LPI-JAPAN
Linux Professional Institute Japan

OSS-DB Exam Silver 技術解説セミナー

2012/7/25

特定非営利活動法人エルピーアイジャパン
テクノロジー・マネージャー
松田　神一

Agenda

OSS-DB

- OSS-DB技術者認定試験の概要
- ポイント解説：運用管理
- ポイント解説：SQL
- OSS-DB Exam Silverの例題

LPI-Japanについて

OSS-DB

- Linux Professional Institute Japan（本部はカナダ）
- Linux/OSS技術者の技術力の認定制度の運用を通じて、日本のLinux/OSS技術者の育成、Linux/OSSビジネスの促進に寄与する活動を展開するNPO法人

- 2000年から、Linux技術者認定試験LPICを実施
- 2011年7月から、オープンソースデータベース技術者認定試験OSS-DBを実施

自己紹介

OSS-DB

- 松田　神一（まつだ　しんいち）
  LPI-JAPAN テクノロジー・マネージャー

- NEC、オラクル、トレンドマイクロなどで約20年間、ソフトウェア開発に従事（専門はアプリケーション開発）
  うち10年間はデータベース、およびデータベースアプリケーションの開発（Oracle、C言語、SQL言語）

- 2010年7月から現職

OSS-DB技術者認定試験の概要

OSS-DB技術者認定試験の概要

OSS-DB

認定の種類

- Silver (ベーシックレベル)
  - OSS-DB Exam Silverに合格すれば認定される
- Gold (アドバンストレベル)
  - OSS-DB Silverの認定を取得し、OSS-DB Exam Goldに合格すれば認定される

Silver認定の基準

- データベースの導入、DBアプリケーションの開発、DBの運用管理ができること
- OSS-DBの各種機能やコマンドの目的、使い方を正しく理解していること

Gold認定の基準

- トラブルシューティング、パフォーマンスチューニングなどOSS-DBに関する高度な技術を有すること
- コマンドの出力結果などから、必要な情報を読み取る知識やスキルがあること

OSS-DB Exam Silverの出題範囲

OSS-DB

- 一般知識 (20%)
  - OSS-DBの一般的特徴
  - ライセンス
  - コミュニティと情報収集
  - RDBMSに関する一般的知識
- 運用管理 (50%)
  - インストール方法
  - 標準付属ツールの使い方
  - 設定ファイル
  - バックアップ方法
  - 基本的な運用管理作業
- 開発/SQL (30%)
  - SQLコマンド
  - 組み込み関数
  - トランザクションの概念

出題範囲に関する注意事項

OSS-DB

- 最新の出題範囲は
  http://www.oss-db.jp/outline/examarea.shtml
  で確認できる

- 前提とするRDBMSはPostgreSQL 9.0

- SilverではOSに依存する問題は出題しないが、記号や用語がOSによって異なるものについては、Linuxのものを採用している
  - OSのコマンドプロンプトには $ を使う
  - 「フォルダ」ではなく「ディレクトリ」と呼ぶ
  - ディレクトリの区切り文字には / を使う

- 出題範囲に関するFAQ
  http://www.oss-db.jp/faq/#n02

傾向と対策

OSS-DB

- Silverの合格基準は、各機能やコマンドについて
  - その目的を正しく理解していること
    - XXXコマンドを使うと何が起きるか
    - YYYをするためにはどのコマンドを使えば良いか
  - 利用法を正しく理解していること
    - コマンドのオプションやパラメータ
    - 設定ファイルの記述方法
- 基本的な出題形式は
  - 最も適切なものを1つ(2つ)選びなさい
  - 誤っているものを1つ(2つ)選びなさい
- 出題範囲にあるすべての項目について、試験問題が用意されている
- 出題範囲詳細に載っている項目すべてについて、マニュアルなどで調査した上で、実際に試して理解する
  - 実機で試すことは極めて重要

ポイント解説：運用管理

データベースのバックアップ

OSS-DB

- データベースでは重要なデータを管理している。ディスクの故障などによるデータの損失に備え、バックアップを取得することが重要

- データベースではメモリ上のデータ（キャッシュ）が最新。キャッシュとディスク上のデータファイルの内容が一致するとは限らない、つまり、OSコマンドを使ってファイルをコピーしてもバックアップにはならない
  - データベースのバックアップには特殊な方法が必要

- データベースがクラッシュしたとき、一週間前のバックアップからデータベースが復元（リストア）できても、ありがたくないかもしれない
  - クラッシュ直前の状態にデータを復旧（リカバリ）するためのバックアップ手段がある

- バックアップの方法とリストア・リカバリの方法をセットで覚えること
  - バックアップを作っても、いざというときに使えなければ役に立たない

バックアップの手段

OSS-DB

- pg_dump コマンド
  - データベース単位でバックアップを作成
  - psql または pg_restore コマンドを使ってリストア
- pg_dumpall コマンド
  - データベースクラスタ全体のバックアップを作成
  - psql コマンドを使ってリストア
- コールドバックアップ (ディレクトリコピー)
  - OS付属のコピー、アーカイブ用コマンドを使ってバックアップを作成
  - 簡単で確実な方法だが、データベースを停止する必要がある
- ポイント・イン・タイム・リカバリ (PITR)
  - 使い方がやや複雑
  - WAL (Write Ahead Logging) 機能と組み合わせて、任意の時点にリカバリ可能
- COPY 文、\copy メタコマンド
  - テーブル単位でCSV形式ファイルの入出力

pg_dumpによるバックアップとリストア

OSS-DB

- データベースを停止せずに、データベース単位のバックアップを取得
  - `$ pg_dump [options] -f dumpfilename dbname` あるいは
  - `$ pg_dump [options] dbname > dumpfilename`
  - `-F` オプションで、出力形式を指定できる。`p` (`plain`) はテキスト形式 (デフォルト)、`c` (`custom`) はカスタム (バイナリ) 形式、`t` (`tar`) はTAR形式
  - データベースクラスタ内のすべてのデータベースのバックアップを取得するには、`pg_dumpall` コマンドを使う。(出力形式はテキストのみ)

- テキスト形式 (p) のバックアップは `psql` コマンドで、バイナリ形式 (c/t) のバックアップは `pg_restore` コマンドでリストアする。
  - `$ psql -f dumpfilename dbname` あるいは
  - `$ psql dbname < dumpfilename`
  - `$ pg_restore -d dbname dumpfilename`

- `pg_dump` が作成するテキスト形式のバックアップはSQLのスクリプト (`CREATE TABLE`, `COPY` など) となっており、エディタで修正可能

pg_dumpall によるバックアップとリストア

OSS-DB

- データベースを停止せずに、データベースクラスタ全体のバックアップを取得
  - `$ pg_dumpall [options] -f dumpfilename` あるいは
  - `$ pg_dumpall [options] > dumpfilename`
- ユーザ情報などのグローバルオブジェクトもバックアップ可能（pg_dump では取得できない）
  - `-g` オプションを指定すると、グローバルオブジェクトのみバックアップする
- 出力フォーマットはテキスト形式のみなので `psql` コマンドでリストアする。データベース名は任意。空のクラスタにロードするときは `postgres` を指定すればよい
  - `$ psql -f dumpfilename postgres` あるいは
  - `$ psql postgres < dumpfilename`

pg_dump/pg_dumpall コマンド（まとめ）

OSS-DB

- pg_dump の -F オプションで出力ファイルのフォーマットを指定
  - p（plain、デフォルト）はテキスト形式
    - CREATE TABLE、COPY などの SQL スクリプトが出力される
    - --inserts オプションを指定すると、COPY 文の代わりに INSERT 文を使うので、他のデータベースへのデータインポートにも利用可能
  - c（custom）は圧縮されたアーカイブ形式（バイナリ）
    - マルチプロセスによる高速リストアが可能
  - t（tar）はLinuxなどのTARによるアーカイブ形式（バイナリ）
    - リストア用の SQL スクリプトと、各テーブルごとのデータファイルがTAR形式で1つのファイルにアーカイブされている
- pg_dumpall はテキスト形式のみ
  - -g オプションを指定すると、グローバルデータのみのバックアップ
- テキスト形式は psql、バイナリ形式は pg_restore でリストア

コールドバックアップ

OSS-DB

ディレクトリコピーによるバックアップ

- データベースを停止すれば、物理的なデータファイルをディレクトリごとコピーすることでバックアップを作成できる。(コールドバックアップ)
- コピーの方法は自由に選んで良い。(cp, tar, cpio, zip…)
  - `$ cp -r data backupdir`
  - `$ tar czf backup.tgz data`
- 簡単で確実な方法だが、頻繁には実行できない

バックアップを、同じ構成の別のマシンにコピーして動かすこともできる

- バックアップ作成と逆のことをすればリストアできる
  - `$ cp -r backupdir data`
  - `$ tar xzf backup.tgz`
- コピー元とコピー先で、PostgreSQLのメジャーバージョンが一致していること

参考:コールドバックアップに対し、データベースの稼働中に取得するバックアップをホットバックアップと呼ぶ

ポイント・イン・タイム・リカバリ (PITR)

OSS-DB

■PITR (Point In Time Recovery)

- 障害の直前の状態までデータを復旧 (リカバリ) できる。
- 間違ってデータを削除した場合でも、任意の時点まで戻すことができる。

■PITRの仕組み

- WAL (Write Ahead Logging) により、データファイルへの書き込み前に、変更操作についてログ出力される。(トランザクションログ)
- WALファイルをアーカイブして保存しておく
- 最後のバックアップ (ベースバックアップ) に対して、障害発生直前までのWALを適用することで、データを復旧できる。

■PITRによるベースバックアップの取得手順

- スーパーユーザで接続し、バックアップ開始をサーバに通知
  - `=# SELECT pg_start_backup('label');`
- `tar`, `cpio` などのOSコマンドでバックアップを取得 (サーバーは止めない)
- 再度、スーパーユーザで接続し、バックアップ終了をサーバに通知
  - `=# SELECT pg_stop_backup();`
- (参考) PostgreSQL 9.1では `pg_basebackup` コマンドにより、上記の手順をまとめて実行できる
- (参考) レプリケーションはPITRと同じ原理で動作している。同じ手順でベースバックアップを取得し、WALデータを転送して適用することでデータベースを複製している

ポイント解説：SQL

主なデータ型（表の列に使用）

OSS-DB

数値型

- SMALLINT（2バイト）、INTEGER（4バイト）、BIGINT（8バイト）
- NUMERIC（最大1000桁）、DECIMAL（NUMERIC と同じ）
- REAL（4バイト）、DOUBLE PRECISION（8バイト）
- SERIAL（自動増分4バイト）、BIGSERIAL（自動増分8バイト）

文字列型

- CHARACTER VARYING（可変長、最大4096文字）、VARCHAR（CHARACTER VARYING と同じ）
- CHARACTER（固定長）、CHAR（CHARACTER と同じ）
- TEXT（可変長、無制限）

日付型

- DATE（日付のみ）
- TIME（時刻のみ）
- TIMESTAMP（日付+時刻）

論理値型

- BOOLEAN（TRUE/FALSE）

データ型（他のRDBMSとの比較）

OSS-DB

- 共通のものが多いが、微妙に仕様が異なることがある
  - INTEGER 型：PostgreSQLでは4バイトの整数、Oracleでは38桁の10進数
  - VARCHAR 型：PostgreSQLでは文字数を指定、最大4096文字、Oracleではバイト数を指定、最大4000バイト
  - DATE 型：PostgreSQLでは日付のみ、Oracleでは日付＋時刻
- 多くのRDBMSでほぼ同じように使えるもの
  - INTEGER, NUMERIC
  - CHAR, VARCHAR
  - TIMESTAMP
- PostgreSQL独自のデータ型
  - SERIAL/BIGSERIAL：自動的にシーケンスが作成され、列値を連番にできる
  - TEXT：可変長文字列だが、最大長を指定しなくて良いので便利
  - BOOLEAN：論理値型
    - TRUE/'t'/'true'/'y'/'yes'/'on'/'1'
    - FALSE/'f'/'false'/'n'/'no'/'off'/'0'
    - 大文字・小文字は区別しない、TRUE/FALSE はキーワード、他は文字列
- （参考）Oracleのデータ型との比較
  - NUMBER, BINARY_FLOAT, BINARY_DOUBLE
  - VARCHAR2, NCHAR, NVARCHAR2, CLOB
  - DATE（DATE 型は日付＋時刻、TIME 型がない）

表のデータを変更するには UPDATE 文を使う

EID	CID	EX_NAME	EX_DATE	SCORE	GRADE
1	1	Silver	2011/7/1	80	Pass
2	2	Silver	2011/7/1	75	Pass
3	3	Silver	2011/7/2	50	Fail
4	1	Gold	2011/7/4	40	Fail
5	2	Gold	2011/7/12	85	Pass
6	1	Gold	2011/7/14	70	Pass

UPDATE

EID		SCORE	GRADE
5		65	Fail

EID	CID	EX_NAME	EX_DATE	SCORE	GRADE
1	1	Silver	2011/7/1	80	Pass
2	2	Silver	2011/7/1	75	Pass
3	3	Silver	2011/7/2	50	Fail
4	1	Gold	2011/7/4	40	Fail
5	2	Gold	2011/7/12	65	Fail
6	1	Gold	2011/7/14	70	Pass

UPDATE 文 - 構文と実施例

OSS-DB

- `UPDATE table_name SET col_name = new_val WHERE condition;`
  - “col_name=new_val” の部分をカンマで区切って複数並べれば、複数の列の値を同時に更新できる
  - WHERE 句を省略すると、すべての行が更新される（要注意）
  - WHERE 句の条件に合致したデータがなければ1行も更新されないが、これ自体はエラーとはならない
- トランザクションの機能を使っていなければ、データは即座に更新され、取り消しできない（OracleやDB2に慣れた人は要注意）
- 例：exam表で、eidが5の行について、scoreとgradeの値を変更
  - `UPDATE exam SET score = 65, grade = 'Fail' WHERE eid = 5;`
  - （参考）同じ更新を、リスト形式を使って
    `UPDATE exam SET (score, grade) = (65, 'Fail') WHERE cid = 5;`
    と書くこともできるが、RDBMSの種類によってはエラーになる

他のテーブルを参照したデータ更新（副問い合わせの利用）

new_exam表

EID	CID	NAME	EX_DATE	SCORE	GRADE
1	1	小沢次郎	2011/7/1		
2	2	石原伸子	2011/7/1		
3	3	戌井玄太郎	2011/7/2		
4	1	小沢次郎	2011/7/4		
5	2	石原伸子	2011/7/12		
6	1	小沢次郎	2011/7/14		

UPDATE

exam表

EID		SCORE	GRADE
1		80	Pass
2		75	Pass
3		50	Fail
4		40	Fail
5		85	Pass
6		70	Pass

EID	CID	NAME	EX_DATE	SCORE	GRADE
1	1	小沢次郎	2011/7/1	80	Pass
2	2	石原伸子	2011/7/1	75	Pass
3	3	戌井玄太郎	2011/7/2	50	Fail
4	1	小沢次郎	2011/7/4	40	Fail
5	2	石原伸子	2011/7/12	85	Pass
6	1	小沢次郎	2011/7/14	70	Pass

UPDATE 文 - 副問い合わせの利用

OSS-DB

■UPDATE 文の SET 句に副問い合わせを書くことができる

- 例:new_exam 表の score 列に、exam 表から該当するデータをコピーする
- ```
  UPDATE new_exam n
  SET score = (SELECT score FROM exam e WHERE n.eid = e.eid);
  ```

■注意事項

- SET 句に記述した SELECT 文が複数の行を返した場合は、UPDATE 文自体がエラーとなり、データは更新されない(RDBMSの種類によっては、一部の行が更新される)
- ```
  UPDATE new_exam n
  SET score = (SELECT score FROM exam e WHERE n.cid = e.cid);
  ```
  → 副問い合わせが複数行を返すのでエラーになる

- SET 句に記述した SELECT 文が行を返さなかった場合、列の値は NULL に更新される。更新されたくない場合は、WHERE 句に適切な条件を記述する必要がある
- ```
  UPDATE new_exam n
  SET score =
  (SELECT score FROM exam e WHERE e.eid = n.eid)
  WHERE EXISTS
  (SELECT * FROM exam e WHERE e.eid = n.eid);
  ```
  ```

UPDATE 文 - 表の結合(参考)

OSS-DB

副問い合わせを利用した UPDATE 文で、複数列を更新したい

- UPDATE table1
  SET col1 = (SELECT …), col2 = (SELECT …)
  WHERE … ;
  とすれば、どの RDBMS でも動作するが、ちょっと冗長

RDBMS依存だが、それぞれに簡潔な記述法がある

- PostgreSQLの場合 ～ FROM 句を使って表を結合できる

```
UPDATE new_exam n
SET (score, grade) = (e.score, e.grade)
FROM exam e WHERE n.eid = e.eid;
```

- Oracleの場合 ～ SET 句で SELECT リストを指定可能

```
UPDATE new_exam n SET (score, grade) =
(SELECT score, grade FROM exam e WHERE e.eid = n.eid);
```

- MySQLの場合 ～ 更新対象表を複数指定することで、表を結合できる

```
UPDATE new_exam n, exam e
SET n.score = e.score, n.grade = e.grade
WHERE n.eid = e.eid;
```

例題解説

例題解説1

OSS-DB

運用管理 - 標準付属ツールの使い方

以下の記述から、誤っているものを2つ選びなさい。

A. `createdb` コマンドでデータベースを作成するには `CREATEDB` 権限が必要である

B. `dropdb` コマンドでデータベースを削除するには `CREATEDB` 権限が必要である

C. `dropdb` コマンドでデータベースを削除する前に、そのデータベース内のテーブルなどすべてのオブジェクトを削除しておく必要がある

D. `dropuser` コマンドでユーザを削除するには、`CREATEROLE` 権限が必要である

E. `dropuser` コマンドでユーザを削除する前に、そのユーザが所有するすべてのテーブルを削除しておく必要がある

例題解説2

OSS-DB

SQL - 集約関数

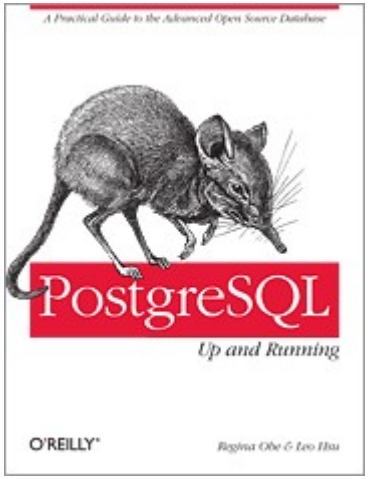
以下のSQL文を順次実行した。最後の SELECT 文が返す値の組み合わせとして適切なものはどれか。

```
CREATE TABLE test1 (id INTEGER, val INTEGER);
INSERT INTO test1 VALUES (1, 10), (2, 20);
INSERT INTO test1 VALUES (3, NULL), (4, 30);
INSERT INTO test1 VALUES (NULL, NULL);
SELECT count(*), count(val), avg(val) FROM test1;
```

A. 5, 5, 12
B. 5, 5, 20
C. 5, 3, 20
D. 4, 4, 15
E. 4, 3, 20

参考資料

- OSS教科書OSS-DB Silver
  - 認定教材
- オープンソースデータベース標準教科書
  - 初心者向けにSQLの初歩からWebアプリケーション開発まで
- PostgreSQL徹底入門
  - PostgreSQL 9.0対応
  - 9.0.1のインストーラ、ソースコード
- PostgreSQL - Up and Running
  - 9.1/9.2対応
  - 現在は英語のみ
- 日本PostgreSQLユーザ会
  http://www.postgresql.jp/
- Let's Postgres
  http://lets.postgresql.jp/
- オンラインマニュアル
  http://www.postgresql.jp/document/9.0/html/



ご清聴ありがとうございました。

■お問い合わせ■
LPI-Japan
テクノロジー・マネージャー
松田　神一
matsuda@lpi.or.jp