

空间数据管理

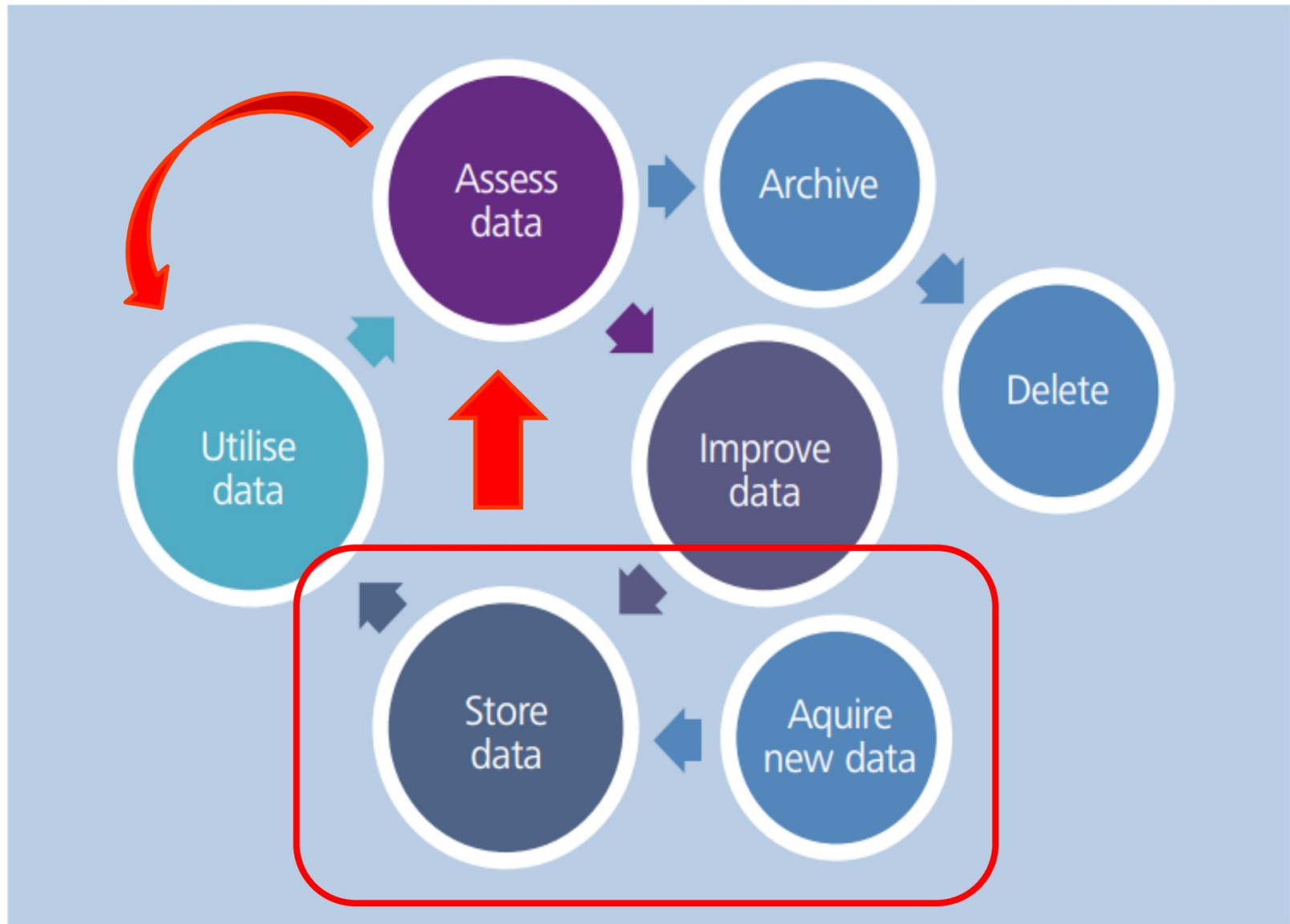
第 3 周

克莱尔·埃鲁尔博士

概述

- 设计表格
 - 概念设计（提醒）
 - 逻辑设计 •

DDL – 创建表 – 物理设计 • DDL – 主键和外键 • DDL – 其他约束



数据库设计

- 数据库设计任务

- 概念设计 图表和文字

- 用户需求的描述,记录为
实体关系图

- 逻辑设计 获取收集的信息

- 概念设计阶段,并对其进行转换以考虑系统性能和预期的操作
条件

- 物理设计 采用规范化的逻辑设计

- 并将其转换为数据库的实际构建脚本(SQL)。

设计表格 - 第 1 步 - 概念

- 确定您感兴趣的高级事物（特征、对象等）

房间、建筑物、道路、电灯开关、大学校园、屋顶、椅子、人、员工、鞋子、奶酪、航班、国家、学校、医院 · 设计将取决于你想要做什么

存储有关数据

- 参见第 1 周“概念设计”

设计表格 - 第 2 步 - 概念

- 第 2 步 您需要收集哪些数据
每个事物/特征/对象
 - 非常详细!
- 对于房间,我们需要
 - 房间的位置
 - 拥有房间的用户
 - 创建房间的时间（这主要是为了我的
标记过程,虽然）

设计表格 - 第 2 步 - 概念

- 对于状况报告,我们需要了解需要维护的事项和/或健康和
安全或舒适问题
 - 因此,首先询问设施经理他们需要什么
 - 作为资产经理,看看他们是否需要汇总级别的任何东西
 - 查看已经存在的现有表格 (纸质表格)
利用
- 这是需求收集

设计表格 - 第 2 步 - 概念

- 是否有相关标准或最佳实践指南：

- <https://www.rics.org/globalassets/rics-website/media/upholding-professional-standards/sector-standards/building-surveying/condition-report-sample-report-rics.pdf>
- <https://www.rics.org/globalassets/rics-website/media/upholding-professional-standards/sector-standards/building-surveying/technical-due-diligence-of-commercial-property.pdf>
- (https://assets.publishing.service.gov.uk/media/6033bb218fa8f543272b4002/SC110008_R2_report.pdf - 环境署)

从概念到逻辑

- 翻译成逻辑模型
 - 注意:因为我们使用的是 UML 符号,所以符号没有太大区别
我们仍然使用类、属性、关联等等!

从概念到逻辑

- 翻译成逻辑模型

- 翻译

- 实体
 - 身份标识
 - 多:多关系

- 进入

- 表格
 - 主键和外键 · ID

从概念到逻辑

- 实体
 - 成为逻辑模型中的表
 - 每个表都与实体同名
 - 每个表都有相同的属性,但现在我们添加
 - 约束
 - 数据类型 (varchar、数字、日期、几何)
 - 作为主键的 ID

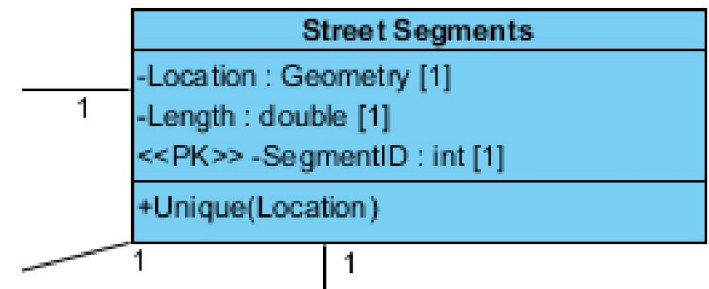
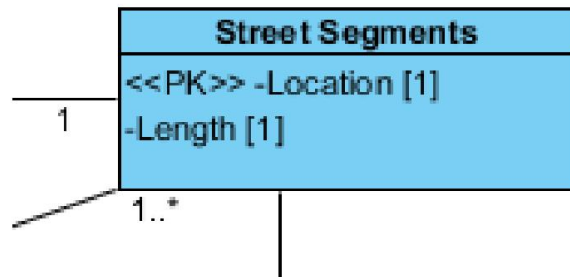
设计表格 - 第 3 步 - 逻辑

- 对于您要捕获的每项信息,确定数据类型
 - 文本 (varchar)
 - 日期
 - 数字
 - 位置 (几何)
- 完整列表:

<https://www.postgresql.org/docs/current/datatype.html> (不包括位置/空间类型)

从概念到逻辑

- 实体



从概念到逻辑

- 一对多关系
 - 为每个实体形成一张表
 - 分配给孩子的关系的任何属性
实体

创建关系模型

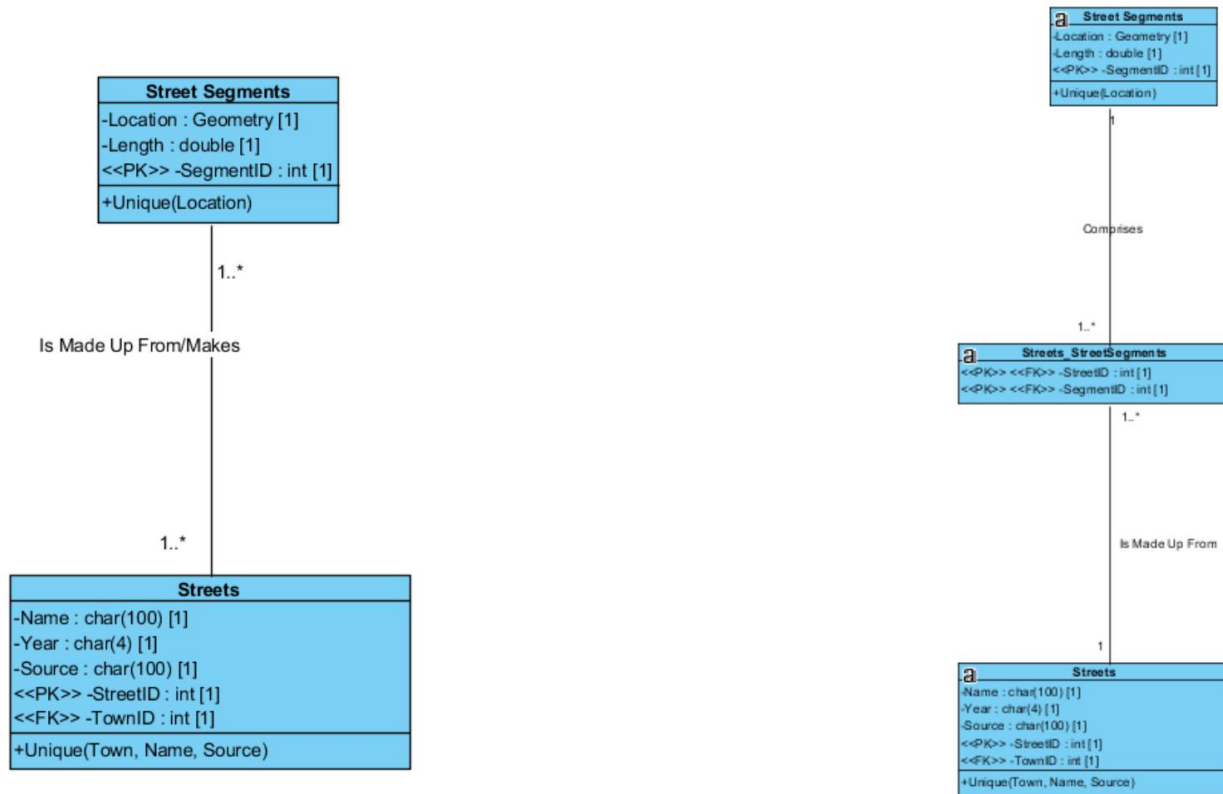
- 一对一的关系
 - 每个实体成为一个关系（表）
 - 来自实体之间关系的属性被分配给一个或另一个新表

ER图

- 解决多:多关系
 - 关系数据库不能处理多:多关系
 - 由于表的工作方式,一个“父”主键可以有許多“子”链接,但反之则不行
 - 然而,在概念层面,您可能在 ER 图中找到 many:man
 - 添加其他实体来解决这些问题

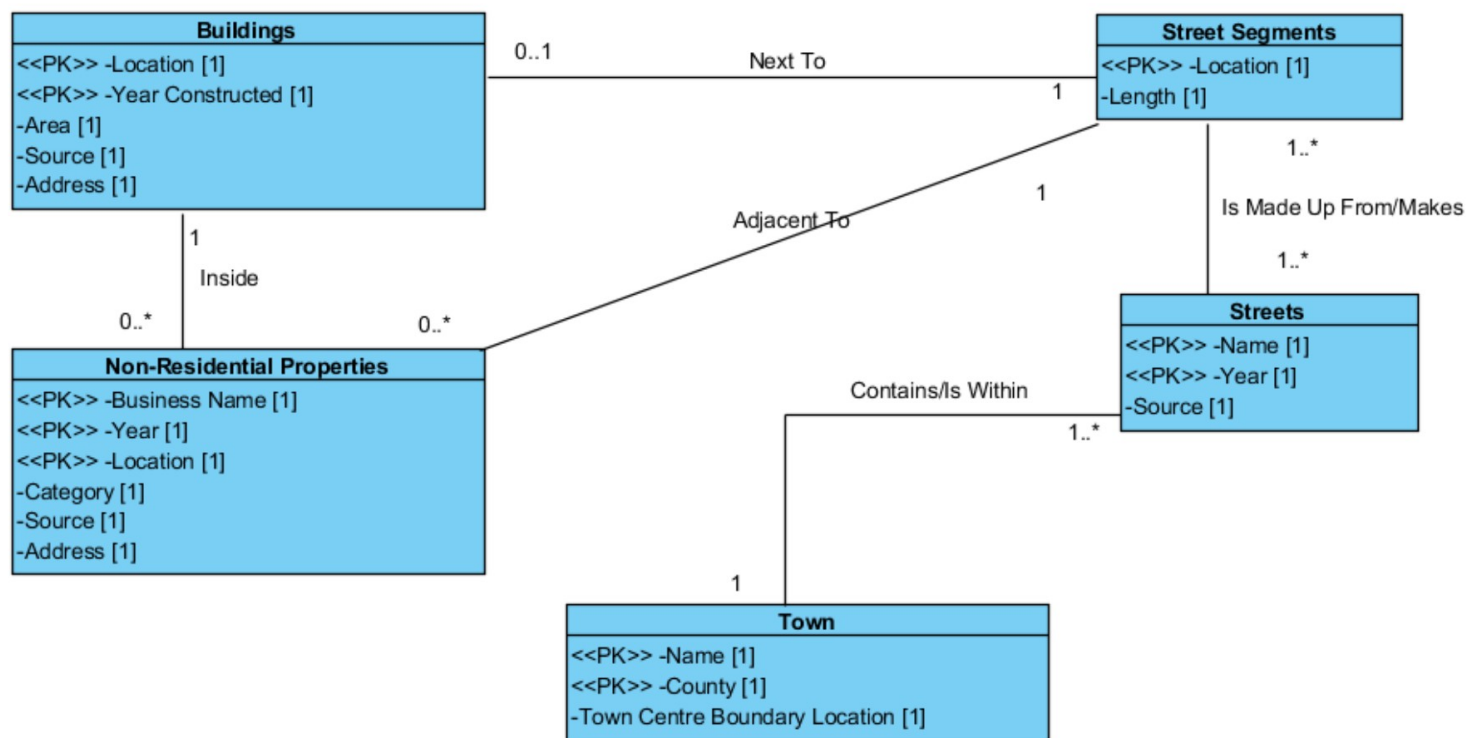
ER图

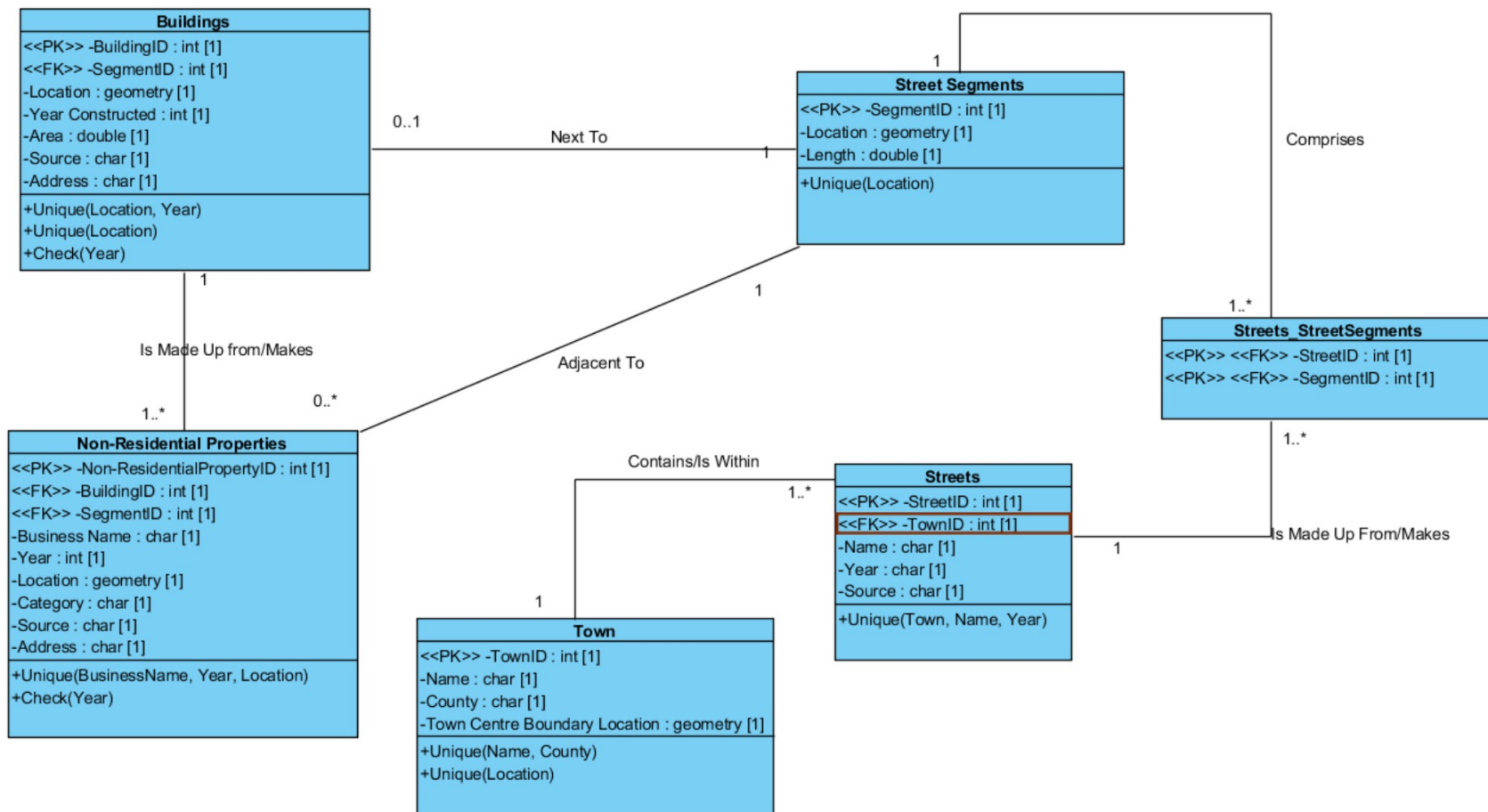
• 解决多:多关系



从概念到逻辑

- 主要区别
 - 完整的主键（标识符）与数字 ID 和唯一约束
 - 多:多关系与 1:多关系
 - 非标准化与标准化
 - 列名与列名 + 数据类型





概述

- 设计表格
 - 概念设计（提醒）
 - 逻辑设计 •

DDL – 创建表 – 物理设计 • DDL – 主键和外键 • DDL – 其他约束

创建表

- 根据ER图,为每个表编写创建表脚本
- 不要忘记包含架构
 - 这是数据库的一部分,您可以在其中根据用途对表进行分组
 - 例如,您可以拥有的设施和资产管理
室内和室外模式
- 对于这个模块,您只能在一个模式中工作 UCXXXXX
或ZCXXXXX

创建表

- 根据ER图,为每个表编写创建表脚本
- 不要忘记包含架构
 - 这是数据库的一部分,您可以在其中根据用途对表进行分组
 - 例如,您可以拥有的设施和资产管理
室内和室外模式

创建表

创建表 Campus_overview_ci.indoor_condition_index (

id 整数 NOT NULL,

room_id 整数,墙壁整数,

天花板整数,门整数,窗户整

数,家具设备整数,加热系统

整数, airing_system 整

数,套接字整数,

lighting_and_switches 整数,

user_id 整数,创建者字符变化

(100) ,没有时区的创建时间时间

戳现在默认 () ,

);

创建表

```
CREATE TABLE Campus_overview_ci.rooms (room_id  
    整数 NOT NULL,user_id 字符不同 (50) ,
```

```
    没有时区的 createtime 时间戳 DEFAULT now() );
```

(注意架构 campus_overview_ci 对于你的所有工作,使用你的 UCL 用户名作为架构)

创建表 - 添加空间列

始终单独执行此操作,因为您可以更好地控制数据类型和维度等

如果存在位置,则更改表 Campus_overview_ci.rooms 删除列;

选择

```
AddGeometryColumn( campus_overview_ci , rooms , location ,  
4326, 点 ,2);
```

创建表格 一些提示

- 始终包含架构
- 始终使用小写的列名和表名字
- 始终使用或表 _ 分隔列中的单词名（不是空格）
- 始终单独添加几何图形（如果需要）
- 始终单独创建约束

概述

- 设计表格
 - 概念设计（提醒）
 - 逻辑设计 •

DDL – 创建表 – 物理设计 • DDL – 主键和外键 • DDL – 其他约束

主键

- 这些是使表格中的每一行都与其他行不同的因素
- 对于房间,这可能是位置 如果每个房间都在不同的位置

房间..

- 但是您可以在同一位置创建 2 个房间
- (这在实践中是一个很好的规则,因为除非在不同的时间,否则你不能在同一个地方有两个房间。但我没有把这个严格的规则放进去,因为你是初学者)

主键

- 那么,也许是位置和user_id?
 - 但同一个用户可以同时创建 2 个房间
地点
- 因此,我们有location、user_id和createtime 因为用户不能同时在同一位置创建房间

主键

- 这是我们房间真正的主键 即每个房间与其他房间的不同之处
- 然后我们使用
 - 主键的 ID 快捷方式
 - 如您所见,它使表之间的数据链接更容易
 - 强制执行我们刚刚的实际规则的约束定义

主键

更改表 Campus_overview_ci.rooms 添加约束

rooms_pk 主键 (room_id) ;

外键



- 确保我们的数据一致

互操作性目的 我们要检查我们作为每个类别的评级输入的值是否有效

- 为了节省空间,我们不复制所有文本..相反,我们将链接到该段文本的ID..

外键

Data Output Explain Messages Notifications

	 id integer	 condition_description character varying (250)
1		1 As new or in good serviceable condition
2		2 Deteriorating; evidence of high usage; age; additional maintenance costs and inefficiency
3		3 Requires replacement within 5 years
4		4 In poor condition; overdue for replacement
5		5 Unable to determine condition (e.g. as item is hidden)
6		6 Item does not exist

外键

- 所以我们需要确保使用有效的 ID
- ...

外键（示例）

更改表

Campus_overview_ci.indoor_condition_index添加
约束 ici_wall_fk外键（墙）引用
polimi.ucl_condition_options(id);

（对我们拥有的每个条件选项重复）

概述

- 设计表格
 - 概念设计（提醒）
 - 逻辑设计 •

DDL – 创建表 – 物理设计 • DDL – 主键和外键 • DDL – 其他约束

其他约束

- 需要在数据库中强制执行规则
- 例如,真正的主键是
保持每个房间与所有其他房间不同

更改表 `Campus_overview_ci.rooms` 添加约束
`room_unique`在 (位置,用户 ID,创建时间)上唯一;

其他约束

- 独特的约束 -
 - 每个表必须至少有一个,以确保执行 REAL 主标识符
- 检查约束
 - 检查值是否在一定范围内