EIE3112 Database System Cheatsheet http://u.pc.cd/boaitalK

表 Table 关系 Relationship 关系模式 schema 域 Domain (D) 基数 cardinality 元组个数 笛卡尔积 Cartesian Product $D_1 \times D_2 \times \ldots \times D_n$ (n个元组 tuple 属度目 关系degree)

PK ——— FK ------ $R(A_1:D_1, A_2:D_2, \ldots, A_n:D_n)$ or $R(A_1, A_2, \ldots, A_n)$ eg. Student(S# char(8), Sname char(10), Ssex char(2), Sage integer, ...)

关系性质: 列同质; 行、列无序; 元组互异(表可同); 属性不可再分(第一范式)

不可: 复合属性 composite; 多值属性

键: 超键 Superkey; 候选键 Candidate key —主属性; 主键 Primary key —非主属性; 外键 Foreign key

完整性: 实体完整性 Entity Integrity 主键不可为NULL ⇔ 不可标识的个体; 参照完整性 Reference Integrity R1.FK必为R2.PK某值或NULL; 用户自定义完整性

NULL: 空值 不知道、不存在、无意义 关系模型: '?' 3+?=? 3>?=? (? and True)=? (有时需特殊处理)

① 关系代数 集合操作: $R \cup S$, $R \cap S$, $R-S$, $R \times S$ (cartesian)
纯关系操作 eg. Select Sname From S, SC Where S.S# = SC.S# and SC.C# = 001 Order by Score desc
$\Leftrightarrow \pi_{Sname}(\sigma_{S.S\#=SC.S\# \text{ and } SC.S\#='001'}(S \times SC))$

Project 投影 $\pi_A(R)$ (distinct)
Select 选择 $\sigma_{condition}(R)$
Join 连接 $R \underset{A\theta B}{\bowtie} S$ (condition) $= \sigma_{t[A]\theta s[B]}(R \times S)$
Division 除 $R \div S$
更名 $\rho_{A_1}(A)$ eg. $\pi_{SC.C\#}(\sigma_{SC.S\#=``"\wedge SC1.S\#=``_2"}(SC \bowtie_{SC.C\#=SC1.C\#} \rho_{SC1}(SC)))$

并、交、差须满足: 并相容性(对应性、可比性) R, S属性数目同, 对应属性域同
笛卡尔积: $R \times S = \{\langle a_1, a_2, \ldots a_n, b_1, b_2, \ldots b_n\rangle \mid \langle a_1, a_2 \ldots a_n\rangle \in R \wedge \langle b_1, b_2 \ldots b_n\rangle \in S\}$
$R \cup S = S \cup R$, $R \cap S = S \cap R$, $R \times S = S \times R$

连接: 等值连接 (θ-连接) 自然连接 R, S有相同属性组
外连接: 左外连接 $R ⟕ S$; 右外连接 $R ⟖ S$; 全外连接 $R ⟗ S$ 自然连接+左表(左外)/右表(右外)/两表失配

关系演算 ①关系元组演算 元组变量: $\{t \mid P(t)\}$
②关系域演算 域变量: $\{\langle x_1, x_2, \ldots, x_n\rangle \mid P(x_1, x_2, \ldots, x_n)\}$ 非过程性 eg. QBE {Print(P.) Delete(D.) Insert(I.) Update(U.)} (P. X)

原子公式 F(t): $t \in R$; $s[A]\,\theta\,c$; $s[A]\,\theta\,u[B]$ 可递归构造 $\wedge$ $\vee$ $\neg$ $\exists(t \in r)(F(t))$ $\forall(t \in r)(F(t))$
$R \cup S = \{t \mid t \in R \vee t \in S\}$ $R \cap B = \{t \mid t \in R \wedge t \in B\}$ $\pi_A(R) = \{t[A] \mid t \in R\}$
$R - S = \{t \mid t \in R \wedge \neg t \in S\}$ $\sigma_{con}(R) = \{t \mid t \in R \wedge F(con)\}$
$R(A) \times S(B) = \{t \mid \exists(u \in R)(\exists s \in S)(t[A] = R[A] \wedge t[B] = S[B])\}$

安全性 不产生无限关系或无穷验证 ⇒ 对关系演算施加约束
反例 $\{t \mid \neg(R(t))\}$, $\{t \mid R(t) \vee t[2] > 3\}$ 可能表示无限关系
$(\exists u)(\omega(u))$, $(\forall u)(\omega(u))$ 可能导致无穷验证

关系代数 ⇔ 安全的元组演算 ⇔ 安全的域演算
都是非过程性, 关系代数 ≤ 元组演算 ≤ 域演算

函数依赖 $R(U)$, $U = \{A_1, A_2 \ldots\}$ $X, Y \subseteq U$, 如 $\forall r \in R$ (¬(有两元组在X属性等而在Y属性不等)) ⇒ $X \to Y$; 如 $X \to Y$, $Y \not\subset X$ ⇒ $X \to Y$ 为非平凡函数依赖; 如 $X \to Y$, $(\forall X' \subset X)(X' \not\to Y)$ ⇒ $X \xrightarrow{f} Y$, 否则 $X \xrightarrow{P} Y$; 候选键 非主属性冗余!
如 $X \to Y$, $Y \to Z$, $Y \not\subset X$, $Z \not\subset Y$, $Z \not\subset X$, $Y \not\to X$ ⇒ Z传递函数依赖于X ⇒ $X \to Z$
逻辑蕴含 如F⊂{R(U)的函数依赖}, $X \in F \Rightarrow X \to Y$, 非 F逻辑蕴含 $X \to Y$ ($F \models X \to Y$)
闭包 被F⊨(X→Y)的所有(X→Y) (F^+); 如 $F^+ = F$ ⇒ F为全函数依赖族/函数依赖完备族
多值依赖 如元组 $t, s, u, v \in r$, $t[X] = s[X]$ ⇒ $\exists t[X] = u[X] = v[X]$, $u[U-X-Y] = s[U-X-Y]$, $v[U-X-Y] = t[U-X-Y]$ ⇒ X多值决定Y ($X \to\to Y$); 函数依赖是多值依赖特例

关系范式 令除外键外的数据只出现一次, 避免: Insertion/deletion/update anomaly
$\forall R(U) \in 0NF$ eg. Star(name, address(street, city))(×)
↓ 消除重复值或属性
$R(U) \in 1NF$: 关系的每个分量都不可分(无复合属性或多值属性)
↓ 拆表消除部分依赖
$R(U) \in 2NF$: U中每一非主属性完全依赖于候选键
↓ 拆表消除传递函数依赖
$R(U) \in 3NF$: 不存在 候选键 经非主属性 传递决定另一非主
↓ 拆表: 不含候选键单独组表, 含者组一表, 合并
$R(U) \in BCNF$: $\forall (X \to Y) \in F$, 当 $Y \not\subset X$ ($A \not\subset X$) 时X含候选键
↓ 消除非主属性对候选键以外的多值依赖
$R(U) \in 4NF$: $\forall (X \to\to Y \in D$ (函数/多值依赖)) 非平凡, $Y \not\subset X$, $XY \neq U$ ⇒ X为超键

SQL ①DDL 定义数据库和表
Create { Create Database [If not exists] <name>
Create [Temperary] Table [If not exists] <name> (<colname> <type> [Primary key|Unique] [Not Null] [Default <const|Null>]
[col_constr <col_constr>], <colname> (...), [Primary key (...)] [Index <name> (FKname) [ASC|DESC]), Index(...),]
[constraint <name> Foreign key (<FKname>) [On Delete|Update Noaction|Restrict|Cascade|Set null|set default], References <tablename>(<colname>)
constraint(...)] [Engine = <name>])
Alter Table <tablename> [Add <colname> <type> [...], (...)] [Drop <constraint> [] (<colname>)] [Modify <colname> <type> [...], ...]
Drop { Database|schema [If exists] <name>
[Temperary] Table [If exists] <name>, <...>, ..., [Restrict|cascade]
Use|close <dbname>

<type> char(n) 固定长字符串; varchar(n) 可变长字符串; int/integer 整型; numeric(P,q) 固定精度数 整P, 小P-q位; real/float(n) 浮点数 (n位精数); date 日期 2003-09-12; time 时间 23:15:03

<col_constr>:
NOT NULL | [Constraint <name>]
UNIQUE | PRIMARY KEY | Check (condition) |
References <tablename> [(colname)] on delete|Update...

② DML 实例
Insert Into <tablename> [(<colname>, (...))] Values (value₁, value₂, ... | Select 子查询)
[Set <colname> = value, (...)] (in Trigger)
Delete From <tablename> [Where <condition> | current of <curname>]
Update <tablename> set <colname> = exp|子查询, (...) [Where <condition>]
Select [Distinct] <colname>|expr|AgFunc [[as] Alias], (...) | * [Into <valname>, (...)] [From <table name> [[as] <alias>]
AgFunc: Avg/Min/Max/Sum/Count()
[Inner|Outer|Left|Right join (<tablename>) on <condition>], <tablename₂>, (...)] [Where <condition>] (for row)
[Group by <colname₁>, (...)] [Having <condition>] (for group) [Order by <colname> [ASC|DESC], (...)]
[with rollup] 逐级分类汇总

Where条件: $=$ $\neq$ $<>$; Attr. [Not] like 'xx' %_\; Attr. Is [Not] Null
子查询: 集合成员资格 / 集合之间比较 / 集合基数测试
row|const [not] in (子查询|枚举)
row|const θ some|all (子查询|枚举)
[NOT] exists (子查询|枚举)

(子查询|枚举) Union|Intersect|Except [All] (子查询|枚举)
相关子查询 (只有单外向内传参)
eg. Select Sname From Student Stud where S# in (Select S# From SC where S# = Stud.S# and C# = 001)

视图 Create view <viewname> [(外模式 = <colname>, (...))] as 子查询

不等值连接 (eg. 有工资差的任意两人)
Select T1.Tname as Teacher1, T2.Tname as Teacher2 From Teacher T1, Teacher T2 where T1.salary > T2.salary;

Useful Func
STR_to_Date(str, Format) std: '2007-1-31' %d %m/M %y/Y
Cast(var as decimal(precision, fraction))
Monthname(date) → str. Now() → datetime
Concat(str, str, ...)
substring(str, startnum, len) (1 2 3 ...)

数据库完整性(DB Integrity)
- 广义完整性 语义完整性、并发控制、安全控制、DB故障恢复
- 狭义完整性 专指语义完整性

- 实体完整性
- 参照完整性
- 用户自定义完整性(语义完整性)

完整性约束条件 Integrity Constraint = (O, P, A, R) R: 响应，不满足时怎么办
O: 数据集合，对象 P: 谓词条件，约束 A: 触发条件，何时

对象
- 列级完整性约束条件：某列
- 表关系完整性~：多列，或跨表

来源
- 结构约束：来自模型
- 内容约束：来自用户

约束状态
- 静态约束
- 动态约束

DBS安全级别：物理控制，网络~，OS~，DBMS~
DBMS安全机制
- 自主安全性机制 存取控制，授权机制 → AccessRule = (S, O, t, P) S: 主体，用户(组) O: 访问对象，t = 权利，P: 约束条件
- 强制安全性机制 数据与用户强制分类
- 推断控制~ 防止从历史或公开信息推断不应公开或个体信息
- 数据加密~ 加解密，密钥，加解密方式与传输

做法
- 最小特权原则
- uptodate on patches
- 删除/禁用无用账户
- 防火墙/访问控制
- Pswd Security

Throttle conn (firewall, no WWW to DB
>>skip-networking
>>flush privileges
Disable 'Local infile
>>set-variable=local-infile=0

SQL Injection
#/--␣, /*...*/ to comment
- no dynamic query
- prevent malicious query
- ⇒ Prepared (parameterized) query

动态 DML (解决连接编程范例(paradigm)：声明型/过程型，数据模型)

断言 Create Assertion <name> check(condition)
(每次更新均检查，! 复杂断言会增加DB负担!)

更改语句分隔符 Delimiter _ (no ';' at the end!)

触发器(Trigger) Create [user] Trigger <name> (A:) Before|After Insert|Delete|Update [OF <colname>, ...] (O:) ON <Tablename>
[Referencing corr_name, ...] [For each row|statement] (P:) [when (condition)] (R:) SQLstatement|Begin ...; ...; End
→ Old|New [Row|Table] [As] name

过程(procedure) Create Procedure <name> ([IN|OUT] <varname> <type>) Begin (.......); End $$
Call <procedurename> (argsIn, argsout)

变量声明 Declare <valname> <type>, (...) Default <value> (有@无须声明，会话结束释放) 赋值 Set <valname> = expr.

条件控制 If (...) Then ...; (......); [Else ...; (......);] End if Case [expr.] when expr.|condition Then ...; (......); when (......); [Else (......);] End case

循环控制 While ( ) Do (......); End While; Repeat (......); Until ( ) End Repeat; <loop_label> Loop (......); End Loop <name> Leave <name>; Iterate <name>;

异常处理 Declare continue|exit handler for <condition|value>, ... SQLstatement;

状态捕获 whenever <condition> <action>
<condition>: SQLError 语句出错 (sqlcode < 0), Not Found 没有结果, SQL Warning 警告 (sqlcode > 0)
<action>: continue 忽略, goto <标签> 转至, Stop 终止、撤销、断开, Do/Call 调用
同条件会使后编写的覆盖 (非执行次序!) 使用goto 应在 handle 内更改为 continue 以防无限循环

游标 (cursor) 声明 Declare <name> cursor for (子查询|表等) [For Read only | Update [OF <colname>, ...]]
打开 Open <curname>
执行 Fetch [next|prior|first|last|[Absolute|relative] value] From <curname> into <varname>, ...
关闭 Close <curname>

DCL
- Grant
- Revoke

Data warehouse = use operational data for decision making (vs: Online Transaction Processing: RT operation)
→ business intelligence
subject-oriented, integrated (to 2NF, consisted data type), time-variant, non-volatile
- Potential High Returns on Investment
- Competitive advantage = decision, untapped info
- Increased productivity for decisioning

Unpredicatble Report, multi-Dim

EU Access Tool
- Trad. reporting & query
- OL Analysis Processing
- Data mining

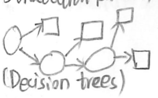
- ETL = Extraction → Transforma → Loading
- Warehouse mngr: Ana, Transfo, merge data,
- Query = query profile

Datamart = subset of data for ana. particular unit
Why? give users access to data need for ana. most often
- provide data in forms that match collective view by group usr.
- improve response, provide structured data
- Simplier to create, lower cost, easier support to targeted usr.

Dimension tbl textual descript — Fact tbl = measures (hist data

OLAP Cube: (Optimized way, quick resp.)

Datamining: structural pattern

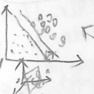


(Decision trees)

Classification/Associative rule
If _ =/>/< _ and/or _ then _ = _

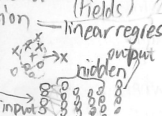
Linear Regression Sol
Result = 10 + 0.5A − 0.7B + 0.5C

Machine Learning
Unsupervised: Clustering & Dim reduction
Supervised: Regression, Decision trees, Classification
Category / Contin



if not linearly separable: non-linear map to high dim

- Decision Tree
- Support vector
- Neural N.
- K-nearest neighbour



Problem: long time (no training stage) Unbalance num, scale attr.

Concept (target)
- Asso. Learning
- Clustering
- Classification
- Prediction — linear regression

Instance (example)
Attributes (Fields)

Asso rule = {, } → {, } conf% w/cond; Itemset; → support% condition/total

Kmeans: Partition data into K clusters based on d
d obser, mean

Big Data
- Volume
- Velocity
- Variety (3V)
- Veracity (accuracy)
- Value

Trad SQL
- Atomic = either completed or nothing
- Consistency: all data cons~ after transact
- Isolated: Modification by transacts isoed
- Durable = effect of transact permanent

problem = hard to join tables, consistency

Sol → Unstructured DB
- too much data
- Volume grow fast
- data not consistent
- Time critical

Key-value store
| key | name: value name: value bins |

Sys don't know data
Good for app get data by key
can be resided distri...

NoSQL
- OK if most data correct
- format unknown or undef.
- will be replaced
- write once read multiple
- grow unpredicable
- dev write query

Docu store (≈ key-v ~)
values are documents
flexible schema
allow query against docs
(K-V only query key)

MapReduce = scaling out (not up)
- Auto parallel & distri
- Fault tolerance
- Status and monitoring
- data locality = co-locate data & node

- map: convert dataset element to break down to k-v pair
- ↓
- reduce: combine k-v pair into smaller k-v~ set

limit
- not all problem
- barrier between map & reduce

imple
- hadoop
- Google mapreduce

Hadoop Dist F S
Files split to blocks
↓ replicate across node
Namenode (master)
Datanode (worker)