

応用情報技術者試験

第6章 データベース

1. データベースの設計手順

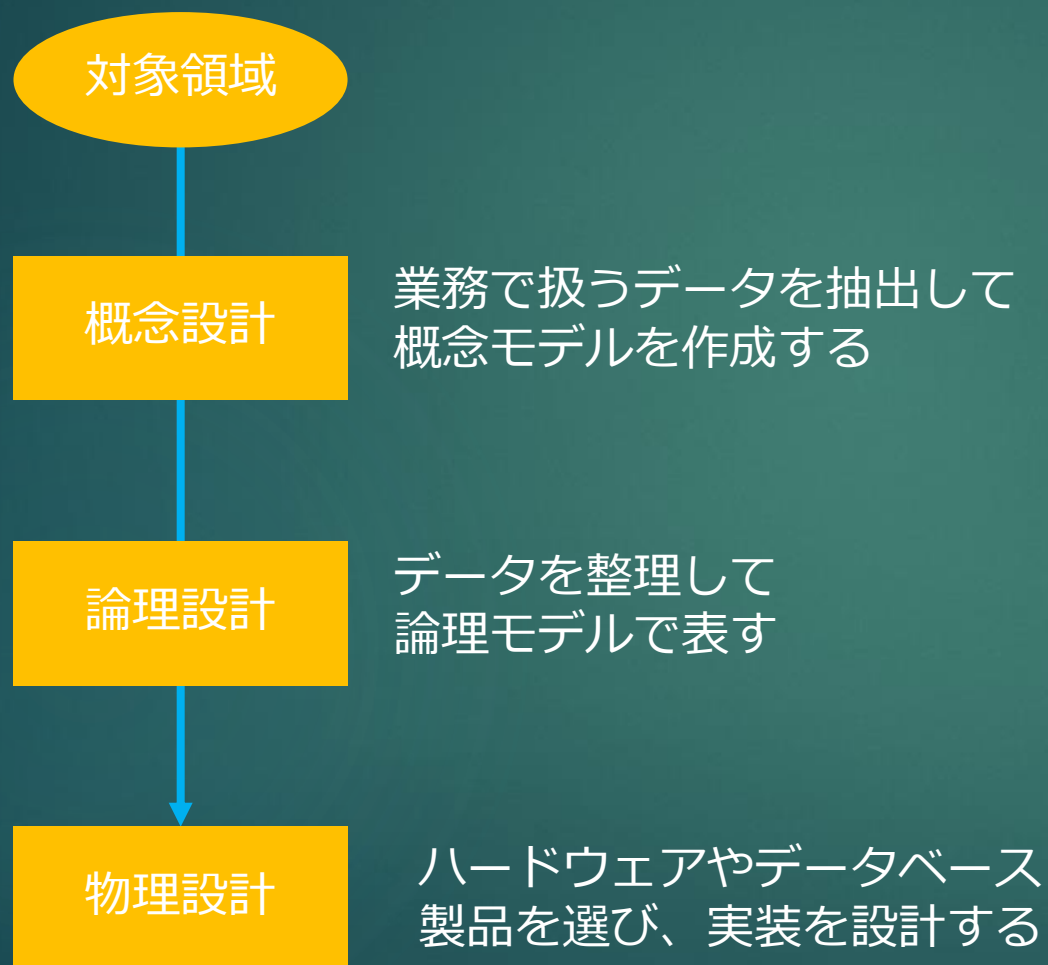
データベースとは

▶ データベース

データをアプリケーションから切り離して集中管理する仕組みのこと。アプリケーションはデータベースから目的のデータを検索し、参考や更新などの処理を行う。

- ▶ データベースの登場により、企業内のデータはデータベースに一元管理され、アプリケーションレベルで共有できるようになった。

データベースの設計手順



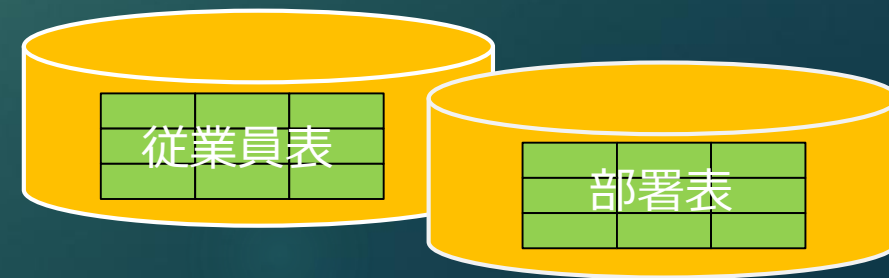
概念モデル



論理モデル（関係モデル）



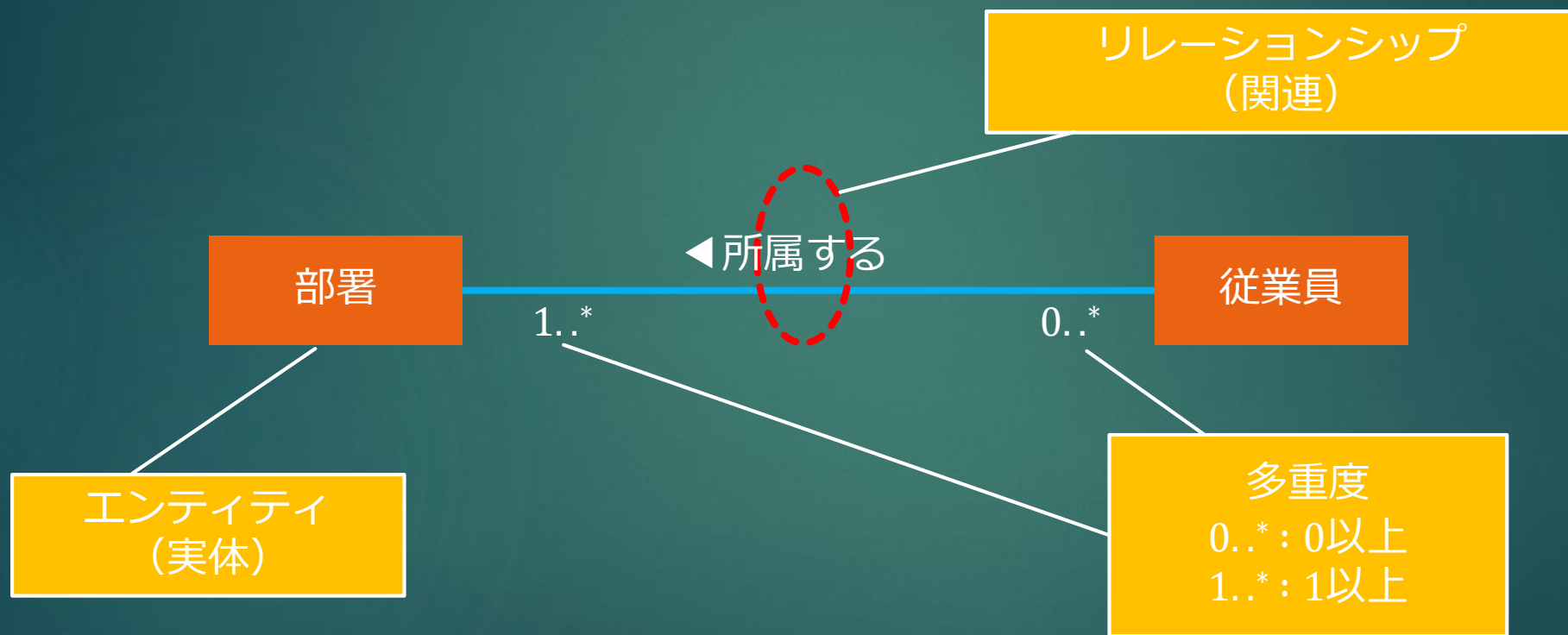
関係データベース



概念設計

- ▶ 概念設計では、データベース化の対象となる領域を分析し、どのようなデータが存在するかを洗い出す。データは、**実体（エンティティ）**ごとにまとめる。たとえば社員番号や氏名は、“従業員”という実態の属性としてまとめられる。関係の深い実体間には、関連（リレーションシップ）がつけられる。
- ▶ 分析の結果は、UMLクラス図や**E-R図(ERD : Entity-Relationship Diagram)**などを用いてモデル化する（**概念モデル**）。いずれの記法でも、実体を四角形で、実体間に関連が存在する場合にはそれらを線で結んで表す。設計の初期段階では、属性を省略することもある。

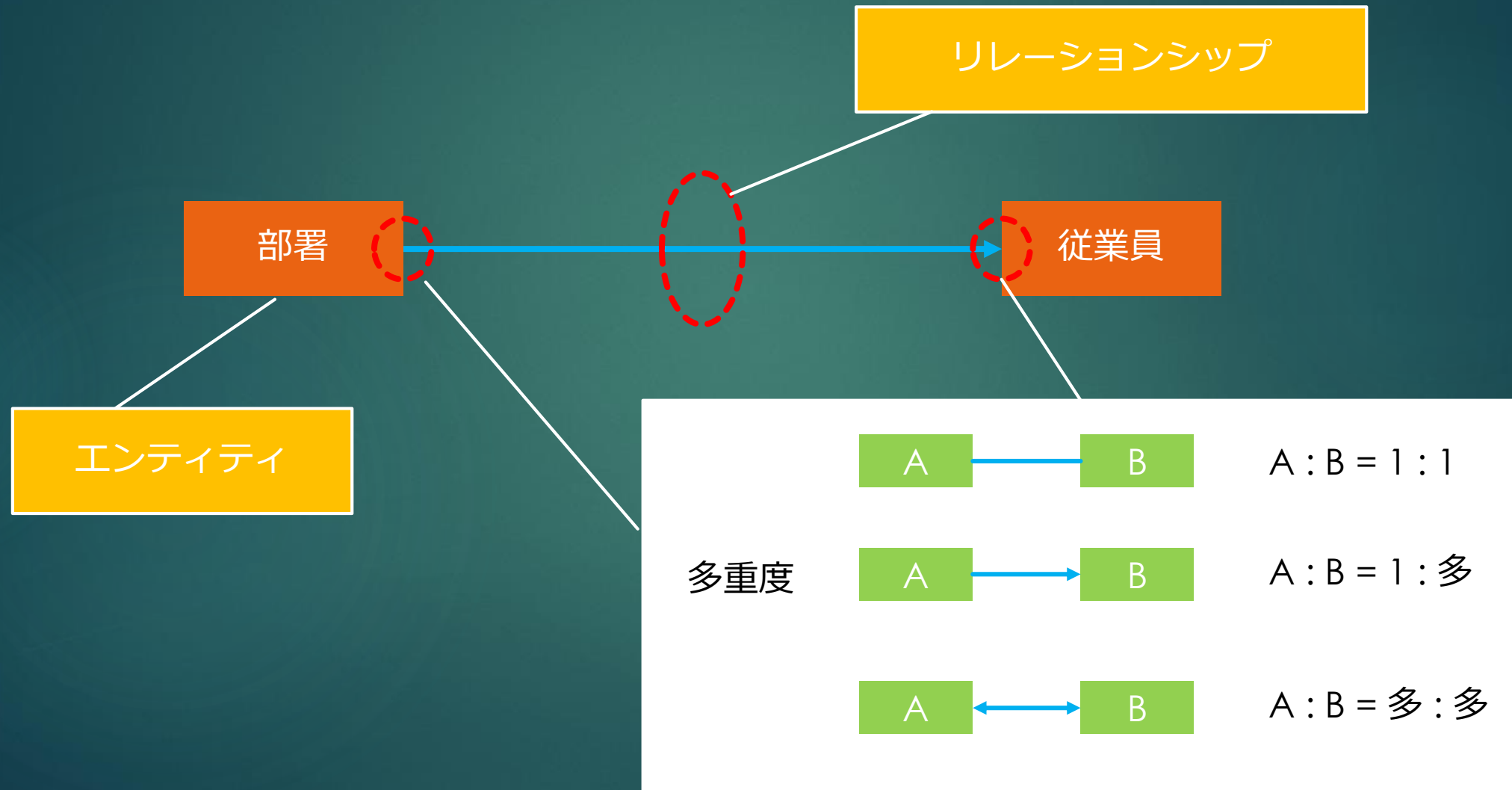
UML



多重度
実体同士
の数的な
対応関係

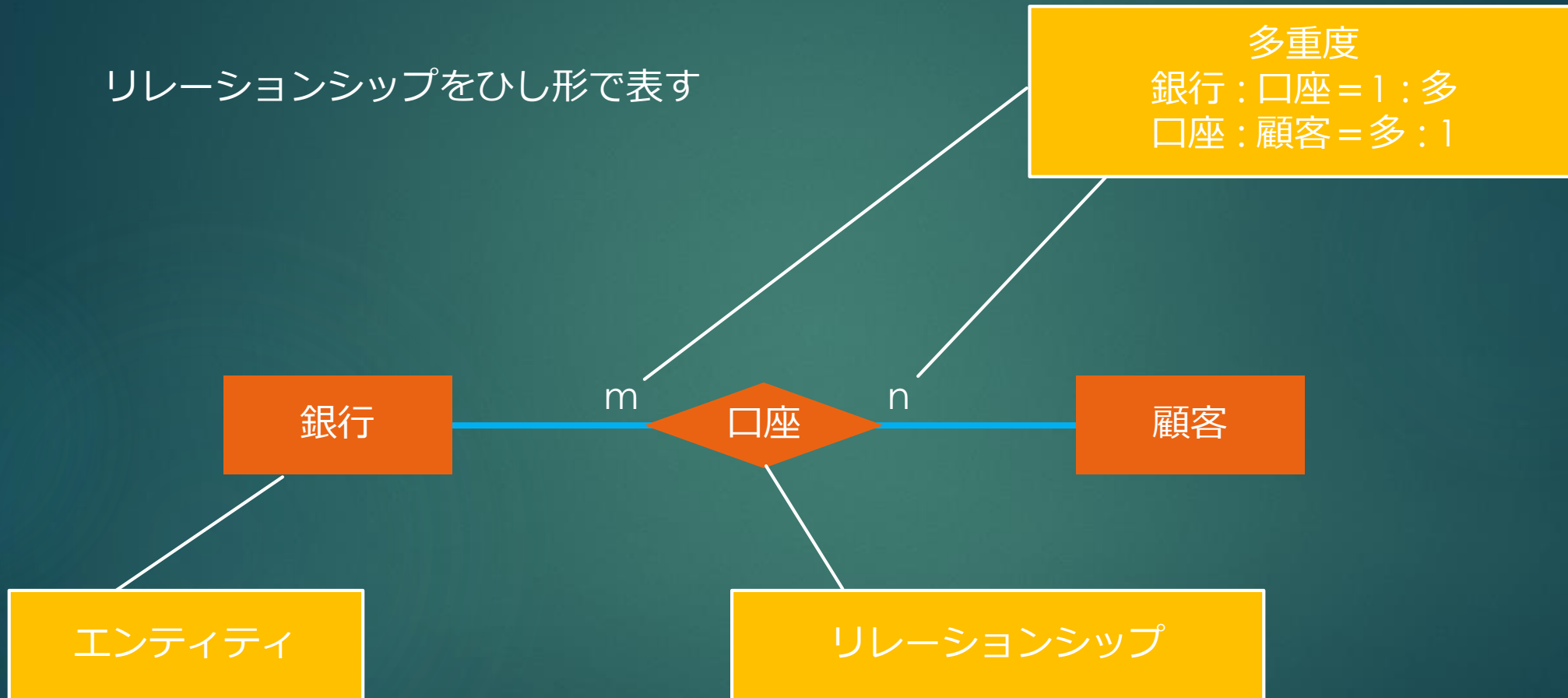
この記法が最も多重度を細かく表現できる

E-R図



E-R図

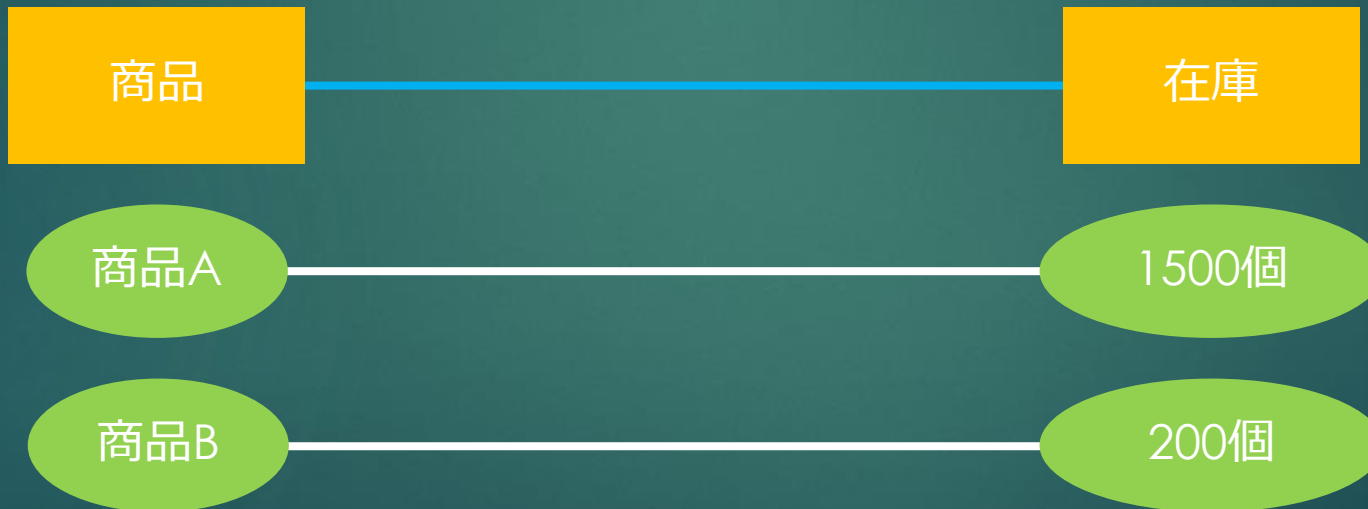
リレーションシップをひし形で表す



多重度とインスタンス

- ▶ 実体をとる**実現値**、例えば実体“部署”に対する“営業部”や“システム部”を**インスタンス**とよぶ。実体間の多重度は、正確には「**インスタンス間の数**的な**対応関係**」である。

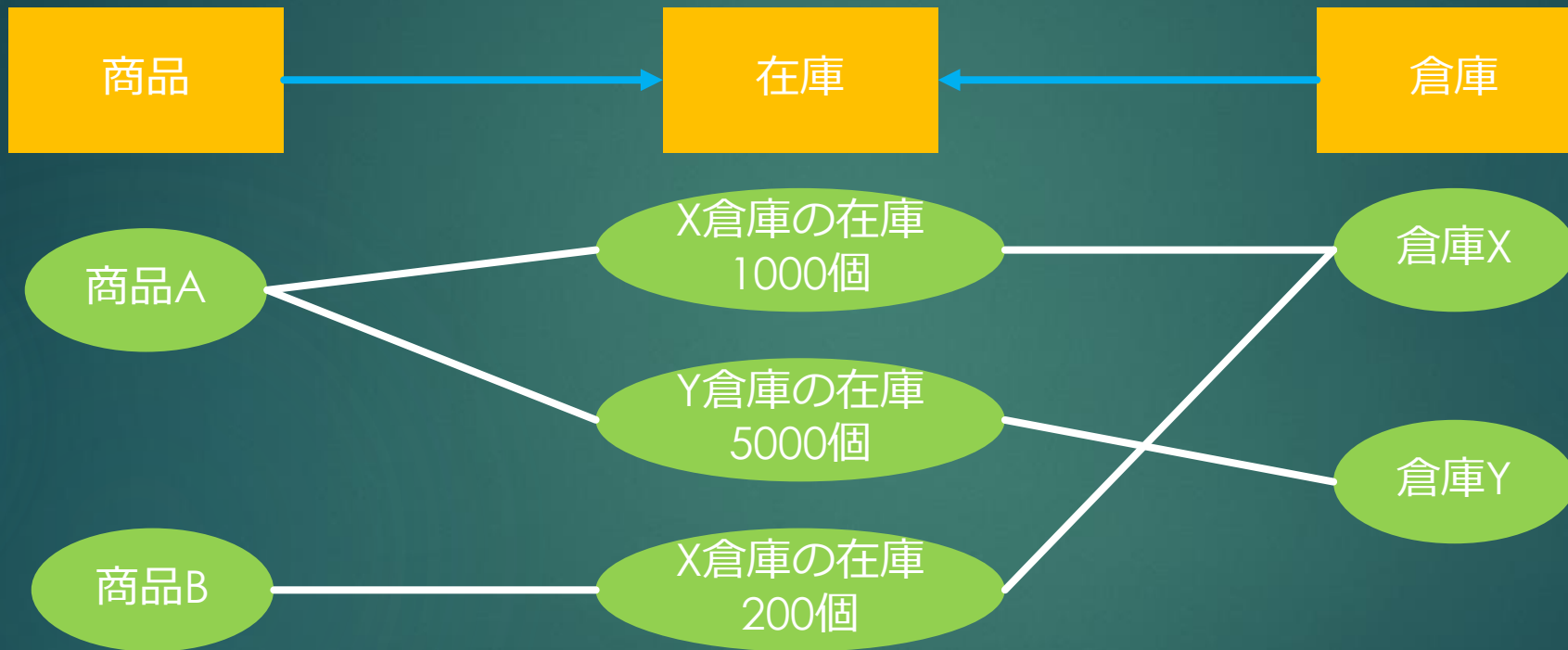
- ▶ 在庫のみの管理



- ▶ 在庫を商品ごとにまとめて管理するだけなら商品と在庫は1:1対応する。

多重度とインスタンス

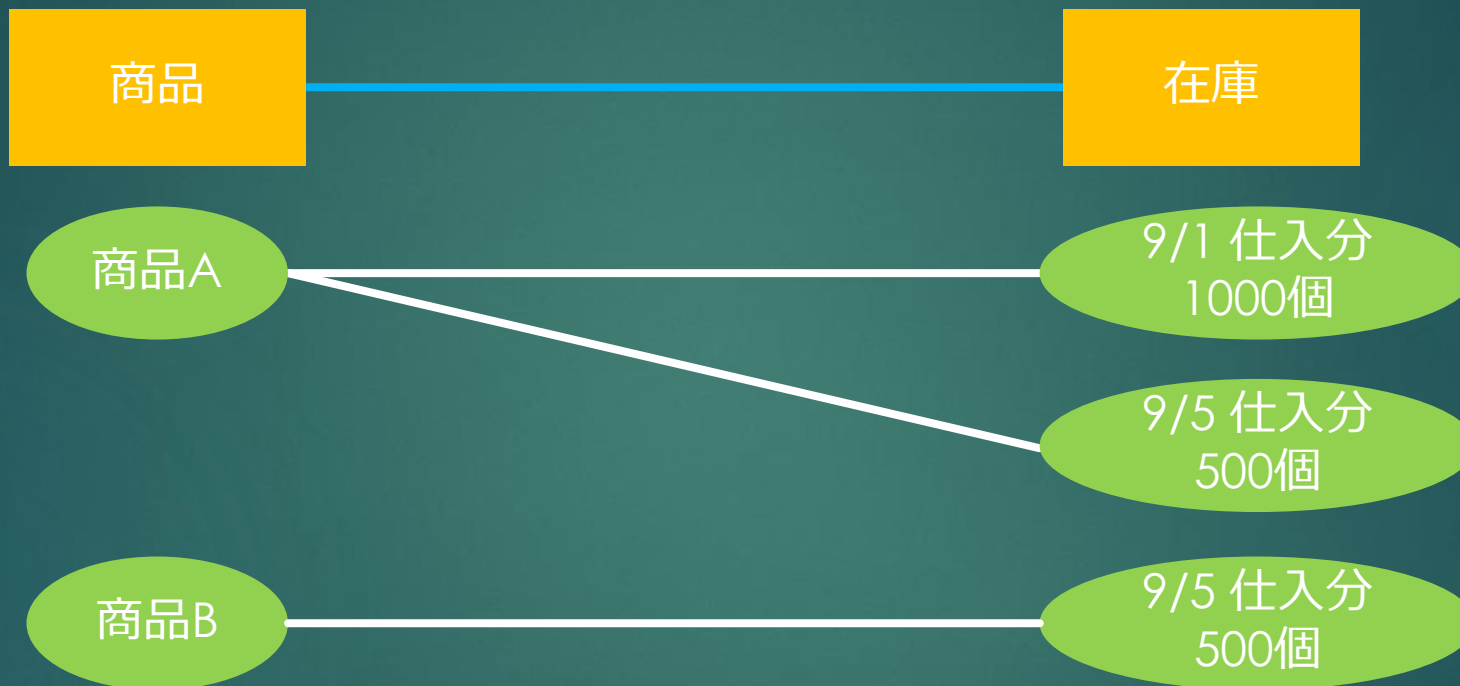
▶ 倉庫ごとの管理



- ▶ 倉庫をいくつか持ち、在庫を倉庫ごとに管理するのであれば、商品と在庫の関係は1:多となる。

多重度とインスタンス

▶ 仕入れごとの管理



- ▶ 在庫事（時系列）に管理するような場合も考えられる。この場合は商品と在庫は1：多に対応する。

論理設計

- ▶ **論理設計**では、概念設計の結果やシステムで用いる画面、帳票イメージをもとにデータの整理・追加を行い、設計内容を詳細化する。また、設計内容を論理モデルで表す。
- ▶ **論理モデル**は、データをどのように管理するかを表すもので、関係モデルが広く用いられている。**関係モデル**は、関係データベースを前提としたモデルで、**データを2次元の表で管理**する。
- ▶ なお、論理モデルには、関係モデル以外にも**階層モデル**や**ネットワークモデル**があるが、現在ほとんど用いられていない。

論理設計

▶ 関係モデル



関係モデル
→対象を2次元の表で表す

物理設計

- ▶ **物理設計**では、最終的なシステム構成とデータベース製品を前提として、データベースを実装するための設計を行う。具体的には性能を確保するためのインデックス構成や、表の配置、耐障害性を高めるための多重化構造などを定める。

データベースの3層スキーマ

- ▶ データベースの設計にあたっては、データのスキーマ（枠組み、構造）を定める。スキーマは、次の三つの階層に分けて考える。

外部スキーマ：ユーザやアプリケーションを提供する構造

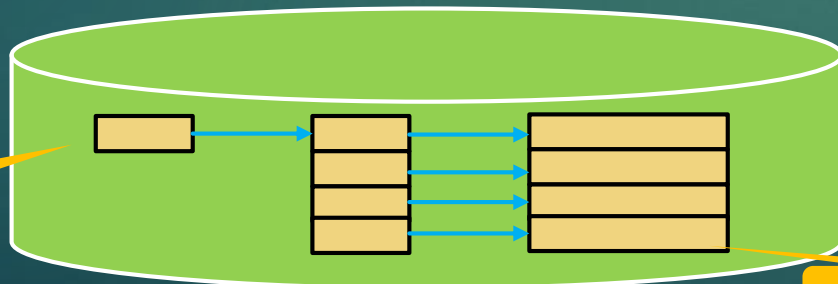
社員番号	氏名	役職	部署
------	----	----	----

概念スキーマ：データの論理的な構造

社員番号	氏名	役職	給与	部署
------	----	----	----	----

内部スキーマ：
データの物理的な
格納構造

インデックス



データファイル

データベースの3層スキーマ

- ▶ 概念設計や論理設計では、データベースの論理的な構造を設計する。これは、スキーマの階層では概念スキーマに相当する。
- ▶ **概念スキーマ**は、データベースの理想的な構造であるが、これをそのままユーザやアプリケーションに提供すると問題が生じる恐れがある。たとえば、取引先に公開する従業員表からは、給与などの個人のデータは省かれるべきである。そこで、概念スキーマから「外部に提供する構造」を改めて作成する。これが、**外部スキーマ**である。
- ▶ **内部スキーマ**は、データファイルの構造やインデックスの構造などデータベースを実装する際の物理的な格納構造である。
- ▶ **3層スキーマ**の形をとることで、データベースは変更に対して強くなる。例えばアプリケーションやハードウェアに変更が生じた場合でも、外部スキーマや内部スキーマの変更で対応できるため、影響が概念スキーマに及びにくくなる。

2. 関係データベースの基礎

候補キーと主キー

- ▶ 関係データベースは、データを2次元の表で表す。表を構成する列を**“列”**または**“属性”**、行を**“行”**または**“タプル”**とよぶ。
- ▶ 関係表には「行を一意（ユニーク）に識別するための列の集まり（1列だけでも良い）」が存在する。そのような列の集まりのうち、必要最小限の列の組み合わせを**候補キー**とよぶ。
- ▶ 候補キーのうち、**意味的に最もふさわしいもの**を選んで**主キー**とする。

候補キーと主キー

The diagram illustrates a database table with four columns: 商品番号 (Product Number), 商品名 (Product Name), 単価 (Unit Price), and 数量 (Quantity). The first two columns are highlighted with red dashed boxes and labeled as '候補キー1' (Candidate Key 1) and '候補キー2' (Candidate Key 2) respectively, with orange arrows pointing to them. A blue arrow labeled '属性名' (Attribute Name) points to the header row. Four blue arrows labeled '行、タプル' (Row, Tuple) point to the four data rows. Four blue arrows labeled '列、属性' (Column, Attribute) point to the four columns. The table data is as follows:

商品番号	商品名	単価	数量
A001	テレビA	50,000	6
A003	テレビB	100,000	4
B002	ビデオカメラ	120,000	5
B004	ビデオデッキ	40,000	8

主キー制約

- ▶ 主キーに選ばれた列には、次の制約（**主キー制約**）が課せられる。

一意性制約	主キーは必ず一意（ユニーク）であり、主キーが重複する（同じ値となる）行は存在しない
非ナル制約	主キーに含まれる列は、空値（ナル値、NULL）をとってはならない

主キー制約

この組み合わせが主キー（複合キー）

受注明細表

<u>伝票番号</u>	<u>商品番号</u>	受注数量	販売単価	...
0001	TV29	5	60,000	...
0001	PC08	10	200,000	...
0001	RD11	2	8,000	...
0002	TV29	3	60,000	...



一意性制約に違反するので追加できない

0001	RD11	...
------	------	-----



非ナル制約に違反するので追加できない

0001	NULL	...
------	------	-----

外部キー

- ある関係表Aの主キーを別の関係表Bが持つような場合、関係表Bの列を**外部キー(foreign key)**とよぶ。外部キーは「他の表を参照する」ための列で、表の結合などに用いられる。

商品表

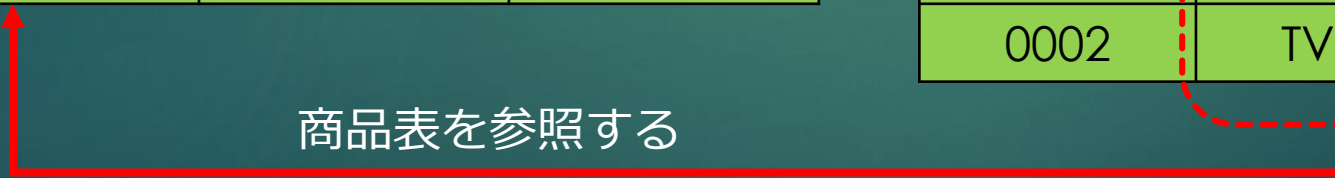
<u>商品番号</u>	商品名	...
TV29	テレビ	...
PC08	パソコン	...
RD11	ラジオ	...

受注明細表

伝票番号	商品番号	...
0001	TV29	...
0001	PC08	...
0001	RD11	...
0002	TV29	...

外部キー

商品表を参照する



参照制約

- ▶ 外部キーにも次のような制約をうける。

外部キーの実現値が参照先（主キー側）の表に必ず存在すること

- ▶ **参照制約**は、**外部キーによる参照を保証する**ための制約であり、**外部キー制約**とも呼ばれる。先の例では受注明細表に登場する商品番号は必ず参照先の商品表に登録されたものでなくてはならない。

参照制約に違反する操作

- ▶ 外部キーで参照関係を結んだ表に対する次の操作は、参照制約に違反する。

参照先（主キー側の表）	（参照先を失う）行の削除、更新
参照元（外部キー側の表）	（参照先のない）行の追加、更新

商品表

<u>商品番号</u>	商品名	...
TV29	テレビ	...
PC08	パソコン	...

受注明細表

伝票番号	商品番号	...
0001	TV29	...
0002	TV29	...

- ▶ 商品表からTV29の行を削除すると受注明細表の参照先が失われることになる。また、受注明細表にTV40をもつ行を追加しようとしても、商品表にTV40がない。これらの操作は、参照関係を保証する参照制約に違反する。

参照制約を守る仕組み

- ▶ 関係データベースは、参照制約を守るための仕組みをもつ。最も基本的な仕組みは、参照制約に違反する操作を「拒否する」ことである。
- ▶ 拒否以外にも「**カスケード(CASCADE)**」とよばれる仕組みがある。
- ▶ カスケードを削除オプションに選んだ場合、受注表から0001の行を削除すると同時に、対応する受注明細表の0001行を削除することで参照制約を守っている。

受注表

伝票番号	受注日	...
0001	20XX/XX/XX	...
0002	20XX/XX/XX	...

0001行の削除と同時に

受注明細表

伝票番号	商品番号	...
0001	TV29	...
0001	PC08	...
0002	TV29	...

伝票番号0001に対応する行をすべて削除

関係データベースの演算

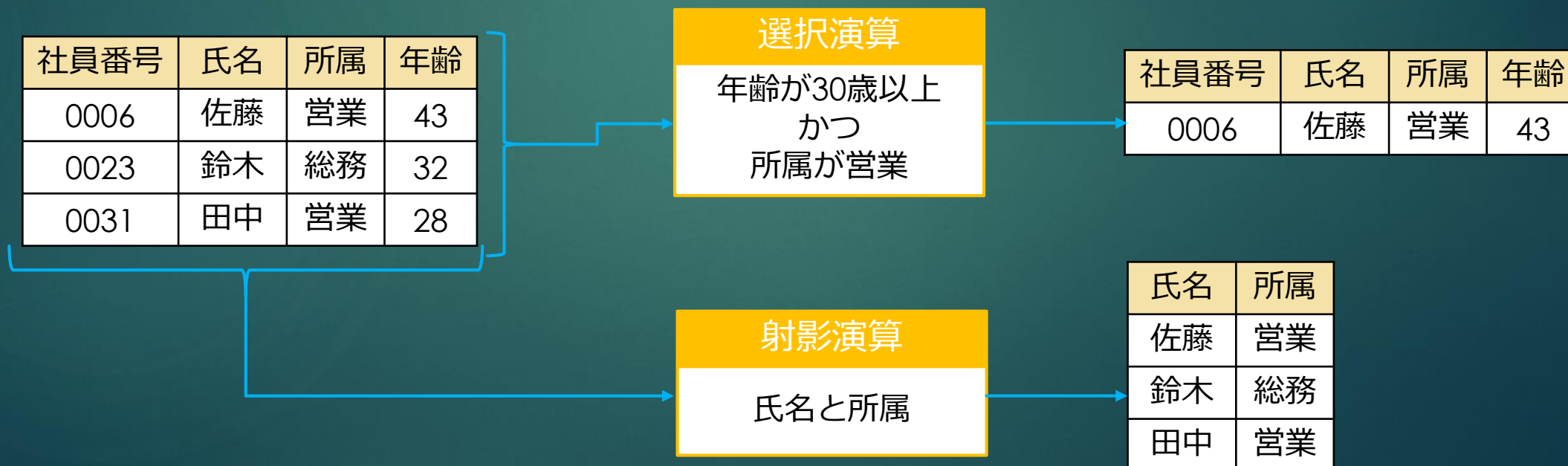
- ▶ 関係データベースのデータ操作は、集合演算及び関係演算を用いて行われる。
- ▶ **集合演算**には、**和**、**差**、**積**、**直積**がある。このうち、和、差、積では演算の対象となる表の形式が同一（属性数や属性の方が等しい）である必要がある。

和	二つの表のいずれかに含まれる行を取り出す
差	一方には含まれ、他方には含まれない行を取り出す
積	双方に含まれる行を取り出す
直積	二つの表に含まれる行の「すべての組み合わせ」を得る

関係データベースの演算

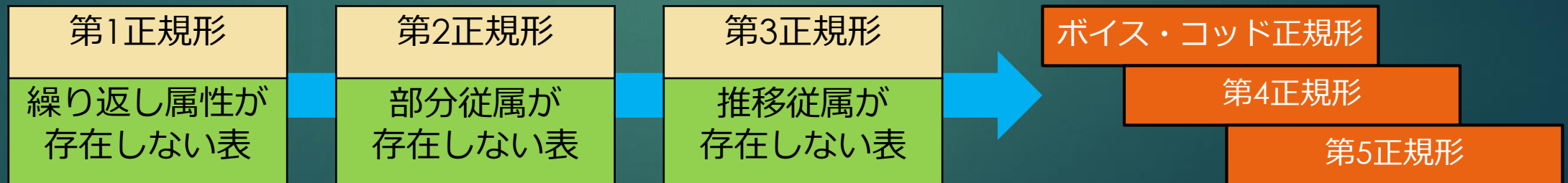
- ▶ 関係演算には、**選択**、**射影**、**結合**がある。

選択	表から条件に合致する行を取り出す
射影	表から指定された列を取り出す
結合	複数の表を組み合わせて一つの表を導出する



関係データベースの正規化

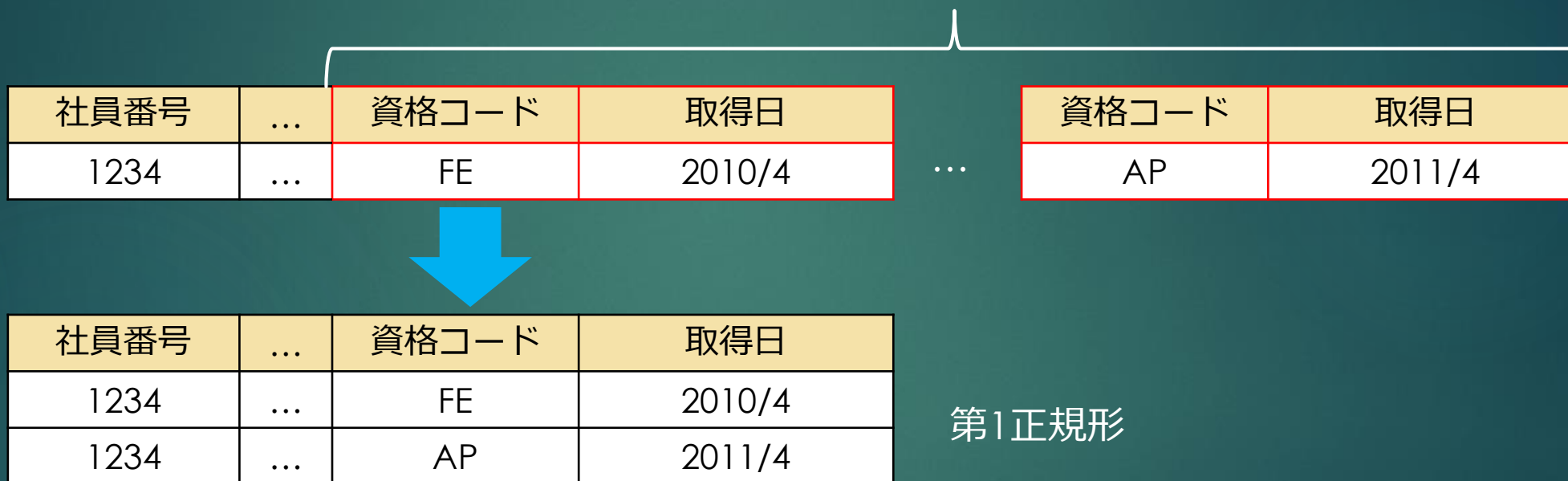
- ▶ 関係データベースにおける**正規化**とは、冗長なデータを排除して**関連の強いデータのみを一つの関係表にまとめ**、表の独立性を高めることである。これによって、矛盾（データの更新時異状）の発生を極力減らし、表の整合性を保つ。
- ▶ 正規系には第1から第5までの種類が大きいものほど正規度は高くなる。なお、一般的なデータベースを扱う場合には、**第3正規化されていれば十分**である。



第1正規形

- ▶ **第1正規形**は、表から**繰返し属性（項目）**を排除した形である。

保有する資格の繰返し



- ▶ 第1正規形を満たさない表は**非正規形**とよばれ、関係データベースでは扱えない

第2正規形

- ▶ **第2正規形**は、候補キーの一部分に従属するような関数従属（**部分関数従属**）が存在しない形である。
- ▶ 関数従属性
Xの値が定まればYの値が定まるという性質。X→Yや「YはXに従属する」などと表す。
- ▶ 第2正規形は、**部分従属を独立させるように表を分割**し、第2正規形を得る。

第2正規形

名称は資格コードにのみに従属する

複合キー

取得表

<u>社員番号</u>	<u>資格コード</u>	名称	取得日
1234	FE	情報基礎	2010/4
1234	AP	情報応用	2011/4

第1正規形

取得表

<u>社員番号</u>	<u>資格コード</u>	取得日
1234	FE	2010/4
1234	AP	2011/4

第2正規形

資格表

<u>資格コード</u>	名称
FE	情報基礎
AP	情報応用

第3正規形

- ▶ **第3正規形**は、候補キーから始まる**推移的関数従属**が存在しない形である。なお、「候補キーから始まる推移的関数従属」とは「**非キー同士**の関数従属」と考える。

- ▶ 推移的関数従属性

$X \rightarrow Y \rightarrow Z$ という連鎖的な従属関係

第3正規形

非キー同士の従属関係
団体コード→団体名称
が存在する

資格試験の実施団体

資格表

<u>資格コード</u>	名称	団体コード	団体名称
K1	経理1級	JAXXX	全国経理...
FE	情報基礎	KSYYY	経済産業...

第2正規形

資格表

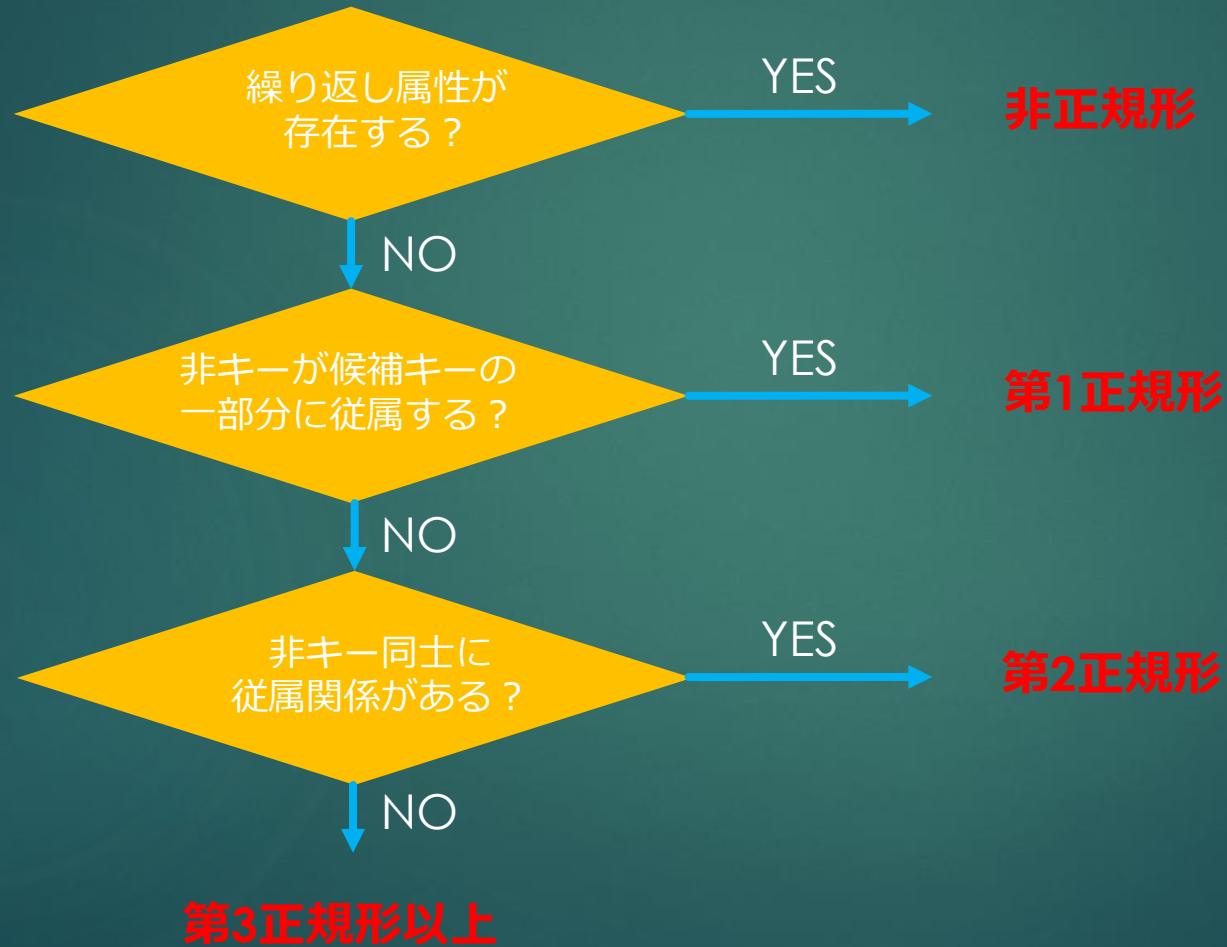
<u>資格コード</u>	名称	団体コード
K1	経理1級	JAXXX
FE	情報基礎	KSYYY

第3正規形

団体表

<u>団体コード</u>	団体名称
JAXXX	全国経理...
KSYYY	経済産業...

正規度の判断手順



3. SQL

SELECT文の基本文法

- ▶ **SELECT文**はデータベースからデータを取り出す操作で、「**問い合わせ機能（クエリ）**」とよぶ。基本的な構文は次のようになる。

SELECT **DISTINCT** 列名のリスト

FROM 表明のリスト

※省略可

WHERE 行の選択条件

GROUP BY グループ化に用いる列名のリスト

HAVING グループ化の条件

ORDER BY 整列に用いる列名、昇降順指定のリスト

DISTINCT句

- ▶ **DISTINCT句**を指定すると、SELECT文で「内容が完全に一致する行」が抽出されたとき、**重複している行を1行にまとめる**。

社員表

社員番号	氏名	...
001	田中一郎	...
002	田中一郎	...
003	山田二郎	...

SELECT 氏名
FROM 社員表

氏名
田中一郎
田中一郎
山田二郎

SELECT **DISTINCT** 氏名
FROM 社員表

氏名
田中一郎
山田二郎

WHERE句

- ▶ **WHERE句**では、**行の選択条件や表の結合条件を指定**する。条件指定では、複数の条件を**AND**や**OR**を用いて結合することができる。条件式には通常の統合や不等号の他にも、次のものを用いることができる。

- ▶ 条件式の指定

構文	意味
列 BETWEEN 値1 AND 値2	列値が値1以上2以下であれば真
列 NOT BETWEEN 値1 AND 値2	列値が値1未満または値2を超えていれば真
列 IN (値リスト)	列値が値リストいずれかに一致すれば真 (例) 列 IN (1, 2, 3)
列 NOT IN (値リスト)	列値が値リストのいずれにも一致しなければ真
列 LIKE パターン文字列	列値がパターン文字列に一致すれば真
列 NOT LIKE パターン文字列	列値がパターン文字列に一致しなければ真
列 IS NULL	列値がNULLであれば真
列 IS NOT NULL	列値がNULLでなければ真

WHERE句

▶ パターン文字列

ワイルドカード（任意の文字を表す特殊記号）を含んだ文字列のこと

▶ ワイルドカード

—	任意の1文字 (例) 'A_': Aで始まる2文字
%	任意の0文字以上の文字列 (例) 'A%': Aから始まる文字列

グループ化

- ▶ **グループ化**とは、表に含まれる行をいくつかのグループに分けることである。グループ化を行うことで、グループごとの合計や平均などの「行の集約値」を問い合わせることができる。
- ▶ どのようにグループを分けるかは、**GROUP BY**句を用いて
GROUP BY 列1, 列2, ...
- ▶ と指定する。このとき、GROUP BY句に指定した列の値がすべて同じ行は同一のグループとして扱われる。
- ▶ グループ化する際は、出力する列は「グループで一つの値」をとるものにかぎる。すなわち、**グループ化に用いた列と集合関数を用いたもの以外は指定できない。**

グループ化

SELECT 担当社員番号, 氏名, 商品コード, SUM (数量 * 単価) AS 売上合計
FROM 売上
GROUP BY 担当社員番号, 氏名, 商品コード

伝票 番号	担当 社員番号	氏名	商品 コード	数量	単価
001	1021	田中	T01	300	10
002	1021	田中	T01	300	20
002	1021	田中	T02	600	20
003	1043	山田	T01	300	40
004	1043	山田	T03	500	50
005	1043	山田	T03	500	15

担当 社員番号	氏名	商品 コード	売上 合計
1021	田中	T01	9,000
1021	田中	T02	12,000
1043	山田	T01	12,000
1043	山田	T03	32,500

各グループごとに出力

集合関数で集計

グループ化

集合関数

- ▶ 集合関数は**グループごとの値を抽出・集計**する関数である。
GROUP BY句がない場合は、表全体を1グループとして扱い、それぞれの値を求める。

- ▶ 集合関数

COUNT (式または*)	行数を求める
SUM (式)	合計値を求める
AVG (式)	平均値を求める
MAX (式)	最大値を求める
MIN (式)	最小値を求める

HAVING句

- ▶ **HAVING句**は、グループに対して選択条件を課す場合に用いる。HAVING句で条件を指定すると、**条件を満たすグループのみ**が問い合わせの対象となる。
- ▶ 条件式は集合関数を用いることができるが、条件式に指定できる列はグループ化を用いた列のみとなる。

HAVING句

社員番号	氏名	給与	部門コード
001	田中	200,000	K01
002	山田	240,000	K01
003	川田	220,000	K02
004	山中	260,000	K02

```
SELECT 部門コード, AVG (給与)
FROM 社員
GROUP BY 部門コード
HAVING AVG (給与) > 220000
```

GROUP BY句
でグループ化

部門コード	AVG (給与)
K01	220,000
K02	240,000

HAVING句で
グループを選択

部門コード	AVG (給与)
K02	240,000

整列

- ▶ SELECT文が返す結果の順序は保証されていない。**特定の順序で整列する**場合には、**ORDER BY**句を用いて、

ORDER BY 列名1 整列順, 列名2 整列順, ...

と指定する。

- ▶ 整列順には昇順(**ASC**)または降順(**DESC**)を指定する。これを省略すると、ASCが指定されたものとみなされる。
- ▶ 複数の列を指定すると、左から順に第一整列キー、第二整列キー、... と見なされて整列される。

副問合わせ(IN / NOT IN)

- ▶ SELECT文の結果を別のSELECT文の選択条件に使用することができる。これを副問い合わせとよぶ。
- ▶ **IN / NOT IN**を用いた副問い合わせは、副問い合わせ部分を実行して値リストに展開してから主問い合わせを実行するという順序になる。
- ▶ IN / NOT IN以外にも、次の句を使うことがある（参考）。
 - ・売上 > ANY（副問い合わせ）
売り上げが副問い合わせの結果のいずれかよりも大きい
 - ・売上 > ALL（副問い合わせ）
売り上げが副問い合わせのどれよりも大きい

副問合わせ(IN / NOT IN)


社員

社員番号	氏名	部門コード
001	田中	JI
002	山田	KE
003	山中	SO
004	吉川	JI

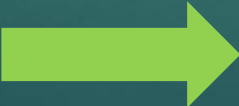
部門

部門コード	部門名	フロア
JI	人事	1F
KE	経理	1F
SO	総務	2F

```
SELECT *  
FROM 社員  
WHERE 社員.部門コード IN( SELECT 部門コード  
FROM 部門  
WHERE フロア = '1F' )
```



```
SELECT *  
FROM 社員  
WHERE 社員.部門コード IN  
(JI, KE)
```



社員番号	氏名	部門コード
001	田中	JI
002	山田	KE
004	吉川	JI

副問い合わせ(EXISTS / NOT EXISTS)

- ▶ **EXISTS / NOT EXISTS**は、**副問い合わせの結果が空かどうか**で真偽を判断する。副問い合わせそのものが一つの条件式になっていると考えればよい。EXISTSは、副問い合わせの結果が1行以上でも存在すれば成立、そうでなければ不成立とみなす。NOT EXISTSはその逆で、副問い合わせの結果が空(0行)であれば、真（成立）、そうでなければ偽（不成立）とみなす。副問い合わせそのものを条件とみなすため、主問い合わせに用いる表の各行に対して、副問い合わせを実行して条件の真偽を判定する。

副問い合わせ(EXISTS / NOT EXISTS)

SELECT * FROM 社員

列名は何でもよい

WHERE EXISTS (SELECT * FROM 受注

WHERE 担当 = 社員.社員番号)

列名は指定しない

主問い合わせ中で指定された表を参照

1行目は...
'5'が受注側にはない → 偽

社員番号	氏名
5	田中
16	山田
46	川田

受注No	受注日	得意先	担当
1	10/1	C209	16
2	10/1	A021	46
3	10/2	R107	16

2行目は...
'16'が受注側にある → 真

社員番号	氏名
16	山田
46	川田

行の挿入 / 削除 / 更新

- ▶ 行の挿入 / 削除 / 更新には、それぞれ**INSERT文**、**DELETE文**、**UPDATE文**を用いる。

INSERT INTO 挿入を行う表名 (列名のリスト)
VALUES (値リスト)

DELETE FROM 削除を行う表名
WHERE 削除する行の条件

UPDATE 更新を行う表名
SET 列名 = 値式
WHERE 更新する行の条件

行の挿入 / 削除 / 更新

- ▶ 社員番号“108”の“田中”という行を社員表に挿入する。部門IDは未配属のためNULLとする。

```
INSERT INTO 社員 (社員番号, 氏名, 部門ID)
VALUES ('108', '田中', NULL)
```

- ▶ 社員表から社員番号“046”の社員を削除する。

```
DELETE FROM 社員
WHERE 社員番号 = '046'
```

- ▶ 社員番号“046”の社員の給与を、現状の1.1倍に更新する。

```
UPDATE 社員
SET 給与 = 給与 * 1.1
WHERE 社員番号 = '046'
```


テーブル定義

- ▶ 表（テーブル）を定義する場合、**CREATE TABLE文**を用いる。CREATE TABLE では、表名と表に含まれる列の名前、データ型、列制約を定義する（列定義）。また、表自体に制約を設ける場合は、表制約を定義することも可能である（表制約定義）。



列制約定義

- ▶ 列制約定義は各列の値に対して制約を設け、整合性を損なう操作を防止する。

書式	意味
PRIMARY KEY	主キー制約。列単体で主キーとなる
NOT NULL	非ナル制約。空値を禁止する
UNIQUE	一意性制約。重複値を禁止する
CHECK 条件式	検査制約。条件式を満たさない列値を禁止する
REFERENCE 表名 (列名)	参照制約。列値が指定した表を参照する

表制約定義

- ▶ 各種の制約を、列の定義を終えた後に**表制約**として設定することもできる。特に**複数列またがる制約は、列ごとに設定できない**ので、**表制約定義**で設定しなければならない。

- ▶ 受注番号と商品コードの組合せを主キーとする。

PRIMARY KEY (受注番号, 商品コード)

- ▶ 受通番号と商品コードの組合せが、受注明細表を参照する外部キーである。

FOREIGN KEY (受注番号, 商品コード)

REFERENCES 受注明細 (受注番号, 商品コード)

権限定義

- ▶ 表に対するアクセス権限を与える場合には**GRANT文**を、アクセス権限を剥奪する場合には**REVOKE文**を用いる。権限は、

SELECT（問い合わせ）、UPDATE（更新）

INSERT（追加）、DELETE（削除）

のほか、**ALL PRIVILEGES**（全権限）を指定することができる。

権限定義

- ▶ 利用者SUZUKI, TANAKAに、社員表への問い合わせ (SELECT), 更新(UPDATE)権限を与える。

GRANT SELECT, UPDATE ON 社員 TO SUZUKI,
TANAKA

- ▶ 利用者SATOUから、社員表に対するすべての権限を剥奪する。

REVOKE ALL PRIVILEGES ON 社員 FROM SATOU

ビュー定義

- ▶ **ビュー**表を作成する場合、**CREATE VIEW**文が用いられる。CREATE VIEW文の基本的な構文は、次の通りである。

CREATE VIEW ビュー表名 AS 問合せ指定

「問合せ指定」では、SELECT文が指定される。その結果導出された**導出表**がビュー表として定義されることになる。

社員表から性別が男(M)である行を抽出し、ビュー男性表とする。

```
CREATE VIEW 男性 AS SELECT * FROM 社員  
WHERE 性別 = 'M'
```

ビュー(VIEW)

実表から問い合わせを用いて導出する仮想的な表。導出表ともよぶ

更新可能なビュー

- ▶ ビューに対する更新が、そのもととなった実装に反映できるとき、そのビューを**更新可能なビュー**とよぶ。ビューが更新可能であるためには、ビューのデータから実装データを特定できる必要がある。そのためには、ビューは次の条件を満たさなければならない。

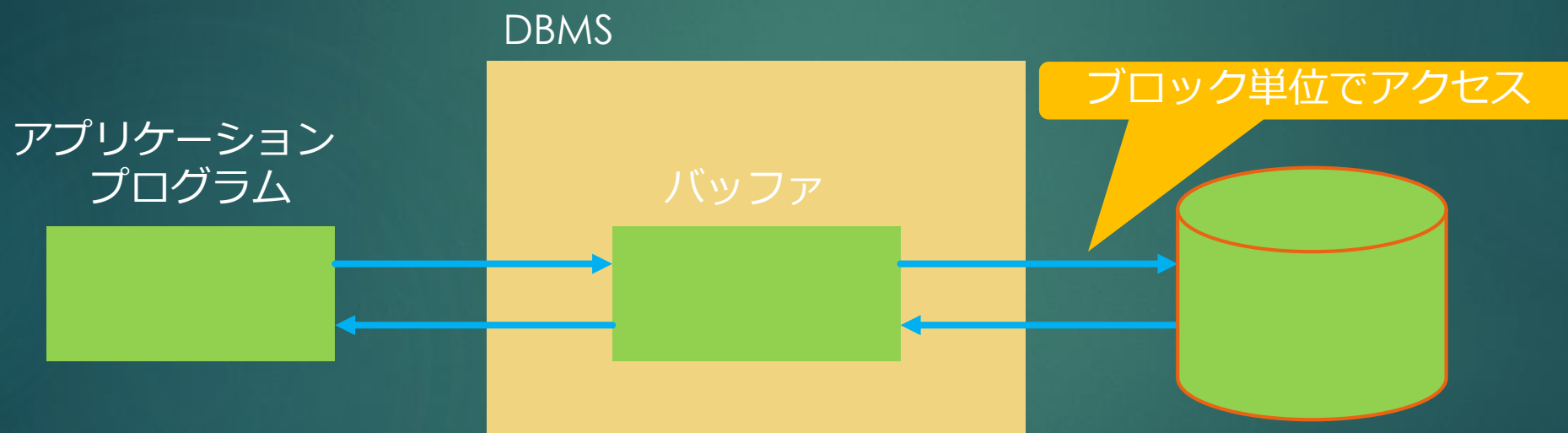
- ・ DISTINCTを使用しないこと
- ・ 問合せ指定中の列は、実表の列を直接参照し、定数・算術演算子・関数を含まないこと
- ・ FROM句に表は一つしか指定しないこと（例外あり）
- ・ WHERE句に副問い合わせを含まないこと
- ・ GROUP BY句やHAVING句を含まないこと

データベースアクセスとインデックス

データベースアクセスの仕組み

- ▶ データベースのデータは補助記憶装置上に構築されているが、データアクセスごとに補助記憶にアクセスするのは効率がよいとはいえない。一般的には主記憶上にバッファを用意し、補助記憶装置とはブロック（ページ）単位でデータを入れ替え、アプリケーションはバッファ上のデータをアクセスする。このような仕組みを用いることで、補助記憶装置への物理アクセスを減少することができる。

データベースアクセスの仕組み



DBMS(Database Management System)

データベースを管理し、データベースへのアクセス機能を提供するソフトウェア

データベースへの反映

- ▶ データベースへの更新は、バッファ上のブロックに対して行われ、補助記憶装置には反映されない。補助記憶装置への反映は、次のいずれかのタイミングで行われる。
 - ・ ブロックがバッファから追い出される（補助記憶装置に戻される）
 - ・ チェックポイントを取得する

▶ チェックポイント

バッファ上のデータを補助記憶装置上のデータを同期させるイベントである。チェックポイントでは、バッファ上の全ブロックが補助記憶装置に書き戻されるため、**チェックポイントを取得した時点で両者の内容は完全に一致**する。

索引（インデックス）の利用

- ▶ **索引**は索引キーと行データの格納位置を対応付けたもので、これを用いた**索引検索(index scan)**では、索引から目的の行データが格納された位置を得るため、**全件検索(full scan)**に比べ物理アクセス回数を減らすことができる。
- ▶ 多くのDBMSでは、主キーに指定した列に自動的に索引がつけられる。また、これ以外の列にも任意の列に索引を指定できるので、検索によく利用する列には索引付けを検討する。

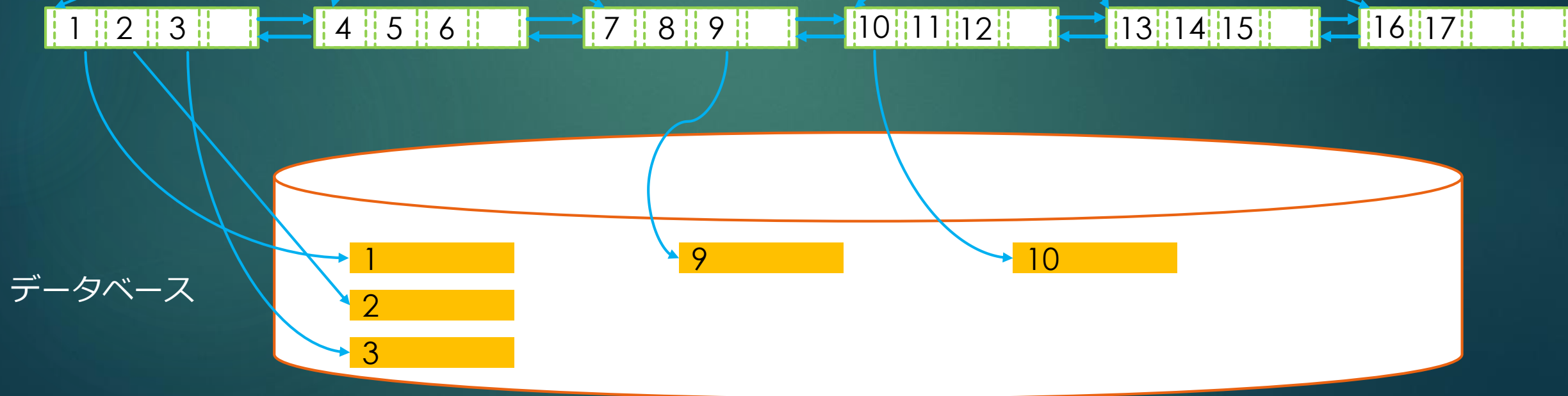
索引（インデックス）の利用

B'木索引

根（ルート）

節（ノード）

葉（リーフ）



索引利用上の注意

- ▶ 索引を利用することで、データベースに対するすべての操作が速くなるわけではない。

- ▶ **索引にも更新が行われること**

テーブルの行データが挿入・更新・削除すると、索引も更新される。このため、**むやみに索引を付与すると、行データの挿入・更新・削除時の応答性能が悪化する**恐れがある。

- ▶ **効果の低い利用は行わない**

索引は、膨大なデータの中からごく少数の行を検索する場合に最高の効果を発揮する。逆に、**データのほとんどが検索条件に当てはまるような場合には効果がない**。

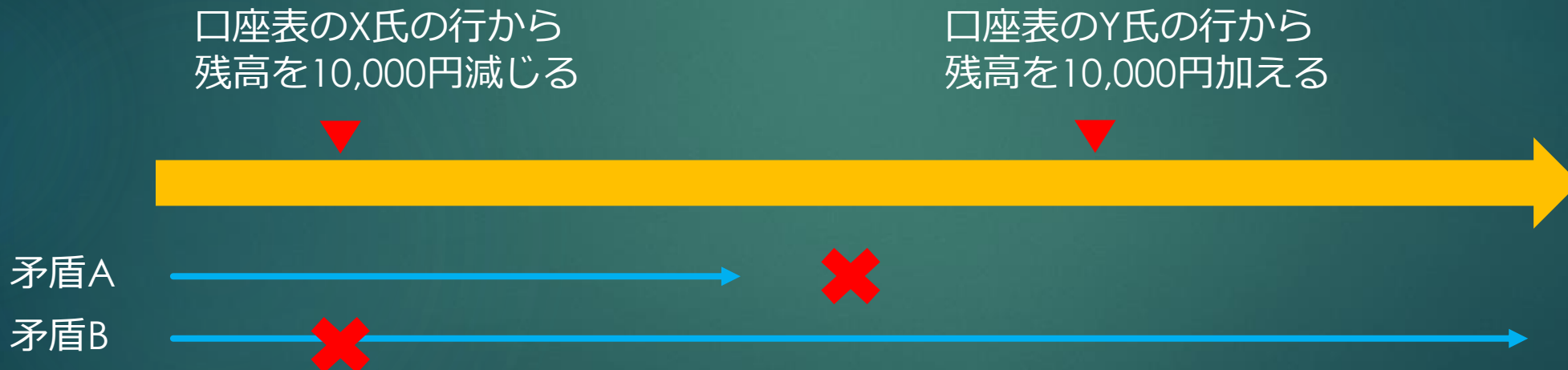
- ▶ **索引が利用されない検索もある**

検索条件に用いる列に演算が行われていたり、**NOTを用いた検索では索引は利用されない**。

5. トランザクション管理

トランザクションとは

- ▶ トランザクション(transaction)は、預金口座への入出金や電車の座席予約といった一連の不可分な処理単位のこと、データベースの参照や更新がともなうことが一般的である。



X氏の口座から金額を減じた時点でトランザクションが異常終了すると、Y氏への振り込みが実行されないため矛盾が生じる。同じように、何らかの不具合によりX氏の残高が減じられないままY氏の残高だけ増えてしまった場合でも、残高に矛盾が生じる。このような矛盾を防ぐために、トランザクションを正しく実行し、正しく終了させる管理が不可欠となる。

ACID特性

- ▶ トランザクションが備えるべき四つの特性を、ACID特性という。ACID特性を満たしている限り、トランザクションはデータベースを矛盾なく更新することができる。

特性	意味
Atomicity (原子性)	トランザクションは「すべて実行される」か「全く実行されない」かのいずれかのじょうたいである
Consistency (一貫性)	トランザクションは、データベースの内容を矛盾させない
Isolation (独立性、隔離性)	トランザクションは他のトランザクションの影響を受けない
Durability (耐久性、永続性)	正常終了したトランザクションの実行結果が失われることはない

コミットメント制御

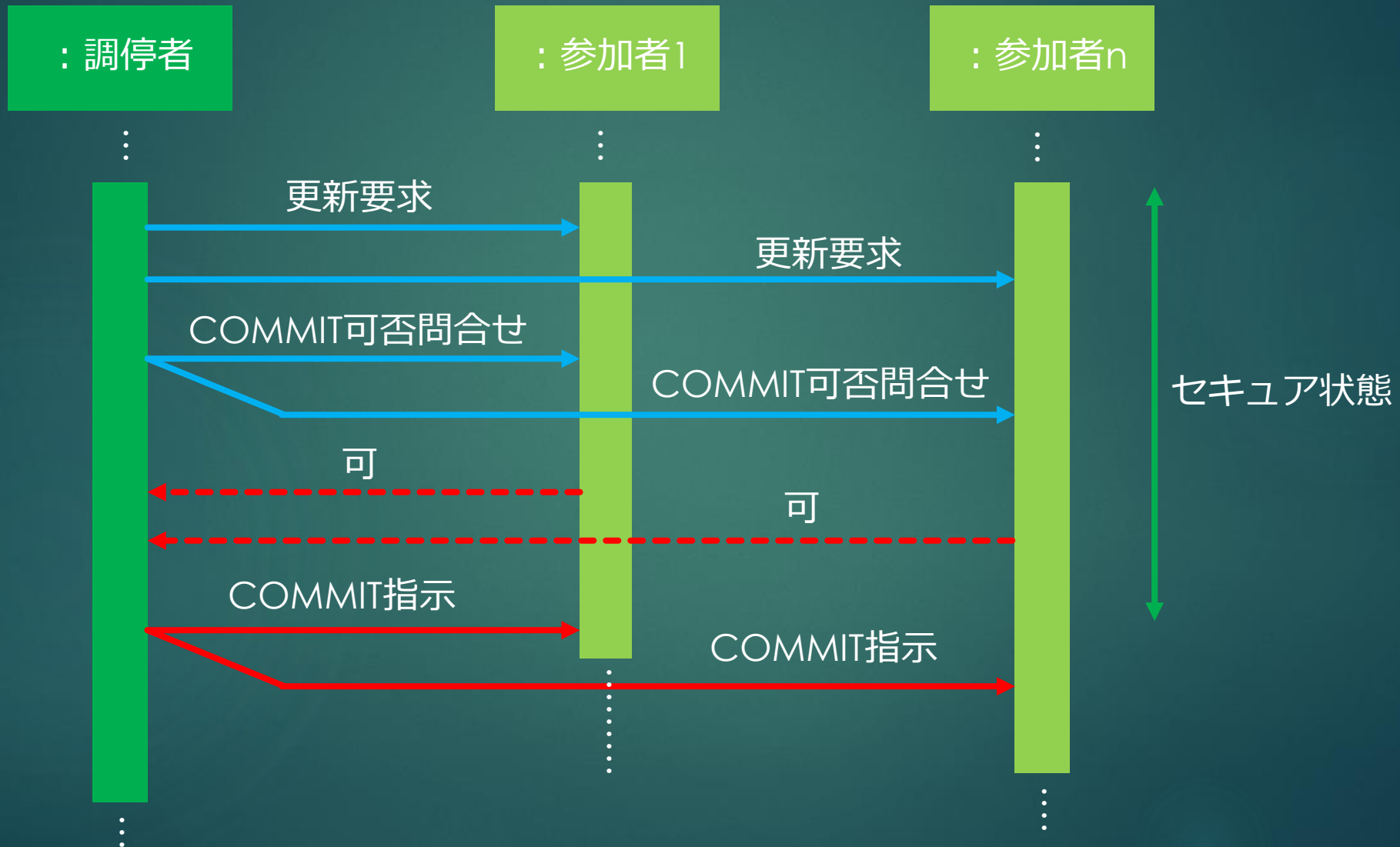
- ▶ コミットメント制御は、トランザクションの原始性を保つための仕組みである。トランザクションが正しく実行された場合は、コミット(COMMIT)を行い更新内容をすべて確定する。一方、正しく完了しなかった場合はロールバック(ROLLBACK)を行い、更新内容を破棄する。



2相コミットメント

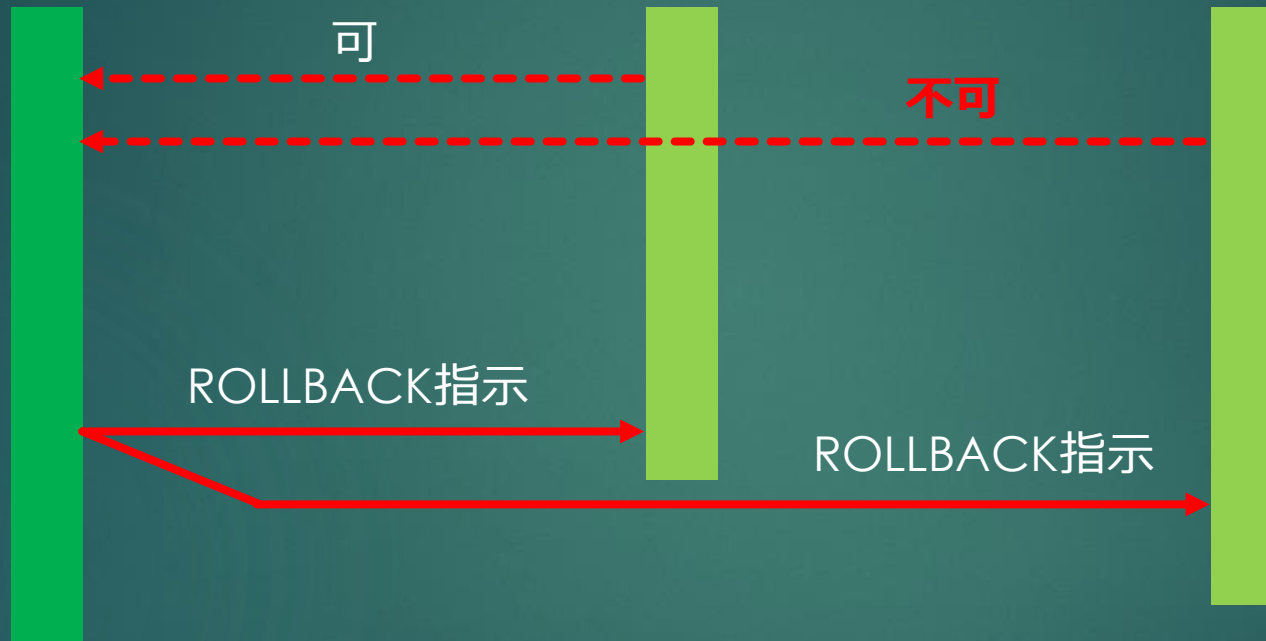
- ▶ データベースが複数のサイトに分散しているような環境では、複数サイトにまたがった制御は行われなければならない、より複雑なコミットメント制御が必要となる。その代表が**2相コミットメント**である。
- ▶ 2相コミットメントは、コミットなどの指示を出す主サイト（**調停者**：Coordinator）と、指示に従ってコミットなどを行う従サイト（**参加者**：Participant）から構成されている。主サイトは、各従サイトをコミットもロールバックも可能な中間状態に設定（**セキユア**指示）し、**すべての従サイトがコミット可能であればコミットを指示**し、そうでなければ、ロールバックを指示する。

2相コミットメント



2相コミットメント

- ▶ 参加者が1つでも不可を返した場合はROLLBACKが指示される



- ▶ 2相コミットメントでは、COMMIT可否を問い合わせしてからCOMMIT指示を出す前に調停者に障害が発生した場合、参加者は調停者の回復を待ち続ける。これを避ける3相コミットメント制御もある。

障害回復制御

- ▶ DBMSは、各種障害が発生しても、データに矛盾を発生させずに障害直前の状態に回復することが求められる。そのため**障害回復制御**が必要となる。障害回復は、次の情報を用いて行う。

- ・バックアップデータ
- ・ログファイル（ジャーナルファイル）

▶ ログファイル

実行中のトランザクションが行った操作の記録を**ログ**といい、これを格納したファイルを**ログファイル**という。ログファイルを構成するレコードには、トランザクションID、操作、データ情報などが含まれ、各トランザクションのログレコードは次のような順序で記録される。

開始情報→更新前情報→更新後情報→コミット情報→終了情報

障害回復制御

▶ ロールバック

ロールバックは、ログファイルの更新前情報を用いて、**トランザクションによる更新を取り消す**処理である。コミット前に中断したトランザクションの更新を取り消す場合に行われる。

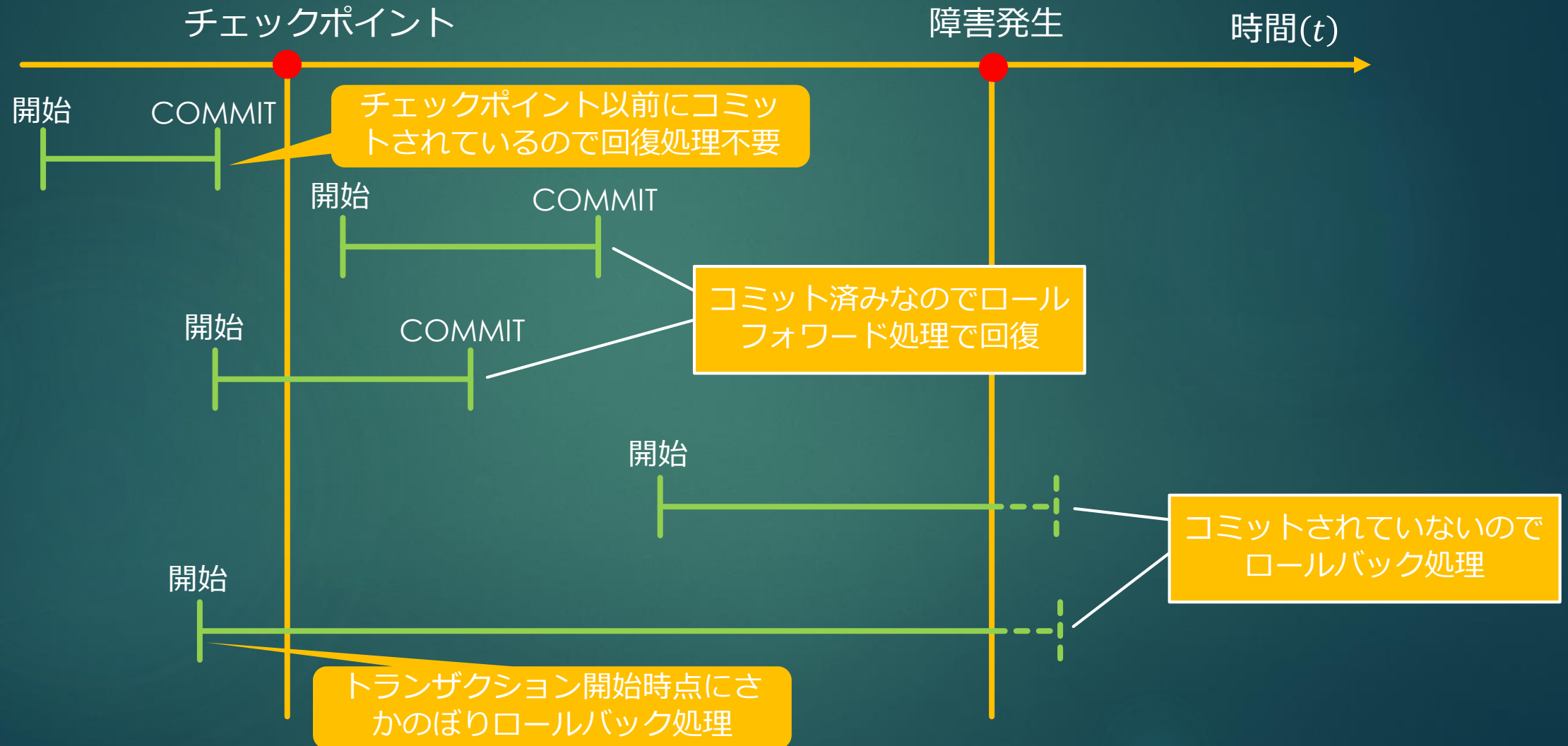
▶ ロールフォワード

ロールフォワードは、ログファイルの更新後情報を用いて、**トランザクションによる更新をデータベースに反映する**処理である。コミットしたのにもかかわらず、データベースに記録されなかった場合の回復に用いる。

システム障害時の回復処理

- ▶ システムダウンや電源断といったシステム障害が発生した場合、再起動したシステムは「バッファの情報は失われているがデータベースには一部の処理結果が反映された状態」となる。そこで、データベースを最新のチェックポイントまで戻したうえで、トランザクションのコミット状況によって次の回復処理を行う。
 - ・ チェックポイント以前にコミット済み→回復の必要はない
 - ・ チェックポイント以降にコミット済み→ロールフォワードでデータベースに反映させる
 - ・ チェックポイント以前に開始されコミットしていない→ロールバックで更新を取り消す

システム障害時の回復処理



媒体障害時の回復処理

- ▶ ディスククラッシュなどの媒体障害（物理障害）が発生した場合、記憶媒体を好感してバックアップデータをロード（リストア）した後、トランザクションによる更新をロールフォワードしてデータベースを回復する。

6. 運用とデータベース応用

データベースの運用

- ▶ データベースの運用にあたっては、性能（アクセス効率）や信頼性を高めるためのさまざまな対策が取られている。

- ▶ データベースの再編成

データベースに対してデータの挿入や削除が繰り返された結果、あふれ域にデータが記憶されたり、索引に偏りが生じるなどの原因により、データベースのアクセス効率が低下することがある。そこで、定期的に**データを基本域に書き戻し索引を再設定する**作業を行う。このようなデータベースのメンテナンスを**再編成**とよぶ。

あふれ域

基本領域に収まりきれないデータを記録するための領域。索引による検索ができないため、あふれ域に多くのデータが記録されると、アクセス効率が低下する。

データの重複保持

- ▶ 応答性能の改善、通信量の削除、信頼性の向上などを目的に、論理的に一つのデータを、物理的には重複させて複数のサイトに分散させることがある。このとき、**主サイトの更新に伴って別サイトのデータも自動的に更新するレプリケーション**という仕組みが必要になる。

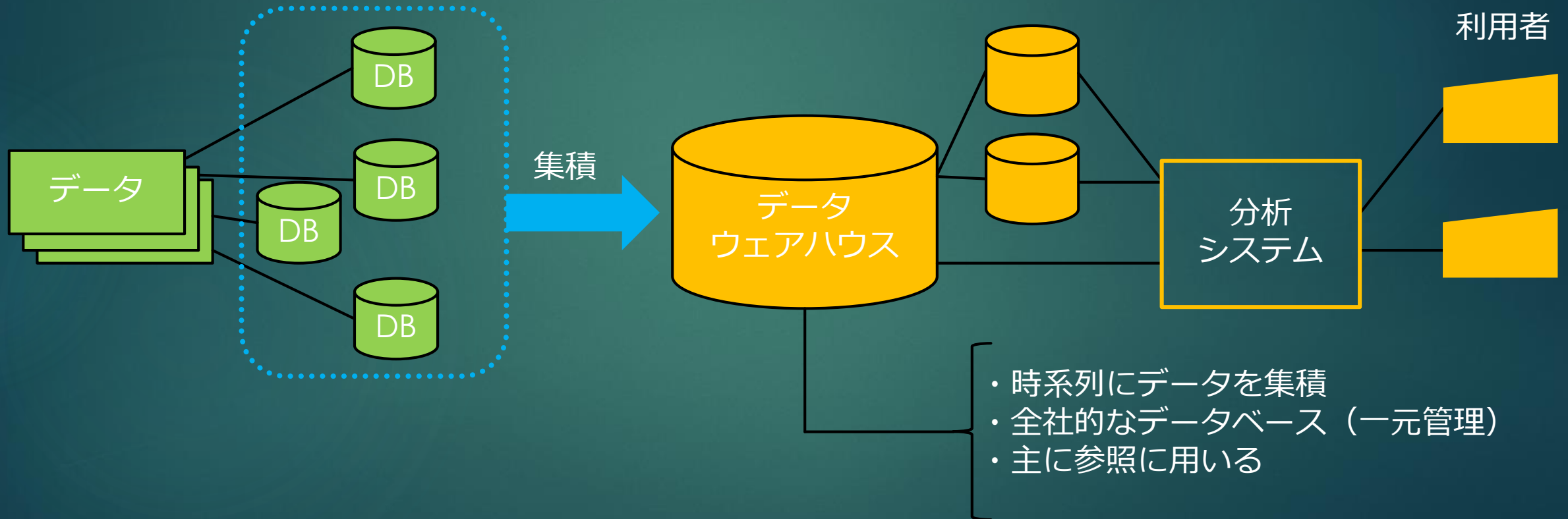
データウェアハウス

- ▶ 日々発生する業務データを一元的に記録するデータベースをデータウェアハウスとよぶ。**データウェアハウス**はいわばデータの倉庫であり、以下のような特徴をもつ。
 - すべての基幹系システムのデータを統合し、集積する
 - データを時系列に蓄積する
 - 一度蓄積されたデータは、通常更新されない

データウェアハウス

基幹業務

分析業務



データマイニング

- ▶ データウェアハウスに蓄積された膨大なデータの中から、**未知の規則性や事実関係を得る**技法を**データマイニング**という。
- ▶ データマイニングの手法

アソシエーションルール	データの中から相関関係を見つける手法。同時に購入される商品进行分析するバスケット分析などが該当する
クラス分類	クラスを定義しておき、蓄積されたデータを各クラスに分類する手法
クラスタリング	異なる性質から持つデータから、類似した性質をもつクラスタ（データの塊）を見つける技法。クラスタはあらかじめ定義されていない点がクラス分類と異なる。セグメンテーションともいう

データクレンジング

- ▶ データウェアハウスは、多様な業務システムからデータを収集する。そのためデータ属性やコード体系が異なるデータや重複データ、陳腐化したデータなど、様々なデータがそれぞれの形のまま含まれている可能性がある。これらのデータの形式や体形を統一し、不要なデータを取り除くことを**データクレンジング**（データ洗淨）とよぶ。
- ▶ データクレンジングによってデータの品質が高まり、精度の高いデータマイニングを行うことができる。

自然言語処理

- ▶ 自然言語とは、我々が日常用いる日本語や英語を指す。マンマシンインタフェースの分野で、自然言語の重要性はますます高まってきている。自然言語の解析のため、さまざまなテキストデータベースが構築され、利用されている。

- ▶ **コーパス**

コーパスは、自然言語の解析のため、文学作品や会話、各種記事などから**大量の文章を収集し記録した大規模なデータベース**である。

- ▶ **シソーラス**

シソーラスは、同義関係や類似関係を元に単語を分類し、体系化した“語彙辞書”のことをいう。データベース化されたシソーラスは、あいまい検索などに利用される。

マンマシン
インタフェース

人間とコン
ピュータとの
間で情報をやり
とりするイ
ンタフェース

ビッグデータ

- ▶ インターネットの普及に伴い発生するようになった、**膨大なデータの集合**を**ビッグデータ**とよぶ。有益な情報が含まれていると考えられるがあまりにも量が多すぎるため、標準的なリレーショナルデータベースやツールでは対応が困難である。クラウドコンピューティングや分散処理技術などを組み合わせた解析方法が考えられている。