WHEREを用いた絞り込み詳細

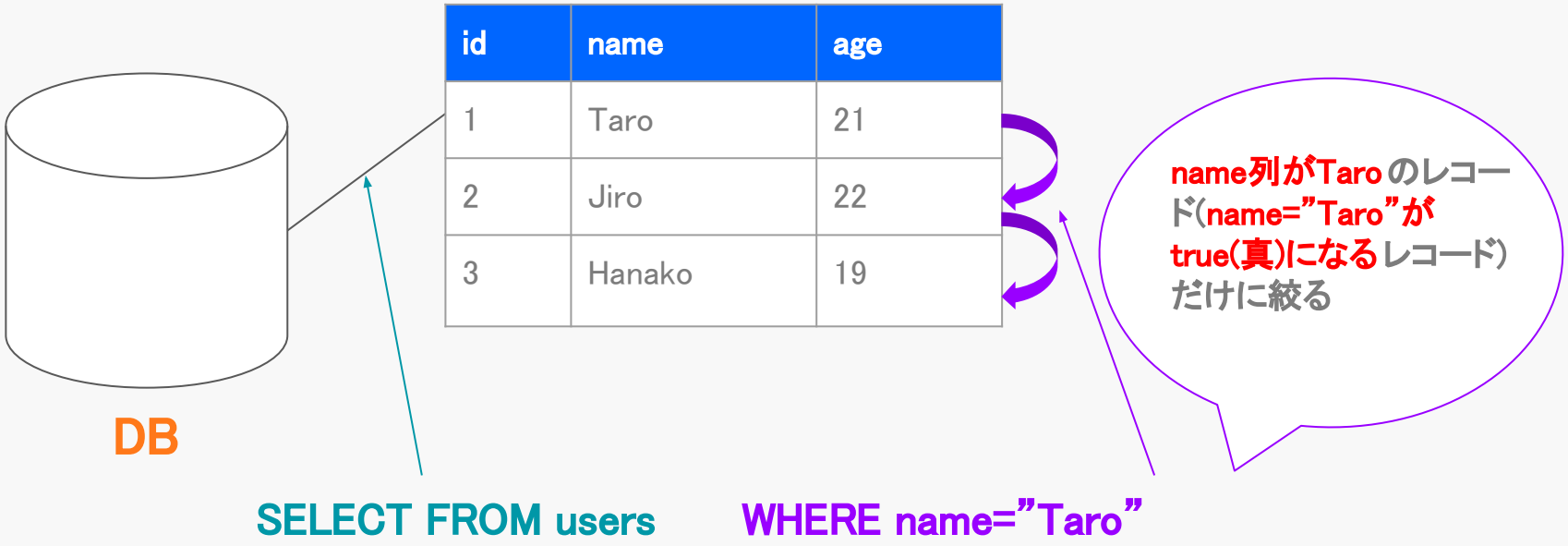
1日目	2日目	3日目	4日目	5日目	6日目	7日目
8日目	9日目	10日目	11日目	12日目	13日目	14日目
15日目	16日目	17日目	18日目	19日目	20日目	21日目

WHEREの基本


出版年が1990年よりも前
作者が村上春樹
WHEREで絞り込む

WHEREについて

テーブル内から特定のレコードを**絞り込む**SQL文。
WHEREの条件式で比較した結果、真となるレコードに絞り込む。



WHEREの基本文

3000 > 1000

1(正しい)

500 > 1000

0(正しくない)

=	左辺と右辺が等しいか
<	左辺が右辺より小さいか
>	左辺が右辺より大きいか
<=	左辺が右辺以下か
>=	左辺が右辺以上か
<>, !=	左辺と右辺が等しくないか

```
SELECT 1 = 1  # 1と表示
SELECT 1 <> 1  # 0と表示
SELECT 100 > 1  # 1と表示
SELECT -10 >= 1  # 0と表示
SELECT “Taro” = “Taro”  # 1と表示
SELECT “Taro” = “Jiro”  # 0と表示
```

```
SELECT “a” < “b”  # 1と表示
SELECT “c” < “b”  # 0と表示
SELECT “A” < “B”  # 1と表示
SELECT “C” < “B”  # 0と表示
SELECT “a” < “B”  # 1と表示
SELECT “Taro” > “Jiro”  # 1と表示
```

テーブルから絞り込む

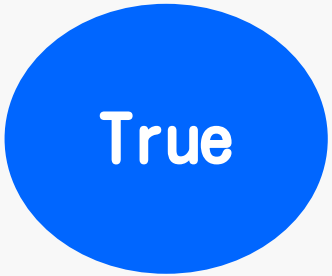
同じようにテーブルからレコードを絞り込むことができる

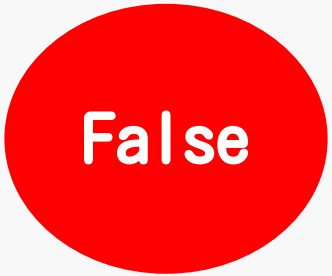
```
# id列が1のレコードだけを選択して表示
SELECT * FROM users WHERE id=1;

# age列が20より大きいレコードだけを選択して表示
SELECT * FROM users WHERE age>20;

# name列がTaroでないレコードだけを選択して表示
SELECT * FROM users WHERE name<>"Taro";

# name列がTaroでないレコードのname列をhogeに変更
UPDATE users SET name = "foge" WHERE name<>"Taro";

# id列が3よりも小さいレコードを削除
DELETE FROM users WHERE id < 3;
```

NULLについて
(IS NULL, IS NOT NULL)

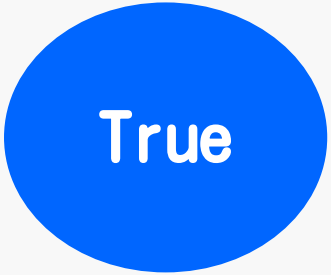
3値論理について

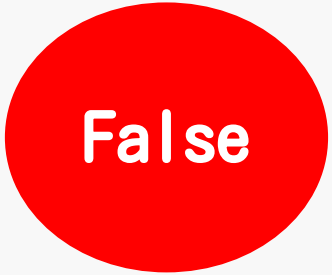
一般的な論理体系では、真か偽か2の論理体系で表される。





DBでは、真か偽が以外に、不明(unknown)であることを表すNULLという値が存在する。この3値の論理値で表す体系を3値論理という。





NULLのレコードを取り出す

NULLは、直接=では取り出せない

SELECT * FROM users WHERE name = NULL

nameの値がNULLであったとしても、
NULL=NULLは不明=不明 で、真(True)にならない

NULLのものを取り出すには、IS NULL, IS NOT NULLを用いる

NULLのレコードを取り出す(IS NULL)
nameがNULLのレコードを、usersテーブルから取り出す
SELECT * FROM users WHERE nameIS NULL;

NULLでないレコードを取り出す(IS NOT NULL)
nameがNULLでないレコードを、usersテーブルから取り出す
SELECT * FROM users WHERE nameIS NOT NULL;

BETWEEN, LIKE
(NOT BETWEEN, NOT LIKE)

BETWEEN, NOT BETWEEN

指定した範囲内に、収まっているレコードを取り出す

式(カラム名) BETWEEN 値1 AND 値2 ― 値1以上、値2以下のレコードを取り出す

```
# ageが10以上17以下のレコードをpeopleテーブルから取り出し
SELECT * FROM people WHERE age BETWEEN 10 AND 17;

# NOTを付けると条件を否定する
# ageが10未満、または17より大きいレコードをpeopleテーブルから取り出し
SELECT * FROM people WHERE age NOT BETWEEN 10 AND 17;

# UPDATEとともに利用する
UPDATE people SET generation="平成世代" WHERE age BETWEEN 5 AND 30;

# DELETEとともに利用する
DELETE FROM people WHERE age NOT BETWEEN 5 AND 30;
```

LIKE, NOT LIKE

指定したパターンに一致するか

式(カラム名) LIKE “パターン”

%	任意の0文字以上の文字列
_	任意の1文字

```
# nameがTで始まるレコードをpeopleテーブルから取り出す
SELECT * FROM people WHERE name LIKE "T%";

# nameがRで終わるレコードをpeopleテーブルから取り出す
SELECT * FROM people WHERE name LIKE "%R";

# nameにpが含まれるレコードをpeopleテーブルから取り出す
SELECT * FROM people WHERE name LIKE "%p%";

# nameがあ〇のレコードをpeopleテーブルから取り出す(あじ、あご　など)
SELECT * FROM products WHERE name LIKE "あ_";
```

IN, NOT IN, ANY, ALL

IN, NOT IN

()内に列挙した複数の値のいずれかに合致したものを取り出す

式(カラム名) IN (値1, 値2, 値3, …)

```
# customersテーブルからcountry列がJapan, US, UKのいずれかに当てはまるものを取り出す
SELECT * FROM customers
WHERE country IN ('Japan', 'US', 'UK');

# suppliersテーブルからcountry列を取り出す。countryに該当するものをcustomersテーブルから取り出す
SELECT * FROM customers
WHERE country IN (SELECT country FROM suppliers);
```

ANY

取得した値のリストと比較して、いずれかが真のものを取り出す

式(カラム名) 比較演算子 ANY (SELECT …)

goodsテーブルからidが5よりも大きなレコードのpriceを取り出し。取り出したpriceのいずれかの値より小さいレコードをproductsテーブルから取り出す。

```
SELECT * FROM products
WHERE price < ANY (SELECT price FROM goods WHERE id > 5);
```

ALL

取得した値のリストと比較して、全てが真のものを取り出す

式(カラム名) 比較演算子 ALL (SELECT …)

goodsテーブルからidが5よりも大きなレコードのpriceを取り出し。取り出したpriceのどの値よりも小さいレコードをproductsテーブルから取り出す。

```
SELECT * FROM products
WHERE price < ALL (SELECT price FROM goods WHERE id > 5);
```

複数の条件を組み合わせる
AND(&&), OR(||)

AND

2つの条件の両方が真の場合だけ、真となる

```
条件式1 AND 条件式2
```

```
# ageが20より大きくnameがAで始まるレコードを、customersテーブルから取り出す
SELECT * FROM customers
WHERE age>20 AND name LIKE “A%”;
```

OR

2つの条件の**どちらかが**真の場合だけ、真となる

```
条件式1 OR 条件式2
```

```
# ageが20より大きい、またはnameがAで始まるレコードを、customersテーブルから取り出す
SELECT * FROM customers
WHERE age>20 OR name LIKE “A%”;
```

AND + OR

ANDとORと()を用いて、複数の条件を繋ぐことができる

```
条件式1 AND (条件式2 OR 条件式3)
```

```
# idが1または5でかつ、salaryが5000よりも大きいレコードを、employeesテーブルから取り出す
SELECT * FROM employees
WHERE salary > 5000 AND (id = 1 OR id = 5);
```

NOT

条件式の直前につけると条件式の否定になる

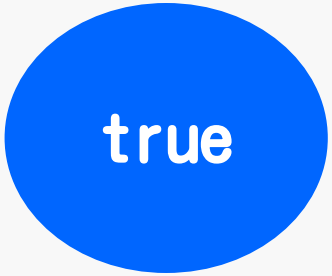
NOT 条件式

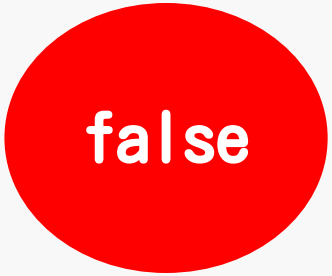
```
# ageが20より大きくない(20以下)のレコードを、studentsテーブルから取り出す
SELECT * FROM students
WHERE NOT age > 20;
```

NULLについて

NULLについて

NULLは値が不明であることを表し、SQLにとって3番目の論理体系である。そのことがSQLの処理に意図しない結果をもたらすことがある。





例えば、以下のSQLの実行結果は、直観的にはすべてのレコードを選択できそうだが、NULLが入っている場合そのようにならない

```
# nameがTAROかTAROでない人を選択するSQL( nameがNULLの人は選択されない )
SELECT * FROM people WHERE name="TARO" OR name<>"TARO"
```

WHEREについて

WHEREでは、式で評価した結果がtrue（真）の場合のレコードが、絞り込まれる。

```
# nameがTAROかTAROでない人を選択するSQL( nameがNULLの人は選択されない )
SELECT * FROM people WHERE name="TARO"  OR name<>"TARO"
```

OR: 左右どちらかの条件がtrue(真)
のときにWHEREの結果true(真)となる

id	name
1	TARO
2	JIRO
3	NULL

name="TARO"はtrue(真)のため、選択される

name<>"TARO"はtrue(真)のため、選択される

name="TARO"はNULL, name<>"TARO"はNULLのため、
選択されない

NULLの評価結果一覧

評価式	結果
NULL = “ABC”	NULL
NULL > 0	NULL
NULL < 0	NULL
NULL <> 1	NULL
NULL = NULL	NULL
NULL <> NULL	NULL
NULL > NULL	NULL
NULL < NULL	NULL
NULL IS NULL	true

AND, ORとNULLと真理値表

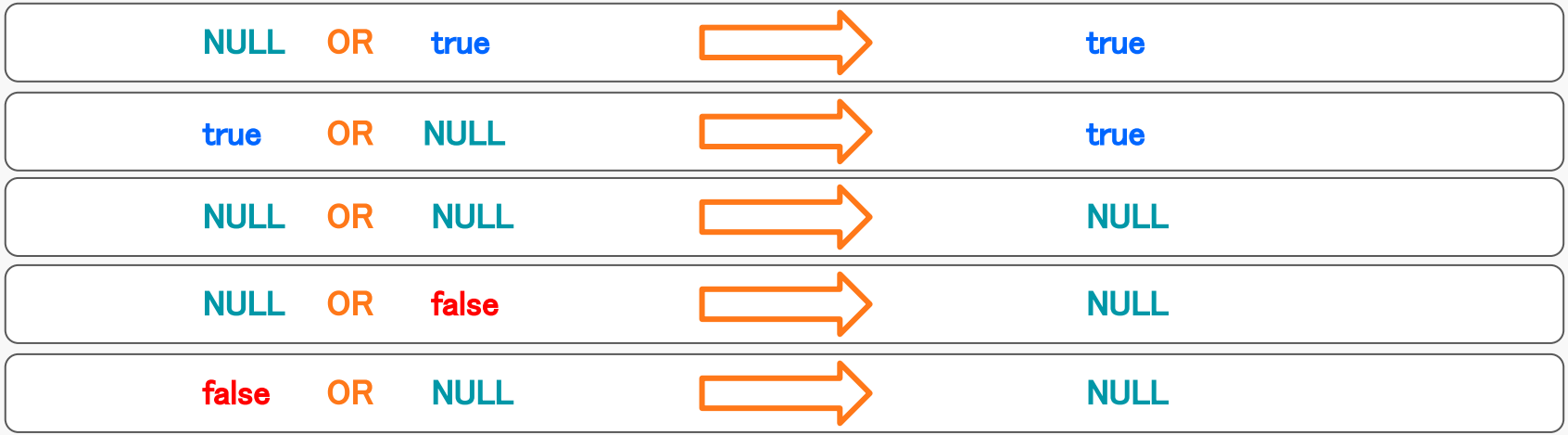
ANDやORを用いて2つの式を繋いだ場合に、左右の式の値に応じて、全体の評価結果が異なるため、
その結果を以下に記述する

```
WHERE 左の評価式 AND(OR) 右の評価式
```

ANDの場合

NULL	AND	true	⇒	NULL
true	AND	NULL	⇒	NULL
NULL	AND	NULL	⇒	NULL
NULL	AND	false	⇒	false
false	AND	NULL	⇒	false

ORの場合



AND	t	N	f
t	t	N	f
N	N	N	f
f	f	f	f

OR	t	N	f
t	t	t	t
N	t	N	N
f	t	N	f

ALLで利用する場合の注意点

```
SELECT * users WHERE age < ALL(SELECT age FROM other_users)
```

users

name	age
Yuko	15
:	:

other_users

name	age
Taro	17
Jiro	18
Saburo	NULL

15 < ALL(17, 18, NULL)

15 < 17 → true
AND
15 < 18 → true
AND
15 < NULL → NULL

NULL

NULLが入るとALLの結果がNULLになる

IN, NOT INとNULL

IN, NOT INでNULLを利用する場合の注意点

下のSQLは、いずれも想定した動作をしない‼

```
# nameがTaro, Jiro, NULLのいずれかのレコードを取り出すはずのSQL
SELECT * people WHERE name  IN (“Taro”, “Jiro”, NULL)
```

```
# nameがTaro, Jiro, NULLのいずれか でないレコード取り出すはずのSQL
SELECT * people WHERE name  NOT IN (“Taro”, “Jiro”, NULL)
```

NULLはSQLを理解する上で重要な概念のため、今後も取り上げていきます