000	10
102	高橋	02-01-01	450000	40
103	田中	03-12-01	400000	30
104	渡辺	07-04-01	300000	40
105	伊藤	04-12-01	350000	30
106	山本	01-04-01	450000	20
107	中村	06-11-01	300000	10
108	小林	05-04-01	360000	20
109	吉田	08-04-01	280000	

基本的なSELECT文
行の選択（NULLの検索）

Point
データは入っていないこと＝NULL
NULLはスペースとは異なり、データが入っていない状態のこと

NULLデータを検索するには「IS NULL」条件

部門に配属されていない社員を検索

```
SQL> SELECT name, dept_id
  2  FROM   emp
  3  WHERE  dept_id is null;

NAME       DEPT_ID
---------- ----------
吉田
```

```
SQL> SELECT name, dept_id
  2  FROM   emp
  3  WHERE  dept_id = null;
レコードが選択されませんでした。
```

EMP（社員表）

EMP_ID	NAME	HIRE_DATE	SALARY	DEPT_ID
100	佐藤	00-04-01	500000	50
101	鈴木	00-10-01	480000	10
102	高橋	02-01-01	450000	40
103	田中	03-12-01	400000	30
104	渡辺	07-04-01	300000	40
105	伊藤	04-12-01	350000	30
106	山本	01-04-01	450000	20
107	中村	06-11-01	300000	10
108	小林	05-04-01	360000	20
109	吉田	08-04-01	280000	

アジェンダ

- リレーショナル・データベースの特徴
- 基本的なSELECT文
- データを加工するSELECT文
- データ集計を行うSELECT文
- 複数の表の結合
- 副問合せ

データの加工

- データを検索する際、データを加工して表示することが可能
  - 計算式
  - SQL関数
  - 複数列の連結

```
SQL>SELECT 給与×12
  2  FROM   emp;
```

給与×12

データベース内のデータが
変わるわけではない。
表示上のみ

EMP_ID	NAME	HIRE_DATE	SALARY	DEPT_ID
100	佐藤	00-04-01	500000	50
101	鈴木	00-10-01	480000	10
102	高橋	02-01-01	450000	40
103	田中	03-12-01	400000	30
104	渡辺	07-04-01	300000	40
105	伊藤	04-12-01	350000	30
106	山本	01-04-01	450000	20
107	中村	06-11-01	300000	10
108	小林	05-04-01	360000	20
109	吉田	08-04-01	280000	

データの加工

SELECT文内での計算

Point

通常の四則演算と同じ
・＋－より×÷を優先
・( )を使って順序入れ替え

```
SELECT   列名 + - * / (四則演算)
FROM     表名
WHERE    列名＝値;
```

社員表から社員名(name)、月収(SALARY)、年収(SALARYの12倍)のデータを検索

```
SQL>SELECT name,salary,salary*12
  2 FROM   emp;

NAME       SALARY SALARY*12
---------- ------ ---------
佐藤       500000   6000000
鈴木       480000   5760000
高橋       450000   5400000
田中       400000   4800000
.....
```

データの加工

列別名(列の見出し)の指定

```
SELECT   列名/式 △列別名
FROM     表名
WHERE    列名＝値;
```

Point

列別名:表示時に、列名と異なる列見出しを表示することが可能

計算結果などを表示する時に、分かりやすくなる

社員表から社員名(name)、月収(SALARY)、年収(SALARYの12倍)のデータを検索

```
SQL>SELECT name,salary,salary*12 △ann_sal
  2 FROM   emp;

     name          SALARY    ANN_SAL ←列別名
---------- ---------- ----------
佐藤              500000    6000000
鈴木              480000    5760000
高橋              450000    5400000
田中              400000    4800000
......
```

データの加工

関数の指定（数値データの加工例）

Point

データを加工して表示するには、
SQL関数を使用
代表的な関数
TRUNC：切捨て
LENGTH：文字数

```
SELECT   関数(列名/式) 列別名
FROM     表名
WHERE    列名＝値;
```

社員表から社員名(name)、日割り給与(SALARY÷23) を検索

```
SQL> SELECT name,salary/23 day_sal
  2  FROM   emp;

NAME       DAY_SAL
---------- ----------
佐藤        21739.1304
```

小数の桁数を指定

```
SQL> SELECT name, TRUNC(salary/23,0) day_sal
  2  FROM   emp;

NAME       DAY_SAL
---------- ----------
佐藤             21739
```

データの加工

関数の指定(文字データの加工例)

Point

データを加工して表示するには、
SQL関数を使用
代表的な関数
　TRUNC:切捨て
　LENGTH:文字数

```
SELECT   関数(列名/式) 列別名
FROM     表名
WHERE    列名＝値;
```

社員表から社員名(name)の文字数を検索

```
SQL> SELECT name,LENGTH(name) name_length
  2  FROM   emp;

NAME       NAME_LENGTH
---------- -----------
佐藤                 2
鈴木                 2
高橋                 2
田中                 2
...
```

その他の関数についての詳細は以下のURLをご参照ください。
「Oracle Database SQL言語リファレンス」
http://download.oracle.com/docs/cd/E16338_01/server.112/b56299/toc.htm

データの加工

複数列の連結

```
SELECT   列名||列名||文字/式 △列別名
FROM     表名
WHERE    列名＝値;
```

Point

列指定の間に || を指定することにより、別々の列や式を一つの列のように表示することができる

社員名に「さん」をつけて表示

```
SQL> SELECT name || 'さん' name
  2  FROM   emp;

NAME
--------------
佐藤さん
鈴木さん
高橋さん
田中さん
.....
```

データの加工

データを加工したレポート形式の出力

社員表から社員の名前と年収を「～さんの年収は～です」という形式で表示

```
SQL> SELECT name || 'さんの年収は'
  2          || salary*12 || 'です。' report
  3  FROM    emp;

REPORT
----------------------------
佐藤さんの年収は6000000です。
鈴木さんの年収は5760000です。
高橋さんの年収は5400000です。
田中さんの年収は4800000です。
.....
```

アジェンダ

- リレーショナル・データベースの特徴
- 基本的なSELECT文
- データを加工するSELECT文
- データ集計を行うSELECT文
- 複数の表の結合
- 副問合せ

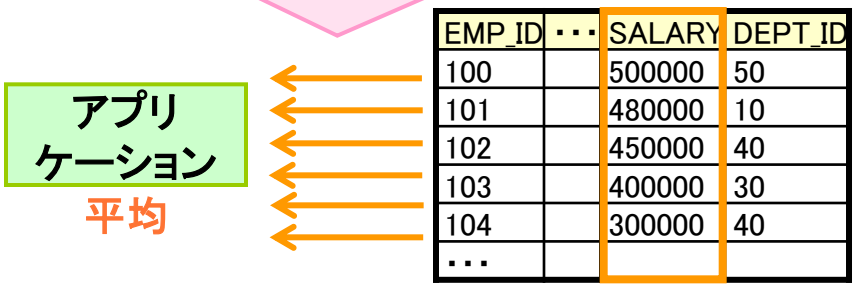
データの集計

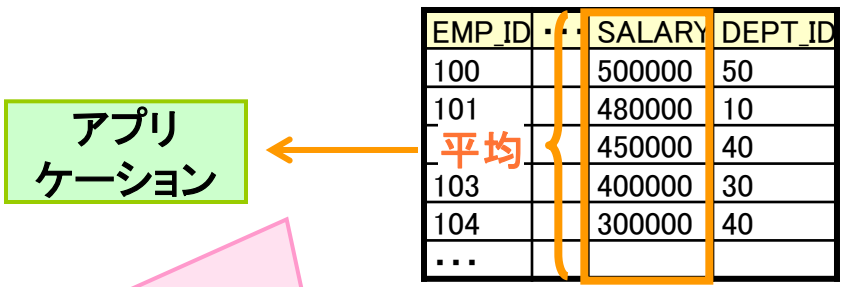
Point

複数のデータを集計して、合計や平均などを出すことが可能
・グループ関数
・グループ化
（部門ごと、地域ごと、商品ごとなど）

- 集計関数を使用し、データベース内でデータを集計して取得
  - 行数のカウント
  - 最大値/最小値/平均値/合計
  - グループに分けて、グループごとに集計結果を取得
  - 部門ごと、商品ごと、地域ごと　etc

全行のデータを取得し、アプリケーション側で平均値などを出すことも可能だが・・・





データベースで平均値を出し、平均値のみをアプリケーションに送ったほうが効率的

データの集計
グループ関数（数値データの集計例）

Point
集計結果データを取得
・グループ関数
・グループ化
（部門ごと、地域ごと、商品ごとなど）

```
SELECT   グループ関数（列名）
FROM     表名
WHERE    条件;
```

社員表から、社員の平均給与と合計給与を調べる

```
SQL>SELECT AVG(salary),
           SUM(salary)
  2  FROM   emp;

AVG(SALARY) SUM(SALARY)
----------- -----------
     387000     3870000
```

グループ関数	説明
COUNT	行数 （条件に合う行が何行あるか）
MAX	最大値
MIN	最小値
AVG	平均
SUM	合計

データの集計
データのグループ化

Point

集計結果データを取得
・グループ関数
・グループ化
（部門ごと、地域ごと、商品ごとなど）

```
SELECT   グループ関数(列名)
FROM     表名
GROUP BY 集計する列(どの項目で集計するか);
```

社員表から、部門ごとの平均給与、合計給与を調べる

```
SQL>SELECT dept_id,AVG(salary),SUM(salary)
  2 FROM   emp
  3 GROUP BY dept_id;

 DEPT_ID    AVG(SALARY) SUM(SALARY)
---------- ----------- -----------
10              390000      780000
20              405000      810000
30              375000      750000
40              375000      750000
50              500000      500000
                280000      280000
```

データの集計
グループ化構文の注意点

```
SELECT   グループ関数(列名)
FROM     表名
GROUP BY 集計する列(どの項目で集計するか);
```

社員表から、部門ごとの平均給与、合計給与を調べる

```
SQL>SELECT name,AVG(salary),SUM(salary)
  2 FROM   emp
  3 GROUP BY dept_id ;

SELECT name,AVG(salary),SUM(salary)
       *
行1でエラーが発生しました。:
ORA-00979: GROUP BYの式ではありません。
```

SELECT句に指定する列（結果として表示する列）は、GROUP BYで指定する必要がある

アジェンダ

- リレーショナル・データベースの特徴
- 基本的なSELECT文
- データを加工するSELECT文
- データ集計を行うSELECT文
- 複数の表の結合
- 副問合せ

複数の表の結合

- リレーショナル・データベースでは、複数の表でデータを管理
  →複数の表に関連するデータが入っている
  - 関連する複数表から、データを取得＝結合
  - 複数の表を関連付けるための「結合条件」を指定

リレーショナル・データベースでは
設計時に関連表に同じ列を持たせ
関係が分かるようにしておく
（ID列などを使用）

社員表と部門表を結合する場合
（社員の所属している部門を調べる等）

EMP（社員表）

EMP_ID	NAME	HIRE_DATE	SALARY	DEPT_ID
100	佐藤	00-04-01	500000	50
101	鈴木	00-10-01	480000	10
102	高橋	02-01-01	450000	40
103	田中	03-12-01	400000	30
104	渡辺	07-04-01	300000	40
105	伊藤	04-12-01	350000	30
…				

Point

DEPT（部門表）

DEPT_ID	DEPT_NAME
10	営業
20	経理
30	人事
40	IT企画
50	経営企画
60	環境対策

複数の表の結合
様々な結合構文

- 結合構文の種類
  - Oracle独自の結合構文
  - SQL:2003(標準規格)に準拠した結合構文

Point
複数の結合構文があり
どちらを使用してもよい

Oracle独自の結合構文

```
SELECT   表1.列名, 表2.列名
FROM     表1,表2
WHERE    表1.列＝表2.列 ;
```

SQL:2003準拠の結合構文

```
SELECT   表1.列名, 表2.列名
FROM     表1 JOIN 表2
  { ON       表1.列＝表2.列
    USING    共通列 ;
```

	Oracle結合構文	SQL:2003準拠構文
SELECT句	複数列から、必要な列名を指定（列名の前に表名を指定）	
FROM句	カンマ区切りで複数の表を指定	表名の間にJOIN句を指定
結合条件	WHERE句で指定	ON、USINGなど複数の指定がある

Oracle構文を使った結合

Oracle構文の基本構文

Oracle構文

```
SELECT   表1.列名, 表2.列名
FROM     表1,表2
WHERE    表1.列＝表2.列;
```

Point

Oracle構文の特徴
- FROM句にカンマ区切りで複数表を指定
- WHERE句で結合条件を指定

社員名(name)と、その人の所属する部門名(DEPT_NAME)を表示

EMP(社員表)

EMP_ID	NAME	･･	DEPT_ID
100	佐藤		50
101	鈴木		10
102	高橋		40
103	田中		30
104	渡辺		40
105	伊藤		30
106	山本		20
107	中村		10
108	小林		20
109	吉田		

DEPT(部門表)

DEPT_ID	DEPT_NAME
10	営業
20	経理
30	人事
40	IT企画
50	経営企画
60	環境対策

```
SQL> SELECT emp.name, dept.dept_name
  2  FROM   emp, dept
  3  WHERE  emp.dept_id=dept.dept_id;

NAME       DEPT_NAME
---------- ----------
佐藤       経営企画
鈴木       営業
高橋       IT企画
田中       人事
渡辺       IT企画
伊藤       人事
山本       経理
中村       営業
小林       経理
```

Oracle構文を使った結合

内部結合と外部結合

- 内部結合
  - 結合列のデータが一致する結果だけを出力

社員名(name)と、その人の所属する部門名(DEPT_NAME)を表示

EMP(社員表)　　DEPT(部門表)

EMP_ID	NAME	‥	DEPT_ID
100	佐藤		50
101	鈴木		10
102	高橋		40
103	田中		30
104	渡辺		40
105	伊藤		30
106	山本		20
107	中村		10
108	小林		20
109	吉田		

DEPT_ID	DEPT_NAME
10	営業
20	経理
30	人事
40	IT企画
50	経営企画
60	環境対策

```
SQL> SELECT emp.name, dept.dept_name
  2  FROM   emp, dept
  3  WHERE  emp.dept_id=dept.dept_id;

name       DEPT_NAME
---------- ----------
佐藤       経営企画
鈴木       営業
高橋       IT企画
田中       人事
渡辺       IT企画
伊藤       人事
山本       経理
中村       営業
小林       経理
```

部門の決まっていない社員(吉田さん)や、従業員のいない部門(環境対策部)は検索されない→内部結合

Oracle構文を使った結合

外部結合（部門データのない社員も表示）

- 外部結合
  - 結合列のデータが一致しない結果も含めて出力

部門の決まっていない社員（吉田さん）も含めて
社員名(name)と、その人の所属する部門名（DEPT_NAME）を表示

EMP（社員表）　　DEPT（部門表）

EMP_ID	NAME	･･	DEPT_ID
100	佐藤		50
101	鈴木		10
102	高橋		40
103	田中		30
104	渡辺		40
105	伊藤		30
106	山本		20
107	中村		10
108	小林		20
109	吉田		

DEPT_ID	DEPT_NAME
10	営業
20	経理
30	人事
40	IT企画
50	経営企画
60	環境対策

```
SQL> SELECT emp.name, dept.dept_name
  2  FROM   emp, dept
  3  WHERE  emp.dept_id=dept.dept_id(+);

name       DEPT_NAME
---------- ----------
中村       営業
鈴木       営業
小林       経理
山本       経理
伊藤       人事
田中       人事
渡辺       IT企画
高橋       IT企画
佐藤       経営企画
吉田
```

外部結合では、WHERE句に（＋）を指定

部門表に対応データがない（社員表にしかない）
→部門表側に（＋）を指定

Oracle構文を使った結合

外部結合（社員データのない部門も表示）

- 外部結合
  - 結合列のデータが一致しない結果も含めて出力

社員の配属されていない部署（環境対策）も含めて
社員名(name)と、その人の所属する部門名（DEPT_NAME）を表示

EMP（社員表）　DEPT（部門表）

EMP_ID	NAME	‥	DEPT_ID
100	佐藤		50
101	鈴木		10
102	高橋		40
103	田中		30
104	渡辺		40
105	伊藤		30
106	山本		20
107	中村		10
108	小林		20
109	吉田		

DEPT_ID	DEPT_NAME
10	営業
20	経理
30	人事
40	IT企画
50	経営企画
60	環境対策

```
SQL> SELECT emp.name, dept.dept_name
  2  FROM   emp, dept
  3  WHERE  emp.dept_id(+)=dept.dept_id;

name       DEPT_NAME
---------- ----------
佐藤       経営企画
鈴木       営業
高橋       IT企画
田中       人事
渡辺       IT企画
伊藤       人事
山本       経理
中村       営業
小林       経理
           環境対策
```

外部結合では、WHERE句に（＋）を指定

社員表に対応データがない（部門表にしかない）
→社員表側に（＋）を指定

SQL構文を使った結合

USING句の使用

SQL構文（USING句）

```
SELECT   表1.列名, 表2.列名
FROM     表1 JOIN 表2
USING    (共通列);
```

Point

SQL構文の特徴
- FROM句にJOIN句で複数表を指定
- 結合条件はいくつかの書き方がある

社員名(name)と、その人の所属する部門名（DEPT_NAME）を表示

EMP（社員表）　　DEPT（部門表）

EMP_ID	NAME	‥	DEPT_ID
100	佐藤		50
101	鈴木		10
102	高橋		40
103	田中		30
104	渡辺		40
105	伊藤		30
106	山本		20
107	中村		10
108	小林		20
109	吉田		

DEPT_ID	DEPT_NAME
10	営業
20	経理
30	人事
40	IT企画
50	経営企画
60	環境対策

```
SQL> SELECT emp.name, dept.dept_name
  2  FROM   emp JOIN dept
  3  USING  (dept_id);

name       DEPT_NAME
---------- ----------
佐藤       経営企画
鈴木       営業
高橋       IT企画
田中       人事
渡辺       IT企画
伊藤       人事
山本       経理
中村       営業
小林       経理
```

USING句で共通列を指定
簡単だが、列名が異なる場合は指定できない

SQL構文を使った結合
ON句の使用

SQL構文（ON句）

```
SELECT   表1.列名, 表2.列名
FROM     表1 JOIN 表2
ON       表1.列＝表2.列;
```

Point

SQL構文の特徴
- FROM句にJOIN句で複数表を指定
- 結合条件はいくつかの書き方がある

社員名(name)と、その人の所属する部門名（DEPT_NAME）を表示

EMP（社員表）

EMP_ID	NAME	‥	DEPT_ID
100	佐藤		50
101	鈴木		10
102	高橋		40
103	田中		30
104	渡辺		40
105	伊藤		30
106	山本		20
107	中村		10
108	小林		20
109	吉田		

DEPT（部門表）

DEPT_ID	DEPT_NAME
10	営業
20	経理
30	人事
40	IT企画
50	経営企画
60	環境対策

```
SQL> SELECT emp.name, dept.dept_name
  2  FROM   emp JOIN dept
  3  ON     emp.dept_id=dept.dept_id;

name       DEPT_NAME
---------- ----------
佐藤       経営企画
鈴木       営業
高橋       IT企画
田中       人事
渡辺       IT企画
伊藤       人事
山本       経理
中村       営業
小林       経理
```

ON句で共通列を「＝」指定

SQL構文を使った結合

外部結合（部門データのない社員も表示）

- 外部結合
  - どの表のデータを全て検索するか、「LEFT」「RIGHT」「FULL」で指定

部門の決まっていない社員（吉田さん）も含めて
社員名(name)と、その人の所属する部門名（DEPT_NAME）を表示

EMP（社員表）

EMP_ID	NAME	‥	DEPT_ID
100	佐藤		50
101	鈴木		10
102	高橋		40
103	田中		30
104	渡辺		40
105	伊藤		30
106	山本		20
107	中村		10
108	小林		20
109	吉田		

DEPT（部門表）

DEPT_ID	DEPT_NAME
10	営業
20	経理
30	人事
40	IT企画
50	経営企画
60	環境対策

```
SQL> SELECT emp.name, dept.dept_name
  2  FROM   emp LEFT OUTER JOIN dept
  3  ON     emp.dept_id=dept.dept_id;
```

```
name       DEPT_NAME
---------- ----------
中村       営業
鈴木       営業
小林       経理
山本       経理
伊藤       人事
田中       人事
渡辺       IT企画
高橋       IT企画
佐藤       経営企画
吉田
```

社員表のデータを全て表示
（部門表に関連データがないものも）
→社員表側を指定
（この場合empがJOINの左側にあるので、LEFTを指定）

SQL構文を使った結合
外部結合（社員データのない部門も表示）

- 外部結合
  - どの表のデータを全て検索するか、「LEFT」「RIGHT」「FULL」で指定

社員の配属されていない部署（環境対策）も含めて
社員名(name)と、その人の所属する部門名（DEPT_NAME）を表示

EMP（社員表）

EMP_ID	NAME	・・	DEPT_ID
100	佐藤		50
101	鈴木		10
102	高橋		40
103	田中		30
104	渡辺		40
105	伊藤		30
106	山本		20
107	中村		10
108	小林		20
109	吉田		

DEPT（部門表）

DEPT_ID	DEPT_NAME
10	営業
20	経理
30	人事
40	IT企画
50	経営企画
60	環境対策

```
SQL> SELECT emp.name, dept.dept_name
  2  FROM   emp RIGHT OUTER JOIN dept
  3  emp.dept_id=dept.dept_id;
```

```
name       DEPT_NAME
---------- ----------
佐藤       経営企画
鈴木       営業
高橋       IT企画
田中       人事
渡辺       IT企画
伊藤       人事
山本       経理
中村       営業
小林       経理
           環境対策
```

部門表のデータを全て表示
（社員表に関連データがないものも）
→部門表側を指定
（この場合deptがJOINの右側に
あるので、RIGHTを指定）

SQL構文を使った結合

外部結合（部門データのない社員も社員データのない部門も表示）

- 外部結合
  - どの表のデータを全て検索するか、「LEFT」「RIGHT」「FULL」で指定

部門の決まっていない社員（吉田さん）と社員の配属されていない部署（環境対策）も含めて、社員名(name)と、その人の所属する部門名（DEPT_NAME）を表示

EMP（社員表）

EMP_ID	NAME	‥	DEPT_ID
100	佐藤		50
101	鈴木		10
102	高橋		40
103	田中		30
104	渡辺		40
105	伊藤		30
106	山本		20
107	中村		10
108	小林		20
109	吉田		

DEPT（部門表）

DEPT_ID	DEPT_NAME
10	営業
20	経理
30	人事
40	IT企画
50	経営企画
60	環境対策

```
SQL> SELECT emp.name, dept.dept_name
  2    FROM   emp FULL OUTER JOIN dept
  3    emp.dept_id=dept.dept_id;

name       DEPT_NAME
---------- ----------
佐藤       経営企画
鈴木       営業
高橋       IT企画
田中       人事
渡辺       IT企画
伊藤       人事
山本       経理
中村       営業
小林       経理
吉田
           環境対策
```

対応するデータがない場合も、全てのデータを表示
→FULL を指定

複数の表の結合
まとめ

	Oracle構文	SQL構文
内部結合	SELECT 表1.列名, 表2.列名 FROM 表1,表2 WHERE 表1.列＝表2.列;	SELECT 表1.列名, 表2.列名 FROM 表1 JOIN 表2 USING (共通列);
		SELECT 表1.列名, 表2.列名 FROM 表1 JOIN 表2 ON 表1.列＝表2.列;
外部結合	SELECT 表1.列名, 表2.列名 FROM 表1,表2 WHERE 表1.列＝表2.列(+);	SELECT 表1.列名, 表2.列名 FROM 表1 LEFT OUTER JOIN 表2 ON 表1.列＝表2.列;
	SELECT 表1.列名, 表2.列名 FROM 表1,表2 WHERE 表1.列(+)＝表2.列;	SELECT 表1.列名, 表2.列名 FROM 表1 RIGHT OUTER JOIN 表2 ON 表1.列＝表2.列;
		SELECT 表1.列名, 表2.列名 FROM 表1 FULL OUTER JOIN 表2 ON 表1.列＝表2.列;

・外部結合では、左右の指定の仕方が異なる
・Oracle構文では「FULL OUTER」に相当する構文はない

アジェンダ

- リレーショナル・データベースの特徴
- 基本的なSELECT文
- データを加工するSELECT文
- データ集計を行うSELECT文
- 複数の表の結合
- 副問合せ

副問合せ

- 他の問合せの結果を使って、ある問合せをするための構文

- 小林さんよりも入社日が遅い人は誰か？
  - 求めたい結果データ＝誰か？（社員名）
  - 条件＝小林さんより入社日が遅い
  - 小林さんの入社日は？
- 社員の中で一番給与が高い人は誰か？
  - 求めたい結果データ＝誰か？（社員名）
  - 条件＝社員の中で一番給与が高い
  - 一番高い給与は？

Point

求めたい結果（主問合せ）を出すための、副次的な問合せが「副問合せ」

2つのSELECT文を入れ子にして記述
副問合せは（ ）で囲んで内側に書く

```
SELECT   列名, 列名・・・
FROM     表名
WHERE    列名  比較演算子 （ SELECT   列名
                            FROM     表名 ・・・ ）;
```

副問合せ

副問合せの構文

小林さんより入社日が遅い(大きい)人は誰か

```
SQL>SELECT name , hire_date
  2 FROM   emp
  3 WHERE  hire_date>( SELECT hire_date
  4                    FROM   emp
  5                    WHERE  name='小林' );

name       HIRE_DAT
---------- --------
渡辺       07-04-01
中村       06-11-01
吉田       08-04-01
```

Oracleが処理する際には、まず副問合せを実行し、その結果を使って主問合せを実行する

社員の中で一番給与の高い人は誰か

```
SQL>SELECT name
  2 FROM   emp
  3 WHERE  salary = ( SELECT max(salary)
                      FROM emp );

name
----------
佐藤
```

総復習

社員表から、部門名と、部門ごとの平均給与、合計給与を調べる

```
SQL>SELECT dept.dept_name,
  2        AVG(emp.salary),SUM(emp.salary)
  3 FROM   emp,dept
  4 WHERE  emp.dept_id=dept.dept_id
  5 GROUP BY dept_name ;

DEPT_NAME   AVG(SALARY) SUM(SALARY)
---------- ----------- -----------
経理             405000      810000
IT企画           375000      750000
営業             390000      780000
人事             375000      750000
経営企画         500000      500000
```

集計関数

給与（社員表）と部門名（部門表）を使うために結合

「部門名」でグループ化し部門名を表示

まとめ

- リレーショナル・データベースの特徴
- 基本的なSELECT文
- データを加工するSELECT文
- データ集計を行うSELECT文
- 複数の表の結合
- 副問合せ

OTN×ダイセミ　でスキルアップ!!

・一般的な技術問題解決方法などを知りたい！
・セミナ資料など技術コンテンツがほしい！

Oracle Technology Network(OTN)を御活用下さい。

http://otn.oracle.co.jp/forum/index.jspa?categoryID=2

一般的技術問題解決にはOTN掲示版の
「データベース一般」をご活用ください

※OTN掲示版は、基本的にOracleユーザー有志からの回答となるため100%回答があるとは限りません。
ただ、過去の履歴を見ると、質問の大多数に関してなんらかの回答が書き込まれております。

http://www.oracle.com/technology/global/jp/ondemand/otn-seminar/index.html

過去のセミナ資料、動画コンテンツはOTNの
「OTNセミナー　オンデマンド コンテンツ」へ

※ダイセミ事務局にダイセミ資料を請求頂いても、お受けできない可能性がございますので予めご了承ください。
ダイセミ資料はOTNコンテンツ　オン　デマンドか、セミナ実施時間内にダウンロード頂くようお願い致します。

ORACLE

OTNセミナー　オンデマンド　コンテンツ

ダイセミで実施された技術コンテンツを動画で配信中!!
ダイセミのライブ感はそのままに、お好きな時間で受講頂けます。

twitter

最新情報つぶやき中
oracletechnetjp
・人気コンテンツは？
・お勧め情報
・公開予告　など

OTN　オンデマンド　検索

※掲載のコンテンツ内容は予告なく変更になる可能性があります。
期間限定での配信コンテンツも含まれております。お早めにダウンロード頂くことをお勧めいたします。

ORACLE

Oracle エンジニアのための技術情報サイト

オラクルエンジニア通信

http://blogs.oracle.com/oracle4engineer/

twitter

最新情報つぶやき中
oracletechnetjp

- 技術資料
- ダイセミの過去資料や製品ホワイトペーパー、スキルアップ資料などを多様な方法で検索できます
- キーワード検索、レベル別、カテゴリ別、製品・機能別
- コラム
- オラクル製品に関する技術コラムを毎週お届けします
- 決してニッチではなく、誰もが明日から使える技術の「あ、そうだったんだ！」をお届けします

オラクルエンジニア通信

先月はこんな資料が人気でした

✓ Oracle Database 11gR2 RAC インストレーション・ガイド ASM 版 Microsoft Windows x86-64

✓ Oracle Database 11gR2 旧バージョンからのアップグレード

Oracle Direct 新サービスができました

新規Oracle Direct Concierge

（無償支援サービス）

•WebLogic Serverバージョンアップ支援サービス

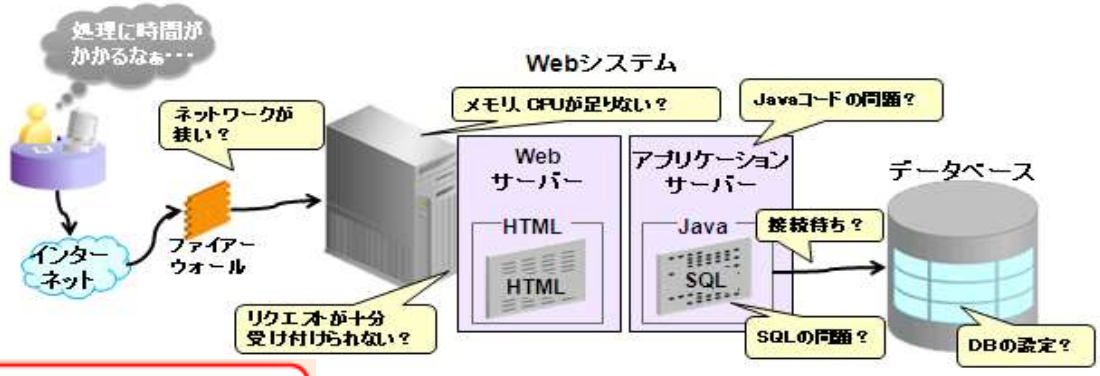
旧WebLogic ServerからWebLogic Server 11g への移行を検討しているお客様へ、
お客様の環境にあった移行の手順や、注意点をアドバイス致します。

•Oracle Application Server、その他アプリケーションサーバーからのOracle WebLogic Server への移行支援サービス

Oracle Application Serverや、その他のアプリケーションサーバーから、WebLogic Server 11g への
移行を検討してるお客様へ、お客様の環境にあった移行の手順や、注意点をアドバイス致します。

•Webシステム　ボトルネック診断サービス

Webシステムの性能劣化に悩まされているお客様へ、
お客様の環境情報を基に問題の切り分けとアドバイスを致します。



Oracle Direct 0120-155-096

お問い合わせフォーム
http://www.oracle.co.jp/inq_pl/INQUIRY/quest?rid=28

オラクル クルクルキャンペーン

2010年
11月30日まで

あのOracle Database Enterprise Editionが超おトク!!

お得な買い方
オラクル5年分

- ライセンス使用期間　を5年間に設定
- 初期のライセンスコストがなんと67%OFF　！
- テクニカル・サポート価格も53%OFF　！

Oracle Databaseの
ライセンス価格を大幅に抑えて
ご導入いただけます

多くのお客様でサーバー使用期間とされる
5年間にライセンス期間を限定
・期間途中で永久ライセンスへ差額移行
・5年後に新規ライセンスを購入し継続利用
・5年後に新システムへデータを移行

Enterprise Editionはここが違う!!
- 圧倒的なパフォーマンス!
- データベース管理がカンタン!
- データベースを止めなくていい!
- もちろん障害対策も万全!

この機能でこの価格
ライセンスパック

- Oracle Databaseの機能を存分に使える！
- 2ノードRAC構成も可能！
- サーバー構成によって計4種類のパックから選べる！

詳しくはコチラ
http://www.oracle.co.jp/campaign/kurukuru/index.html

Oracle Direct 0120-155-096

お問い合わせフォーム
http://www.oracle.co.jp/inq_pl/INQUIRY/quest?rid=28

ORACLE

あなたにいちばん近いオラクル

Oracle Direct

まずはお問合せください

Oracle Direct　検索

システムの検討・構築から運用まで、ITプロジェクト全般の相談窓口としてご支援いたします。
システム構成やライセンス/購入方法などお気軽にお問い合わせ下さい。

Web問い合わせフォーム

専用お問い合わせフォームにてご相談内容を承ります。
http://www.oracle.co.jp/inq_pl/INQUIRY/quest?rid=28

※フォームの入力には、Oracle Direct Seminar申込時と同じ
ログインが必要となります。
※こちらから詳細確認のお電話を差し上げる場合がありますので、ご登録さ
れている連絡先が最新のものになっているか、ご確認下さい。

フリーダイヤル

0120－155－096

※月曜~金曜　9:00~12:00、13:00~18:00
（祝日および年末年始除く）

ORACLE