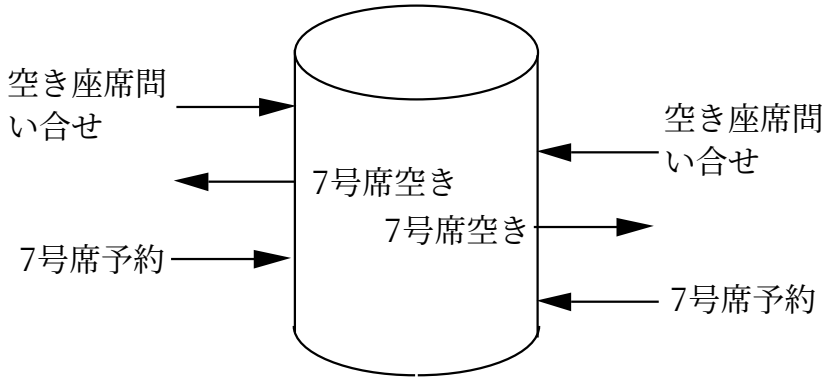
情報科学 II 講義資料-2　データベース 2　　大島正毅

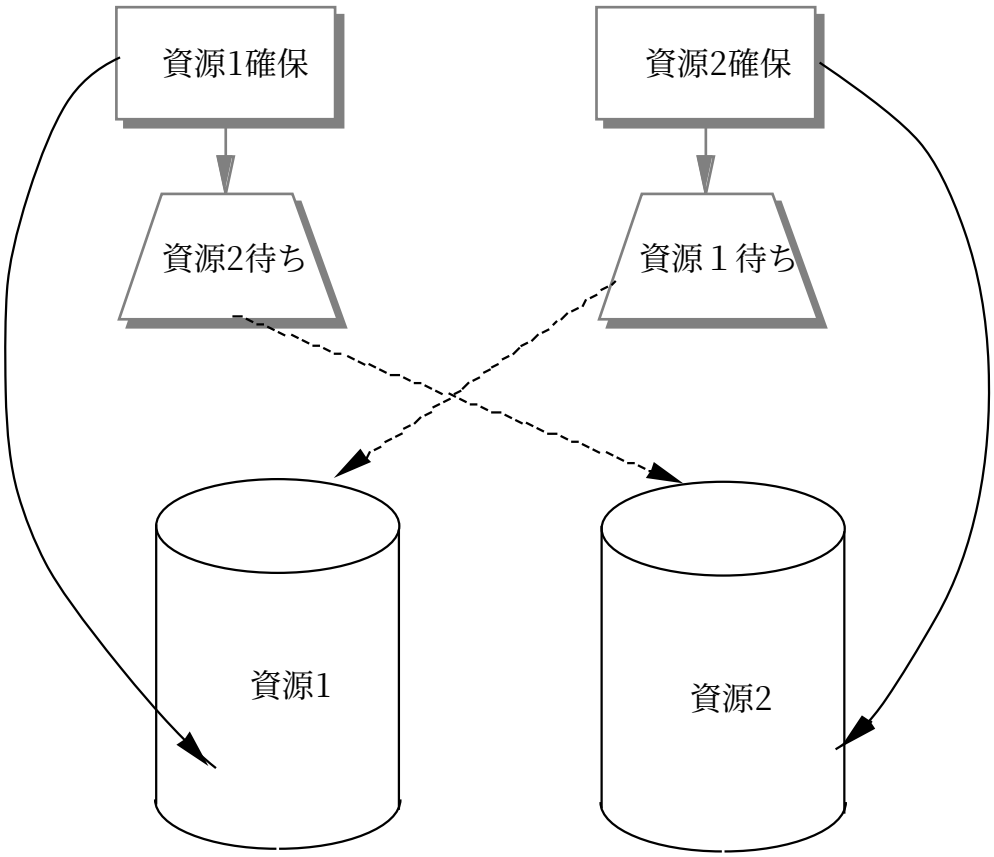
データの一貫性制御（integrity control)
(1)正当性確認（データの一貫性（Data Integrity）チェック）　ファイル（一般にはデータベース）の中のデータがあるべき正しい状態にあることをデータ定義に従ってチェックすること（誤入力対策などに有効、ただし完璧は尽せない）。フィールド値が一定の範囲にあることなど（例年齢は正の値、免許証は有りか無しかのいずれかなど）。（キーについて主キーは空値であったり重複してはならない。）複数フィールド間の関係定義とチェックもあり得る。
(2) 障害回復　万一の事故でデータを失ったり、データが矛盾したとき正常状態に復帰させる

バックアップ（ダンプ）：データの保存
リストア　保存したデータを基に、バックアップ時点にデータを戻すこと。
チェックポイント（検査時点）（バックアップの上）データ操作を開始する時点。
トランザクション
データベースに対する論理的にまとまりのある一連の操作。（データベースに対する読み、書き、正常終了操作（コミット）または異常終了操作（ロールバック）などを含む）。トランザクションは有効か無効かいずれかになるように処理する（トランザクションの原子性）。
ログファイル（ジャーナル）　データベースの操作を記録したもの
コミット　トランザクションの結果が正常の場合データベースにそれを反映すること。
ロールバック（後退復帰）：異常時等に現トランザクションで行った操作をすべて取り消し、トランザクション前（チェックポイント）の状態に復帰すること
前進復帰：ログファイルを用いて、チェックポイント時点より先の時点までの操作をやり直すこと
(3) アクセス管理機能　データベースの部分ごと、ユーザ（ユーザグループ）ごとに、読みだけ、読み書きを許可する機能
(4) 同時実行制御　データベースを同時に操作するとき、矛盾する結果を生まないように制御すること（例、空き座席の問い合せと予約）鍵、デッドロック



同時実行制御関連の話題

ロック（排他制御）について
・読み終わらない内に他人に書き換えられないよう他人に読みは許すが書込は許さないロック（共有ロック）を掛ける。
・書込中は他人に読み書きとも許さないロック（排他ロック）を掛ける。
・ある人が共有ロックまたは排他ロックしようとするとき、他人がすでに同じ対象にロックを掛けているときは、そのロックが開くまで待たされる。



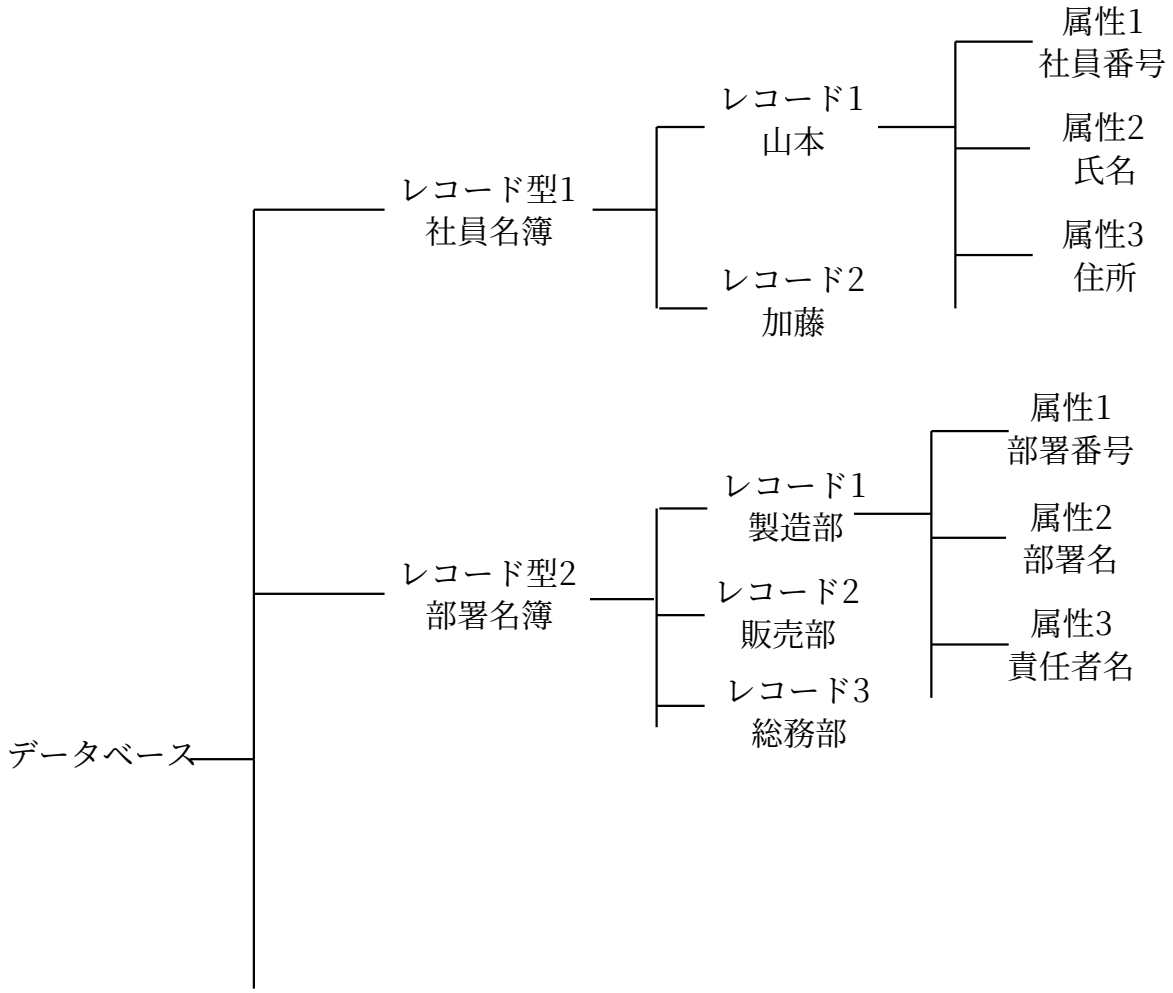
ある資源を確保するために待ちとなるとき、すでにその資源に掛かっている鍵の解除条件を自分が有している場合には永久に待ちとなってしまう場合（デッドロック）が存在する。一切デッドロックが発生しないようにすると効率が落ちるので、デッドロックを検出して状況を解消するようにする工夫がなされることが多い。

データモデルの類型、データベースの歴史、集合論についての復習

（復習　何らかの意味で独立した対象を属性（フィールド）の集まりで表したものをレコードと呼ぶ）

１つのファイルのレコード内の属性が、別のファイルと関係するとき、関係の仕方をレコード間情報と呼ぶ。

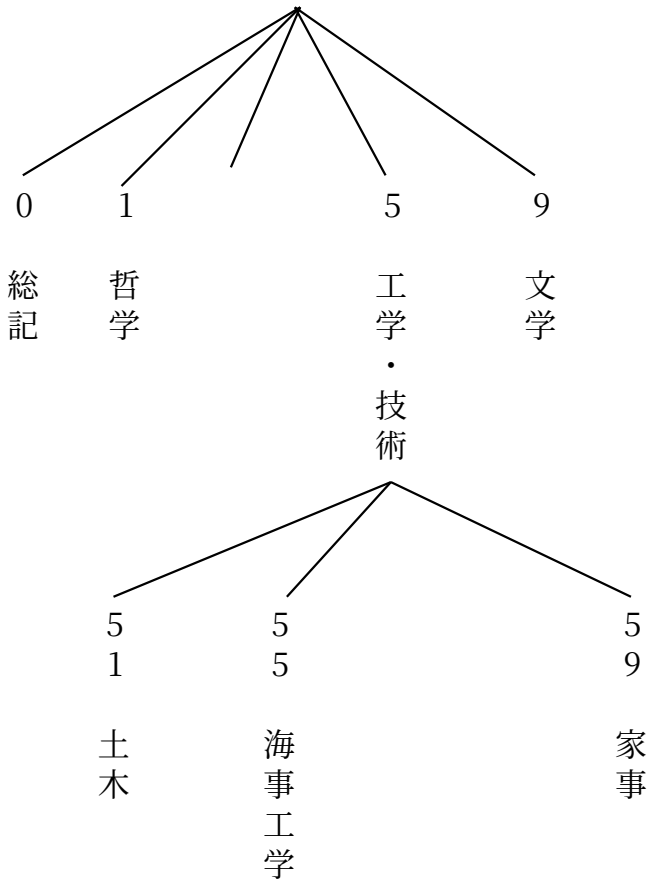
（データベースの構造例）



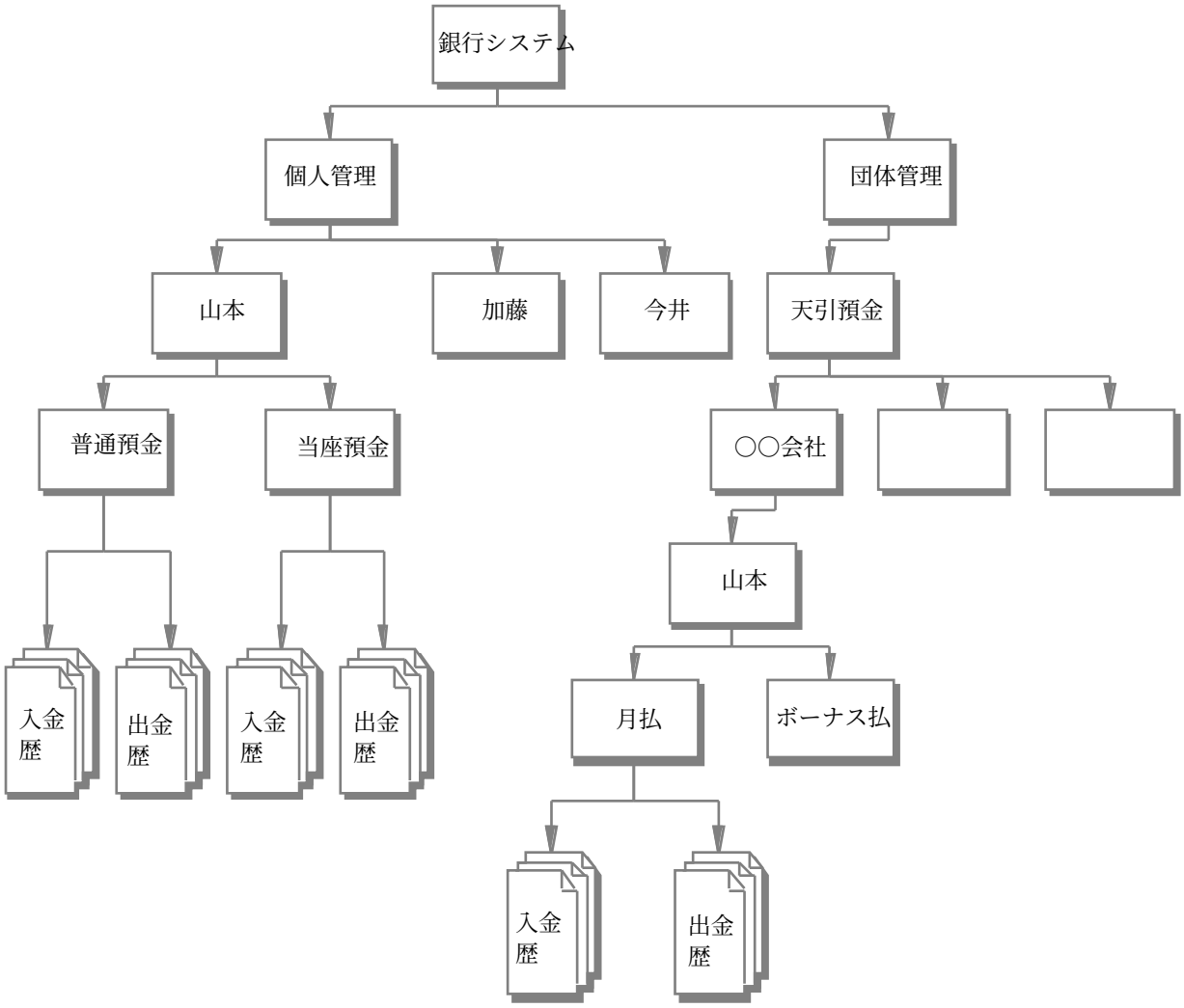
データモデルの類型
1.階層モデル
2.網モデル
3.関係（リレーショナル）モデル

階層モデル　木を用いる

1:n の関係しか表せない



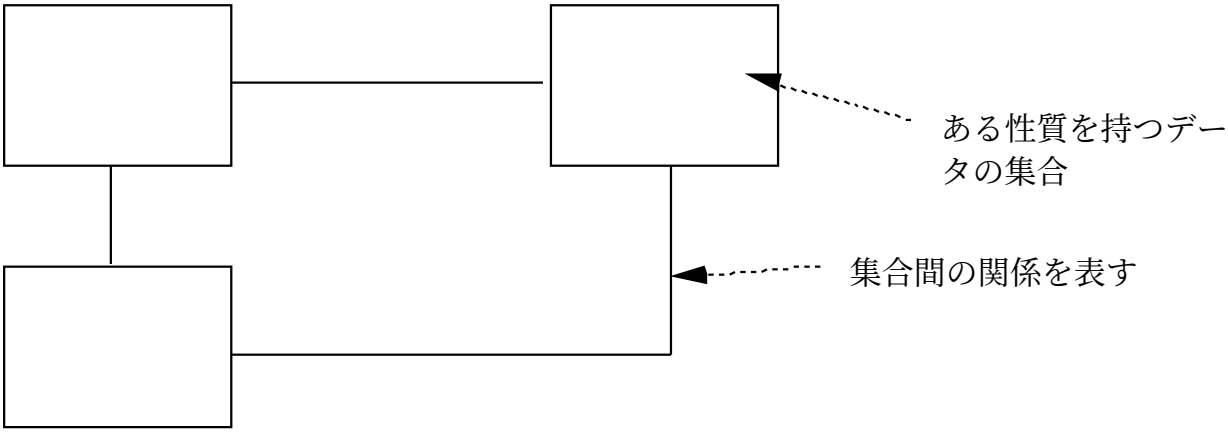
階層モデルの例

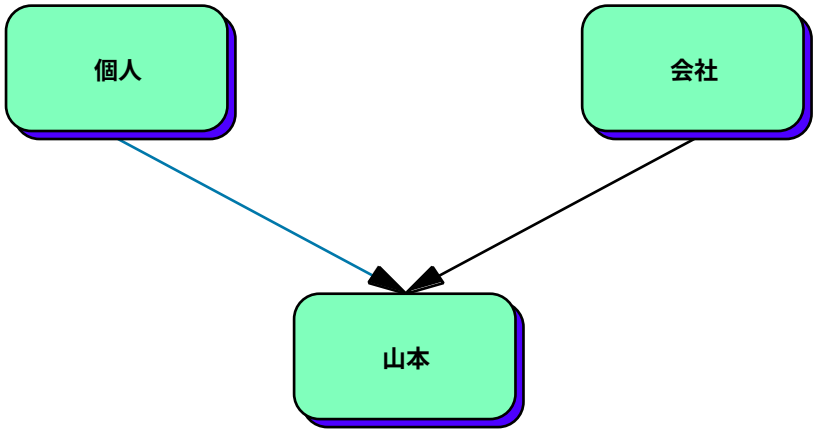


枝分れが一意に定まらないと不都合が生じる。

上記の例で山本（個人管理、団体管理）は実際には同一人物でもデータ構造上は別。
山本氏が引越すとデータを 2 箇所とも直さねばならない、矛盾が起ってもシステムは知らない。

網モデル　親子集合を用いる



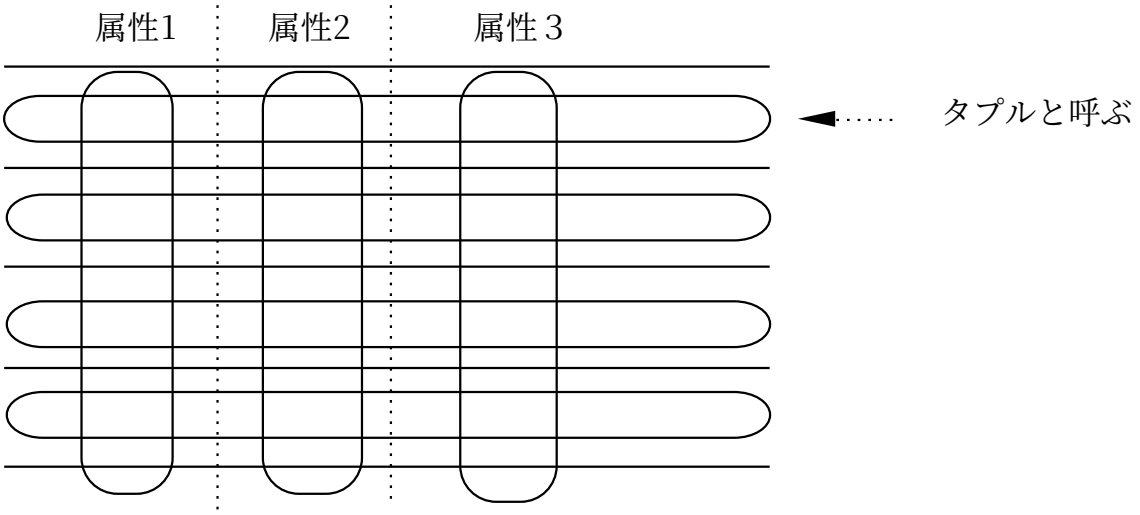


網モデルでは

網型モデルではレコード型のほかリンク型があり、このようなレコード間のつながりも許される。基本とするのは親子集合型で、例えばある組織とその組織に属する構成員を各組織レコードから、構成員に向かってのリンクで表す。

網型データ構造　網型データ構造は柔軟性があり階層モデルを表すこともでき、効率の追求もできる。ただし内部構造が複雑になり維持管理に負担が掛りがち。

関係モデル
表（だけ）によって表す



行（row）と列（column）によってデータを構成し、行をタプル（tuple）と呼びひとまとまりの実体（名簿の場合 1 人分のデータに相当）に、列を属性（attribute）と呼ぶ（名簿の場合、学籍番号、名前など）。

1970 年 IBM コッド　理論研究（集合論に基礎を置く）から生れたもので、表現の仕方が理想的。ただし、初期には効率のよいシステムの実現は困難と見られた。現在は主流となっている。

データベースの歴史ほか
・1959 年 W.C. McGee 今日のデータベースの源流となる考え方を提唱。
・1959 年 CODASYL（The Conference on Data Systems Languages）発足、事務用計算機言語 COBOL（Common Buisiness Oriented Language) に精力的に取り組む。データベースに関し CODASYL 方式を推進（CODASYL 方式の源流は GE 社の IDS と言われている）。CODASYL 方式には網モデルの考え方が取り入れられている。
・CODASYL 方式による多数の商用データベースが作られた。
・商用データベースシステムには階層構造モデルによるものもある。
・関係モデルに基づく言語 SQL が開発されている。CODASYL 方式と SQL の統一を目指す立場から NDL が開発された。

3 つのモデルに属さないものもある。

計算機の発展初期にはメインフレーム計算機と呼ばれるものが主流の時代があり、そこで大きな影響力を持っていた IBM 社が提供した階層構造モデルによるデータベースが

使われた。この時代、CODASYL が提唱した CODASYL 方式が次第に主流になった。CODASYL 方式の規格は次第に洗練されていったが、システム作成には相当の熟練を要した。関係モデルに基づくシステムは単純で分かりやすいので、初期に非難された効率の悪さを克服して次第に主流となり今日にいたっている。今日でも一部に CODASYL 方式は使われているが、新たに作られるシステムは大部分関係モデルに基づくシステムである。

（実際のシステム例：ACCESS、オラクル、4th Dimension、ファイルメーカーPro、桐）

参考　カード型データベース
パソコンなどでデータベースソフトと呼ぶものにはカード型データベースが多い。概念的には１つのレコードに相当するデータをカードの上に書き、それを多数寄せ集め自由に検索や書き換えできる。またレイアウトを選んだり作ったりできるものがある。これらはそれなりに便利なものであるが、この授業で単ファイルと呼ぶものに相当する。本格的データベースの簡易版と見ることもできる。ユーザインタフェースに関しては充実している例が多い。

集合論についての復習

集合 A が要素 $a_1, a_2, \cdots, a_n$ からなることを

$A=\{ a_1, a_2, \cdots, a_n\}$ と書く（外延的記述）。

また

$A=\{ a \mid a$ は…である$\}$ と要素に共通の性質を書いて表す（内包的記述）。

集合では　$A=\{ 1, 2, 3 \}$ も　$A=\{ 3, 1, 2 \}$ も同じ意味となる（順番に意味がない、要素重複せず）。

集合の例

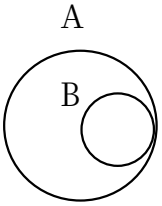
課程={商船システム, 流通情報, 交通機械}

偶数= $\{ N \mid N \div 2$ の余り$= 0 \}$

a が集合 A の要素であることを、a は集合 A に属するといい、$a \in A$ と書く。

a が集合 A に属さないとき、$a \notin A$ と書く。

集合 A の要素が集合 B のすべての要素を含んでいるとき、B は A の部分集合といい、$A \supset B$ または $B \subset A$ と書く。



2 つの集合 A、B があるとき、A から 1 要素、B から 1 要素を選んで組にする、組合わせ要素から得られる集合を A と B の直積といい $A \otimes B$ で表す。

$A=\{ a_1, a_2, a_3 \}$

$B=\{ b_1, b_2 \}$ とすると

$A \otimes B = \{ (a_1, b_1), (a_1, b_2), (a_2, b_1), (a_2, b_2), (a_3, b_1), (a_3, b_2) \}$

直積の要素を組（tuple タプル）と呼ぶ。