Basic SQL

```
SELECT [DISTINCT] target-list
FROM relation-list
WHERE qualification
```

- 其实WHERE是可以省略，来表示不晒选行
- 显示指明DISTINCT：**消除重复列**
- 别名：Range Variables
  - 自身连接 SELF-JOIN 的情况：

    ```
    SELECT  x.sname, x.age, y.sname, y.age
    FROM    Sailors x, Sailors y
    WHERE   x.age > y.age
    ```
- SELECT、WHERE可以加入算入表达式
- AS关键字：由于给算数表达式起名
- LIKE关键字：用在WHERE，满足模板
  - `_` 表示任何一个字符
  - `%` 表示上一个字符有0个或多个
- UNION：将两个查询结果并在一起
- INTERSECT：将两个查询结果交在一起

  ```
  SELECT S.sid
  FROM   Sailors S, Boats B, Reserves R
  WHERE  S.sid=R.sid
          AND R.bid=B.bid
          AND B.color=‘red’
  INTERSECT
  SELECT S.sid
  FROM   Sailors S, Boats B, Reserves R
  WHERE  S.sid=R.sid
          AND R.bid=B.bid
           AND B.color=‘green’
  ```

  > 或者使用自交来完成上述逻辑

> ```
> SELECT R1.sid
> FROM   Boats B1, Reserves R1,
>  Boats B2, Reserves R2
> WHERE R1.sid=R2.sid
>   AND R1.bid=B1.bid
>   AND R2.bid=B2.bid
>   AND (B1.color=‘red’ AND B2.color=‘green’)
> ```

- EXCEPT：减表（上表减去下表）
- IN / NOT IN：实现嵌套查询

  ```
  SELECT S.sname
  FROM   Sailors S
  WHERE  S.sid IN
     (SELECT  R.sid
      FROM    Reserves R
      WHERE  R.bid=103)
  ```

- EXISTS / NOT EXISTS：以下示例与上一段代码逻辑相同，但以下代码具有主外键的检查

  ```
  SELECT  S.sname
  FROM    Sailors S
  WHERE EXISTS
        (SELECT  *
         FROM  Reserves R
         WHERE R.bid=103 AND S.sid=R.sid)
  ```

  看作函数：外层传入参数S，调用内层

- ANY / ALL

  ```
  SELECT *
  FROM   Sailors S
  WHERE  S.rating > ANY
     (SELECT  S2.rating
      FROM  Sailors S2
      WHERE S2.sname=‘Horatio’)
  ```

  这个例子使用了别名

- GROUP BY 聚合成组（后面接参数为分组依据）
  - For each rating, find the average age of the sailors

```
SELECT  S.rating,  AVG (S.age)
FROM  Sailors S
GROUP BY S.rating
```

- HAVING 对分组进行限制

  ```
  SELECT  S.rating, MIN (S.age)
  FROM  Sailors S
  WHERE  S.age >= 18
  GROUP BY  S.rating
  HAVING  COUNT (*) > 1
  ```

- ORDER BY 排序

  ```
  SELECT  S.rating, S.sname, S.age
      FROM  Sailors S, Boats B, Reserves R
      WHERE  S.sid=R.sid
          AND R.bid=B.bid AND B.color='red'
      ORDER BY  S.rating, S.sname;
  ```

- null
- JOIN 连接

  ```
  SELECT (column_list)
  FROM  table_name
    [INNER | {LEFT |RIGHT | FULL } OUTER] JOIN table_name
      ON qualification_list
  WHERE …
  ```

  - INNER JOIN 内连接

    ```
    SELECT s.sid, s.name, r.bid
        FROM Sailors s INNER JOIN Reserves r
        ON s.sid = r.sid
    ```

  - LEFT OUTER JOIN 左外连接：以关键词左侧的表进行连接
    - 连接：不满足符合连接条件的元组也输出（**用空值来表示**）
  - RIGHT OUTER JOIN 右外连接
  - FULL OUTER JOIN：两边没获得匹配的元素都输出，空值代替
- 一些可用的数学函数
  - COUNT(*) / COUNT([DISTINCT] A)
  - SUM([DISTINCE] A)

    - AVG([DISTANCE] A)
    - MAX(A) MIN(A)
- CREATE VIEW 创建视图，可以看作创建了一个临时变量

  ```
  CREATE VIEW Reds
  AS SELECT  B.bid,  COUNT (*) AS scount
       FROM Boats B, Reserves R
       WHERE  R.bid=B.bid AND   B.color='red'
        GROUP BY  B.bid
  ```

- 权限
    - 可以作用在表、视图上，
    - 特权包括：选择、插入、删除、引用、ALL
    - 授予、回收
    - 指定用户、用户组
- CHECK 约束（创建表时）

  ```
  CREATE TABLE   Sailors
      ( sid  INTEGER,
      sname  CHAR(10),
      rating  INTEGER,
      age  REAL,
      PRIMARY KEY  (sid),
      CHECK  ( rating >= 1
          AND rating <= 10 ))
  ```

  > <>：不等于
  >
  > ```
  > CREATE TABLE  Reserves
  >     ( sname  CHAR(10),
  >     bid  INTEGER,
  >     day  DATE,
  >     PRIMARY KEY  (bid,day),
  >     CONSTRAINT  noInterlakeRes
  >     CHECK  (`Interlake' <>
  >             ( SELECT  B.bname
  >             FROM  Boats B
  >             WHERE  B.bid=bid)))
  > ```
  >
  > 上述的含义是：不允许叫Interlake的船被预约。

- 正则表达式符号：`~`
    - 以AB开头：`^AB`

- 以CD结尾：`CD$`
- 或：`|`