数据库原理与实践

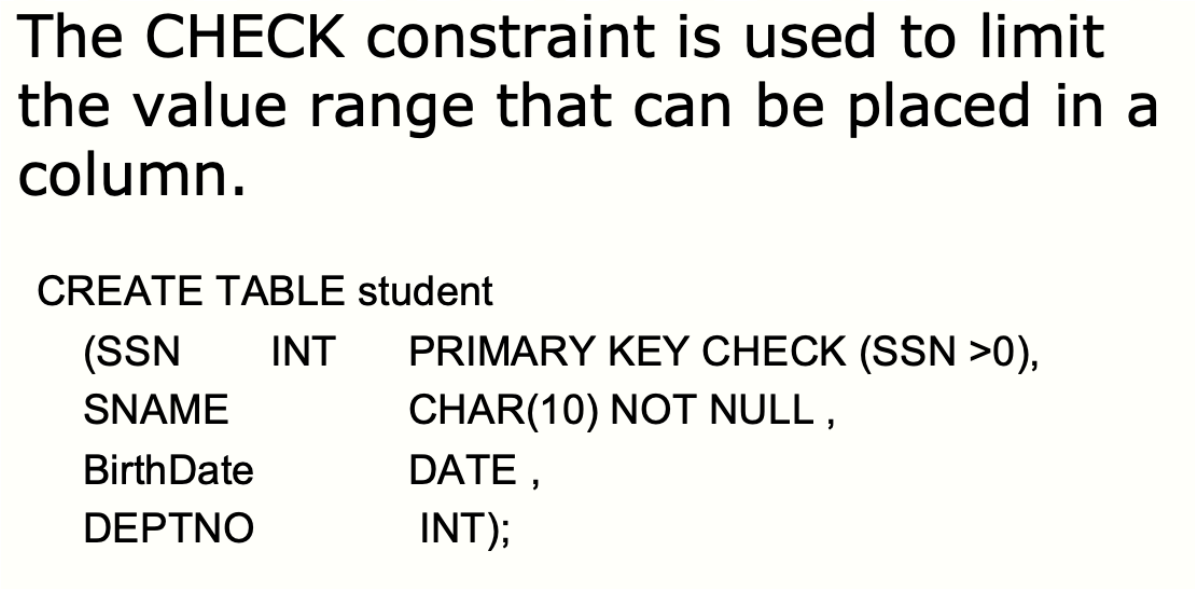
该部分建议看书，然后看PPT的具体例子。

SQL 语言基础

1. **发展历程**：SQL 全称为 Structural Query Language，原名 SEQUEL，1974 年由 IBM 研发。ANSI 于 1986 年发布首个标准 SQL - 86，ISO 于 1987 年采用，后续有 SQL - 89、SQL2 等标准更新，SQL3 仍在讨论中。市场上有超 100 种基于 SQL 的产品，如 Oracle、Sybase 等。
2. 主要组成部分
    - **DDL（Data Definition Language）**：用于定义表结构，如创建（CREATE）、修改（ALTER）、删除（DROP）数据库和表等操作。
    - **DML（Data Manipulation Language）**：包含交互式（如直接操作数据）和嵌入式（在高级编程语言中操作数据）两种，用于数据的增删改查。
    - **View Definition**：创建不同视角的数据库视图。
    - **Authorization（DCL）**：涉及数据访问权限管理。
    - **Integrity Constraints**：定义数据的各种约束条件。
    - **Transaction Definition**：确定原子命令组。

数据库与表的操作

1. 数据库操作
    - **创建数据库**：使用 `CREATE DATABASE <database_name>` 语句，如 `CREATE DATABASE STUDENT_DB`，不同 RDBMS 实现方式略有差异。
    - **删除数据库**：通过 `DROP DATABASE <database_name>` 实现，如 `DROP DATABASE STUDENT_DB`。
2. 表操作
    - **创建表**：`CREATE TABLE <table_name> (<attribute_name 1> <data_type 1>,...)`，可定义列的数据类型（如 VARCHAR、CHAR、NUMBER、DATE 等）及完整性约束（如 PRIMARY KEY、UNIQUE、NOT NULL、FOREIGN KEY、CHECK、DEFAULT 等）。例如创建学生表 `student`，包含 `SSN`（INT 型，非空且为主键）、`SNAME`（CHAR (10) 型）、`BirthDate`（DATE 型）、`DEPTNO`（INT 型）等列。
        - 如果想跨列创建主键，应该使用如下语句`CONSTRAINT pk PRIMARY KEY(col1, col2)`。
        - 如果想创建外键，可以使用`FOREIGN KEY(<本表的列名>) REFERENCES <外表名>(<外表的列名>)`。
        - 此外还可以加上check限制。



- 修改表结构
  - **添加列**：`ALTER TABLE <tablename> ADD (column_name datatype)`，如给 `student` 表添加 `Address` 列（CHAR (30) 型）。
  - **删除列**：`ALTER TABLE table_name DROP column_name`，如删除 `student` 表的 `Address` 列。
  - **修改列属性**：`ALTER TABLE table_name MODIFY column_name datatype`，如修改 `student` 表 `Address` 列的数据类型为 CHAR (50)。
  - **删除表限制**：`ALTER TABLE <table_name> DROP CONSTRAINT <constraint_name>`。
  - **修改列名**：`ALTER TABLE <table_name> RENAME CONSTRAINT <old_col_name> TO <new_col_name>`。
- **删除表**：`DROP TABLE <table_name>`，如 `DROP TABLE student`。

3. **索引操作**：使用 `CREATE INDEX index_name ON table_name (column_name)` 创建索引，如在 `student` 表的 `SNAME` 列创建索引 `StudentIndex`，可提高查询效率。

数据查询

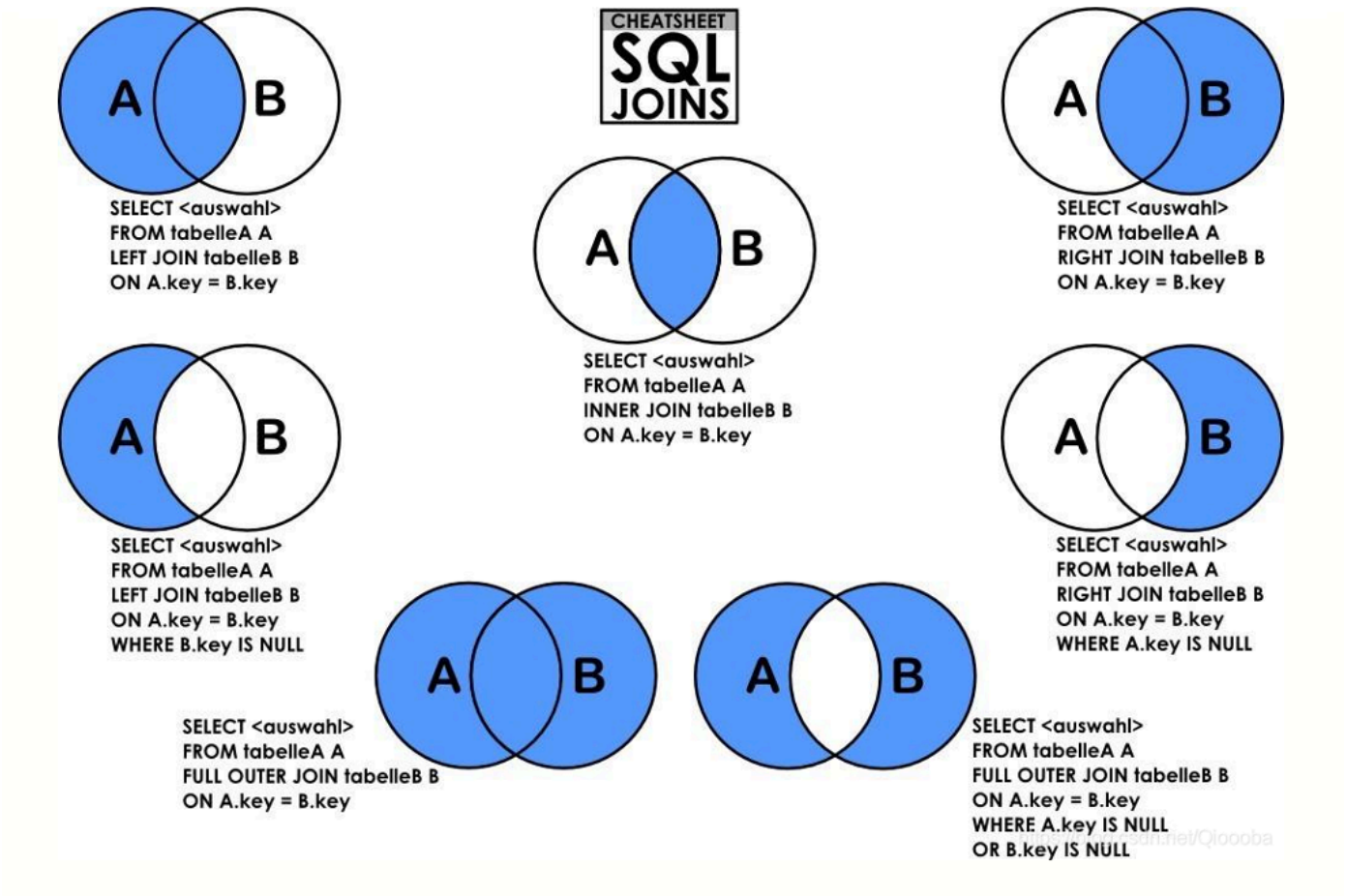
1. 基本查询语句（SELECT）：基本语法为

```
SELECT select_list FROM table_source [WHERE search_condition ] [GROUP BY group_by_expression ] [HAVING search_condition ] [ORDER BY order_expression [ASC | DESC ] ]
```

- 单表查询
  - **检索特定列或所有列**：如 `SELECT SSN, SName, GPA FROM Student` 检索学生表的指定列；`SELECT * FROM Student` 检索所有列；还可通过计算检索列，如 `SELECT SName, 2023 - AGE FROM Student` 计算学生年龄并与姓名一起检索，可使用别名（如 `SELECT SName, 2023 - AGE as Birthdate FROM Student`）使结果更易读，用 `DISTINCT` 关键字去除重复值（如 `SELECT DISTINCT AGE FROM Student`）。

- **条件筛选（WHERE 子句）**：支持比较（如 `AGE = 20`）、范围（`BETWEEN...AND`、`NOT BETWEEN...AND`）、集合成员（`IN`、`NOT IN`）、模式匹配（`LIKE`、`NOT LIKE`，通配符 `%` 匹配任意字符，`_` 匹配单个字符，可使用转义字符）、空值判断（`IS NULL`、`IS NOT NULL`）、复合条件（`AND`、`OR`）等搜索条件，如 `SELECT * FROM Student WHERE AGE = 20 AND GPA > 3`。如果模式匹配中包含通配符(%,_) 本身（如查询'20%', 我们可以使用如下语句：`Like '20#%' ESCAPE '#'`。
- **排序结果（ORDER BY 子句）**：按指定列升序（默认）或降序（使用 `DESC` 关键字）排序，如 `SELECT * FROM Student ORDER BY GPA` 或 `SELECT * FROM Student ORDER BY AGE DESC`。
- **聚合函数**：包括 `COUNT`（计数）、`AVG`（求平均）、`MAX`（求最大值）、`MIN`（求最小值）、`SUM`（求和），可与 `GROUP BY` 子句结合对分组数据进行操作，如 `SELECT AVG(GPA) FROM Student WHERE DEPTNO = '33'` 计算特定系学生的平均绩点；`GROUP BY` 语句将结果集按一列或多列分组，`HAVING` 子句用于筛选分组后的结果，如 `SELECT Avg(grade) FROM SC GROUP BY SSN HAVING Avg(grade)>81` 查找平均成绩大于 81 的学生。

- **多表查询**：通过 `FROM` 子句列出多个表，在 `SELECT` 和 `WHERE` 子句中引用表属性实现连接查询。支持内连接（`INNER JOIN`，基于连接谓词组合两表列值，如 `SELECT * FROM student, department WHERE student.DeptID = department.DeptID`）、外连接（`LEFT OUTER JOIN` 保留左表所有行、`RIGHT OUTER JOIN` 保留右表所有行、`FULL OUTER JOIN` 保留两表所有行），还可使用表别名简化查询，如 `SELECT SName, GPA FROM Students s, SC e, Courses c WHERE cname = 'CourseA' AND s.SSN = e.SSN AND e.Course_no = c.Course_no`。
- **集合操作**：支持 `UNION`（合并结果集，去除重复行，`UNION ALL` 保留重复行）、`INTERSECT`（取交集）、`MINUS`（取差集），要求操作的表具有兼容性，如 `(select SName from student WHERE ssn = any (select ssn from sc WHERE cno = 21001001) ) UNION (select SName from students WHERE ssn = any (select ssn from sc WHERE cno = 22003002) )`。
- **子查询**：嵌套在其他 SQL 语句中的 `SELECT` 语句，可出现在列列表、`FROM`、`GROUP BY`、`HAVING` 等子句中，如 `SELECT SNAME FROM student WHERE DEPTNO = (SELECT DEPTNO FROM student WHERE SNAME = 'JONES')`，还可使用比较运算符（`ALL`、`ANY`、`IN`、`NOT IN` 等）进行复杂查询，如 `SELECT sname FROM student WHERE gpa > some (SELECT gpa FROM student WHERE sex = 'female')`。
- **相关子查询**：外层查询的 `FROM` 子句中关系的某些属性在内存查询的 `WHERE` 子句中被引用，如 `SELECT * FROM Student s WHERE EXISTS (SELECT * FROM sc WHERE ssn = s.ssn)` 查找至少选了一门课的学生，执行时外层查询每一行都要对内层查询求值一次，遵循特定的属性名作用域规则。



视图操作

1. **创建视图**：`CREATE VIEW view_name AS SELECT column_list FROM table_name [WHERE condition]`，如 `CREATE VIEW view_student AS SELECT SSN, SNAME FROM student WHERE SNAME LIKE ‘ZHANG%’ WITH CHECK_OPTION`，视图是基于基表或其他视图的动态“虚拟表”，`CHECK_OPTION` 用于限制可更新视图的数据修改。
2. **修改视图**：使用 `ALTER VIEW view_name [(column_list)] [WITH ENCRYPTION] AS select_statement [WITH CHECK OPTION]`，如修改 `view_student` 视图增加 `Address` 列。
3. **删除视图**：`DROP VIEW view_name`，删除视图不影响基表，但查询引用已删除视图会出错。

数据更新

1. **插入数据（INSERT）**：如 `INSERT INTO emp (empno, ename, job, mgr, hiredate, sal, comm, deptno) VALUES(7500, 'CAMPBELL', 'ANALYST', 7566, ‘1992 - 3 - 5', 24500,0, 40)`，还可从其他表查询数据插入新表，如 `INSERT INTO Deptage(Sdept, Avgage) SELECT Sdept, AVG(Sage) FROM Student GROUP BY Sdept`。
2. **更新数据（UPDATE）**：如 `UPDATE emp SET comm = 0`，可结合条件更新特定数据，如 `UPDATE emp SET sal = sal * 1.15 WHERE (job = 'ANALYST' OR job = 'CLERK') AND deptno = 20`。
3. **删除数据（DELETE）**：如 `DELETE FROM emp WHERE job = 'SALESMAN' AND comm < 100`，删除表中所有数据使用 `DELETE FROM <tablename>`，删除操作可能会因数据依赖关系产生问题，如删除学生成绩低于平均绩点的记录时，平均绩点会随删除操作改变，SQL 中通常先计算要删除的元组，再一次性删除。

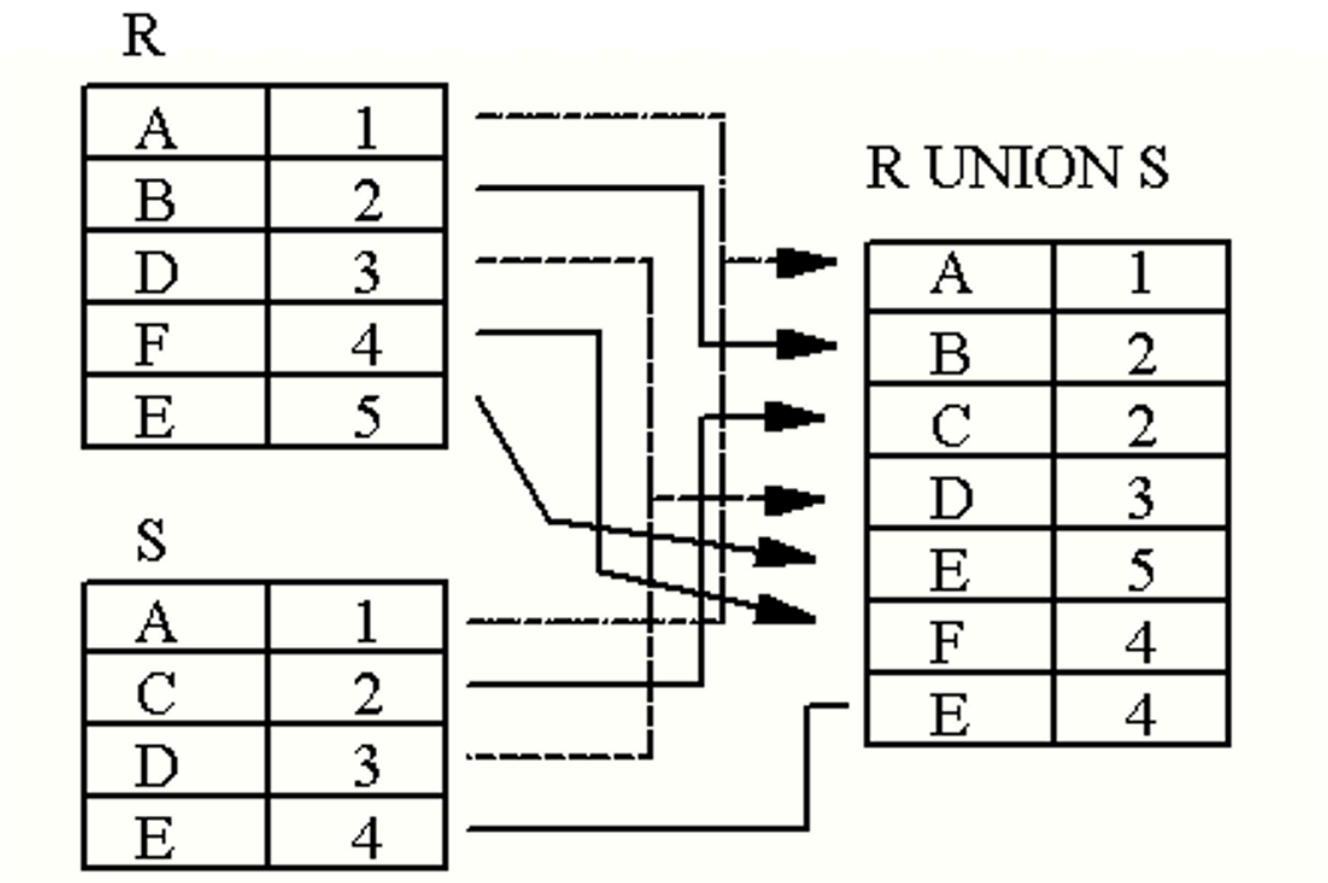
关系代数

1. 基本概念与操作

   ：是用于操作关系集的操作集合，输出结果仍是关系。操作包括集合操作（

   并

```
UNION
```



   、交

```
INTERSECTION
```

R

A	1
B	2
D	3
F	4
E	5

S

A	1
C	2
D	3
E	4

R INTERSECTION S

A	1
D	3

、差

```
DIFFERENCE
```

R

A	1
B	2
D	3
F	4
E	5

S

A	1
C	2
D	3
E	4

R DIFFERENCE S

B	2
F	4
E	5

S DIFFERENCE R

C	2
E	4

Notice that Difference has a directionality to it, e.g. R-S and S-R are not the same thing.

、笛卡尔积

```
CARTESIAN PRODUCT
```

R

A	1
B	2
D	3
F	4
E	5

S

A	1
C	2
D	3
E	4

R CROSS S

A	1	A	1
A	1	C	2
A	1	D	3
A	1	E	4
B	2	A	1
B	2	C	2
B	2	D	3
B	2	E	4
D	3	A	1
D	3	C	2
D	3	D	3
D	3	E	4

F	4	A	1
F	4	C	2
F	4	D	3
F	4	E	4
E	5	A	1
E	5	C	2
E	5	D	3
E	5	E	4

）和特殊关系操作（

选择

```
SELECT
```

、投影

```
PROJECTION
```

、连接

```
JOIN
```

，连接又分为 theta 连接、等值连接、自然连接、外连接等）。

- **并操作（UNION）**：结果包含参与运算关系中所有元组，需满足并兼容性（关系度数相同且对应属性域相同），如 `R UNION S`。
- **差操作（DIFFERENCE）**：如 `R - S` 结果是在 `R` 中但不在 `S` 中的元组，具有方向性。
- **交操作（INTERSECTION）**：返回同时在两个关系中的元组，可由差操作推导（`R ∩ S = R - (R - S) = S − (S - R)`）。
- **笛卡尔积（CARTESIAN PRODUCT）**：将两个关系的属性组合成新关系，新关系的度数为两关系度数之和，元组数为两关系元组数之积，如 `R × S`。
- **选择操作（SELECT）**：检索满足选择条件的元组子集，如 `σ<selection - condition> (Relation)`，具有交换律。
- **投影操作（PROJECTION）**：选择关系中的指定列，如 `π<attributes> (Relation)`。
- **连接操作（JOIN）**：`R ⨝ join - condition S` 返回满足连接条件的 `R × S` 中的元组，等值连接是连接条件只用等号的特殊情况，自然连接省略连接条件，会自动去除重复属性列，外连接（左外连接、右外连接、全外连接）用于保留未匹配的元组并以空值表示。

- **除法操作（DIVISION）**：虽不是基本操作，但可用于表达如查找选了所有课程的学生等查询，需满足一定前提条件。

应用示例

对于给定的关系数据库（如读者 `R`、书籍 `B`、借阅 `RB` 关系），可分别用关系代数和 SQL 实现以下查询：

1. 列出 “人民出版社” 所出版的书名和作者（关系代数：通过对 `B` 关系进行选择和投影操作；SQL：使用 `SELECT` 和 `WHERE` 子句结合条件查询）。
2. 列出有书籍超期未还的读者姓名、书名、应还日期（关系代数：涉及多表连接和条件筛选操作；SQL：通过连接 `R`、`B`、`RB` 表并设置条件查询）。
3. 列出借阅 “莫言” 所有书籍的读者姓名（关系代数：可能需要除法操作或复杂的连接和筛选；SQL：可使用子查询和相关子查询实现）。还可创建反映女性读者姓名和年龄的视图 `Female`，以及写出创建 `R` 和 `RB` 表的 SQL 语句。