约束应明确说明

- 数据中存在隐含约束是一种不良设计
- 字段的性质随着环境变化而变化时设计的错误和不稳定性

- 数据语义属于DBMS，别放到应用程序中

```
configuration(parameter_name, parameter_value)
```

```
configuration(parameter_id, parameter_name, parameter_type)
      configuration_numeric (parameter_id, parameter_value)
```

过于灵活的危险性

- “真理向前跨一步就是谬误”
- 不可思议的四通用表设计
  - Objects(oid, name), Attributes(attrid, attrname,type)
  - Object_Attributes(oid,attrid,value)
  - Link(oid1,oid2)
- 随意增加属性，避免NULL
- 成本急剧上升，性能令人失望

如何处理历史数据

- 历史数据：例如：商品在某一时刻的价格
- Price_history
  - (article_id，effective_from_date，price)
- 缺点在于查询当前价格比较笨拙
- 设置为价格终止时间呢?

```
select a.article_name, h.price
from articles a,
     price_history h
where a.article_name = some_name
  and h.article_id = a.article_id
  and h.effective_from_date =
     (select max(b.effective_from_date)
      from price_history b
      where b.article_id = h.article_id)
```

如何处理历史数据

```
select a.article_name, h.price
   from articles a,
        price_history h
   where a.article_name = some_name
     and h.article_id = a.article_id
     and h.effective_from_date =
        (select max(b.effective_from_date)
         from price_history b
         where b.article_id = h.article_id
           and b.effective_from_date <= sysdate)
```

处理流程

- 操作模式（operating mode）
  - 异步模式处理（批处理）
  - 同步模式处理（实时交易）
- 处理数据的方式会影响我们物理结构的设计

数据集中化（Centralizing）

- 分布式数据系统复杂性大大增加
  - 远程数据的透明引用访问代价很高
  - 不同数据源数据结合极为困难
    - Copy的数据传输开销
    - 无法从数据规划中获益（物理结构，索引）
- 数据库该如何部署呢?
  - 中庸、分析、决策
- 离数据越近，访问速度越快

系统复杂性

- 数据库的错误很多
  - 硬件故障
  - 错误操作...
- 数据恢复往往是RD和DBA争论焦点
  - DBA，即便确保数据库本身工作正常，依然无法了解数据是否正确
  - RD，在数据库恢复后进行所有的功能性的检查

小结

- 错误的设计是导致灾难性后果的源泉
- 解决设计问题会浪费惊人的精力和智慧
- 性能问题非常普遍
- 打着“改善性能”的旗号进行非规范化处理，常常使性能问题变得更糟

- 成果的数据建模和数据操作应严格遵循基本的设计原则。

End