بسم الله الرحمن الرحيم

Database Programming

برمجة قواعد البيانات

عدد الساعات: ٢ نظري+٢عملي

الرمز: ٣١٤ حسب

المتطلبات: ٢٢٣ حسب (مبادئ قواعد البيانات)

أستاذات/المادة: م. لندا عمر البدري - م. نجلاء حسن

- **المرجع الأساسي: Ramez Elmasri, Navvathe, shamakant b, "Fundamentals of Database Systems", Addison-Wesley , fifth Edition**
- المرجع الإضافي : مفاهيم وتطبيقات تكنولوجيا قواعد البيانات

الطبعة الأولى (2008)

تأليف: د/ علاء حسين الحمامي

د/ سعد عبد العزيز العاني

الناشر: اثراء للنشر والتوزيع -الأردن

المحاضرة الأولى والثانية

التطبيع (التسوية)

Normalization

مقدمة

- التسوية هي تقنية لتصميم قاعدة بيانات ، وتبدأ باختيار العلاقات بين الصفات .
- الجدول (Table) هو كتلة البناء الأساسية في عملية تصميم قاعدة البيانات ، لذلك فإن التركيب الجيد للجدول يؤدي الى عملية تصميم قاعدة بيانات جيدة .

كيف يمكننا تمييز تركيب الجدول الضعيف؟ وكيف يمكننا انتاج تركيب جدول جيد ؟

الجواب لكلا السؤالين هو التطبيع (Normalization)

- التسوية هي عملية تقويم و تصحيح هياكل الجداول لتقليل تكرار البيانات Redundancies ، من خلال المساعدة في القضاء على البيانات الشاذة
- التسوية هي عملية تخصيص الصفات Attributes الى كينونات Entities وذلك من خلال :
- تقليص تكرار البيانات .
- المساعدة في القضاء على البيانات الشاذة .
- انتاج تكرارات مسيطرة عليها لربط الجداول .
- تعمل التسوية خلال سلسلة من المراحل تسمى الصيغ الطبيعية Normal Forms
- تسمى المراحل الثلاثة الأولى بما يلي :
- الصيغة الطبيعية الأولى First Normal From (1NF) .
- الصيغة الطبيعية الثانية Second Normal Form (2NF) .
- الصيغة الطبيعية الثالثة Third Normal Form (3NF) .

- يجب أن لا نفترض بأن أعلى مستوى من التسوية هو دائماً المفضل جداً ، فأننا نتوقع من حين الى آخر فتح التسوية Denormalize ، لبعض اجزاء تصميم قاعدة البيانات من أجل تلبية متطلبات الاداء Performance .
- تنتج فتح التسوية Denormalization صيغ طبيعية أقل ، مثلاً ، تحول 3NF الى 2NF .
- قد تؤدي عملية فتح التسوية الى زيادة كفاءة الأداء مع كمية أكبر من تكرارات البيانات .

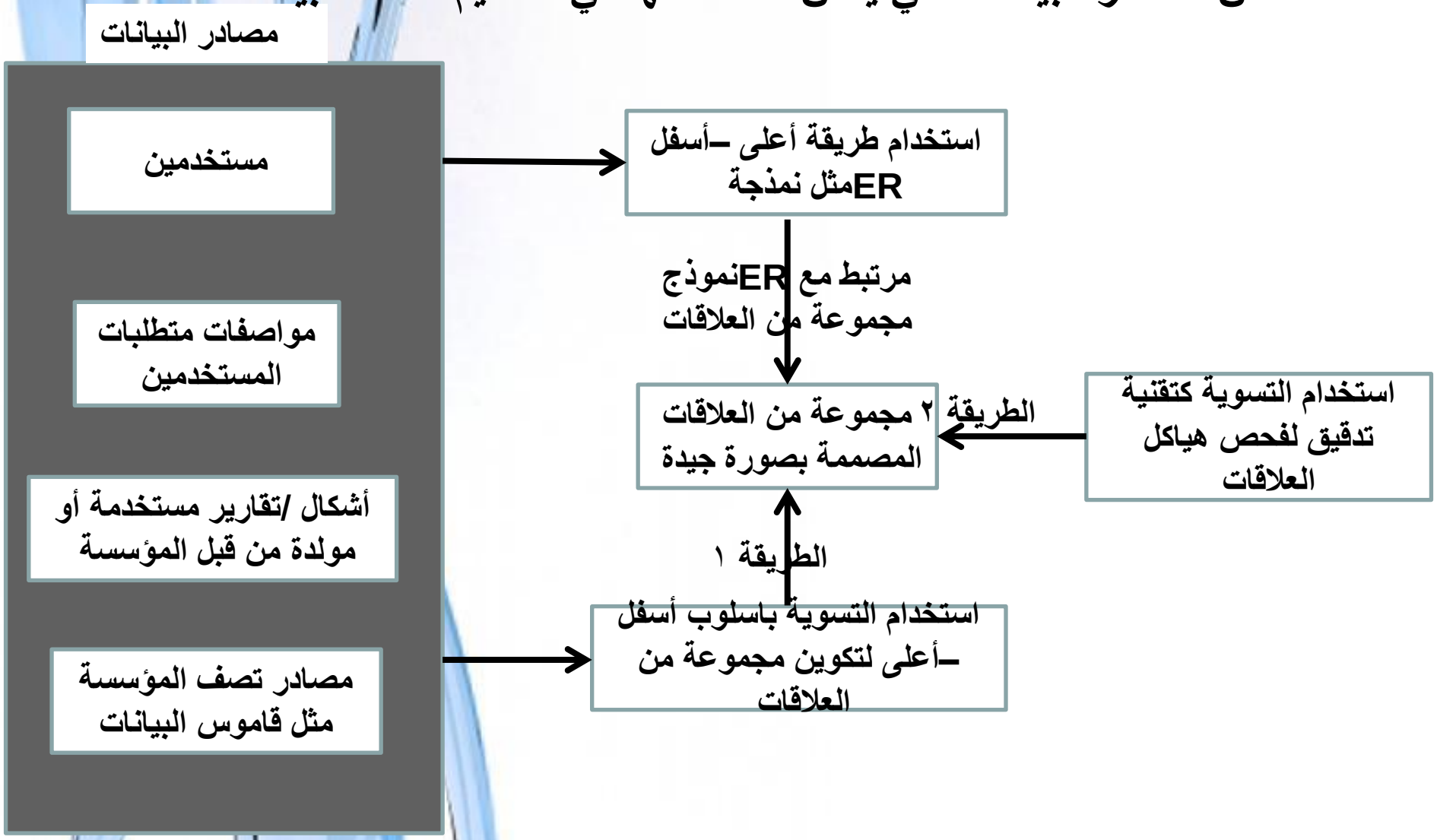
الحاجة الى التسوية
The need for Normalization

- أن الغرض من عملية التسوية هي تحديد مجموعة من العلاقات (Relations) الملائمة و التي تسند متطلبات البيانات لأي مؤسسة .
- **خصائص مجموعة العلاقات الملائمة ما يلي :**
- أقل عدد من الصفات الضرورية لإسناد متطلبات البيانات .
- صفات ذات علاقة منطقية قريبة (موصوفة على شكل اعتمادية وظيفية) في نفس العلاقة .
- اقل تكرار لكل صفة ممثلة لمرة واحدة فقط مع استثناء مهم للصفات التي تكون جزء أو جميع المفاتيح الأجنبية Foreign keys ، التي هي ضرورية لربط العلاقات ذات العلاقة .

توجد طريقتين لاستخدام التسوية

- الطريقة الاولي : أستخدام التسوية كتقنية تصميم قاعدة بيانات بمفردها بالاستخدام من أسفل الى أعلى .
- الطريقة الثانية : استخدام التسوية كتقنية تدقيق لفحص تركيب العلاقات ، و التي تم تكوينها باستخدام اسلوب الأعلى – الأسفل Top-down مثل نمذجة ER .

أمثلة عن مصادر البيانات التي يمكن استخدامها في تصميم قاعدة البيانات



تكرار البيانات ومشاكل التحديث
Data redundancy & update Anomalies

- الهدف الرئيسي من تصميم قاعدة البيانات العلائقية هو تقليل تكرار البيانات ، وإذا تم تحقيق هذا الهدف فإن فوائد تنفيذ قاعدة البيانات تتضمن :
- تحديث البيانات المخزونة في القاعدة بأقل عدد من العمليات .
- تقليص سعة خزن الملفات وبالتالي تقليل الكلفة .
- تعتمد قواعد البيانات العلائيقية ايضاُ على وجود كمية معينة من تكرارالبيانات على شكل نسخ من المفاتيح الرئيسية Primary key تعمل على شكل مفاتيح اجنبية Foreign Key

مثال لتوضيح المشاكل المرتبطة مع تكرار البيانات غير المرغوب بها

- Staff(Staff No, sName, position, salary, branch no)
- Branch(branch No, bAddress)
- staffBranch (staffNo, sName, position, salary, branch No, bAddress)

تابع:مثال لتوضيح المشاكل المرتبطة مع تكرار البيانات غير المرغوب بها

staffBranch

Staff No	sName	position	salary	branch no	bAddress
Sl21	john	Manager	3000	B005	22 deer rd, London
SG37	Ann	Assistance	1200	B003	163 Main st, Glasgow
SG14	David	Supervisor	1800	B003	163 Main st, Glasgow
SA9	Marry	Assistance	9000	B007	16 Argyll st, Aberdeen
SG5	Susan	Manager	2400	B003	163 Main st, Glasgow
SL41	Julie	Assistance	9000	B005	22 deer rd. London

- BRANCH

Branch No	bAddress
B005	22deer rd, London
B007	16 Argyll st, Aberdeen
B003	163 Main st, Glasgow

- STAFF

Staff No	sName	position	salary	branch no
Sl21	john	Manager	3000	B005
SG37	Ann	Assistance	1200	B003
SG14	David	Supervisor	1800	B003
SA9	Marry	Assistance	9000	B007
SG5	Susan	Manager	2400	B003
SL41	Julie	Assistance	9000	B005

العلاقات التي لها بيانات مكررة يكون لها مشاكل تسمى مشاكل التحديث Update Anomalies وتصنف الى:

- مشاكل الادخال
- مشاكل الحذف
- مشاكل التعديل

مشاكل الادخال Insertion Anomalies

- لادخال تفاصيل موظف جديد في علاقة Staff Branch ، يجب ادخال تفاصيل الفرع الصحيحة الذي سيكون فيه الموظف الجديد حتى تكون متناغمة مع قيم القيود الأخري.
- لادخال تفاصيل فرع جديد ليس فيه موظفين حالياً في علاقة Staff Branch من الضروري ادخال قيم فارغة في صفات الـ staff (مثل staff no)، وبما أن staffno مفتاح اساسي لعلاقة staffbranch ،فهذا غير مسموح به .

مشاكل الحذف Deletion Anomalies

- أذا حذفنا بيانات الموظف الذي رقمه SA9 من علاقة Staff Branch (Marry) ، فإن التفاصيل ذات العلاقة مع الفرع ذو الرقم B007 سوف تفقد من قاعدة البيانات

مشاكل التعديل Modification Anomalies

- إذا اردنا تعديل قيمة واحدة من الصفات لفراع معين في العلاقة Branch staff مثلاً عنوان الفرع المرقم B003 فيجب تحديث قيود جميع الموظفين المنتسبين لذلك الفرع ، إذا لم ينفذ هذا التعديل على كل البيانات فسوف تصبح قاعدة البيانات غير متناغمة (يظهر الفرع B003 بإنه يمتلك عناوين مختلفة).

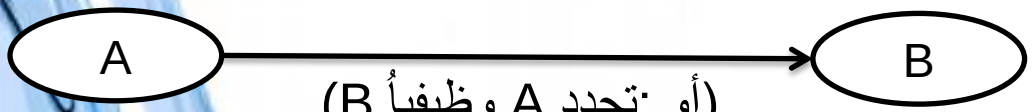
الإعتمادية الوظيفية Functional Dependencies

- تصف العلاقة بين الصفات.
- مثلاُ:

اذا كان A و B صفات في علاقة R تكون B وظيفياُ معتمدة على A: اذا كانت كل قيمة من A مرتبطة مع قيمة واحدة من B

- تُمثل الاعتمادية بين الصفات A و B مخططياً كالتالي:

B وظيفياً معتمدة على A

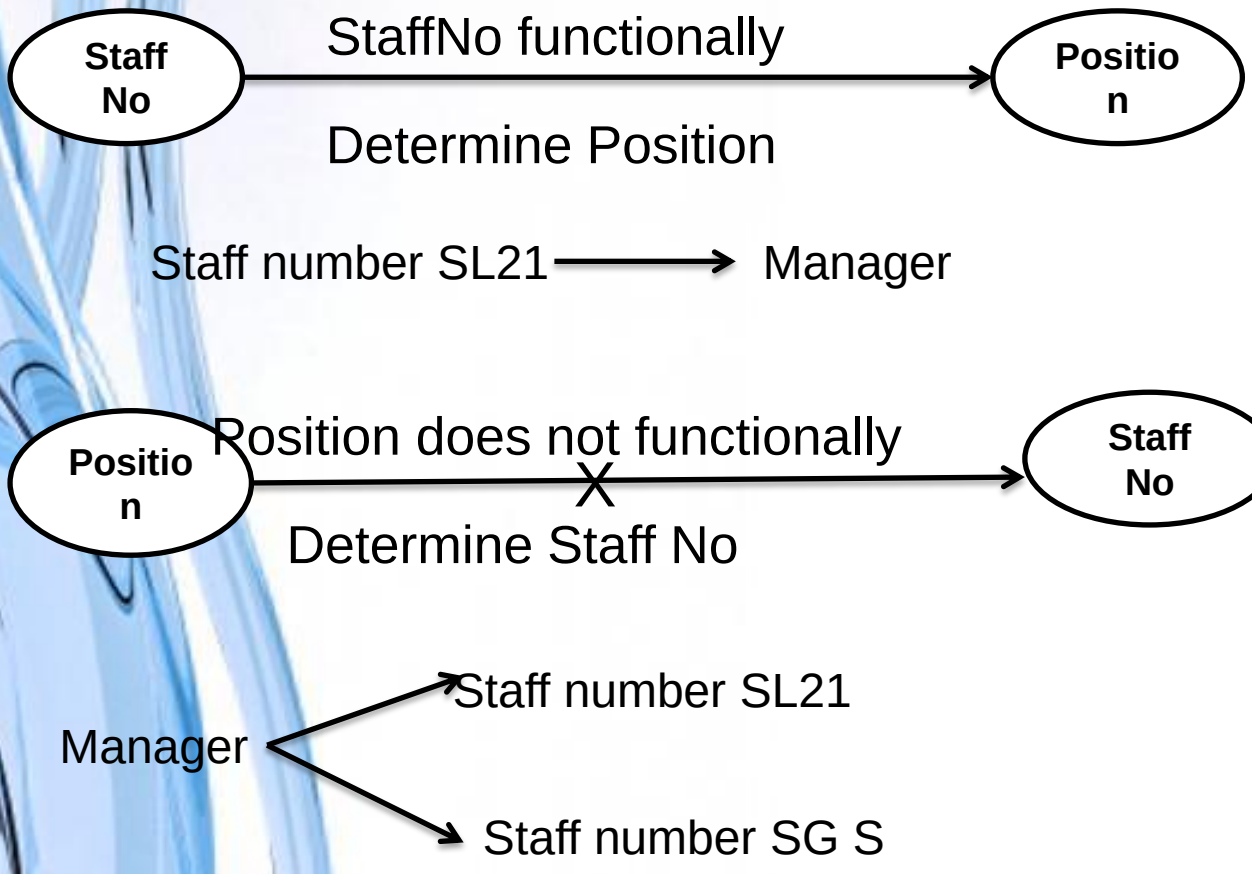


(أو :تحدد A وظيفياُ B)

- عندما تكون الاعتمادية الوظيفية موجودة، فان الصفة أو مجموعة الصفات على جانب الشمال من السهم يسمى المقرر Determinant (من الشكل: A هو المقرر الى B).

Functional Dependencies تابع:الإعتمادية الوظيفية

- مثال:



مثال عن الاعتمادية الوظيفية الموجودة دائماُ

staffNo ⟶ sName

sName ⟶ staffNo

- العلاقة بين staffNo و sName هي (1:1) (لكل موظف اسم واحد فقط)
- العلاقة بين sName و staffNo هي (1:M)(يوجد عدد من الموظفين لديهم نفس الاسم)
- تبقى الاعتمادية الوظيفية صحيحة بعد الأخذ في الاعتبار جميع القيم المحتملة الى صفات staffNo وsName من علاقة staff هي: sName ⟶ staffNo

أنواع الاعتمادية الوظيفية

❖الاعتمادية الوظيفية الكاملة:

اذا كان A وB صفات في علاقة، يكون B معتمد وظيفياً بصورة كاملة على A اذا كان B معتمد وظيفياً على A ، لكن ليس على مجموعة فرعية ملائمة من A .(ازالة أي صفة من A تؤدي الى عدم وجود الاعتمادية)

مثال: Branch No ⟵ staffNo

❖الاعتمادية الوظيفية الجزئية (Partially dependency):

A ⟶ B هي اعتمادية جزأية اذا كانت هناك صفة يمكن ازالتها من A وتبقى الاعتمادية موجودة

مثال: Branch No ⟵ staffNo, sName

تابع:أنواع الاعتمادية الوظيفية

❖الاعتمادية الوظيفية المتعدية (transitive dependency):

اعتمادية صفة واحدة ليست رئيسية على صفة أخرى ليست رئيسية (شرط ان لاتكون أي منهما مفتاح أو جزاء منه)

* مطلوب تمييز الاعتمادية المتعدية لأن وجوده في علاقة يمكن أن يسبب في حدوث مشاكل تحديث.

مثال:

Staff No ⟶ (sName, position, salary, branchNo, bAddress)

branchNo ⟶ bAdress

تحديد المفتاح الرئيسي لعلاقة باستخدام الاعتمادية الوظيفية

- مثال:

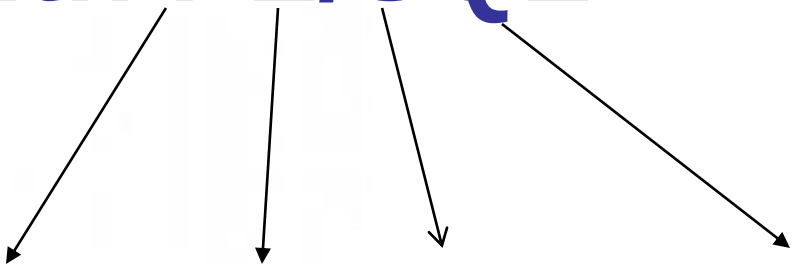
تحديد المفتاح الرئيسي للعلاقة Staffbranch ، المقررات للاعتمادية الوظيفية هي: staffNo, bAddress, (branchNo ,position), (bAddress ,position)

- لتحديد المفاتيح المرشحة يجب تحديد الصفة التي تعرف بصورة فريدة كل قيد في العلاقة ، واذا كان للعلاقة اكثر من مفتاح مرشح نحدد المفتاح المرشح أن يعمل كمفتاح رئيسي للعلاقة ، جميع الصفات التي هي ليست جزء من المفتاح الرئيسي يجب أن تكون معتمدة وظيفياً على المفتاح .
- المفتاح المرشح الوحيد الذي يمكن استخدامه كمفتاح رئيسي لعلاقة ال Staffbranch هو ال Staffno .

PL/SQL تطبيق

Introduction to PL/SQL

Programming in Oracle with PL/SQL



Procedural Language Structured Query Language

What is PL/SQL?

- Oracle's procedural extension to SQL.
- Supplements SQL with several high-level programming features such as block structure, variables, constants and types, the assignment statement, conditional statements, loops,
- customized error handling, and structured data.

Using SQL Queries in PL/SQL Programs

- • Action queries can be used as in SQL*Plus
- • May use variables in action queries
- • DDL commands may not be used in PL/SQL

Fundamentals of PL/SQL

- Full-featured programming language
- An interpreted language
- Type , execute in SQL*Plus editor

PL/SQL Program Blocks

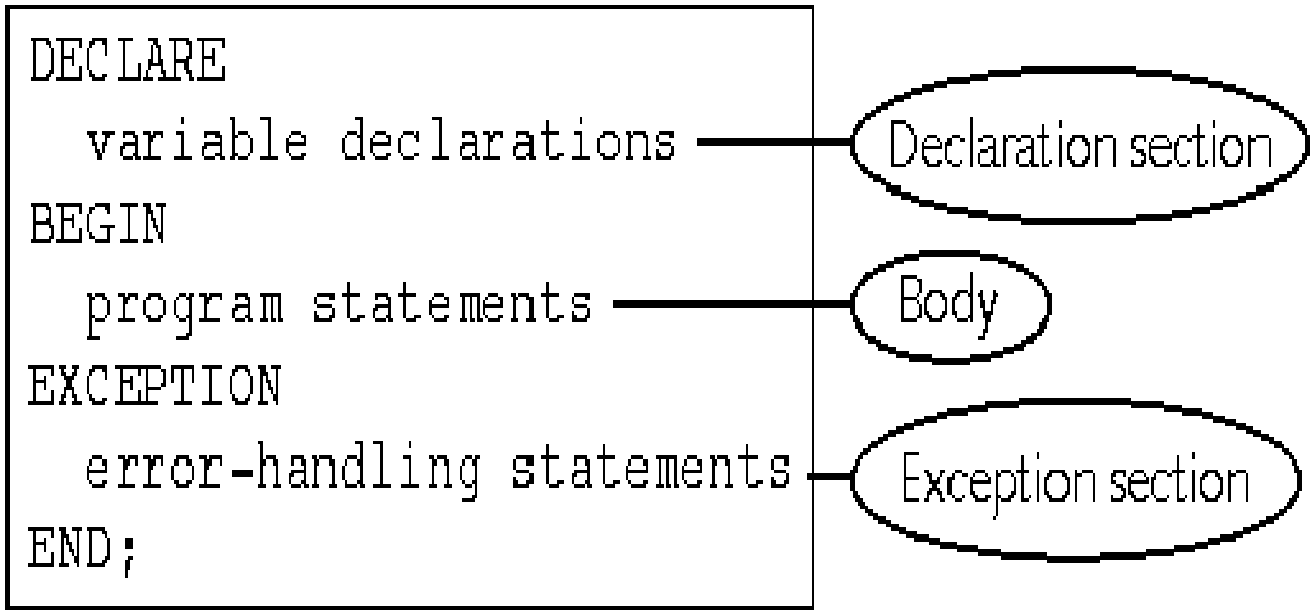


Figure 4-1 Structure of a PL/SQL program block

PL/SQL Program Blocks

- Declaration part :is where objects are defined . The declaration part is optional .
- Executable Part : consists of executable statement (SQL statement , PL/SQL statement , or both).
- Exception Handling Part :In PL/SQL , a warning or error condition is called an exception .The exception handling part consists of code for handling errors .this part is optional .

Comments:

- – Not executed by interpreter
- – Enclosed between /* and */
- – On one line beginning with --

Arithmetic Operators

Operator	Description	Example	Result
**	Exponentiation	2 ** 3	8
* /	Multiplication Division	2 * 3 9/2	6 4.5
+ –	Addition Subtraction	3 + 2 3 – 2	5 1
–	Negation	-5	-5

Table 4-5 PL/SQL arithmetic operators in describing order of precedence

Assignment Statements

- Assigns a value to a variable

variable_name := value;

X:=5;

- Value can be a literal:

first_name := 'John';

- Value can be another variable:

current_first_name := first_name;

The first PL/SQL program – Anonymous blocks

- You can execute this from the SQL*PLUS command prompt:

```
BEGIN
    DBMS_OUTPUT.PUT_LINE('Hello World!');
END;
/
```

- This is called an anonymous block – that is a block without a name.
  - A block is surrounded by BEGIN, END keywords.
  - The built-in procedure PUT_LINE, part of the DBMS_OUTPUT package. This procedure takes a string as input and displays that string on screen.
  - The / indicates that we are finished.
- However, in order to really see the “Hello world” message on the screen, you should set the SQL*PLUS environment variable:

  SQL> **SET SERVEROUTPUT ON**

THE END